



РАЗРАБОТКА ПРИЛОЖЕНИЙ ДЛЯ МОБИЛЬНЫХ ОПЕРАЦИОННЫХ СИСТЕМ «ANDROID»

Электронный учебно-методический комплекс
Учебная программа учреждения высшего образования по учебной
дисциплине для специальности высшего образования второй ступени
(магистратуры):

1-31 81 06 «Веб-программирование и интернет-технологии»
физико-математического факультета

Брест
БрГУ имени А.С. Пушкина
2016

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 1 из 469

Назад

На весь экран

Закрыть



Рецензенты:

Проректор по научной работе

Брестского государственного университета имени А.С. Пушкина,
кандидат физико-математических наук, доцент

А.Е. Будько

**Кафедра информатики и прикладной математики Брестского
государственного технического университета**

Кондратюк, А.П.

Разработка приложений для мобильных операционных систем «Android» : ЭУМК для студ. второй ступени (магистратуры) специальности 1-31 81 06 «Веб-программирование и интернет-технологии» физ.-мат. фак. / А.П. Кондратюк ; Брест. гос. ун-т им. А.С. Пушкина, каф. ПМиИ. – Брест : электрон. издание БрГУ, 2016. – 469 с.

ЭУМК написан в соответствии с действующей базовой программой по дисциплине «Разработка приложений для мобильных операционных систем "Android"» и ставит своей целью облегчить самостоятельную работу студентов с теоретическим материалом при подготовке к лекциям, практическим занятиям и зачету.

Предназначено для студентов специальности 1-31 81 06 «Веб-программирование и интернет-технологии».

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 2 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



СОДЕРЖАНИЕ

Предисловие	8
Глава 1 Программное обеспечение	9
§1. Введение	9
§2. Устройство платформы Android	25
§3. Обзор сред программирования	34
§4. Эмуляторы	47
Глава 2 Разработка приложений	51
§5. Возможности отладки на реальных устройствах	51
§6. Примеры приложений	52
§7. Введение в разработку	55
§8. Основные виды Android-приложений	57
§9. Безопасность	60
§10. Архитектура приложения, основные компоненты	62
10.1 Активности (Activities)	71
10.2 Сервисы (Services)	76
10.3 Контент-провайдеры (Content Providers)	82
10.4 Приемники широковещательных сообщений (Broadcast Receivers)	86
§11. Манифест приложения	88
§12. Ресурсы	92

Начало

Содержание

◀ ▶

◀▶

Страница 3 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 4 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

§13. Основы разработки интерфейсов мобильных приложений	97
13.1 Визуальный дизайн интерфейсов	98
13.2 Строительные блоки визуального дизайна интерфей- сов	101
13.3 Элементы управления и дизайн навигации	107
13.4 Рекомендации по проектированию GUI под Android .	127
§14. Основы разработки многооконных приложений	137
14.1 Многооконные приложения	137
14.2 Работа с диалоговыми окнами	139
14.3 Особенности разработки приложения, содержащего несколько активностей	148
14.4 Перелистывание (Swipe)	151
§15. Использование возможностей смартфона в приложениях . .	153
15.1 Отличительные особенности смартфонов	154
15.2 Сенсорное (touch) управление	157
15.3 Работа с мультимедиа	164
15.4 Использование встроенной камеры	172
15.5 Взаимодействие с системами позиционирования . .	173
15.6 Другие сенсоры и датчики	177
§16. Использование библиотек	182
16.1 Библиотеки	182
16.2 Обзор популярных библиотек	185



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 5 из 469

Назад

На весь экран

Закрыть

16.3	Безопасность использования подключаемых библиотек	198
§17.	База данных и мультимедия в Android	200
17.1	Основы работы с базами данных, SQLite	201
17.2	Анимация	207
17.3	2D и 3D графика	213
17.4	Основные принципы разработки игровых приложений для смартфонов	216
Глава 3	Лабораторный практикум	218
§18.	Лабораторная работа №1	218
18.1	Введение	218
18.2	Установка среды	219
18.3	Создание проекта	227
18.4	Запуск проекта на эмуляторе устройства	240
18.5	Запуск проекта на устройстве	247
§19.	Лабораторная работа №2	260
19.1	Введение	260
19.2	Создание приложения и изучение его структуры	261
19.3	Настройка интерфейса приложения	275
19.4	Реализация логики приложения	287
19.5	Задания для самостоятельной работы:	292
19.6	Немного о работе с эмулятором	293



Кафедра ПМиИ

19.7	Заключение	295
§20.	Лабораторная работа №3	298
20.1	Введение	298
20.2	Создание прототипа интерфейса	299
20.3	BuildingBlocks или элементы для построения интерфейса	348
20.4	Задание	362
§21.	Лабораторная работа №4	362
21.1	Введение	363
21.2	Создание многоэкранного приложения со списком	363
21.3	Создание диалогового окна	373
21.4	Создание приложения со слайдингом из шаблона	383
21.5	Задание для самостоятельного выполнения	389
§22.	Лабораторная работа №5	389
22.1	Введение	389
22.2	Распознавание всех поддерживаемых жестов	390
22.3	Распознавание только части поддерживаемых жестов	394
§23.	Лабораторная работа №6	401
23.1	Введение	401
23.2	Создание набора жестов	402
23.3	Использование созданных жестов в приложении	406
23.4	Заключение	411
§24.	Лабораторная работа №7	415

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 6 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 7 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

24.1	Введение	416
24.2	Создание приложения	416
24.3	Настройка интерфейса и реализация логики активности для работы с камерой	417
24.4	Настройка интерфейса и реализация логики активности для воспроизведения аудио и видео	423
24.5	Настройка интерфейса и реализация логики активности для просмотра изображений	429
24.6	Настройка интерфейса и реализация логики главной активности приложения	433
§25.	Лабораторная работа №8	453
25.1	Введение	453
25.2	Разработка приложения, получающего координаты устройства и отслеживающего их изменение	453
25.3	Заключение	458
	Вопросы и задания для самоконтроля	461
	Литература	462

ПРЕДИСЛОВИЕ

Настоящий электронный учебно-методический комплекс предназначен для студентов специальности 1-31 81 06 «Веб-программирование и интернет-технологии» физико-математического факультета. Он написан в соответствии с действующей базовой программой по дисциплине «Разработка приложений для мобильных операционных систем "Android"».

В электронном издании излагается теоретический материал, содержащий вопросы: устройство платформы Android, обзор сред программирования и эмуляторов, основные виды Android-приложений и их безопасность, архитектура приложения и его компоненты, манифест и ресурсы приложения, основы разработки многооконных мобильных приложений и их интерфейсов, использование возможностей смартфона, обзор библиотек Android, работа с базами данных и мультимедиа. Теоретический материал иллюстрируется примерами.

Учебно-методический комплекс ставит своей целью облегчить самостоятельную работу студентов с теоретическим и практическим материалом при подготовке к лекциям, лабораторным занятиям и зачету.

Автор.



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 8 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

ГЛАВА 1

Программное обеспечение

§1. Введение

Аннотация: Целью лекции является описание основных принципов разработки для ОС Android. В лекции рассказывается об устройстве платформы Android, приводится обзор сред программирования, описываются возможности отладки на эмуляторах и реальных устройствах. Имеется большое количество разнообразных примеров и иллюстраций. В конце приведен список дополнительных источников. Лекция является обязательной для понимания следующих тем курса.

Скриншоты приложений взяты из магазина приложений Google Play из [магазина приложений Google Play](#) или сделаны самостоятельно с использованием смартфона Мегафон SP-A20i Mint на платформе Intel Medfield.

Презентацию к данной лекции можно скачать [здесь](#).

Android - операционная система для мобильных устройств: смартфонов, планшетных компьютеров, КПК. В настоящее время именно *Android* является самой широко используемой операционной системой для мобильных устройств. Подтверждение этого факта можно найти в таблице, составленной по данным аналитической компании Gartner.



Кафедра
ПМИИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 9 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 10 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Таблица 1.1. Мировые продажи смартфонов конечным пользователям, распределение по ОС

Операционная система	Продано (тыс.ед.) III кв. 2013	Доля рынка (%) III кв. 2013	Продано (тыс.ед.) III кв. 2012	Доля рынка (%) III кв. 2012
Android	205022,7	81,9	124552,3	72,6
iOS	30330,0	12,1	24620,3	14,3
Microsoft	8912,3	3,6	3993,6	2,3
BlackBerry	4400,7	1,8	8946,8	5,2
Bada	633,3	0,3	4454,7	2,6
Symbian	457,5	0,2	4401,3	2,6
другие	475,2	0,2	683,7	0,4
Общее колво:	250231,7	100,0	171652,7	100,0

Источник: Gartner (ноябрь 2013)

Внимательное изучение таблицы позволяет увидеть подавляющую популярность смартфонов под управлением ОС *Android* в мире, доля таких устройств не первый год превышает половину от общего числа купленных смартфонов. Кроме всего прочего, эта популярность продолжает расти. Очевидно, что армия пользователей смартфонов под управлением *Android* будет искать дополнительные приложения для своих устройств, в связи с этим умение разрабатывать эти самые приложения может принести много пользы своему владельцу. Например, можно разрабатывать



для себя полезные, интересные, занимательные (нужное подчеркнуть) приложения, а можно, разведав обстановку и осмотревшись, сделать разработку мобильных приложений своей профессиональной деятельностью, основной или дополнительной.

Курс "Разработка приложений для смартфонов на ОС *Android*" предоставляет возможность приобрести начальные навыки разработки мобильных приложений, если остановиться только на первой его части. Изучение полной версии курса позволит сделать серьезный шаг к тому, чтобы профессионально разрабатывать мобильные приложения и получать от этой деятельности не только моральное, но и материальное удовлетворение.

Данная лекция является первой для всего курса, призвана ввести читателя в курс дела. В первую очередь в ней рассматриваются вопросы становления и развития ОС *Android*. Для успешного программирования под *Android* необходимо понимать внутреннюю организацию и архитектуру этой платформы, а также полезно знать, какие инструменты и среды разработки можно использовать. Этим вопросам посвящена основная часть лекции. Кроме того, в лекции рассматриваются особенности запуска и отладки мобильных приложений.

Немного истории

Рассмотрим, как все начиналось. В 2003 году в Пало Альто, штат Калифорния Энди Рубин с единомышленниками (Рич Майнер, Ник Сирс

Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 11 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 12 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

и Крис Уайт) основали компанию Android Inc. Поначалу в компании занимались проектированием мобильных гаджетов, которые на основе геолокационных данных автоматически подстраивались под нужды пользователей.

В августе 2005 года Android Inc. стала дочерней компанией Google. Энди Рубин, Рич Майнер и Крис Уайт остались в Android Inc. и начали работать над операционной системой, базирующейся на ядре Linux. В Google задумали реализовать мощнейшую платформу, пригодную к использованию на тысячах различных моделей телефонов. В связи с этим был создан Open Handset Alliance (ОНА) - консорциум, состоящий из более 80 компаний, направляющий свои усилия на разработку открытых стандартов для мобильных устройств. В состав ОНА входят такие гиганты, как Google (организатор и идейный вдохновитель), HTC, Sony, Dell, Intel, Motorola, Qualcomm, Texas Instruments, Samsung Electronics, LG Electronics, T-Mobile, Sprint Corporation, NVIDIA и многие другие.

Первая версия Android была представлена 23 сентября 2008 года, версии было дано название Apple Pie (можно заметить созвучие с прямым конкурентом). Далее так повелось, что название каждой очередной версии представляет какой-либо десерт, при этом первые буквы наименований в порядке версий соответствуют буквам латинского алфавита по порядку. С развитием обновлений Android можно познакомиться в приведенной ниже таблице.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 13 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Версия, логотип, дата выхода	Основные возможности
Android 1.0 Apple Pie	<p>Первый стабильный релиз, основан на ядре Linux 2.6.25.</p> <p>Поддерживается:</p> <p>файловая система FAT32, стек интернет-протоколов TCP/IP;</p> <p>протоколы передачи данных: 802.11 b/g Wi-Fi, Bluetooth 2.0 EDR, GPRS, EDGE, UMTS, HSDPA;</p> <p>фото и видео съемка, однако недостаточно опций для настройки разрешения камеры, баланса белого и др.;</p> <p>сенсорные дисплеи и landscape режим отображения данных на экране, максимальная цветность дисплея - 16 бит (тип HVGA);</p> <p>виджеты и ярлыки на рабочем столе (Home Screen), сменные обои;</p> <p>регулярные телефонные функции, контроль вызова, конференц-связь, легкая интеграция с контактами;</p> <p>полноценный web-браузер на движке WebKit, HTML, XHTML;</p> <p>e-mail клиент, протоколы POP3, IMAP4, SMTP;</p>



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 14 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

медиа проигрыватель, позволяющий управлять, импортировать, проигрывать медиа контент в различных форматах.

Базовые приложения:

будильник; калькулятор; календарь; камера; контакты; сообщения (в том числе MMS); настройки; голосовой набор.

Минимальные системные требования для запуска и работы: архитектура ARM, 128 MB RAM, 256 MB ROM.

Вideo презентация: [здесь](#)

Android 1.1
Banana
Bread
февраль
2009
(API level:
2)

Нововведения:
Исправлены проблемы:
с будильником; со спящим режимом; с вызовом дисплея набора номера; в IMAP ошибки запроса пароля и др.
Изменения API.
Добавлены подробности и отзывы к картам.
Добавлена поддержка вложений из MMS.
Локализации:
Английская US (en_US)
Немецкая (de)
Подробности: [здесь](#)

Android 1.5
Cupcake
апрель 2009
(API level:
3)

Нововведения:

Поддержка экранной клавиатуры (); акселерометра; видеозапись и воспроизведение видео; приложение для работы с YouTube; стерео Bluetooth; функция копирования и вставки между приложениями (copy& paste).

Локализации:

добавились очень многие, в том числе и русская (ru_RU).

Система:

новое Linux ядро (версия 2.6.27); автоматическая проверка и восстановление файловой системы на SD card; новое приложение для просмотра СТК меню оператора (SIM Application Toolkit 1.0).

Изменения в пользовательском интерфейсе (UI): изменено большинство UI-элементов, добавлены новые виджеты; определение режима (книжный или портретный) работы программы; анимированное переключение между окнами.

Подробности: [здесь](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 15 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 16 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Android 1.6
Donut сен-
тябрь 2009
(API level:
4)

Нововведения:

Система:

новое ядро Linux (версия 2.6.29); поддержка сотового стандарта CDMA; поддержка разрешений дисплеев: QVGA и WVGA; обновленный медиа-движок OpenCore 2; движок синтеза речи (многоязыковой); Gesture Builder поддержка возможности (для разработчиков) создавать, сохранять, загружать и распознавать жесты, прикреплять к определенным действиям.

Пользовательские возможности:

строка быстрого поиска (прямо с рабочего стола); история и закладки в браузере, контакты и поиск в интернете; возможность подключаться к видам VPN: L2TP/IPSEC pre-shared key based VPN, L2TP/IPSEC certificate based VPN, L2TP only VPN, PPTP only VPN; ускорение работы камеры; индикатор работы батареи позволяет увидеть сколько энергии потребляют работающие программы и сервисы. Обновленный Android Market.

Подробности: [здесь](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 17 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Android 2.0,
2.0.1, 2.1
Eclair ок-
тябрь 2009
(API level:
5)

(API level:
6)
(API level:
7)

Нововведения в 2.0:
поддержка работы нескольких почтовых аккаунтов одновременно, возможность использования совместных папок (входящие, исходящие) для всех аккаунтов;
быстрый способ работы с контактами Quick Contact; поиск по всем сохраненным SMS и MMS сообщениям, удаление старых после заданного срока;

возможности камеры: вспышка, цифровой зум, сценические режимы, баланс белого, цветовые эффекты, макрофокусировка;

улучшенное расположение виртуальных клавиш клавиатуры, поддержка комбинированных нажатий клавиш (технология мультитач), усовершенствованная функция автодополнения;

поддержка HTML5, версии Bluetooth 2.1, новых профилей OPP и PBAP.

Подробности:[здесь](#)

Нововведения 2.0.1:

подрелиз версии 2.0, включающий в себя незначительные изменения в функционале и по большей части bugfix-ом версии 2.0.

Подробности: [здесь](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 18 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Нововведения 2.1:

основным новшеством, представляющим интерес для конечного пользователя, стало добавление анимированных (живых) обоев, остальные изменения в Framework API, представляют интерес для разработчиков.

Подробности: [здесь](#)

Android 2.2
Froyo май
2010
(API level:
8)

Нововведения:
рост производительности примерно в 3-5 раз за счет использования Dalvik Virtual Machine Just-in-Time компилятора;
возможности установки приложений на SD-карту, переноса приложений из внутренней памяти на карту и обратно;
возможность использовать смартфон в качестве точки доступа к интернету, в качестве модема для других устройств;
поддержка Adobe Flash;
V8 javascript существенно повысил скорость работы штатного браузера.
Подробности: [здесь](#)



Кафедра ПМиИ

Android 2.3,
2.3.3
Gingerbread
декабрь
2010
(API level:
9)
(API level:
10)

До весны 2013 года самая массовая версия на рынке.
Нововведения:
новое ядро Linux 2.6.35; поддержка открытых мультидийных стандартов (VP8 и WebM), форматов ACC/AMR, звуковых эффектов и эквалайзера, фронтальной камеры (интеграция с VOIP(SIP)); обновленный GUI: уменьшение времени доступа к функциям, повышение общей энергоэффективности системы;
улучшение стандартной клавиатуры системы: поддержка словарей, технологии мультитач, упрощенное выделение и копирование текста;
поддержка технологии NFC; расширение возможностей работы с датчиками положения телефона.

Подробности:[здесь](#)

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 19 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 20 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Android 3.0-3.2	Специальная версия для работы на планшетах (MID, tablets).
Honeycomb февраль 2011 (API level: 11)	Нововведения 3.0: новое ядро Linux 2.6.36; поддержка файловой системы ext4, файловой системы FUSE для MTP устройств; поддержка режима USB-хост для работы с клавиатурой, мышью и USB-хабами; поддержка MTP/PTP; виртуальная машина Dalvik: поддержка и оптимизация SMP, множество улучшений JIT, улучшенный сборщик мусора;
(API level: 12)	совершенно новый интерфейс с полноценной оптимизацией под устройства с большими экранами; поддержка виртуальных рабочих столов, каждый из которых может иметь свой набор виджетов и ярлыков; улучшенные и переработанные базовые приложения: Browser, e-mail и др.
(API level: 13)	Подробности: здесь
	Нововведения 3.1: поддержка работы кардридера; усовершенствован GUI: доработан менеджер задач, позволяющий переключаться между множеством различных приложений (в 3.0 только 5 программ



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 21 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

одновременно);

возможность менять размер виджетов, как по горизонтали, так и по вертикали.

Подробности: [здесь](#)

Нововведения 3.2:

расширен спектр поддерживаемых планшетов; возможность автоматического масштабирования приложений для отображения на более крупных экранах.

Подробности: [здесь](#)

Android 4.0, 4.0.3 Ice Cream Sandwich
ноябрь 2011 (API level: 14)
(API level: 15)

Нововведения:
поддержка и смартфонов, и планшетов; поддержка новых процессорных архитектур, помимо ARM поддержка Intel x86 и MIPS;
возможность разблокировки экрана: при помощи функции определения лица; жестами: перетащить замочек из центра экрана на иконку приложения и оно запустится;
многозадачность: кнопка Recent Apps позволяет мгновенно переходить от одной задачи к другой с помощью списка в системной панели;
новые элементы управления передачей данных через сеть: в приложении Настройки можно увидеть общее использование данных по каждому типу сети,



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 22 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

объем данных, используемых каждым работающим приложением;

доступность Android 4.0 для слепых и слабовидящих пользователей, браузер поддерживает экранного чтеца, который воспроизводит все видимое активное содержимое на экране;

AndroidBeam - удобное средство обмена между двумя NFC-устройствами;

Wi-Fi Direct и Bluetooth H DP, HFP: возможность прямого подключения к соответствующим устройствам.

Подробности: [здесь](#)

Android 4.1-4.3 Jelly Bean	Нововведения 4.1: увеличена скорость прорисовки интерфейса, улучшен поиск, добавлено несколько полезных сервисов; улучшена работа со словарями, возможно использовать голосовой ввод без подключения к интернету; специальные возможности: возможность управления смартфоном с помощью жестов и голосовых подсказок, подключения устройств ввода, поддерживающих шрифт Брайля; существенно доработана функция передачи данных Beam; переработан поиск (вместо ссылок ответ на
июль 2012	
(API level: 16)	



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 23 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

запрос); голосовой поиск; Google Now: нужная информация в нужное время.

Подробности: [здесь](#)

Нововведения 4.2:

реализована поддержка нескольких пользователей (планшеты); поддержка wireless display: возможность трансляции видео и изображений на внешний экран;

возможность отображения полезной информации в режиме сна, при подключении к док-станции или на зарядке; улучшена панель уведомлений.

Подробности: [здесь](#)

Нововведения 4.3:

ускорение работы системы; более точный набор на клавиатуре; скрытая возможность управления процессами программ (необходима активация); поддержка OpenGL/ES 3.0 (не на всех устройствах).

Подробности: [здесь](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 24 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Android 4.4
Kit Kat
октябрь
2013
(API level:
19)

Нововведения:
многозадачность, оптимизация распределения ре-
сурсов между приложениями;
определитель номера работает не только с адресной
книгой (например, Google maps);
серьезная интеграция приложения Hangouts (от-
правка SMS, MMS, голосовые и видеозвонки);
в состав вошел Quickoffice, интегрированный с
Google Drive;
поддержка принтеров, подключение через прило-
жения поддерживающие печать (например, Google
Cloud Print, HP ePrint);
поддержка стандарта Wi-Fi Miracast, позволяющий
вещать изображение на телевизор;
возможность захвата экрана для записи видео.

Подробности: [здесь](#)

Таблица 1.2. История обновлений ОС Android

§2. Устройство платформы Android

Платформа *Android* объединяет операционную систему, построенную на основе ядра ОС Linux, промежуточное *программное обеспечение* и встроенные мобильные приложения. Разработка и развитие мобильной платформы *Android* выполняется в рамках проекта AOSP (*Android Open Source Project*) под управлением ОНА (*Open Handset Alliance*), руководит всем процессом поисковый гигант Google.

Android поддерживает фоновое выполнение задач; предоставляет богатую библиотеку элементов пользовательского интерфейса; поддерживает 2D и 3D графику, используя OpenGL стандарт; поддерживает *доступ* к файловой системе и встроенной базе данных SQLite.

С точки зрения архитектуры, система *Android* представляет собой полный программный *стек*, в котором можно выделить следующие уровни:

- **Базовый уровень (Linux Kernel)** - уровень абстракции между аппаратным уровнем и программным стеком;
- **Набор библиотек и среда исполнения (Libraries & Android Runtime)** обеспечивает важнейший базовый функционал для приложений, содержит виртуальную машину Dalvik и базовые библиотеки Java необходимые для запуска Android приложений;
- **Уровень каркаса приложений (Application Framework)** обес-



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 25 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

печивает разработчикам доступ к API, предоставляемым компонентами системы уровня библиотек;

- **Уровень приложений (Applications)** - набор предустановленных базовых приложений.

Наглядное изображение архитектуры на рисунке 1.

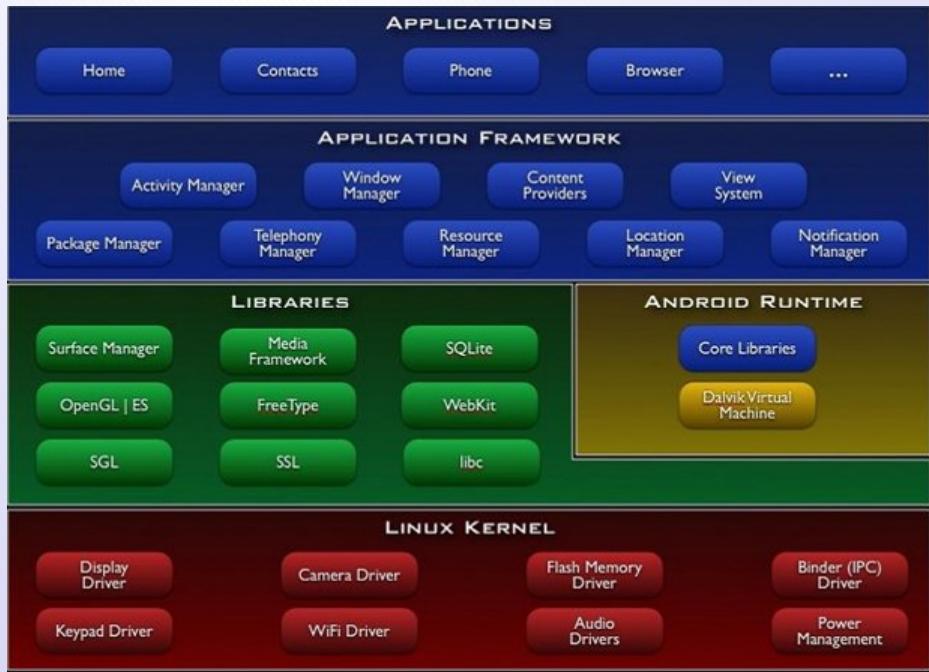


Рисунок 1. Архитектура Android

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 26 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 27 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Рассмотрим компоненты платформы более подробно.

В основании компонентной иерархии лежит *ядро* ОС Linux 2.6 (несколько урезанное), служит промежуточным уровнем между аппаратным и программным обеспечением, обеспечивает функционирование системы, предоставляет системные службы ядра: *управление памятью*, энергосистемой и процессами, обеспечение безопасности, работа с сетью и драйверами.

Уровнем выше располагается набор библиотек и среда исполнения. Библиотеки реализуют следующие функции:

- предоставляют реализованные алгоритмы для вышележащих уровней;
- обеспечивает поддержку файловых форматов;
- осуществляет кодирование и декодирование информации (например, мультимедийные кодеки);
- выполняет отрисовку графики и т.д.

Библиотеки реализованы на C/C++ и скомпилированы под конкретное *аппаратное обеспечение* устройства, вместе с которым они и поставляются производителем в предустановленном виде.

Рассмотрим некоторые библиотеки:



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 28 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Surface Manager

- композитный менеджер окон. Поступающие команды отрисовки собираются в закадровый буфер, где они накапливаются, составляя некую композицию, а потом выводятся на экран. Это позволяет системе создавать интересные бесшовные эффекты, прозрачность окон и плавные переходы.

Media Framework

- библиотеки, реализованные на базе PacketVideo OpenCORE. Используются для записи и воспроизведения аудио и видео контента, а также для вывода статических изображений. Поддерживаются форматы: MPEG4, H.264, MP3, AAC, AMR, JPG и PNG.

SQLite

- легковесная и производительная реляционная СУБД, используется в Android в качестве основного движка для работы с базами данных.

3D биб- лиотеки

- используются для высокооптимизированной отрисовки 3D-графики, при возможности используют аппаратное ускорение. Библиотеки реализованы на основе API OpenGL|ES. OpenGL|ES (OpenGL for Embedded Systems) - подмножество графического программного интерфейса OpenGL, адаптированное для работы на встраиваемых системах.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 29 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

FreeType

- библиотека для работы с битовыми картами, для растеризации шрифтов и осуществления операций над ними.

LibWebCore- библиотеки браузерного движка WebKit, используемого также в известных браузерах Google Chrome и Apple Safari.

SGL (Skia Graphics Engine)

- открытый движок для работы с 2D-графикой. Графическая библиотека является продуктом Google и часто используется в других программах.

SSL

- библиотеки для поддержки одноименного криптографического протокола.

Libc

- стандартная библиотека языка C, а именно ее BSD реализация, настроенная для работы на устройствах на базе Linux.

Среда исполнения включает в себя библиотеки ядра, обеспечивающие большую часть низкоуровневой функциональности, доступной библиотекам ядра языка *Java*, и виртуальную машину *Dalvik*, позволяющую запускать приложения. Каждое *приложение* запускается в своем экземпляре виртуальной машины, тем самым обеспечивается изоляция работающих приложений от ОС и друг от друга. Для исполнения на виртуальной машине *Dalvik* *Java*-классы компилируются в исполняемые файлы с расширением *.dex* с помощью инструмента *dx*, входящего



в состав *Android* SDK. DEX (Dalvik EXecutable) - формат исполняемых файлов для виртуальной машины Dalvik, оптимизированный для использования минимального объема памяти. При использовании *IDE* Eclipse и плагина *ADT(Android Development Tools)* компиляция классов Java в формат .dex происходит автоматически.

Архитектура *Android* Runtime такова, что работа программ осуществляется строго в рамках окружения виртуальной машины, что позволяет защитить ядро ОС от возможного вреда со стороны других ее составляющих. Поэтому код с ошибками или вредоносное ПО не смогут испортить *Android* и устройство на его базе, когда сработают.

На еще более высоком уровне располагается каркас приложений (*Application Framework*), архитектура которого позволяет любому приложению использовать уже реализованные возможности других приложений, к которым разрешен доступ. В состав каркаса входят следующие компоненты:

- богатый и расширяемый набор представлений (**Views**), который может быть использован для создания визуальных компонентов приложений, например, списков, текстовых полей, таблиц, кнопок или даже встроенного web-браузера;
- контент-провайдеры (**Content Providers**), управляющие данными, которые одни приложения открывают для других, чтобы те могли их использовать для своей работы;

Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 30 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 31 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

- менеджер ресурсов (**Resource Manager**), обеспечивающий доступ к ресурсам без функциональности (не несущим кода), например, к строковым данным, графике, файлам и другим;
- менеджер оповещений (**Notification Manager**), позволяющий приложениям отображать собственные уведомления для пользователя в строке состояния;
- менеджер действий (**Activity Manager**), управляющий жизненными циклами приложений, сохраняющий историю работы с действиями, предоставляющий систему навигации по действиям;
- менеджер местоположения (**Location Manager**), позволяющий приложениям периодически получать обновленные данные о текущем географическом положении устройства.

Application Framework предоставляет в распоряжение приложений в ОС *Android* вспомогательный функционал, благодаря чему реализуется принцип многократного использования компонентов приложений и ОС. Естественно, в рамках политики безопасности.

И, наконец, самый высокий, самый близкий к пользователю уровень приложений. Именно на этом уровне *пользователь* взаимодействует со своим устройством, управляемым ОС *Android*. Здесь представлен набор базовых приложений, который предустановлен на ОС *Android*. Например, *браузер*, *почтовый клиент*, *программа для отправки SMS*, *карты*,



календарь, менеджер контактов и др. Список интегрированных приложений может меняться в зависимости от модели устройства и версии *Android*. К этому уровню также относятся все пользовательские приложения.

Разработчик обычно взаимодействует с двумя верхними уровнями архитектуры *Android* для создания новых приложений. Библиотеки, система исполнения и ядро Linux скрыты за каркасом приложений.

Повторное использование компонентов других приложений приводит к идее задач в *Android*. Приложение может использовать компоненты другого *Android* приложения для решения задачи, например, если разрабатываемое приложение предполагает использование фотографий, оно может вызвать приложение, управляющее фотографиями и зарегистрированное в системе *Android*, выбрать с его помощью фотографию и работать с ней.

Для пополнения коллекции приложений своего мобильного устройства пользователь может воспользоваться приложением Google Play, которое позволяет покупать и устанавливать приложения с сервиса Google Play. Разработчики, в свою очередь, могут выкладывать свои приложения в этот сервис, Google Play отслеживает появление обновлений приложения, сообщает пользователям этого приложения об обновлении и предлагает установить его. Также Google Play предоставляет разработчикам доступ к услугам и библиотекам, например, доступ к использованию и отображению Google Maps.

Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 32 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Для установки приложения на устройствах с ОС *Android* создается *файл* с расширением *.apk (*Android package*), который содержит исполняемые файлы, а также вспомогательные компоненты, например, файлы с данными и *файлы ресурсов*. После установки на устройство каждое *приложение* "живет" в своем собственном изолированном экземпляре виртуальной машины Dalvik.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)



Страница 33 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

§3. Обзор сред программирования

Прежде чем начать разрабатывать приложения под *Android*, рассмотрим существующие инструменты, подходящие для этих целей. Можно выделить необходимые инструменты, без которых разработка мобильных приложений под *Android* просто невозможна. С другой стороны, существует большое количество вспомогательных систем, в какой-то мере упрощающих процесс разработки.

К обязательным инструментам относится *Android SDK* - набор средств программирования, который содержит инструменты, необходимые для создания, компиляции и сборки мобильного приложения. Рассмотрим кратко наиболее важные инструменты, входящие в состав *Android SDK*:

- **SDK Manager** - инструмент, позволяющий загрузить компоненты *Android SDK*. Показывает пакеты *Android SDK* и их статус: установлен (Installed), не установлен (Not Installed), доступны обновления (Update available).



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 34 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

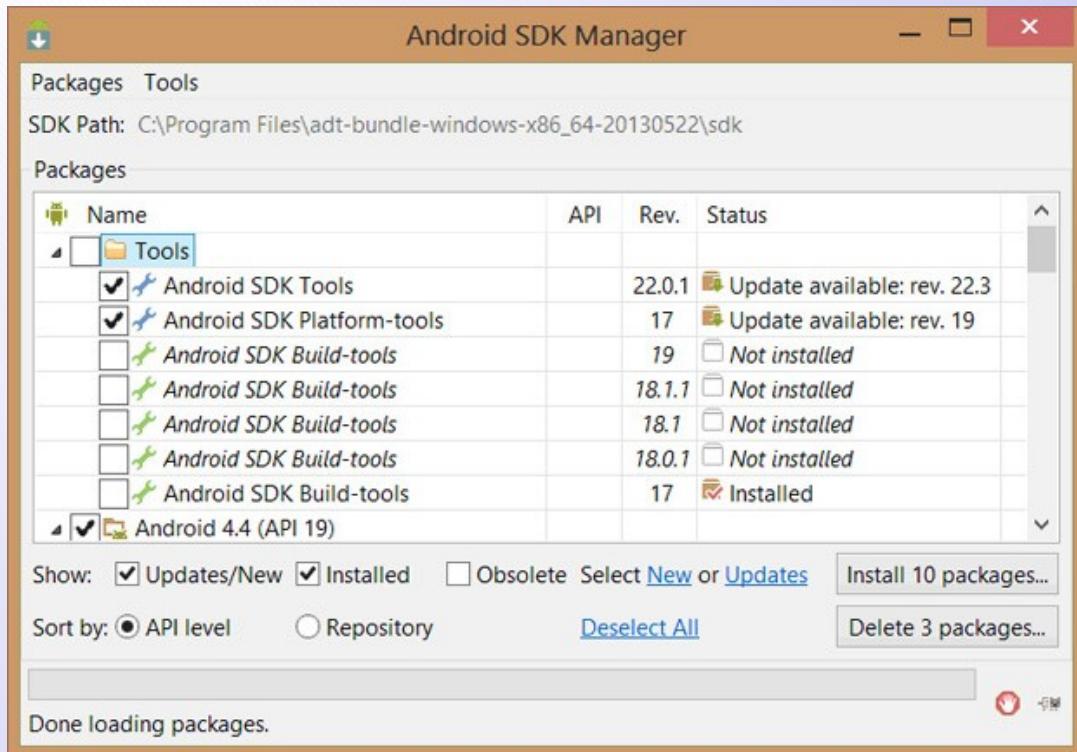


Рисунок 1. Android SDK Manager

- **Debug Monitor** - самостоятельный инструмент, предоставляющий графический интерфейс к нескольким инструментам, предназначенный для анализа и отладки Android приложений:
 - DDMS (Dalvik Debug Monitor Server) предоставляет услуги пе-

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 35 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)



[Страница 36 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

реброса портов, захват экрана устройства, информацию о потоках и динамической памяти устройства, вывод информации о действиях Android в реальном времени (logcat) и многое другое.

- Hierarchy Viewer позволяет отлаживать и оптимизировать пользовательский интерфейс Android приложения.
- Tracer for OpenGL ES - инструмент для анализа OpenGL|ES кода, используемого в мобильном приложении, позволяет захватывать команды OpenGL|ES и демонстрировать их по отдельным кадрам, что помогает понять как исполняются графические команды.

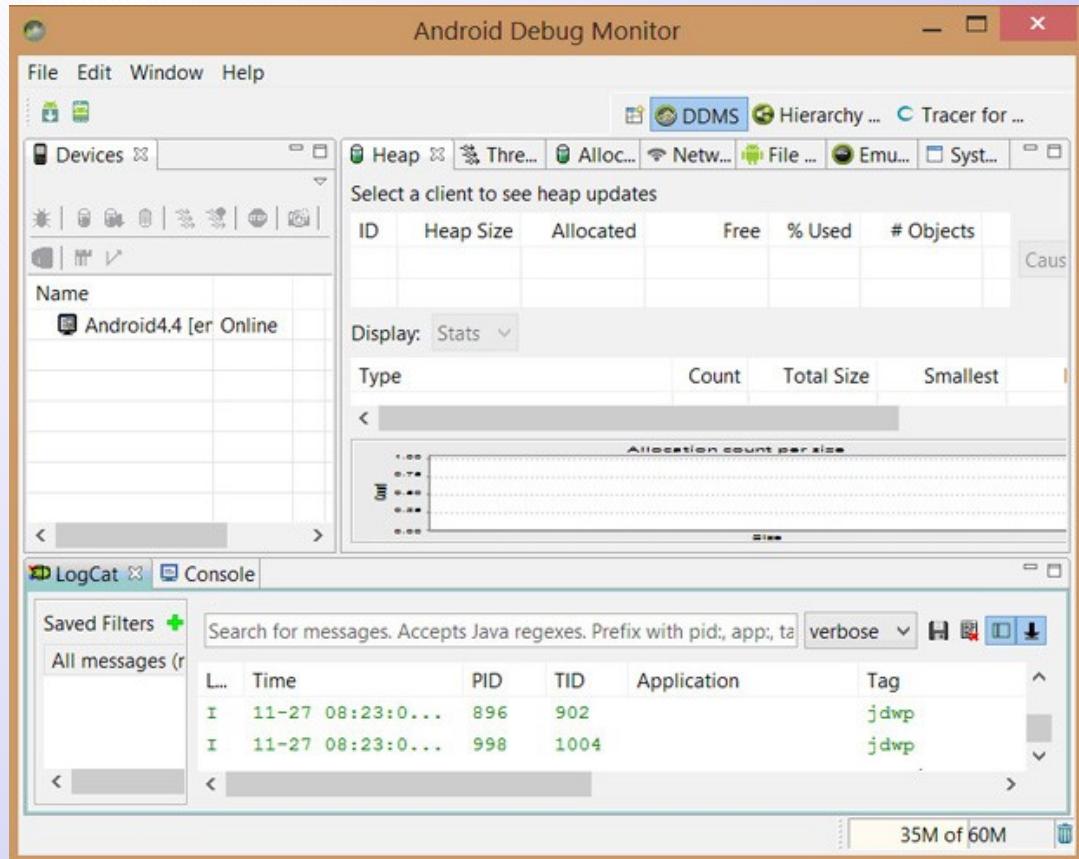


Рисунок 2. Окно инструмента Monitor

- **Android Emulator (emulator)** - виртуальное мобильное устройство, которое создается и работает на компьютере разработчика,



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 37 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 38 из 469

Назад

На весь экран

Закрыть

используется для разработки и тестирования мобильных приложений без привлечения реальных устройств.

- **AVD Manager** - предоставляет графический интерфейс для создания виртуальных Android устройств (AVDs), предусмотренных Android Emulator, и управления ими.

(В ЛР №1 подробно рассматривается создание и использование виртуального устройства).

- **Android Debug Bridge (adb)** - гибкий инструмент, позволяющий управлять состоянием эмулятора или реального Android устройства, подключенного к компьютеру. Также может использоваться для установки Android приложения (.apk файл) на реальное устройство.

Мы рассмотрели основные инструменты, входящие в состав *Android SDK*, разумеется, не все и недостаточно подробно. Для более серьезного изучения инструментов имеет смысл обратиться к сайту разработчиков [здесь](#). Для разработки мобильных приложений под *Android* уверенного владения инструментами из *SDK* вполне достаточно. Если же возникают какие-то вопросы, дополнительные инструкции по созданию проектов, компиляции, запуску из командной строки содержатся в руководстве от Google [здесь](#).

В современных условиях разработка *ПО* в большинстве случаев ведется с использованием интегрированных сред разработки (*IDE*). *IDE*



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 39 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

имеют несомненные достоинства: процесс компиляции, сборки и запуска приложения обычно автоматизирован, в связи с чем для начинающего разработчика создать свое первое *приложение* труда не составляет. Но чтобы заниматься разработкой всерьез, необходимо потратить силы и время на изучение возможностей самой среды. Рассмотрим IDE, пригодные для разработки под Android1. Для начала поговорим о двух средах разработки, которые рекомендует Google: *Android IDE (ADT)* и *Android Studio*.

Android IDE - среда разработки под *Android*, основанная на Eclipse. Предоставляет интегрированные инструменты для разработки, сборки и отладки мобильных приложений. В данном курсе *Android IDE* выбрана в качестве основной среды разработки. Возможности этой среды более подробно рассмотрены в первой лабораторной работе. Также там даны рекомендации по установке и настройке среды, созданию и запуску первого приложения как на эмуляторе, так и на реальном устройстве.

Android Studio - среда разработки под *Android*, основанная на IntelliJ IDEA. Подобно *Android IDE*, она предоставляет интегрированные инструменты для разработки и отладки. Дополнительно ко всем возможностям, ожидаемым от IntelliJ, в *Android Studio* реализованы:

- поддержка сборки приложения, основанной на Gradle;
- специфичный для *Android* рефакторинг и быстрое исправление дефектов;



Кафедра ПМиИ

- lint инструменты для поиска проблем с производительностью, с юзабилити, с совместимостью версий и других;
- возможности ProGuard (утилита для сокращения, оптимизации и обfuscации кода) и подписи приложений;
- основанные на шаблонах мастера для создания общих Android конструкций и компонентов;
- WYSIWYG редактор, работающий на многих размерах экранов и разрешений, окно предварительного просмотра, показывающее запущенное приложение сразу на нескольких устройствах и в реальном времени;
- встроенная поддержка облачной платформы Google.

Загрузить последнюю версию *Android Studio*, а также получить рекомендации по установке, настройке и началу работы можно [здесь](#).

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 40 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 41 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

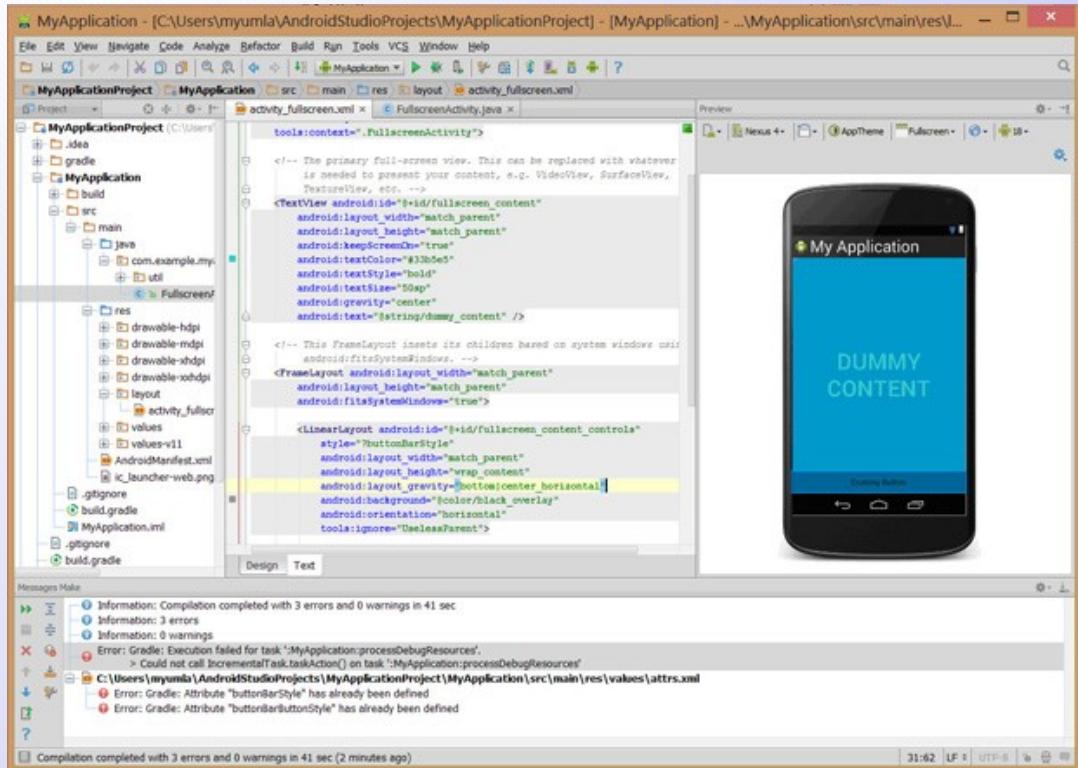


Рисунок 3. Среда разработки Android Studio

Перейдем к рассмотрению других инструментов, пригодных для разработки мобильных приложений под *Android*. Начнем с инструментов от Intel - Intel XDK и Intel Beacon Mountain.

Intel XDK позволяет легко разрабатывать кроссплатформенные мо-

бильные приложения; включает в себя инструменты для создания, отладки и сборки *ПО*, а также эмулятор устройств; поддерживает разработку для *Android*, Apple iOS, Microsoft Windows8, Tizen; поддерживает языки разработки: HTML5 и JavaScript.

Последняя тема данного курса полностью посвящена изучению нового поколения инструментальных средств разработки мобильных HTML5-приложений и Intel XDK, предполагается разработка мобильного приложения с использованием этих инструментов.

Intel Beacon Mountain - среда разработки, позволяющая создавать приложения для устройств, работающих под управлением ОС *Android*. Предоставляет инструменты необходимые для проектирования, разработки, отладки и оптимизации приложений под *Android*. Освобождает разработчика от необходимости поддерживать систему разработки в актуальном состоянии, следит за обновлениями и добавляет их в среду разработки по мере появления. Поддерживает разработку для целевых платформ на основе процессоров Intel Atom и ARM. Beacon Mountain построена на основе [Android IDE](#) (*Eclipse*, *Android ADT*, *Android SDK*), для более серьезной разработки и оптимизации добавлены следующие инструменты Intel:

- **Intel* Hardware Accelerated Execution Manager (Intel* HAXM)**
 - аппаратно поддерживаемый процессор виртуализации, использующий технологию виртуализации Intel* (Intel* VT) для ускорения



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 42 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 43 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

работы эмулятора в среде разработки.

- **Intel* Graphics Performance Analyzers (Intel* GPA) System Analyzer** поддерживает мобильные устройства с процессором Intel Atom под управлением ОС Android. Позволяет разработчикам оптимизировать загруженность системы при использовании процедур OpenGL, предоставляя возможность получать множество системных метрик в реальном времени, отображающих загруженность CPU, GPU и OpenGL ES API. Разработчик может запустить несколько графических экспериментов для выявления узких мест в обработке графики.
- **Intel* Integrated Performance Primitives (Intel* IPP) Preview** - библиотека оптимизированной обработки данных и изображений, поддерживающая мобильные устройства с платформой Intel под управлением ОС Android. Preview версия является частью полной версии Intel IPP, которая тоже поддерживает ОС Android.
- **Intel* Threading Building Blocks (Intel* TBB)** - широко используемая, признанная библиотека шаблонов C++ для создания масштабируемых приложений и увеличения производительности. Поддерживает мобильные устройства с платформой Intel под управлением Android. Проверенные алгоритмы позволяют разработчикам эффективно распараллелить C++ мобильные приложения, что повышает производительность при снижении энергетических за-

трат.

Загрузить *Intel Beacon Mountain* можно по ссылке [здесь](#).

The screenshot shows the Intel Developer Zone website. At the top, there's a navigation bar with links for 'Разработка' (Development), 'Вспомогательные программы' (Auxiliary programs), 'Ресурсы' (Resources), and a search bar. On the right side of the header, there are social media links for Facebook, LinkedIn, Twitter, and Google+. Below the header, the main content area has a title 'Beacon Mountain версии 0.6 для Android*' with a small image of a smartphone displaying the app. To the left of the title are icons for the Android logo and the Intel Inside Atom logo. Below the title is a list of features:

- Поддержка Jelly Bean или выше.
- Выполнение в средах систем Apple OS X* и 64-разрядных ОС Microsoft Windows® 7 и 8.
- Наличие плагинов Eclipse®, а также поддержка Android SDK, NDK, и т. п.

Below this list are tabs for 'OVERVIEW', 'DOCUMENTATION & SUPPORT', 'BEACON MOUNTAIN DETAILS', and 'UPGRADE NOTES'. A red banner below the tabs says 'Теперь поддерживаются системы под управлением Apple OS X* и Microsoft Windows® 7, 8'. Another section below discusses 'Быстрая разработка приложений Android для устройств на базе процессоров ARM* и Intel® Atom™'. It mentions that the software is designed for productivity and includes tools for programming and debugging original applications for devices running on the Intel Atom processor or the ARM architecture. It also supports smartphones and tablets running the Android operating system. The page ends with a section on 'Основные функциональные возможности:' (Main functional features) with a list of bullet points.

Рисунок 4. Страница поддержки Intel* Beacon Mountain

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 44 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 45 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Нельзя обойти вниманием *инструментарий Marmalade SDK*.

Marmalade SDK - кроссплатформенное *SDK* от Ideaworks3D Limited.

Представляет собой набор библиотек, образцов, инструментов и документации, необходимых для разработки, тестирования и развертывания приложений для мобильных устройств. Используется, в основном, для разработки игр. Многие получившие признание игры, такие как *Cut the Rope* и *Plants vs. Zombies*, были разработаны с использованием этого программного средства. К сожалению, *Marmalade SDK* представляет собой проприетарное *программное обеспечение* (самая дешевая лицензия \$15 в месяц) и не может быть рекомендована в данном учебном курсе, но читатель может самостоятельно попробовать бесплатную 30-дневную версию, доступную по ссылке [здесь](#).

Нельзя не сказать об отечественных разработках. Например, компания 1С идет в ногу со временем, версия платформы 1С 8.3 позволяет разрабатывать мобильные приложения. *Программный продукт "1С: Предприятие 8. Расширение для карманных компьютеров"* обеспечивает возможность работы с данными информационных баз 1С: Предприятия 8 на мобильных устройствах (карманных компьютерах, коммуникаторах, терминалах сбора данных), а также на персональных компьютерах (в том числе ноутбуках), не имеющих прямого доступа к информационным базам 1С:Предприятия 8.

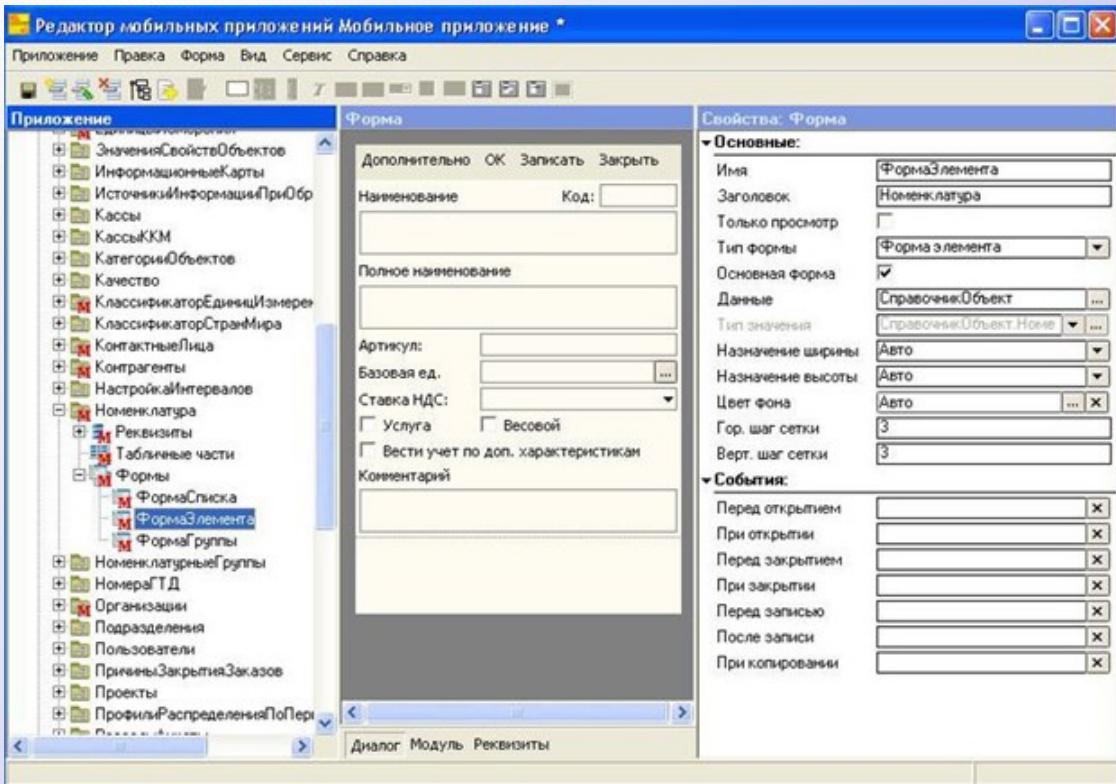


Рисунок 5. Редактор мобильных приложений 1С

Разумеется, данный *программный продукт* имеет очень узкую сферу применения, однако в некоторых случаях может являться наиболее удачным решением. Подробности по ссылке [здесь](#).



Кафедра
ПМИИ

Начало

Содержание



Страница 46 из 469

Назад

На весь экран

Закрыть



§4. Эмуляторы

Эмуляция. Стандартный эмулятор Android

Эмуляция (англ. *emulation*) в вычислительной технике - комплекс программных, аппаратных средств или их сочетание, предназначенное для копирования (или **эмулирования**) функций одной вычислительной системы (*гостя*) на другой, отличной от первой, вычислительной системе (*хосте*) таким образом, чтобы эмулированное поведение как можно ближе соответствовало поведению оригинальной системы (*гостя*). Целью является максимально точное воспроизведение поведения в отличие от разных форм компьютерного моделирования, в которых имитируется поведение некоторой абстрактной модели **Википедия**.

Эмулятор - виртуальное мобильное устройство, которое запускается на компьютере. При помощи эмулятора можно разрабатывать и тестировать приложения без использования реальных устройств. На рисунке 1 приведен пример запущенного стандартного эмулятора. Подробно работа с эмуляторами рассмотрена в лабораторной работе.

Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 47 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

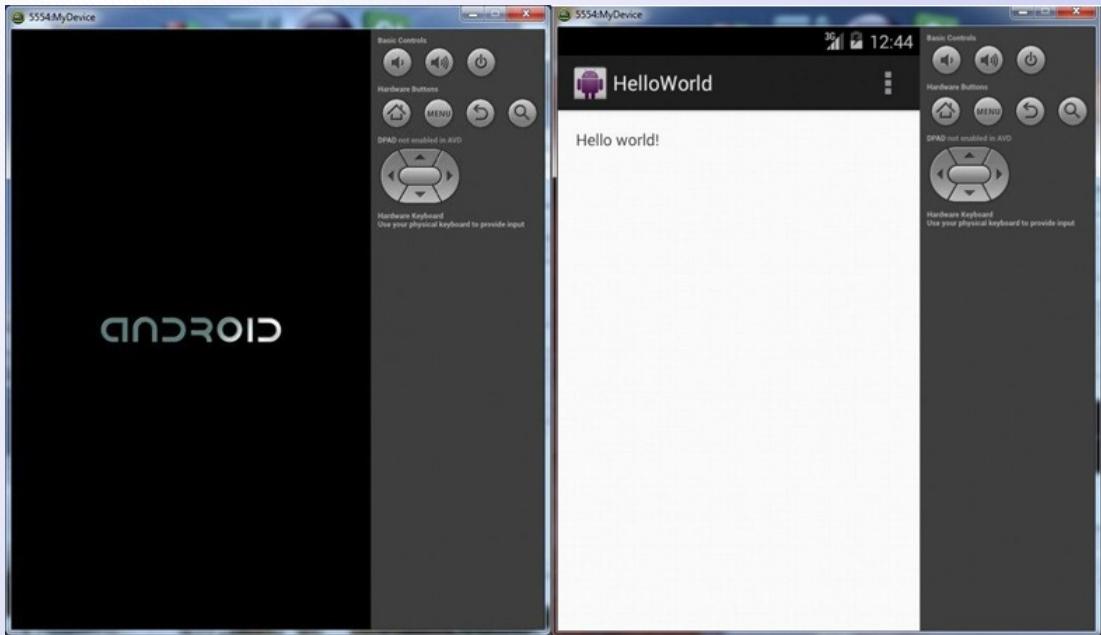


Рисунок 1. Эмулятор Android SDK в процессе запуска и приложение Hello, world!

К достоинствам использования эмуляторов можно отнести простоту их использования и нулевую стоимость. Разработчику не нужно покупать огромное количество устройств с различными характеристиками, чтобы проверить работоспособность приложения на различных смартфонах. Достаточно создать несколько эмуляторов с требуемыми характеристиками и запустить на них приложение. К сожалению, эмуляторы имеют и ряд недостатков:



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 48 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 49 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

- Требуют много системных ресурсов.
- Из-за различий в архитектуре процессоров компьютера и смартфона медленно запускаются. Современные персональные компьютеры построены на архитектурах x86 и x64, а большинство процессоров смартфонов на Android - ARM. Процесс эмуляции одной архитектуры на другой чрезвычайно сложен и происходит довольно медленно.
- В некоторых случаях стандартного эмулятора недостаточно. Речь идет о возможностях смартфонов, которыми обычные компьютеры не обладают (например, наличие датчика gps или акселерометра). В таких случаях полноценную отладку можно провести только с использованием реального устройства.

Альтернативные эмуляторы

Стандартный эмулятор, поставляемый вместе с Android SDK, не устраивает многих. Существуют проекты, поддерживающие разработку и развитие альтернативных эмуляторов. В качестве примера можно привести Genymotion (см. рис. 2) - быстрый эмулятор Android (по мнению его разработчиков). Он содержит предварительно настроенные образы Android (x86 с аппаратным ускорением OpenGL). Genymotion доступен для Linux, Windows и Mac OS X и требует для своей работы VirtualBox. Иными словами, Genymotion представляет собой виртуальную машину с установленной ОС Android, которую пользователь запускает так

же, как и другие виртуальные машины. Проблема высокого потребления системных ресурсов, конечно, не исчезает, однако скорость запуска существенно увеличивается.

В настоящее время активно развивается.

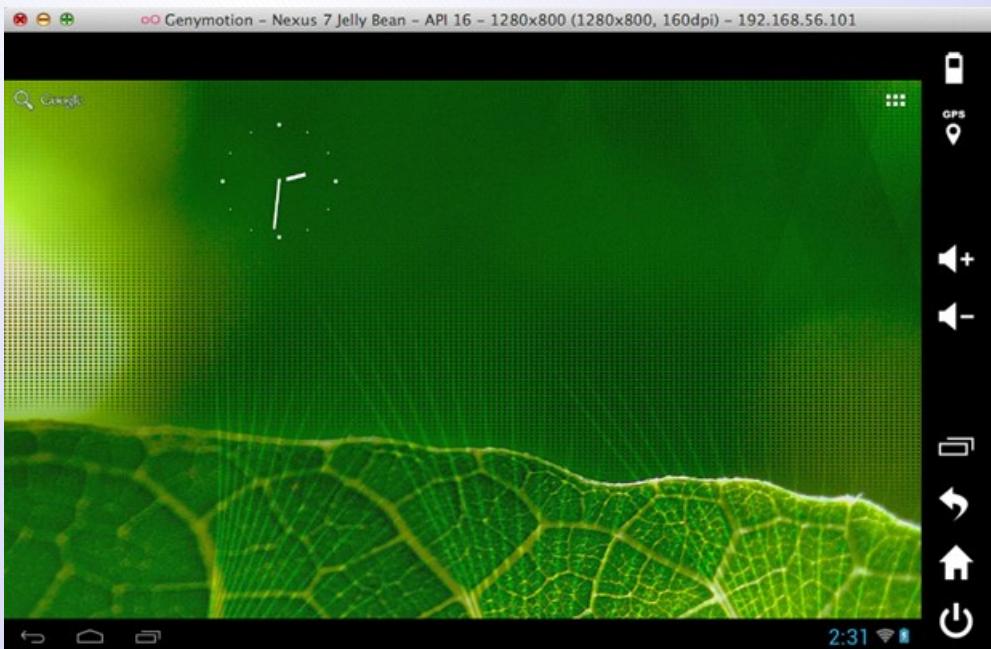


Рисунок 2. Альтернативный эмулятор Genymotion



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 50 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

ГЛАВА 2

Разработка приложений

§5. Возможности отладки на реальных устройствах

Разработанное *приложение* можно запустить на реальном устройстве, например, на смартфоне. Для этого необходимо проделать предварительную работу. Для запуска приложений, разработанных в *Android IDE*, необходимо:

- Настроить устройство (включить режим отладки по USB).
- Настроить компьютер (для Windows необходимо установить нужный драйвер вручную, нужны права администратора).
- Настроить среду и запустить проект на устройстве.

Подробности отладки на реальных устройствах описаны в лабораторной работе.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 51 из 469

Назад

На весь экран

Закрыть

§6. Примеры приложений

Google Play - это магазин приложений от Google, позволяющий владельцам устройств с операционной системой *Android* устанавливать и приобретать различные приложения. Учётная запись разработчика, которая даёт возможность публиковать приложения, стоит \$25. В настоящее время Google Play насчитывает более миллиона различных приложений, каждый месяц пользователями загружается несколько миллиардов. Разумеется, далеко не все из них высокого качества и поддерживаются разработчиками, встречается и вредоносное *программное обеспечение*.

В настоящий момент доступно более 30 различных категорий приложений. Внутри каждой категории приложения упорядочены на основании рейтинга, отзывов, количества скачиваний, страны распространения и других факторов.

Время от времени редакция Google Play собирает коллекции приложений или игр, основанных на теме или сезонном событии. Коллекции пользуются популярностью у клиентов за счет своевременности и актуальности.

Приведем примеры интересных и удобных приложений, заслуженно удерживающих высокие места в своих категориях уже долгое время.

Популярное игровое приложение *Cut the Rope* позволяет разобраться в правилах игры прямо в ее процессе и не требует чтения сложных



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 52 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

инструкций (см. рисунок 1). Идея игры предельно проста - в коробке сидит маленький зелёный монстр Ам Ням, которого надо кормить леденцами. Леденцы болтаются на веревках, и их надо правильно перерезать, чтобы леденец попал точно в рот Ам Няма. По ходу игры сложность уровней возрастает, появляются дополнительные препятствия. Попутно надо собирать звездочки, которые позволяют открывать новые локации.



Рисунок 1. Первый уровень игры Cut the Rope



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 53 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Если вам нужен интерактивный помощник, который способен понимать сделанные устно указания и напоминать о делах в нужное время, *приложение "Помнить все"*(см. рис. 2) - то, что вам нужно. Используя библиотеку для распознавания речи, оно анализирует полученную информацию, отображает ее в виде текста, который при необходимости можно исправить, и устанавливает время для напоминания.



Рисунок 2. Приложение "Помнить все"распознает русскую речь

Кафедра
ПМиИ

§7. Введение в разработку

Аннотация: В данной теме обсуждаются вопросы, связанные непосредственно с разработкой мобильных приложений для устройств, работающих под управлением Android. Рассматривается еще несколько общих вопросов: во-первых, какие виды мобильных приложений существуют и каковы особенности каждого вида; во-вторых, как организовано исполнение приложений в ОС Android и каким образом обеспечивается безопасная среда их функционирования. Понимание этих вопросов позволяет вести более осознанную разработку приложений. В лекции рассматривается архитектура Android приложений, основанная на идее многократного использования компонентов, которые являются основными строительными блоками. Подробно описываются основные компоненты, а также такие важные понятия для мобильных приложений, работающих под управлением Android, как манифест приложения и ресурсы.

В "Введение в разработку мобильных приложений" были рассмотрены общие вопросы, связанные с операционной системой *Android*, а также инструменты, используемые для разработки мобильных приложений. Изучены основы разработки интерфейсов мобильных приложений. В данной теме обсуждаются вопросы, связанные, непосредственно, с разработкой мобильных приложений для устройств, работающих под управлением *Android*.

Для начала предполагается рассмотреть еще несколько общих вопро-



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 55 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 56 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

сов: во-первых, какие виды мобильных приложений существуют и каковы особенности каждого вида; во-вторых, как организовано исполнение приложений в ОС *Android* и каким образом обеспечивается безопасная среда их функционирования. Понимание этих вопросов позволяет вести более осознанную разработку приложений.

Невозможно создать осмысленное *приложение*, не изучив внутреннюю организацию, присущую приложениям, работающим на определенной платформе. В данном курсе, очевидно, необходимо изучить структуру и основные компоненты приложений, разрабатываемых для работы на смартфонах под управлением ОС *Android*. От типа мобильного устройства внутренняя организация приложений не зависит, т. е. *Android*-приложения, разработанные для смартфонов вполне смогут выполняться и на планшетах. В данной лекции рассматривается *архитектура Android* приложений, основанная на идее многократного использования компонентов, которые являются основными строительными блоками. Подробно описываются основные компоненты, а также такие важные понятия для мобильных приложений, работающих под управлением *Android*, как *манифест* приложения и ресурсы.

§8. Основные виды Android-приложений

Приступая к разработке мобильных приложений хорошо бы иметь представление о том, какие виды приложений существуют. Дело в том, что если удастся определить к какому типу относится *приложение*, то становится понятнее на какие моменты в процессе его разработки необходимо обращать основное внимание. Можно выделить следующие виды приложений:

- **Приложения переднего плана** выполняют свои функции только, когда видимы на экране, в противном же случае их выполнение приостанавливается. Такими приложениями являются, например, игры, текстовые редакторы, видеопроигрыватели. При разработке таких приложений необходимо очень внимательно изучить жизненный цикл активности, чтобы переключения в фоновый режим и обратно проходили гладко (бесшовно), т. е. при возвращении приложения на передний план было незаметно, что оно вообще куда-то пропадало. Для достижения этой гладкости необходимо следить за тем, чтобы при входе в фоновый режим приложение сохраняло свое состояние, а при выходе на передний план восстанавливало его. Еще один важный момент, на который обязательно надо обратить внимание при разработке приложений переднего плана, удобный и интуитивно понятный интерфейс.
- **Фоновые приложения** после настройки не предполагают взаи-



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 57 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 58 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

модействия с пользователем, большую часть времени находятся и работают в скрытом состоянии. Примерами таких приложений могут служить, службы экранирования звонков, SMS-автоответчики. В большинстве своем фоновые приложения нацелены на отслеживание событий, порождаемых аппаратным обеспечением, системой или другими приложениями, работают незаметно. Можно создавать совершенно невидимые сервисы, но тогда они будут неуправляемыми. Минимум действий, которые необходимо позволить пользователю: санкционирование запуска сервиса, настройка, приостановка и прерывание его работы при необходимости.

- **Смешанные приложения** большую часть времени работают в фоновом режиме, однако допускают взаимодействие с пользователем и после настройки. Обычно взаимодействие с пользователем сводится к уведомлению о каких-либо событиях. Примерами таких приложений могут служить мультимедиа-проигрыватели, программы для обмена текстовыми сообщениями (чаты), почтовые клиенты. Возможность реагировать на пользовательский ввод и при этом не терять работоспособности в фоновом режиме является характерной особенностью смешанных приложений. Такие приложения обычно содержат как видимые активности, так и скрытые (фоновые) сервисы, и при взаимодействии с пользователем должны учитывать свое текущее состояние. Возможно потребуется обновлять



Кафедра ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 59 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

графический интерфейс, если приложение находится на переднем плане, или же посыпать пользователю уведомления из фонового режима, чтобы держать его в курсе происходящего. И эти особенности необходимо учитывать при разработке подобных приложений.

- **Виджеты** - небольшие приложения, отображаемые в виде графического объекта на рабочем столе. Примерами могут служить, приложения для отображения динамической информации, такой как заряд батареи, прогноз погоды, дата и время. Разумеется, сложные приложения могут содержать элементы каждого из рассмотренных видов. Планируя разработку приложения, необходимо определить способ его использования, только после этого приступать к проектированию и непосредственно разработке.

§9. Безопасность

Обратим внимание на организацию исполнения приложений в ОС *Android*. Как уже было отмечено приложения под *Android* разрабатываются на языке программирования *Java*, компилируется в *файл* с расширением .apk, после этого *файл* используется для установки приложения на устройства, работающие под управлением *Android*. После установки каждое *Android* приложение "живет" в своей собственной безопасной "песочнице рассмотрим, как это выглядит:

- операционная система *Android* является многопользовательской ОС, в которой каждое приложение рассматривается как отдельный пользователь;
- по умолчанию, система назначает каждому приложению уникальный пользовательский ID, который используется только системой и неизвестен приложению;
- система устанавливает права доступа ко всем файлам приложения следующим образом: доступ к элементам приложения имеет только пользователь с соответствующим ID;
- каждому приложению соответствует отдельный Linux процесс, который запускается, как только это необходимо хотя бы одному компоненту приложения, процесс прекращает работу, когда ни один компонент приложения не использует его или же системе требуется



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 60 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 61 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

освободить память для других (возможно, более важных) приложений;

- каждому процессу соответствует отдельный экземпляр виртуальной машины Dalvik, в связи с этим код приложения исполняется изолировано от других приложений.

Перечисленные идеи функционирования приложения в ОС *Android* реализуют принцип минимальных привилегий, т. е. каждому приложению, по умолчанию, разрешен *доступ* только к компонентам, необходимым для его работы и никаким больше. Таким образом обеспечивается очень безопасная среда функционирования приложений.

Однако, в случае необходимости приложения могут получить *доступ* к данным других приложений и системным сервисам (услугам). В случае, когда двум приложениям необходимо иметь *доступ* к файлам друг друга, им присваивается один и тот же пользовательский *ID*. Для экономии системных ресурсов такие приложения запускаются в одном Linux процессе и делят между собой один и тот же экземпляр виртуальной машины, в этом случае приложения также должны быть подписаны одним сертификатом. В случае же, когда приложению требуется *доступ* к системным данным, например, контактам, *SMS* сообщениям, картам памяти, камере, Bluetooth и т. д., пользователю необходимо дать приложению такие полномочия во время установки его на устройство.

§10. Архитектура приложения, основные компоненты

Вот и пришла пора поговорить непосредственно о внутренней организации приложений под *Android*: обсудить их архитектуру и основные компоненты.

Архитектура *Android* приложений основана на идее многократного использования компонентов, которые являются основными строительными блоками. Каждый компонент является отдельной сущностью и помогает определить общее *поведение приложения*.

Система *Android* выстроена таким образом, что любое *приложение* может запускать необходимый компонент другого приложения. Например, если *приложение* предполагает использование камеры для создания фотографий, совершенно необязательно создавать в этом приложении *активность* для работы с камерой. Наверняка на устройстве уже есть *приложение* для получения фотографий с камеры, достаточно запустить соответствующую *активность*, сделать фотографию и вернуть ее в *приложение*, так что *пользователь* будет считать, что камера часть приложения, с которым он работает.

Когда система запускает компонент, она запускает процесс приложения, которому принадлежит компонент, если он еще не запущен, и создает экземпляры классов, необходимых компоненту. Поэтому в отличие от большинства других систем, в системе *Android* приложения не имеют единой точки входа (нет метода `main()`, например). В силу запус-



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 62 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

ка каждого приложения в отдельном процессе и ограничений на *доступ* к файлам, *приложение* не может напрямую активировать *компонент* другого приложения. Таким образом для активации компонента другого приложения необходимо послать системе сообщение о намерении запустить определенный *компонент*, система активирует его.

Можно выделить четыре различных типа компонентов, каждый тип служит для достижения определенной цели и имеет свой особый *жизненный цикл*, который определяет способы создания и разрушения соответствующего компонента. Рассмотрим основные компоненты *Android*-приложений.

Активности (Activities). *Активность* - это видимая часть приложения (экран, окно, форма), отвечает за *отображение* графического интерфейса пользователя. При этом *приложение* может иметь несколько активностей, например, в приложении, предназначенном для работы с электронной почтой, одна *активность* может использоваться для отображения списка новых писем, другая *активность* - для написания, и еще одна - для чтения писем. Несмотря на то, что для пользователя *приложение* представляется единым целым, все активности приложения не зависят друг от друга. В связи с этим любая из этих активностей может быть запущена из другого приложения, имеющего *доступ* к активностям данного приложения. Например, *приложение* камеры может запустить *активность*, создающую новые письма, чтобы отправить только что сделанную фотографию адресату, указанному пользователем.



Кафедра ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 63 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Сервисы (Services). Сервис - компонент, который работает в фоновом режиме, выполняет длительные по времени *операции* или работу для удаленных процессов. Сервис не предоставляет пользовательского интерфейса. Например, сервис может проигрывать музыку в фоновом режиме, пока пользователь использует другое *приложение*, может загружать данные из сети, не блокируя взаимодействие пользователя с активностью. Сервис может быть запущен другим компонентом и после этого работать самостоятельно, а может остаться связанным с этим компонентом и взаимодействовать с ним.

Контент-провайдеры (Content providers). Контент-провайдер управляет распределенным множеством данных приложения. Данные могут храниться в файловой системе, в базе данных SQLite, в сети, в любом другом доступном для приложения месте. Контент-провайдер позволяет другим приложениям при наличии у них соответствующих прав делать запросы или даже менять данные. Например, в системе *Android* есть контент-провайдер, который управляет информацией о контактах пользователя. В связи с этим, любое *приложение* с соответствующими правами может сделать *запрос* на чтение и запись информации какого-либо контакта. Контент-провайдер может быть также полезен для чтения и записи приватных данных приложения, не предназначенных для доступа извне.

Приемники широковещательных сообщений (Broadcast Receivers). Приемник - компонент, который реагирует на широкове-

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 64 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 65 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

щательные извещения. Большинство таких извещений порождаются системой, например, извещение о том, что экран отключился или низкий заряд батареи. Приложения также могут инициировать *широковещание*, например, разослать другим приложениям сообщение о том, что некоторые данные загружены и доступны для использования. Хотя приемники не отображают пользовательского интерфейса, они могут создавать уведомление на панели состояний, чтобы предупредить пользователя о появлении сообщения. Такой приемник служит проводником к другим компонентам и предназначен для выполнения небольшого объема *работ*, например, он может запустить соответствующий событию сервис.

Все рассмотренные компоненты являются наследниками классов, определенных в *Android SDK*.



Кафедра ПМиИ

Начало

Содержание



Страница 66 из 469

Назад

На весь экран

Закрыть

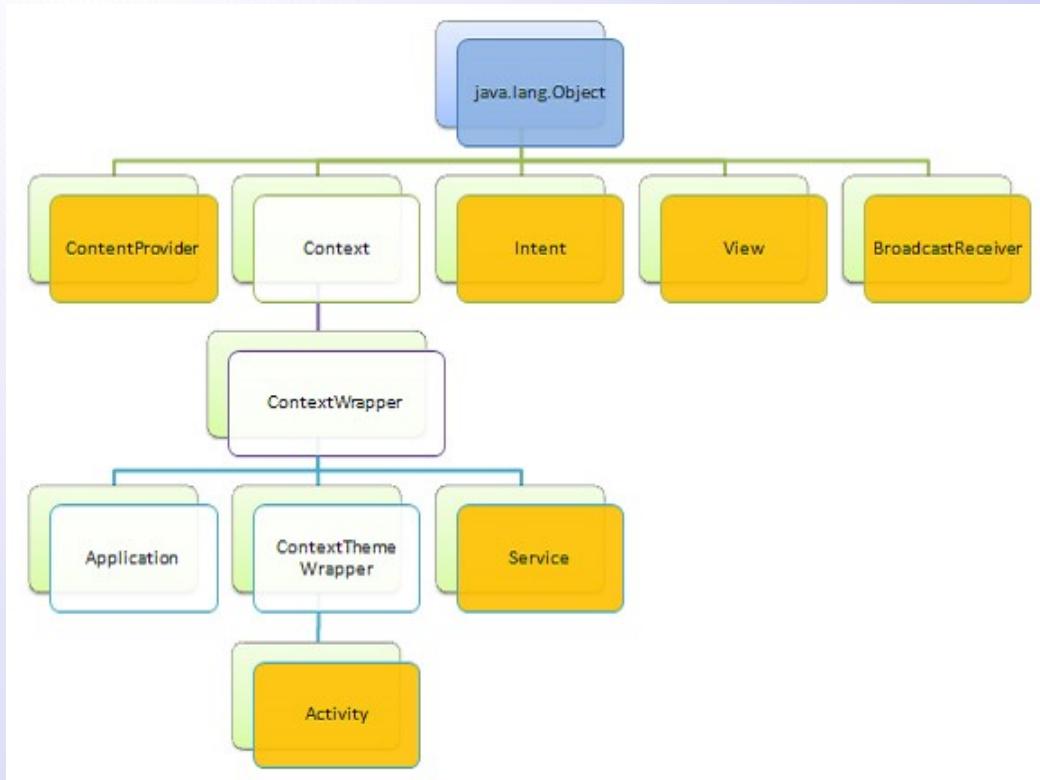


Рисунок 1. Иерархия классов Android SDK

(источник: [здесь](#))

На рис. 1 показана иерархия основных классов *Android SDK*, с которыми обычно имеет дело разработчик. На самом деле классов намного больше, желтым цветом выделены классы, с которыми разработчик ра-



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 67 из 469

Назад

На весь экран

Закрыть

ботает непосредственно, наследует от них свои классы. Остальные классы не менее важны, но они реже используются напрямую. Для начала рассмотрим классы Intent и View.

Класс View является основным строительным блоком для компонентов пользовательского интерфейса (UI), он определяет прямоугольную область экрана и отвечает за прорисовку и обработку событий. Является базовым классом для виджетов (GUI widgets), которые используются для создания интерактивных компонентов пользовательского интерфейса: кнопок, текстовых полей и т. д. А также является базовым классом для класса ViewGroup, который является невидимым контейнером для других контейнеров и виджетов, определяет свойства расположения компонентов пользовательского интерфейса. Интерфейс Android-приложения представляет собой иерархию UI компонентов (см. рис. 2), можно описать эту иерархию программно, но более простым и эффективным способом задать расположение элементов интерфейса является XML файл, который предоставляет удобную для восприятия структуру компоновки (*layout file*). Во время исполнения XML файл автоматически превращается в дерево соответствующих объектов. Подробнее о классе View, свойствах и методах: [здесь](#).



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 68 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

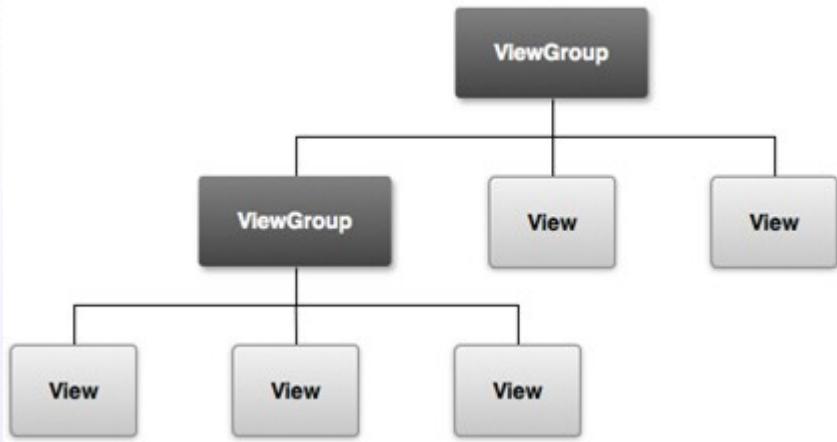


Рисунок 2. Иерархия компонентов, определяющая компоновку интерфейса пользователя

Объекты-экземпляры класса **Intent** используются для передачи сообщений между основными компонентами приложений. Известно, что три из четырех основных компонентов: активности, сервисы и приемники широковещательных сообщений, могут быть активированы с помощью сообщений, которые называются намерениями. Такие сообщения являются инструментом позднего связывания компонентов одного или нескольких приложений. Экземпляр класса **Intent** представляет собой структуру данных, содержащую описание *операции*, которая должна быть выполнена, и обычно используется для запуска активности или сервиса. В случае с приемниками широковещательных сообщений *объ-*



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 69 из 469

Назад

На весь экран

Закрыть

ект Intent содержит описание события, которое произошло или было объявлено.

Для каждого типа компонентов существуют свои механизмы передачи намерений.

- Чтобы запустить активность или вызвать у работающей активности новое действие, необходимо передать объект-намерение в метод `Context.startActivity()` или `Activity.startActivityForResult()`.
- Чтобы запустить сервис или доставить новые инструкции работающему сервису, необходимо передать объект-намерение в метод `Context.startService()`. Также объект-намерение может быть передан в метод `Context.bindService()`, чтобы связать между собой вызывающий компонент и сервис.
- Чтобы доставить объект-намерение всем заинтересованным приемникам широковещательных сообщений, необходимо передать его в любой из широковещательных методов:
`Context.sendOrderedBroadcast()`, `Context.sendStickyBroadcast()`,
`Context.sendBroadcast()`.

В каждом случае система *Android* в ответ на намерение находит соответствующий компонент: активность, сервис или множество широковещательных приемников и запускает его если необходимо. В этой системе сообщений не случается накладок: сообщение-намерение, отправ-



ленное определенному компоненту, будет получено именно этим компонентом и никем другим.

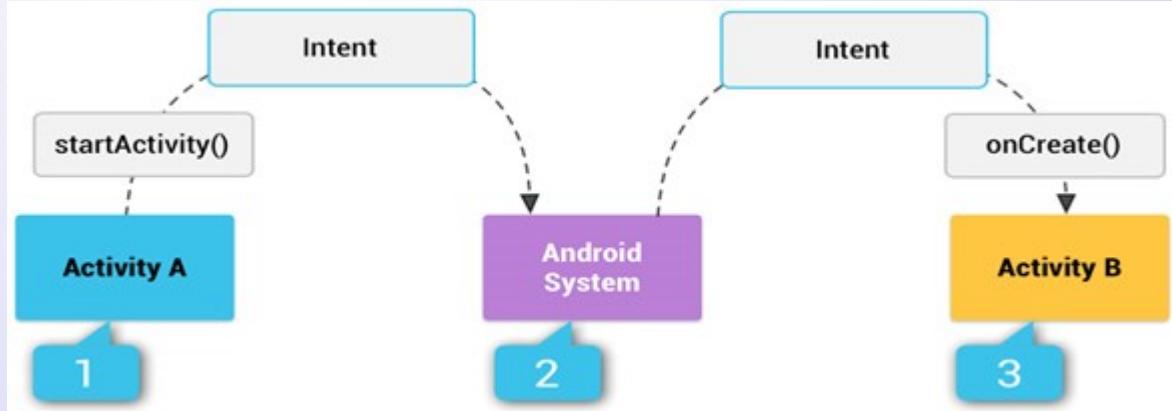


Рисунок 3. Передача намерений (Intent)

На рис. 3 можно увидеть как происходит передача намерений (Intent), в данном случае одна *активность* запускает другую. [6] Активность А создает намерение (Intent) с описанием действия и передает его в метод `startActivity()`. [7] Система *Android* проверяет все приложения на совпадение с намерением, когда совпадение найдено, [8] система запускает соответствующую *активность*, для чего вызывает метод `onCreate()` и передает в него *объект*-намерение Intent. Подробнее о классе Intent: [здесь](#) и [здесь](#).



Пришло время более серьезно рассмотреть основные компоненты: активности, сервисы, контент-провайдеры, приемники широковещательных сообщений. В первую очередь нас будет интересовать *жизненный цикл* этих компонентов. Что же такое этот *жизненный цикл*? *Жизненный цикл* можно рассматривать, как процесс функционирования компонента: начиная с момента создания и запуска, включая активный и неактивный периоды работы, и, заканчивая уничтожением и освобождением ресурсов.

10.1 Активности (Activities)

Активность - окно, несущее графический интерфейс пользователя. Окно активности обычно занимает весь экран устройства, однако вполне возможно создавать полупрозрачные или плавающие диалоговые окна. Мобильные приложения обычно являются многооконными, т. е. содержат несколько активностей, по одной на каждое окно. Одна из активностей определяется как "главная" и именно ее пользователь видит при первом запуске приложения.

Каждый экран приложения является наследником класса **Activity**. Для создания активности необходимо создать класс-наследник класса **Activity** напрямую или через любого его потомка. В этом классе необходимо реализовать все методы, вызываемые системой для управления жизненным циклом активности. Таких методов семь:

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 71 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

onCreate()

- метод, вызываемый системой при создании активности. В реализации метода необходимо инициализировать основные компоненты активности и в большинстве случаев вызвать метод `setContentView()` для подключения соответствующего XML-файла компоновки (layout file). После метода `onCreate()` всегда вызывается метод `onStart()`.

onRestart()

- метод, вызываемый системой при необходимости запустить приостановленную активность. После этого метода всегда вызывается метод `onStart()`.

onStart()

- метод, вызываемый системой непосредственно перед тем, как активность станет видимой для пользователя. После этого метода вызывается `onResume()`.

onResume()

- метод, вызываемый системой непосредственно перед тем, как активность начнет взаимодействовать с пользователем. После этого метода всегда вызывается `onPause()`.

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 72 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

onPause()

- метод, вызываемый системой при потере активностью фокуса. В этом методе необходимо фиксировать все изменения, которые должны быть сохранены за пределами текущей сессии. После этого метода вызывается `onResume()`, если активность вернется на передний план, или `onStop()`, если активность будет скрыта от пользователя.

onStop()

- метод, вызываемый системой, когда активность становится невидимой для пользователя. После этого метода вызывается либо `onRestart()`, если активность возвращается к взаимодействию с пользователем, либо `onDestroy()`, если активность уничтожается.

onDestroy()

- метод, вызываемый системой перед уничтожением активности. Этот метод вызывается либо когда активность завершается, либо когда система уничтожает активность, чтобы освободить ресурсы. Можно различать эти два сценария с помощью метода `isFinishing()`. Это последний вызов, который может принять активность.

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 73 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 74 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

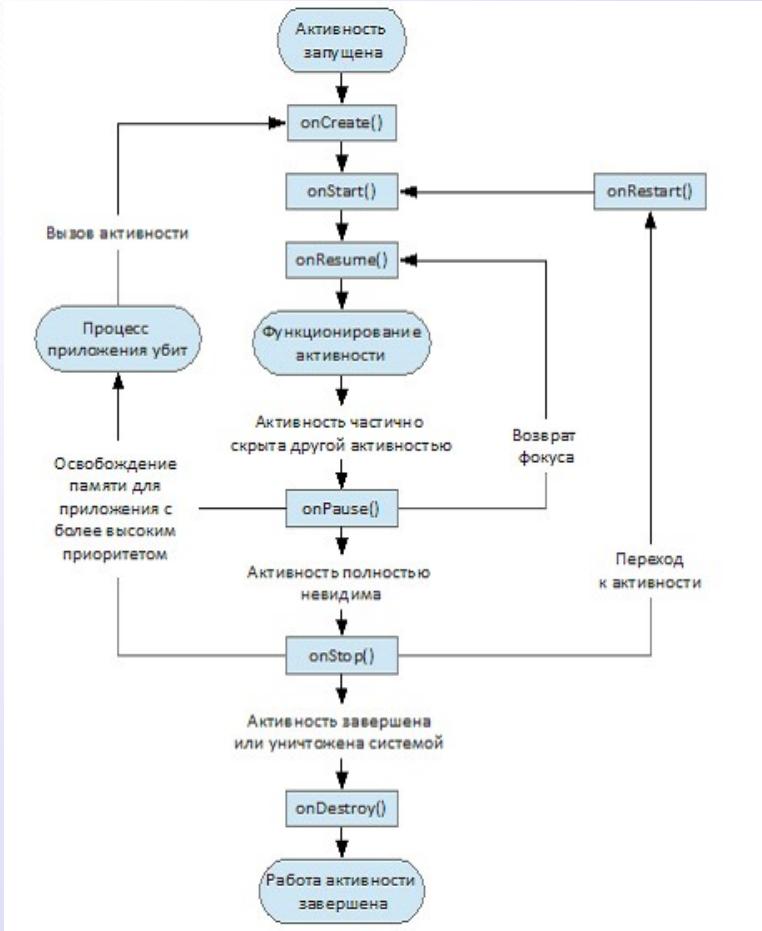


Рисунок 4. Жизненный цикл активности

(источник: [здесь](#)).

При реализации вышеперечисленных методов первым делом всегда необходимо вызывать соответствующий метод предка.

Рассмотренные методы определяют жизненный цикл активности. На рис. 4 можно увидеть пути, по которым активность может переходить из одного состояния в другое. В прямоугольниках указаны методы, которые вызываются при смене состояний активности.

Фактически активность может существовать в одном из трех состояний:

- **Выполняется (running).** Активность находится на переднем плане и удерживает фокус ввода. Если внимательно рассмотреть (рис. 4) можно заметить, что в это состояние активность попадает после вызова метода `onResume()`. Пока активность находится в этом состоянии ее процесс не может быть уничтожен системой.
- **Приостановлена.** Активность частично видима, однако фокус ввода потерян. В это состояние активность попадает после вызова метода `onPause()` (рис. 4). В этом состоянии активность поддерживается в "боевой готовности" т.е. в любой момент может получить фокус ввода и стать активной. Однако в этом состоянии процесс активности может быть уничтожен системой, в случае экстремальной нехватки памяти.
- **Остановлена.** Активность полностью невидима. В это состояние



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 75 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 76 из 469

Назад

На весь экран

Закрыть

активность попадает после вызова метода `onStop()` (рис. 4). В этом состоянии активность может быть "вызвана к жизни она сохраняет все состояния и необходимую для восстановления информацию, однако процесс активности может быть уничтожен, если память понадобится для других целей.

10.2 Сервисы (Services)

Сервис (Service) является компонентом приложения, предназначенным для выполнения длительных операций в фоновом режиме. Существует два способа существования сервисов:

- первый заключается в том, что сервис запущен (`started`) и работает самостоятельно в фоновом режиме, так он может работать неопределенно долго, пока не выполнит свою задачу;
- второй заключается в том, что сервис привязан (`bound`) к некоторому компоненту или нескольким компонентам, в этом случае сервис предлагает интерфейс для взаимодействия с компонентом и работает пока привязан хотя бы к одному компоненту, как только связь со всеми компонентами разрывается сервис завершает свою работу.

Для создания сервиса необходимо создать класс-наследник класса `Service` напрямую или через любого его потомка. При этом в реализации класса необходимо переопределить (т. е. написать свою реализа-

цию) некоторые методы, управляющие ключевыми аспектами жизненного цикла *сервиса* и обеспечивающие механизм связывания компонентов с сервисом, в соответствующем случае. Рассмотрим наиболее важные методы требующие реализации при создании сервиса.

onStartCommand() - метод, вызываемый системой, когда некоторый компонент, например *активность*, вызывает метод **startService()**. В этом случае сервис запускается и может работать в фоновом режиме неопределенно долго, поэтому необходимо позаботиться об остановке сервиса, когда он выполнит свою работу. Для остановки сервиса используется метод **stopSelf()** в случае, когда сервис сам прекращает свою работу, или **stopService()** в случае, когда работу сервиса прекращает некоторый компонент. Нет необходимости писать реализацию метода **onStartCommand()**, если не предполагается самостоятельной работы сервиса (т. е. он будет работать только в связке с некоторыми компонентами).



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 77 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 78 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

onBind()

- метод, вызываемый системой, когда некоторый компонент желает привязать к себе сервис и вызывает метод `bindService()`. Этот метод должен возвращать реализацию интерфейса `IBinder`, которая может быть использована компонентом-клиентом для взаимодействия с сервисом. Метод `onBind()` необходимо реализовать в любом случае, но, если не предполагается связывания сервиса с какими-либо компонентами, возвращаемое значение должно быть равным `null`.

Необходимо отметить, что сервис может быть запущен как самостоятельная единица, а в последствии может быть привязан к некоторым компонентам. В этом случае в сервисе должны быть обязательно реализованы оба метода `onStartCommand()` и `onBind()`.

onCreate()

- метод, вызываемый системой, при первом обращении к сервису для выполнения первоначальных настроек. Этот метод вызывается до вызова методов `onStartCommand()` и/или `onBind()`.

onDestroy()

- метод, вызываемый системой, когда сервис либо выполнил все действия, для которых создавался, либо больше не связан ни с одним компонентом, т. е. его услуги больше не требуются. В реализации этого метода необходимо предусмотреть освобождение всех ресурсов, таких как потоки, зарегистрированные слушатели, приемники и т. д. Вызов этого метода является последними вызовом, который может получить сервис.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)



Страница 79 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 80 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

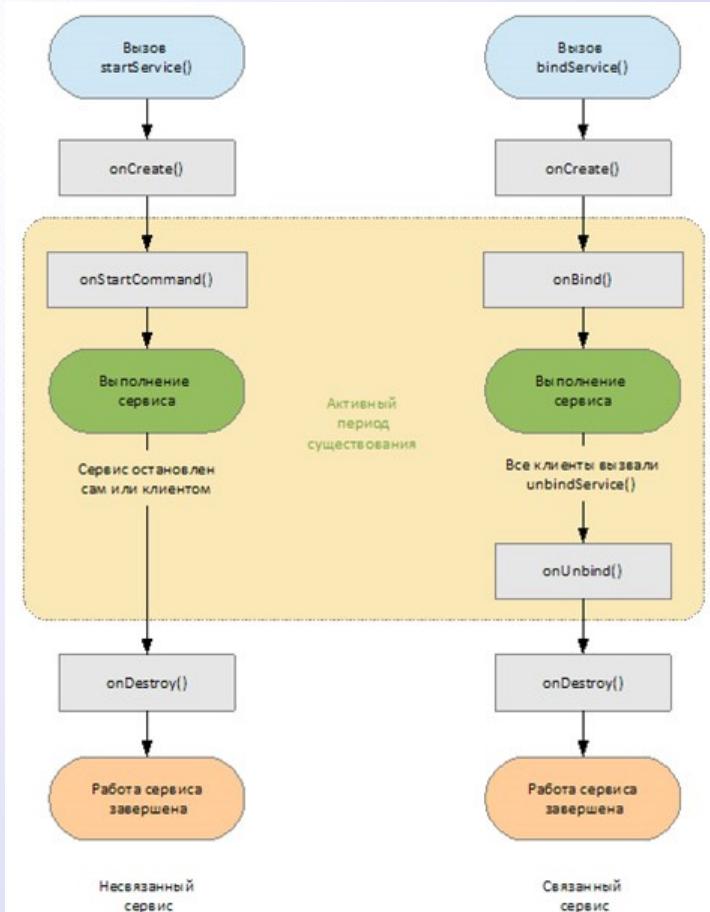


Рисунок 5. Жизненный цикл сервиса



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 81 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

(источник: [здесь](#)).

На рис. 5 показан жизненный цикл сервиса, левая диаграмма показывает жизненный цикл самостоятельного сервиса, правая - жизненный цикл сервиса, привязанного к некоторым компонентам. На рисунке хорошо видно, что жизненный цикл сервиса намного проще жизненного цикла активности. Однако для разработчика понимание того, как именно сервис создается, запускается и завершает свою работу, может оказаться даже более важным, т. к. сервис работает в фоновом режиме и пользователь может и не осознавать, что в некоторых случаях он имеет дело с работой сервисов.

Android принудительно останавливает работу сервисов только, когда ресурсов системы не хватает для активности, которая работает в данный момент на переднем плане. Приоритет работающих сервисов всегда выше, чем у приостановленных или полностью невидимых активностей, а если сервис привязан к выполняющейся активности, то его приоритет еще выше. С другой стороны, со временем приоритет самостоятельно работающего сервиса понижается и его шансы быть принудительно остановленным системой в случае нехватки ресурсов повышаются. В связи с этим имеет смысл проектировать сервис таким образом, чтобы через некоторое время он требовал у системы перезапуска. В случае если система все таки экстренно завершила работу сервиса, она перезапустит его как только освободятся ресурсы.

Подробнее о создании, использовании и удалении сервисов: [здесь](#) и

здесь.

10.3 Контент-провайдеры (Content Providers)

Контент-провайдер управляет доступом к хранилищу данных. Для реализации провайдера в Android приложении должен быть создан набор классов в соответствии с манифестом приложения. Один из этих классов должен быть наследником класса **ContentProvider**, который обеспечивает интерфейс между контент-провайдером и другими приложениями. Основное назначение этого компонента приложения заключается в предоставлении другим приложениям доступа к данным, однако ничто не мешает в приложении иметь активность, которая позволит пользователю запрашивать и изменять данные, находящиеся под управлением контент-провайдера.

В мобильных приложениях контент-провайдеры необходимы в следующих случаях:

- приложение предоставляет сложные данные или файлы другим приложениям;
- приложение позволяет пользователям копировать сложные данные в другие приложения;
- приложение предоставляет специальные варианты поиска, используя поисковую платформу (framework).



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 82 из 469

Назад

На весь экран

Закрыть

Если приложение требует использования контент-провайдера, необходимо выполнить несколько этапов для создания этого компонента:

1. Проектирование способа хранения данных. Данные, с которыми работают контент-провайдеры, могут быть организованы двумя способами:

- Данные представлены файлом, например, фотографии, аудио или видео. В этом случае необходимо хранить данные в собственной области памяти приложения. В ответ на запрос от другого приложения, провайдер может возвращать ссылку на файл.
- Данные представлены некоторой структурой, например, таблица, массив. В этом случае необходимо хранить данные в табличной форме. Стока таблицы представляет собой некоторую сущность, например, сотрудник или товар. А столбец - некоторое свойство этой сущности, например, имя сотрудника или цена товара. В системе Android общий способ хранения подобных данных - база данных SQLite, но можно использовать любой способ постоянного хранения.

Больше о хранении данных в Android можно узнать по ссылке: [здесь](#).

2. Создание класса-наследника от класса ContentProvider на-



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 83 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 84 из 469

Назад

На весь экран

Закрыть

прямую или через любого его потомка. При этом в реализации класса необходимо переопределить (т. е. написать свою реализацию) обязательные методы.

query()

- метод, извлекающий данные из провайдера, в качестве аргументов получает таблицу, строки и столбцы, а также порядок сортировки результата, возвращает объект типа [Cursor](#).

insert()

- метод, добавляющий новую строку, в качестве аргументов получает таблицу, и значения элементов строки, возвращает URI добавленной строки.

update()

- метод, обновляющий существующие строки, в качестве аргументов получает таблицу, строки для обновления и новые значения элементов строк, возвращает количество обновленных строк.

delete()

- метод, удаляющий строки, в качестве аргументов принимает таблицу и строки для удаления, возвращает количество удаленных строк.

getType()

- метод, возвращающий String в формате MIME, который описывает тип данных, соответствующий URI. Подробнее: [здесь](#).



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 85 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

onCreate()

- метод, вызываемый системой, сразу после создания провайдера, включает инициализацию провайдера. Стоит отметить, что провайдер не создается до тех пор, пока объект **ContentResolver** не попытается получить к нему доступ.

Созданный **контент-провайдер** управляет доступом к структурированным данным, выполняя обработку запросов от других приложений. Все запросы, в конечном итоге, вызывают объект **ContentResolver**, который в свою очередь вызывает подходящий метод объекта **ContentProvider** для получения доступа. Все вышеперечисленные методы, кроме **onCreate()**, вызываются приложением-клиентом. И все эти методы имеют такую же сигнатуру, как однотипные методы класса **ContentResolver**.

Подробнее о классе ContentProvider: [здесь](#).

3. **Определение строки авторизации провайдера, URI для его строк и имен столбцов.** Если от провайдера требуется управление намерениями, необходимо определить действия намерений, внешние данные и флаги. Также необходимо определить разрешения, которые необходимы приложениям для доступа к данным провайдера. Все эти значения необходимо определить как константы в отдельном классе, этот класс в последствии можно предоставить другим разработчикам.

Подробнее об URI: [здесь](#). Подробнее о намерениях: [здесь](#).

10.4 Приемники широковещательных сообщений (Broadcast Receivers)

Каждый широковещательный приемник является наследником класса `BroadcastReceiver`. Этот класс рассчитан на получение объектов-намерений отправленных методом `sendBroadcast()`.

Можно выделить две разновидности широковещательных сообщений:

- **Нормальные широковещательные сообщения** передаются с помощью `Context.sendBroadcast` в асинхронном режиме. Все приемники срабатывают в неопределенном порядке, часто в одно и то же время.
- **Направленные широковещательные сообщения** передаются с помощью `Context.sendOrderedBroadcast` только одному приемнику в один момент времени. Как только приемник сработает, он может передать сообщение следующему приемнику, а может прервать вещание так, что больше ни один приемник это сообщение не получит.

Даже в случае нормального широковещания могут сложиться ситуации, в которых система будет передавать сообщения только одному приемнику в один момент времени. Особенно это актуально для приемников, которые требуют создания процессов, чтобы не перегружать



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 86 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

систему новыми процессами. Однако в этом случае ни один приемник не может прервать широковещание.

Объект типа `BroadcastReceiver` действителен только во время вызова метода `onReceive()`, как только метод выполнен, система завершает работу объекта и больше не активирует его.

Подробнее о приемниках широковещательных сообщений: [здесь](#).



Кафедра ПМиИ

[Начало](#)

[Содержание](#)



Страница 87 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

§11. Манифест приложения

Корневой каталог каждого приложения под *Android* должен содержать файл *AndroidManifest.xml* (в точности с таким названием).

Манифест приложения содержит всю необходимую информацию, используемую системой для запуска и выполнения приложения. Основная информация, содержащаяся в манифесте:

- Имя Java пакета приложения, которое используется как уникальный идентификатор приложения.
- Описание компонентов приложения: активностей, сервисов, приемников широковещательных сообщений и контент-провайдеров, которые составляют приложение. Для каждого компонента приложения определено имя соответствующего класса и объявлены их основные свойства (например, с какими сообщениями-намерениями они могут работать). Эта информация позволяет системе *Android* узнать какие компоненты и при каких условиях могут быть запущены.
- Определение процессов, в которых будут выполняться компоненты приложения.
- Объявление полномочий, которыми должно обладать приложение для доступа к защищенным частям API и взаимодействия с другими приложениями.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 88 из 469

Назад

На весь экран

Закрыть

- Объявление полномочий, которыми должны обладать другие приложения для взаимодействия с компонентами данного.
- Список вспомогательных классов, которые предоставляют информацию о ходе выполнения приложения. Эти объявления содержатся в *манифесте* пока идет разработка и отладка приложения, перед публикацией приложения они удаляются.
- Определение минимального уровня Android API для приложения.
- Список библиотек связанных с приложением

В файле манифеста только два элемента: `<manifest>` и `<application>` являются обязательными и при этом встречаются ровно по одному разу. Остальные элементы могут встречаться несколько раз или не появляться совсем, в этом случае *манифест* определяет пустое *приложение*.

Следующий листинг демонстрирует общую структуру файла манифеста.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
<uses-permission />
<permission />
<permission-tree />
<permission-group />
<instrumentation />
<uses-sdk />
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 89 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 90 из 469

Назад

На весь экран

Закрыть

```
<uses-configuration />
<uses-feature />
<support-screens />
<compatible-screens />
<supports-gl-texture />
<application>
<activity>
<intent-filter>
<action />
<category />
<data />
</intent-filter>
<meta-data />
</activity>
<activity-alias>
<intent-filter> ... </intent-filter>
<meta-data />
</activity-alias>
<service>
<intent-filter> ... </intent-filter>
<meta-data />
</service>
<receiver>
```

```
<intent-filter> ... </intent-filter>
<meta-data />
</receiver>
<provider>
<grant-uri-permission />
<meta-data />
<path-permission />
</provider>
<uses-library />
</application>
</manifest>
```

Листинг 3.1. Структура файла AndroidManifest.xml

В манифесте элементы одного уровня, такие как `<activity>`, `<service>`, `<receiver>`, `<provider>`, могут следовать друг за другом в любой последовательности. Элемент `<activity-alias>` является исключением из этого правила, он должен следовать за соответствующей активностью.

Более предметно разговор о файле манифеста и его основных элементах пойдет в лабораторных работах.

Подробности: [здесь](#).



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 91 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

§12. Ресурсы

При разработке мобильных приложений необходимо выработать привычку отделять ресурсы приложения от кода. К ресурсам приложения могут относиться: изображения, строки, цвета, компоновки элементов пользовательского интерфейса (*layout*) и т. д. Отделение ресурсов от кода позволяет использовать *альтернативные* ресурсы для различных конфигураций устройств: язык, разрешение экрана и т. д. Для обеспечения совместимости с различными конфигурациями, ресурсы необходимо сгруппировать в директории по типу ресурсов и конфигурации устройства, полученные директории поместить в папку **res/**.

Для любого типа ресурсов можно определить две группы. Первая определяет ресурсы, которые будут использоваться независимо от конфигурации устройства или в том случае, когда под конфигурацию нет подходящих альтернативных ресурсов. Эта *группа* называется ресурсы по умолчанию (*default*). Вторая *группа* определяет ресурсы, подходящие для определенной конфигурации устройства, размещается в директории с названием, обозначающим данную конфигурацию. Такие ресурсы называются альтернативными.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 92 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 93 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

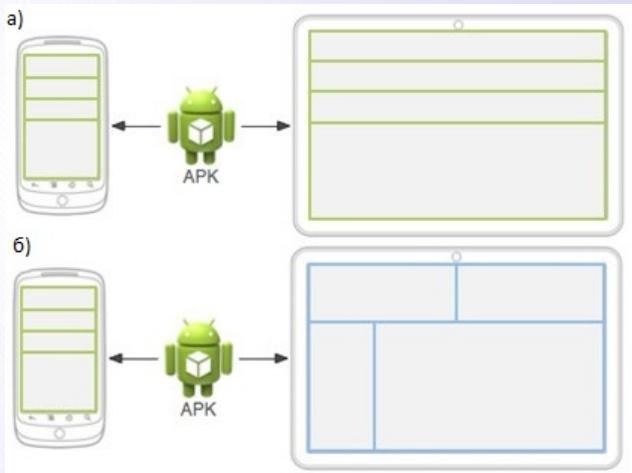


Рисунок 1. Использование ресурсов: а) используется компоновка по умолчанию (приложение не содержит альтернативы); б) каждое устройство использует соответствующую компоновку

Каждый тип ресурсов необходимо размещать в специальной поддиректории папки `res/`. Рассмотрим основные из этих поддиректорий:

animator/ - содержит XML файлы, которые определяют свойства анимации;

anim/ - содержит XML файлы, которые определяют анимацию преобразований;

color/ - содержит XML файлы, которые определяют списки цветов;



Кафедра ПМиИ

- drawable/** - содержит графические файлы или XML файлы, которые компилируются в графические ресурсы;
- layout/** - содержит XML файлы, которые определяют компоновку элементов пользовательского интерфейса;
- menu/** - содержит XML файлы, которые определяют все меню приложения;
- values/** - содержит XML файлы, которые определяют простые значения, таких ресурсов как, строки, числа, цвета.

Следует отметить, что *файлы ресурсов* нельзя размещать в папку **res/** напрямую, они обязательно должны размещаться в соответствующем каталоге, иначе будет выдана ошибка компиляции.

Все ресурсы, которые содержатся в рассмотренных поддиректориях являются ресурсами по умолчанию. Понятно, что различные типы устройств могут требовать различных типов ресурсов. Например, для устройств с разными размерами экрана компоновки элементов пользовательского интерфейса должны отличаться. Рис. 1 показывает варианты внешнего вида приложения с использованием только компоновки по умолчанию (а) и с использованием альтернативных компоновок (б). Даже на схеме понятно, что при правильном *подходе* приложение, изменяющее свой внешний вид в зависимости от размера экрана привлекательнее, чем остающееся неизменным.

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 94 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Чтобы определить зависимые от конфигурации альтернативы для множества ресурсов:

1. необходимо создать директорию в каталоге **res/**, присвоить этой директории имя в следующей форме:
имя_ресурса-спецификатор_конфигурации, где

- имя_ресурса - имя директории, соответствующего ресурса по умолчанию (см. выше);
- спецификатор_конфигурации - имя, определяющее конфигурацию, для которой используются данные ресурсы. Полный список доступных спецификаторов: [здесь](#);

2. необходимо сохранить ресурсы в новой директории, файл ресурсов должен называться в точности так же, как соответствующий файл ресурсов по умолчанию.

Например, если *компоновка* элементов пользовательского интерфейса сохранена, как ресурс по умолчанию, в папке **res/layout/**, можно (скорее даже нужно) определить альтернативную компоновку элементов пользовательского интерфейса, соответствующую горизонтальной (альбомной) ориентации экрана смартфона и сохранить ее в папке **res/layout-land/**. *Android* автоматически определит подходящую компоновку, сверяя текущее состояние устройства с именами папок в каталоге **/res**.

Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

◀ ▶

◀◀ ▶▶

Страница 95 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Все ресурсы после определения могут быть доступны по ссылке на их *ID*, которые определены в автоматически генерируемом классе `R`. Для каждого типа ресурсов в `R` классе существует *подкласс*, например, `R.drawable` для всех графических ресурсов. *ID* ресурса всегда имеет две составляющие:

- тип ресурса - все ресурсы группируются по типам, например, `string`, `drawable`, `layout`;
- имя ресурса - либо имя файла без расширения, либо значение атрибута `android:name` в XML файле для простого значения.

Получить *доступ* к ресурсу можно двумя способами:

- в коде: можно использовать выражения вида `R.тип_ресурса.имя_ресурса`, например, `R.string.hello`;
- в XML: используется специальный XML синтаксис, который соответствует ID определенному в `R` классе, например, `@string/hello`.

Более предметно разговор об использовании ресурсов в лабораторных работах.

Подробности: [здесь](#).

Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 96 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

§13. Основы разработки интерфейсов мобильных приложений



Аннотация: Большинство современных мобильных устройств имеют сенсорные дисплеи. Между традиционным оконным и тачевым интерфейсами существует огромная разница. Разработка удобного интерфейса для мобильных приложений является довольно сложной проблемой. Основной целью лекции является рассмотрение основ разработки интерфейсов мобильных приложений. В лекции рассказывается об особенностях визуального дизайна интерфейсов, строительных блоках и элементах управления. Приведены рекомендации по проектированию GUI под Android, а также имеется большое количество разнообразных примеров. В конце приведен список дополнительных источников. Описанные принципы помогут при разработке удобных пользовательских интерфейсов для мобильных приложений. Лекция может быть использована как часть курса или же отдельно от него для лучшего понимания особенностей интерфейса мобильных приложений.

Для подготовки данной лекции в качестве основного источника информации была использована книга "Алан Купер об интерфейсе" [11], однако задекларированные в ней принципы были переработаны в контексте программирования для мобильных устройств, а примеры заменены на более подходящие в рамках данного курса. Скриншоты приложений взяты из магазина приложений Google Play или сделаны са-

Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 97 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



мостоятельно с использованием смартфона Мегафон SP-A20i Mint на платформе Intel Medfield.

Презентацию к данной лекции можно скачать [здесь](#).

13.1 Визуальный дизайн интерфейсов

Силы, вложенные в разработку модели поведения программного продукта, будут потрачены впустую, если вы не сумеете должным образом донести до пользователей принципы этого поведения. В случае мобильных продуктов это делается визуальными средствами - путем отображения объектов на дисплее (в некоторых случаях целесообразно использовать тактильные ощущения от нажатия).

Визуальный дизайн интерфейсов - очень нужная и уникальная дисциплина, которую следует применять в сочетании с проектированием взаимодействия и промышленным дизайном. Она способна серьезно повлиять на эффективность и привлекательность продукта, но для полной реализации этого потенциала нужно не откладывать визуальный дизайн на потом, а сделать его одним из основных инструментов удовлетворения потребностей пользователей и бизнеса.

Изобразительное искусство, визуальный дизайн интерфейсов и прочие дисциплины дизайна

Художники и визуальные дизайнеры работают с одними и теми же

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 98 из 469

Назад

На весь экран

Закрыть



изобразительными средствами, однако их деятельность служит различным целям. Цель художника - создать объект, взгляд на который вызывает эстетический отклик. Изобразительное искусство - способ само выражения художника. Художник не связан почти никакими ограничениями. Чем необычнее и своеобразнее продукт его усилий, тем выше он ценится.

Дизайнеры создают объекты, которыми будут пользоваться другие люди. Если говорить о дизайнерах визуальных интерфейсов, то они ищут наилучшее представление, доносящее информацию о поведении программы, в проектировании которой они принимают участие. Придерживаясь целеориентированного подхода, они должны стремиться представлять поведение и информацию в понятном и полезном виде, который поддерживает маркетинговые цели организации и эмоциональные цели персонажей. Разумеется, визуальный дизайн пользовательских интерфейсов не исключает эстетических соображений, но такие соображения не должны выходить за рамки функционального каркаса.

Графический дизайн и пользовательские интерфейсы

Графические дизайнеры обычно очень хорошо разбираются в визуальных аспектах и хуже представляют себе понятия, лежащие в основе поведения программного продукта и взаимодействия с ним. Они способны создавать красивую и адекватную внешность интерфейсов, а кроме

Кафедра
ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 99 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 100 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

того - привносить фирменный стиль во внешний вид и поведение программного продукта. Для таких специалистов дизайн или проектирование интерфейса есть в первую очередь тон, стиль, композиция, которые являются атрибутами бренда, во вторую очередь - прозрачность и понятность информации и лишь затем - передача информации о поведении посредством ожидаемого назначения.

Дизайнерам визуальной части интерфейса необходимы некоторые навыки, которые присущи графическим дизайнерам, но они должны еще обладать глубоким пониманием и правильным восприятием роли поведения. Их усилия в значительной степени сосредоточены на организационных аспектах проектирования. В центре их внимания находится соответствие между визуальной структурой интерфейса с одной стороны и логической структурой пользовательской ментальной модели и поведения программы - с другой. Кроме того, их заботит вопрос о том, как сообщать пользователю о состояниях программы и что делать с когнитивными аспектами пользовательского восприятия функций.

Визуальный информационный дизайн

Информационные дизайнеры работают над визуализацией данных, содержимого и средств навигации. Усилия информационного дизайнера направлены на то, чтобы представить данные в форме, способствующей их верному истолкованию. Результат достигается через управление визуальной иерархией при помощи таких средств, как цвет, форма, распо-



ложение и масштаб. Распространенными объектами информационного дизайна являются всевозможные графики, диаграммы и прочие способы отображения количественной информации.

Чтобы создавать привлекательные и удобные пользовательские интерфейсы, дизайнер интерфейсов должен владеть базовыми визуальными навыками - пониманием цвета, типографики, формы и композиции - и знать, как их можно эффективно применять для передачи поведения и представления информации, для создания настроения и стимулирования физиологических реакций. Дизайнеру интерфейса также требуется глубокое понимание принципов взаимодействия и идиом интерфейса, определяющих поведение продукта.

13.2 Строительные блоки визуального дизайна интерфейсов

Дизайн интерфейсов сводится к вопросу о том, как оформить и расположить визуальные элементы таким образом, чтобы внятно отразить поведение и представить информацию. Каждый элемент визуальной композиции имеет ряд свойств, и сочетание этих свойств придает элементу смысла. Пользователь получает возможность разобраться в интерфейсе благодаря различным способам приложения этих свойств к каждому из элементов интерфейса. В тех случаях, когда два объекта обладают общими свойствами, пользователь предположит, что эти объекты связаны

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 101 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

или похожи. Когда пользователи видят, что свойства отличаются, они предполагают, что объекты не связаны.

Создавая пользовательский *интерфейс*, проанализируйте перечисленные ниже визуальные свойства каждого элемента или группы элементов. Чтобы создать полезный и привлекательный пользовательский *интерфейс*, следует тщательно поработать с каждым из этих свойств.

Форма

Форма - главный признак сущности объекта для человека. Мы узнаем объекты по контурам. Если мы увидим на картинке синий ананас, мы его сразу опознаем, потому что мы помним его форму. И лишь потом мы удивимся странному цвету (см. рис. 1). При этом различение форм требует большей концентрация внимания, чем анализ цвета или размера. Поэтому форма - не лучшее свойство для создания контраста, если требуется привлечь внимание пользователя.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 102 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Рисунок 1. В первую очередь мы видим ананас, а уже потом начинаем задумываться, почему он синий

Размер

Более крупные элементы привлекают больше внимания, особенно если они значительно превосходят размерами окружающие элементы. Люди автоматически упорядочивают объекты по размеру и склонны оценивать их по размеру; если у нас есть текст в четырех размерах, предполагается, что относительная важность текста растет вместе с размером и что полужирный текст более важен, чем текст с нормальным начертанием. Таким образом, размер - полезное свойство для обозначения



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 104 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

информационных иерархий.

Цвет

Цветовые различия быстро привлекают внимание. В некоторых профессиональных областях цвета имеют конкретные значения, и этим можно пользоваться. Так, для бухгалтера красный цвет - отрицательные результаты, а черный - положительные.

Цвета приобретают смыслы и благодаря социальным контекстам, в которых проходит наше взросление. Например, белый цвет на Западе ассоциируется с чистотой и миром, а в Азии и арабских странах - с похоронами и смертью. При этом цвет изначально не обладает свойством упорядоченности и не выражается количественно, поэтому далеко не идеален для передачи информации такого рода. Кроме того, не следует делать цвет единственным способом передачи информации, поскольку цветовая слепота встречается довольно часто.

Применяйте цвет с умом. Чтобы создать эффективную визуальную систему, позволяющую пользователю выявлять сходства и различия объектов, используйте ограниченный набор цветов - эффект радуги перегружает восприятие пользователя и ограничивает возможности по передаче ему информации.

Выбор цветовой палитры для программы необходимо проводить очень осторожно. По разным данным, той или иной формой цветовой слепоты



[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 105 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

страдают до 10% мужчин, и использование, например, красного и зеленого цветов для указания контраста затрудняет работу с приложением для этих людей.

Яркость

Понятия темного и светлого обретают смысл преимущественно в контексте яркости фона. На темном фоне темный текст почти не виден, тогда как на светлом он будет резко выделяться. Контрастность люди воспринимают легко и быстро, так что значение яркости может стать хорошим инструментом привлечения внимания к тем элементам, которые требуется подчеркнуть. Значение яркости - также упорядоченная переменная, например, более темные (с более низкой яркостью) цвета на карте легко интерпретируются: они обозначают большие глубины или большие значения других параметров.

Направление

Направление полезно, когда требуется передавать информацию об ориентации (вверх или вниз, вперед или назад). Помните, что восприятие направления может быть затруднено в случае некоторых форм и при малых размерах объектов, поэтому ее лучше использовать в качестве вторичного признака. Так, если требуется показать, что рынок акций пошел вниз, можно использовать направленную вниз стрелку красного цвета.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 106 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Текстура

Разумеется, изображенные на экране элементы не обладают настоящей текстурой, но способны создавать ее видимость. Текстура редко бывает полезна для передачи различий или привлечения внимания, поскольку требует значительной концентрации на деталях. И тем не менее текстура может быть важной подсказкой. Засечки и выпуклости на элементах пользовательского интерфейса обычно указывают, что элемент можно перетаскивать, а фаски или тени у кнопки усиливают ощущение, что ее можно нажать.

Расположение

Расположение - это переменная, упорядоченная и выражаемая количественно, а значит, полезная для передачи иерархии. Расположение также может служить средством создания пространственных отношений между объектами на экране и объектами реального мира (например, небо в верхней половине, земля в нижней).

Расположение элементов мобильного приложения очень сильно влияет на удобство использования и зависит от того, как пользователь будет держать устройство (см. рис. 2). Подробнее об этом будет рассказано в продолжении данного курса.

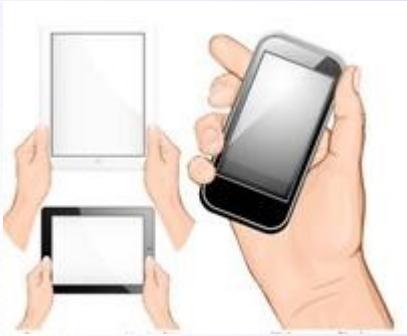


Рисунок 2. Различные варианты удержания смартфона

13.3 Элементы управления и дизайн навигации

Элементы управления - это доступные для манипулирования само-достаточные экранные объекты, посредством которых люди взаимодействуют с цифровыми продуктами. *Элементы управления* (controls, другое название - widgets - сокращение от windows gadgets - оконные при-способления) - это базовые строительные блоки графического пользо-вательского интерфейса. Рассматривая *элементы управления* с учетом целей пользователя, их можно разбить на четыре основные категории:

- **командные элементы управления**, применяемые для выполне-ния функций;
- **элементы выбора**, позволяющие выбирать данные или настрой-ки;

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 107 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



- **элементы ввода**, применяемые для ввода данных;
- **элементы отображения**, используемые для наглядного непосредственного манипулирования.

Некоторые элементы управления сочетают в себе свойства более чем одной категории.

Командные элементы управления

Командные элементы управления выполняют действия, причем делают это немедленно. Главным и по сути единственным командным элементом является кнопка, которая обладает множеством вариантов отображения. Элементы меню также являются командными идиомами.

Кнопки

Кнопки обычно легко опознаются благодаря их псевдотрехмерности (рис. 3). Действие выполняется сразу после нажатия на кнопку. Часто особым образом выделяется кнопка по умолчанию, соответствующая наиболее часто используемому действию.

Кнопка - удобный и привлекательный с визуальной точки зрения элемент управления. Весь ее облик подсказывает, что на нее можно нажать, и это характеризует ее ожидаемое назначение. Рекомендуется изменять внешний вид нажатой кнопки, так как это облегчает понимание работы программы пользователем.

Кафедра
ПМИ

Начало

Содержание

◀ ▶

◀▶

Страница 108 из 469

Назад

На весь экран

Закрыть



Рисунок 3. Скриншоты популярной игры "Cut the Rope". Кнопки кажутся выпуклыми, а нажатая кнопка меняет цвет

Кнопки-значки

Кнопки, помещенные на панель инструментов, обычно становятся квадратными, теряют текстовую надпись и обзаводятся пиктограммой - пояснением в виде графического значка (рис. 4).

Считается, что кнопки-значки очень удобны: они постоянно на виду и взаимодействовать с ними проще, чем с элементами раскрывающегося меню. Поскольку они постоянно видны, то легко запоминаются. У большинства пользователей не возникает вопросов относительно ожидаемого назначения кнопки. Проблема в том, что изображение на кнопке иногда



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 109 из 469

Назад

На весь экран

Закрыть

бывает непонятным. Например, пиктограмма с изображением дискеты, традиционно обозначающая сохранение, часто непонятна молодым пользователям, которые никогда не работали с реальными дискетами.

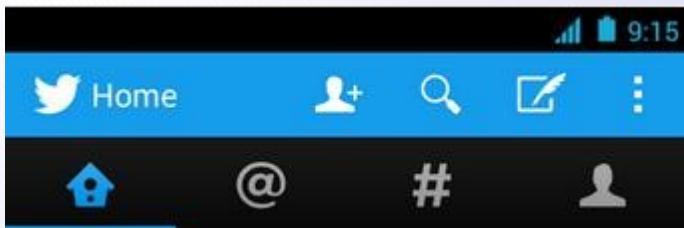


Рисунок 4. Кнопки-значки в Android-приложении Twitter

Гиперссылки

Текстовые гиперссылки - распространенный способ навигации в сети и веб-приложениях, однако при программировании для мобильных устройств их следует избегать. Дело в том, что попасть по ссылке пальцем с первого раза часто затруднительно и пользователей раздражает необходимость повторения этих действий.

Элементы управления выбором

Элементы выбора позволяют пользователю выбрать из группы допустимых объектов тот, с которым будет совершено действие. Элементы



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 110 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 111 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

выбора применяются также для действий по настройке. Распространенными элементами выбора являются флажки и списки.

Раньше использование элементов управления выбором не приводило к немедленному выполнению действий - требовалась еще и активация командного элемента. Сейчас возможны оба варианта. Если желательно дать пользователю возможность несколько раз осуществить выбор перед выполнением действия, следует создать явный командный элемент управления (кнопку). Если же пользователю полезно сразу видеть результат своих действий, и эти действия легко отменить, разумно сделать так, чтобы элемент выбора играл также и роль командного элемента.

Флажки

Назначение флажка очевидно. Щелкнув по флажку, пользователь немедленно увидит появившуюся галочку. Флажок прост, нагляден и изящен, однако основан на тексте. Качественный текст может исключить возможность неоднозначного толкования флажка. Однако этот же поясняющий текст вынуждает пользователя замедляться для прочтения, а также занимает значительное экранное пространство.

Традиционно флажки имеют квадратную форму. Не забывайте, что пользователи распознают визуальные объекты по форме, и квадратная форма флажков - важный стандарт.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 112 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Рисунок 5. Один из самых распространенных выключателей - кнопка "Pause" в играх

Состояние выключателя остается неизменным до следующего щелчка. Выключатели экономно расходуют экранные пространство: они занимают меньше места, потому что их назначение описывается не с помощью текста, а посредством визуальных средств. Разумеется, это озна-



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 113 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

чает, что им присущ тот же недостаток обычных кнопок-значков - неоднозначность пиктограмм.

Триггеры

Кнопки-триггеры - это разновидность элементов управления. Они призваны экономить экранное пространство, к сожалению, ценой значительной дезориентации пользователя. Классический пример - размещение на одной кнопке функций воспроизведения и паузы для музыкального проигрывателя. Подводным камнем такого подхода является то, что элемент управления можно ошибочно посчитать индикатором состояния проигрывателя ("на паузе" или "идет воспроизведение"). Элемент управления может служить либо индикатором состояния, либо кнопкой переключения состояний, но не тем и другим одновременно (рис. 6).



Рисунок 6. Если кнопка говорит "ВКЛЮЧЕНО" когда находится в состоянии "выключено то непонятно, в каком же состоянии находится кнопка. Если кнопка говорит "ВЫКЛЮЧЕНО" когда находится в состоянии "выключено тогда возникает вопрос, где искать кнопку "ВКЛЮЧЕНО"?



Кафедра ПМиИ

Радиокнопки

Радиокнопки внешне похожи на флагшки (рис. 7), но являются взаимоисключающими, то есть выбор одного из вариантов автоматически аннулирует предыдущий выбор. В каждый момент времени может быть выбрана только одна кнопка. Радиокнопки всегда объединяются в группы из двух или более радиокнопок, причем в каждой группе одна радиокнопка всегда выбрана. Радиокнопки всегда круглые по той же причине, по которой флагшки всегда имеют квадратную форму: именно такими они были изначально.

Радиокнопки занимают даже больше места, чем флагшки, однако в некоторых случаях такой расход экранного пространства оправдан.

Кнопка-значок преобразовала радиокнопки так же, как флагшки, заменив их в основном интерфейсе приложения. Если два или более выключателя объединены схемой взаимного исключения - так, чтобы в каждый момент мог быть включен лишь один из них, - они ведут себя точно так же, как радиокнопки. Так образуются радиокнопки со значками. Элементы управления цветом в Adobe Photoshop - хороший пример **радиокнопок со значками** (рис. 8).

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 114 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

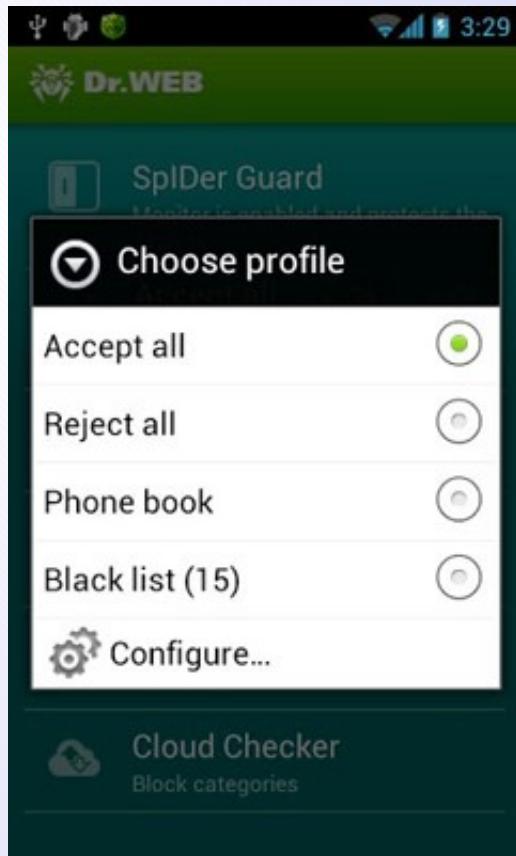


Рисунок 7. Радиокнопки Android-приложения Dr.Web



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 115 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 116 из 469

Назад

На весь экран

Закрыть

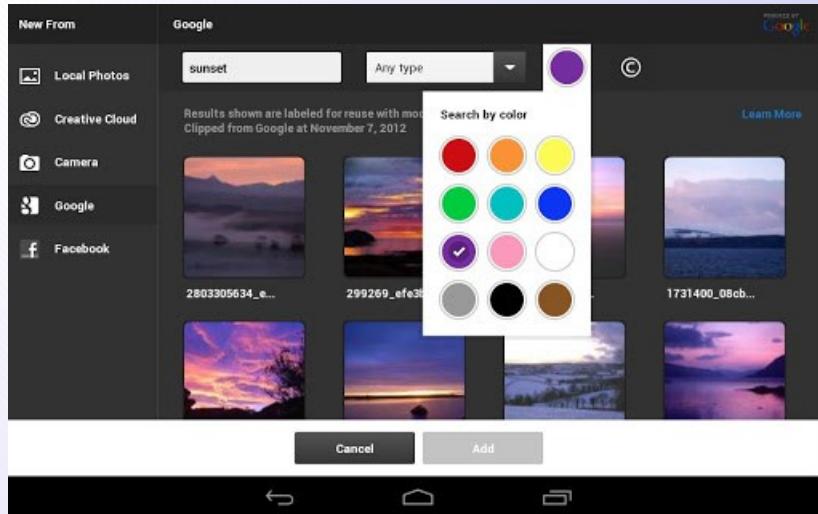


Рисунок 8. Элементы управления цветом в Android-приложении Adobe Photoshop представляют собой группу радиокнопок со значками

Списки

Элементы управления типа "список" позволяют осуществлять выбор из конечного множества текстовых строк, каждая из которых представляет команду, объект или признак. Подобно радиокнопкам, списки - мощный инструмент, упрощающий взаимодействие за счет устранения возможности неправильного выбора. Списки - это небольшие текстовые области с полосой прокрутки, автоматически подключаемой при необходимости (рис. 9). Пользователь может выбрать единственную строку текста, нажав на нее.



Кафедра ПМиИ

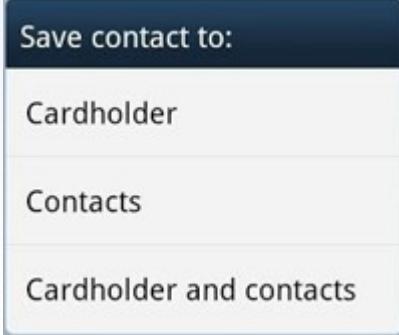


Рисунок 9. Стандартный список в Android

Раскрывающийся список - повсеместно встречающийся вариант обычного списка. Он показывает лишь выбранный элемент в одну строку, но если нажать на стрелку, открываются другие варианты выбора.

Элемент управления **представление в виде списка** предоставляет возможность сопровождать каждую строку текста пиктограммой. Такая возможность весьма полезна - существует множество ситуаций, когда можно упростить работу пользователя, располагая графические идентификаторы рядом со строками важных вариантов выбора (рис. 10).





Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 118 из 469

Назад

На весь экран

Закрыть



Рисунок 10. Элемент управления типа "список" с пиктограммами в Android-приложении, позволяющий визуально оценивать погоду в различных городах

Комбо-списки и комбо-кнопки

Комбо-элементы представляют собой сочетание элементов. Комбо-кнопка - разновидность радиокнопки со значком. Обычно она выглядит как кнопка-значок с небольшой стрелкой, но если нажать на стрелку и удерживать ее в нажатом состоянии, разворачивается меню.

Комбо-список представляет собой сочетание списка и поля редактирования (рис. 11).



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 119 из 469

Назад

На весь экран

Закрыть



Рисунок 11. Элемент управления типа "список" с пиктограммами в Android-приложении, позволяющий визуально оценивать погоду в различных городах

Вариант с раскрывающимся списком значительно экономит экранное пространство. Комбо-список хорошо подходит для тех случаев, когда необходимо организовать выбор единственного объекта.

Элементы ввода

Элементы ввода дают пользователю возможность не только выбирать существующие сведения, но и вводить новую информацию. Самый



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 120 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

простой элемент - поле редактирования текста (поле ввода). В эту категорию попадают также такие элементы управления, как счетчики и ползунки.

Ограничивающие элементы ввода

Любой элемент управления, ограничивающий набор значений, доступных для ввода пользователем, является ограничивающим элементом ввода. Так, например, ползунок со шкалой значений от 0 до 100 является ограничивающим элементом ввода. Независимо от действий пользователя не может быть введено число, выходящее за диапазон определенных программой значений. Проще говоря, ограничивающие элементы ввода должны использоваться везде, где необходимо ограничить множество допустимых значений.

Ограничивающий элемент ввода должен четко информировать пользователя о допустимых границах. Текстовое поле, которое отвергает ввод пользователя после того, как он выполнил ввод, не может считаться ограничивающим элементом управления. Если пользователь должен выразить выбор числовым значением в определенных границах, предоставьте ему элемент управления, сообщающий об этих границах и предотвращающий ввод недопустимых значений. Такую возможность дает ползунок. Ползунок позволяет пользователю определять числовые значения в относительных терминах, а не в результате непосредственно го ввода с клавиатуры. Но для ввода точных значений лучше подходят



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 121 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

счетчики.

Счетчики

Счетчик состоит из небольшого поля ввода и двух прикрепленных к нему кнопок (рис. 12). Благодаря счетчикам грань между ограничивающими и неограничивающими элементами ввода данных становится размытой. Маленькие кнопки со стрелками позволяют пользователю изменять значение в поле редактирования небольшими шагами. Эти шаги могут выполняться до определенного предела: значение не может превысить максимум, установленный программой, или стать меньше установленного минимума. Если пользователь пожелает ввести определенное число, он может сделать это за счет прямого ввода числа в поле редактирования.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 122 из 469

Назад

На весь экран

Закрыть

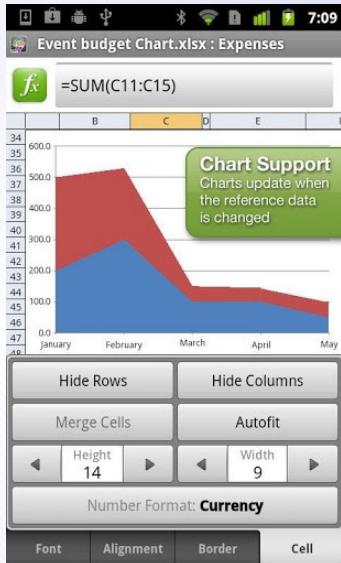


Рисунок 12. Реализация счетчиков в Android-приложении Quickoffice

Рукоятки и ползунки

Рукоятки и ползунки очень эффективно расходуют экранное пространство, и оба этих элемента управления замечательно справляются с задачей обеспечения визуальной обратной связи по настройкам (рис. 13). Ползунки и рукоятки применяются в основном в качестве ограничивающих элементов управления ввода. Например, ползунки - превосходное средство для действий, связанных с масштабированием.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 123 из 469

Назад

На весь экран

Закрыть



Рисунок 13. Ползунок в Android-приложении Winamp осуществляет перемотку воспроизведимой композиции

Неограничивающие элементы ввода

Пожалуй, главный неограничивающий элемент ввода - поле ввода текста. Этот простейший элемент управления позволяет пользователям набирать любые алфавитно-цифровые строки. Как правило, поля ввода - это небольшие области, внутри которых можно набрать одно-два слова, но они могут быть реализованы и в виде довольно сложных текстовых редакторов.

Когда пользователю предложено неограничивающее текстовое поле



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 124 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

ввода, которое при этом принимает лишь строки определенного формата, вероятно, имеется необходимость помочь пользователям вводить "допустимые" строки. Имеется множество стандартных форматов вводимых данных - даты, телефонные номера, почтовые индексы, номера социального страхования. Ключ к успешному проектированию элемента ввода с проверкой данных - в хорошо развитой обратной связи с пользователем.

Элементы управления отображением

Элементы управления отображением используются для управления визуальным представлением информации на экране. Типичными примерами элементов отображения являются разделители и полосы прокрутки. Сюда же входят разделители страниц, линейки, направляющие, сетки и рамки.

Текстовые элементы

Вероятно, самый простой элемент управления отображением - элемент вывода текстовой информации, который отображает текстовое сообщение в некоторой позиции на экране. Он предоставляет текстовые метки для других элементов управления и выводит данные, которые не могут или не должны быть изменены пользователем. Единственная серьезная проблема этого элемента состоит в том, что он зачастую используется там, где должны присутствовать элементы ввода (и наоборот).



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 125 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Полосы прокрутки

Полосы прокрутки служат важной цели - они позволяют осмысленным образом помещать большие объемы информации внутри рамок окон и панелей. К сожалению, они расходуют экранное пространство и ими сложно манипулировать. Однако замечательное преимущество полосы прокрутки состоит в создании контекста текущего положения в окне. Бегунок полосы прокрутки указывает текущее положение и нередко масштаб "территории доступной для прокрутки.

Разделители

Разделители - удобный инструмент для разделения главного окна приложения на несколько связанных между собой панелей, в каждой из которых можно просматривать, изменять или переносить ту или иную информацию. Подвижные разделители всегда должны сообщать о своей подвижности посредством изменения формы курсора. Однако следует проявлять осторожность, выбирая, какие именно разделители должны стать подвижными. В общем случае разделитель не должен перемещаться таким образом, чтобы содержимое панели становилось непригодным к использованию.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 126 из 469

Назад

На весь экран

Закрыть



Рисунок 14. Скриншоты популярной игры "Cut the Rope". Новостная панель появляется, если потянуть за кольцо

13.4 Рекомендации по проектированию GUI под Android

Рекомендации разработчиков. Android Guideline

Когда платформа Android только появилась, не было никаких рекомендаций по разработке дизайна, поэтому все разработчики проектировали внешний вид приложений по своему вкусу. Отсутствие единого стиля сказалось на интерфейсах не лучшим образом, многие программы были откровенно некрасивы и неудобны. Кроме того, операционная система Android работает на устройствах с различными экранами, и разработчику необходимо помнить, что его приложение должно масштабироваться под различные параметры смартфонов и планшетов.

В настоящее время существует стандарт Android Design, и, если вы хотите, чтобы ваше приложение стало по-настоящему популярным и нужным, настоятельно рекомендуем его придерживаться. Далее мы рассмотрим основные принципы дизайна. Разумеется, в рамках этого курса невозможно учесть все нюансы. В списке источников есть ссылка на рекомендации от Android User Experience Team, к сожалению, все на английском языке.

Приведем выдержки из рекомендаций по дизайну:

- Реальные объекты гораздо веселее, чем кнопки и меню. Позвольте людям манипулировать знакомыми вещами! Тогда работа будет эффективнее.
- Картинки работают быстрее, чем слова.



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 127 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

- Используйте короткие фразы, состоящие из простых слов. Люди часто пропускают предложения, если они слишком длинные.
- Никогда не теряйте пользовательскую информацию. Если человеку придется вводить данные повторно, велика вероятность того, что он откажется использовать ваше приложение.
- Если объекты похожи, они должны выполнять сходные действия.
- Показывайте только то, что необходимо пользователю именно в этот момент.
- Выводите пользователю сообщения, только если вопрос действительно важен.
- Делайте важные вещи быстро.
- Разбивайте сложные задачи на несколько простых шагов.
- Будьте вежливы и корректны в общении с пользователем.
- Пользователь всегда должен быть уверен в том, что он знает, где сейчас находится. На любом шаге он должен иметь возможность вернуться назад, даже если это прервёт выполнение какой-то задачи.
- Используйте интерфейсные элементы, которые будут работать в любой ситуации.
- Самый главный принцип - НЕ УСЛОЖНЯЙТЕ пользователю жизнь!

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 128 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Обзор интерфейса

Приведем выдержки из рекомендаций по дизайну приложений для Android. Сделаем краткий обзор интерфейса операционной системы.

Домашний экран - это настраиваемая пользователем область, которая может содержать иконки приложений, папки и виджеты. Смартфон может иметь несколько домашних экранов, навигация между ними осуществляется с помощью перелистывания влево или вправо.

На домашнем экране в нижней в центре нижней части есть кнопка для открытия экрана приложений. Экран приложений позволяет пользователю запустить любую из установленных программ. Если устройство было использовано для отладки в процессе разработки, то приложение тоже окажется в этом списке и его можно будет вызвать даже после отключения от компьютера. Если приложение было использовано недавно, его можно найти в списке недавно использованных приложений, который вызывается нажатием на третью кнопку на панели внизу (см. рис. 15).

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 129 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 130 из 469

Назад

На весь экран

Закрыть



Рисунок 15. Домашний экран, экран всех приложений и список недавно использовавшихся приложений

В нижней и верхней частях экрана находятся системные панели, предназначенные для размещения уведомлений и навигации по устройству. Нижняя панель (Navigation Bar) предназначена для навигации на тех устройствах, которые не имеют аппаратных навигационных клавиш (все современные устройства). Верхняя часть экрана (Status Bar) предназначена для вывода различных сведений, например, времени, уровня заряда батареи, сигнала сотовой сети, а так же информационных сообщений.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 131 из 469

Назад

На весь экран

Закрыть

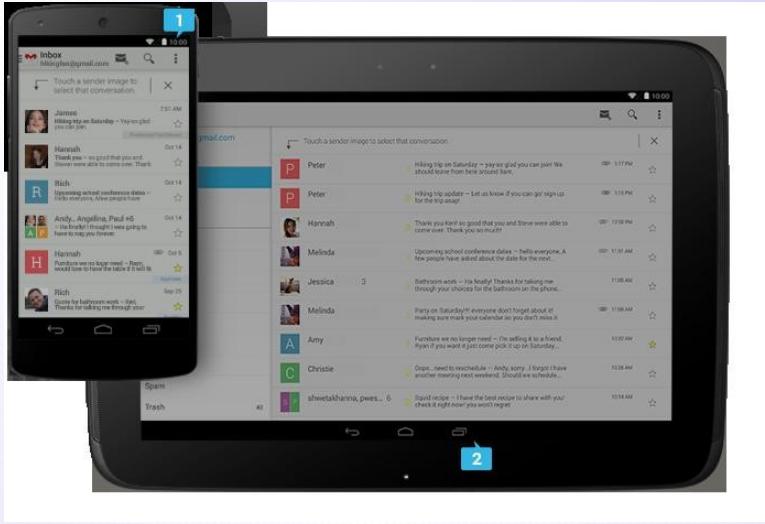


Рисунок 16. Информационная (1) и навигационная (2) панели

Уведомления - это быстрые сообщения, к которым пользователь может получить доступ в любое время из информационной панели. Это могут быть сообщения об обновлениях или других важных сообщениях, которые не настолько серьезны, чтобы прерывать работу пользователя. Но к ним легко можно получить доступ, потянув вниз верхнюю панель. Нажатие на уведомление вызывает соответствующее сообщение.

Шрифты

В дизайне Android используются традиционные типографические инструменты, такие как масштаб, разреженность и выравнивание по сет-

ке. Успешное применение этих выразительных средств помогает пользователю воспринимать информацию быстрее. В версии Android 4.0 Ice Cream Sandwich была представлена шрифтовая гарнитура без засечек Roboto, специально разработанная для экранов с высоким разрешением. Набор шрифтов доступен для бесплатной загрузки. Гарнитура включает в себя прямое и наклонное начертания для шрифтов различной ширины (см. рис. 17).

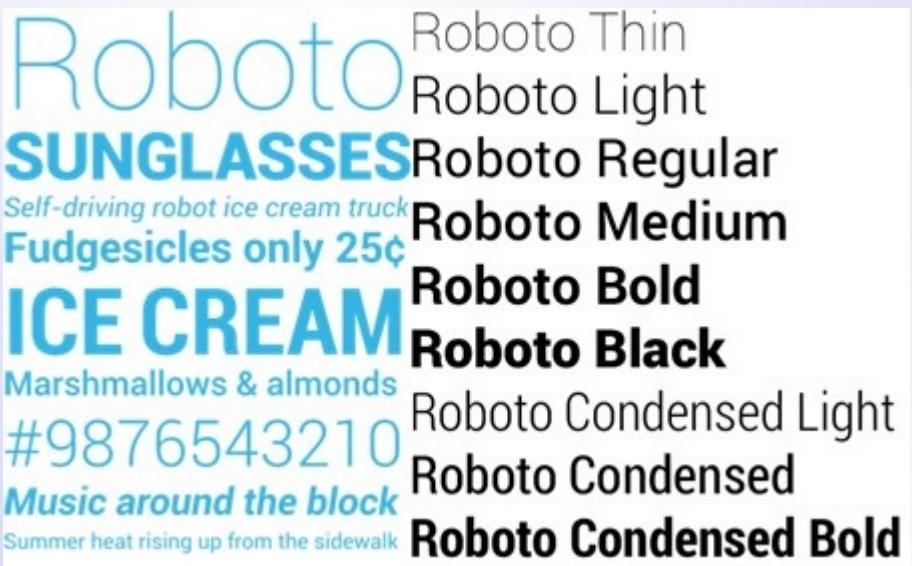


Рисунок 17. Шрифт Roboto и его возможные варианты

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 132 из 469

Назад

На весь экран

Закрыть





Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 133 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Масштабирование

Устройства различаются не только физическими размерами. Важным параметром является плотность экрана (DPI - количество точек на дюйм). Выделяют несколько категорий плотности экрана для Android-устройств: LDPI, MDPI, HDPI, XHDPI, XXHDPI, и XXXHDPI. Чтобы элементы интерфейса имели одинаковый физический размер на экранах разных устройств, компания Google ввела абстрактную единицу измерения - DP (независимый от разрешения пиксель). Один DP равен одному пикселью на экране типа MDPI. Устройства, имеющие меньше 600dp по короткой стороне, считаются телефонами, в противном случае мы говорим о планшетах (см. рис. 18).

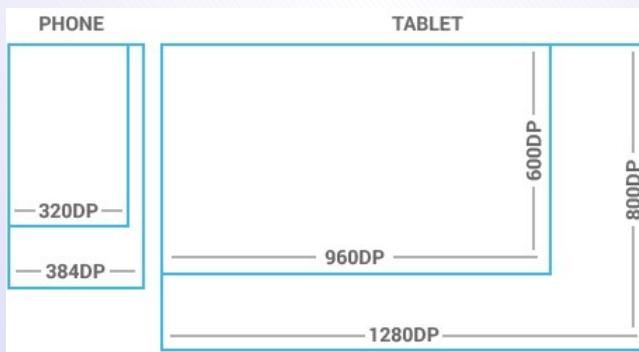


Рисунок 18. Размеры экранов телефонов и планшетов

Соответствие размеров экранов и их плотностей представлено в таблице 2.9:

Таблица 2.9. Плотности и размеры экранов

№	Обозначение	Название	Соответствие	1 dp =
1	LDPI	Low density	120 dpi	0,75 пикселя
2	MDPI	Medium density	160 dpi	1 пиксель
3	HDPI	High density	240 dpi	1,5 пикселя
4	XHDPI	Extra-high density	320 dpi	2 пикселя
5	XXHDPI	Extra-extra!-high density	480 dpi	3 пикселя
6	XXXHDPI	Extra-extra-extra!-high density	640 dpi	4 пикселя



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 134 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 135 из 469

Назад

На весь экран

Закрыть

Минимальный размер элемента управления - 48dp. Такое значение обусловлено тем, что на реальном устройстве оно соответствует 7-10 миллиметрам. При управлении кончиками пальцев такой размер является минимальным для отделения нужного элемента от всех остальных. Если какой-то из размеров элемента управления должен быть больше, чем 48dp, рекомендуется делать его размеры кратным этому значению (см. рис. 19).

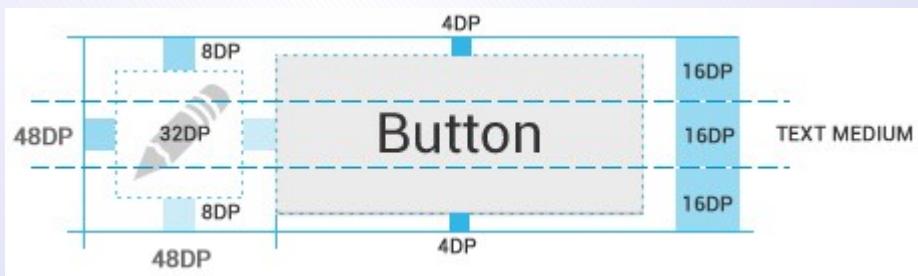
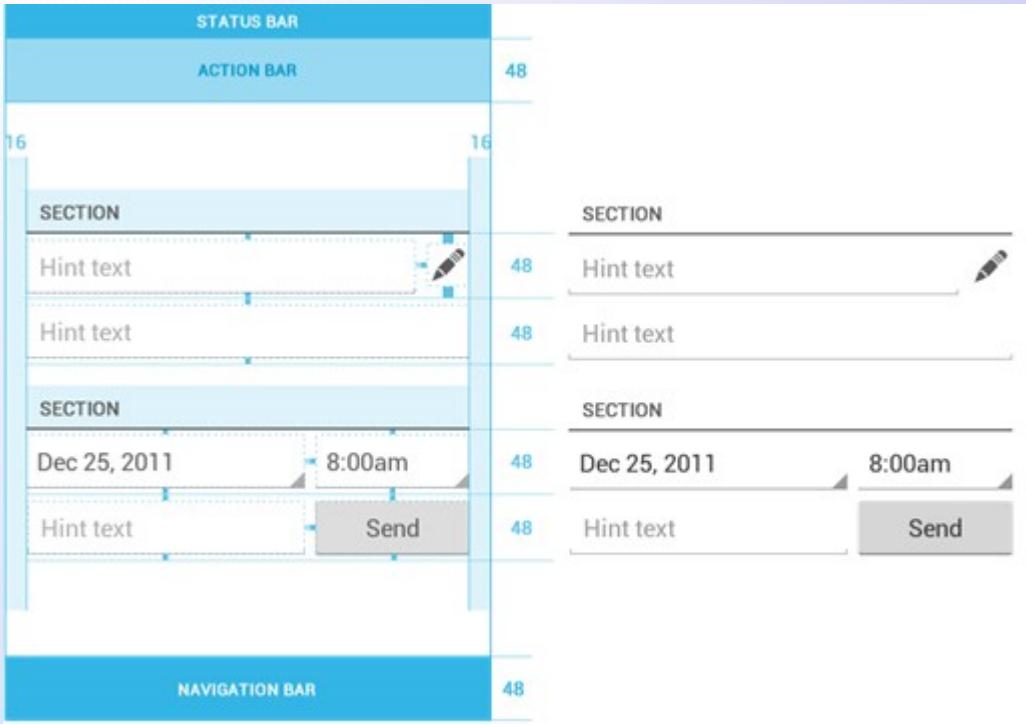


Рисунок 19. Размеры элемента управления кратны 48dp

Расстояние между элементами управления рекомендуется делать кратным 8dp (см. рис. 20).



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 136 из 469

Назад

На весь экран

Закрыть

Рисунок 20. Пример расположения элементов управления

Прочие нюансы дизайна для Android рассмотрены в лабораторной работе.

§14. Основы разработки многооконных приложений

Аннотация: В прошлых лекциях мы рассмотрели особенности разработки приложений для ОС Android и настройки их интерфейсов. Однако все рассмотренные примеры вписывались в рамки экрана отдельно взятого устройства. Что делать в случаях, когда это условие не может быть соблюдено? В лекции рассказывается о работе с диалоговыми окнами, уведомлениями и всплывающими подсказками. Приведены особенности разработки приложений, содержащих несколько *активностей*, а так же способы перемещения между ними в запущенном приложении. Лекция может быть использована как часть курса, так и отдельно от него в целях углубления знаний по разработке многооконных Android-приложений.

Скриншоты приложений взяты из магазина приложений Google Play, официального сайта для разработчиков Android или сделаны самостоятельно с использованием смартфона Мегафон SP-A20i Mint на платформе Intel Medfield.

Презентацию к данной лекции можно скачать [здесь](#).

14.1 Многооконные приложения

Для мобильных приложений главным ограничением является размер экрана устройства. Очень часто невозможно разместить все элементы



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 137 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 138 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

полнофункционального приложения так, чтобы их можно было увидеть одновременно. Очевидным решением этой проблемы является разделение интерфейса на части по какому-либо принципу. Основные пути решения этой проблемы:

- Использовать различные сообщения (диалоговые окна, уведомления, всплывающие подсказки). Этот способ наиболее прост и не требует редактирования файла манифеста, однако очевидно, что так можно решить только часть задач.
- Использовать в одном приложении несколько активностей. Способ универсальный и подходит для любых приложений, однако прежде чем его реализовывать, необходимо очень хорошо продумать структуру будущего приложения. Здесь требуется редактировать манифест и организовать переключение между различными активностями удобным для пользователя способом.
- Разместить компоненты на активности таким образом, что в нужный момент можно будет легко переключиться на работу с другой частью интерфейса.

Каждый способ имеет свои нюансы использования. Рассмотрим их более подробно.

14.2 Работа с диалоговыми окнами

Диалоговые окна

Диалог - это небольшое окно, позволяющее пользователю получить или ввести дополнительную информацию. Диалоговое окно занимает только часть экрана и обычно используется в модальном режиме. Это означает, что работа приложения приостанавливается до момента, пока пользователь не закроет диалоговое окно. При этом ему, возможно, потребуется ввести какие-то данные или просто выбрать один из вариантов ответа (см. рис. 1).

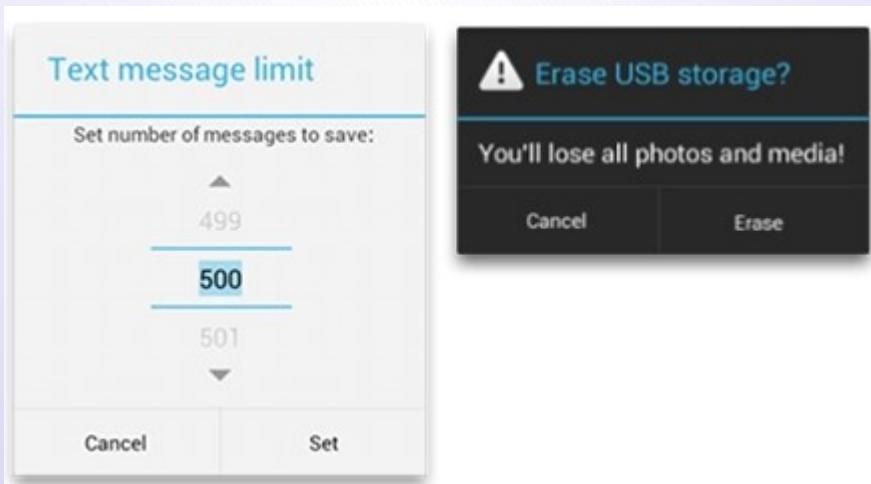


Рисунок 1. Примеры диалоговых окон



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 139 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 140 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

В ОС Android можно выделить три вида диалоговых окон:

- **Класс Dialog и его производные.** Помимо традиционного набора диалоговых окон, он содержит несколько дополнительных вариантов, в которых используются возможности сенсорного интерфейса (см. рис. 1 слева). Диалоги этого типа не создают новых активностей и их не нужно регистрировать в файле манифеста (см. следующие разделы лекции), что существенно упрощает разработку. Однако они работают в модальном режиме и требуют немедленного ответа пользователя, поэтому для простого информирования рекомендуется использовать сообщения следующих двух типов.
- **Уведомления (notifications).** Это сообщения, которые отображаются в верхней панели в области уведомлений. Для того чтобы прочитать это сообщение, необходимо на домашнем экране потянуть вниз верхнюю шторку. Пользователь может это сделать в любой момент времени, следовательно, уведомления стоит использовать, когда сообщение является важным, однако не требует немедленного прочтения и ответа.
- **Всплывающие подсказки (toasts).** Сообщения, которые появляются прямо на экране приложения, перекрывая его интерфейс, и через некоторое время (обычно несколько секунд) автоматически пропадают. Их рекомендуется использовать для простых уведомлений, не требующих ответа пользователя, но важных для продолжения.



ния его работы.

Использование класса Dialog

Класс **Dialog** является базовым для диалогов и редко используется напрямую. Рекомендуется применять производные от этого класса:

- **AlertDialog**. Диалоговое окно может содержать заголовок, до трех кнопок, список выбираемых значений или настраиваемое содержимое. Пример на рис. 1 справа.
- **DatePickerDialog** или **TimePickerDialog**. Диалоговое окно с предопределенным интерфейсом, позволяющее выбрать дату или время.
- **ProgressDialog**. Показывает диалоговое окно, содержащее линейку процесса выполнения какого-то действия. В рекомендациях по дизайну для Android советуют использовать вместо него компонент **ProgressBar**.

Существует возможность создавать собственные диалоговые окна с использованием класса **DialogFragment** в качестве контейнера. В таком случае можно контролировать его поведение. Обратите внимание, что минимальной версией, поддерживающей **DialogFragment** напрямую, является Android 3.0 (API level 11). Если вы хотите использовать возможности этого класса на более ранних версиях, необходимо добавить библиотеку Support Library в ваше приложение.



Рассмотрим создание диалогового окна на примере класса `AlertDialog`. Существует множество вариантов диалоговых окон этого класса, однако все они содержат следующие три части (см. рис. 2):

- Заголовок.** Не является обязательным элементом и должен быть использован, только если содержательная часть занята детализированным сообщением, списком или чем-то еще. Если нужно сделать небольшое сообщение или вопрос, не стоит снабжать его выделенным заголовком.
- Содержательная часть.** Здесь может быть сообщение, список или какой-то другой настраиваемый компонент.
- Управляющие кнопки.** Диалог может содержать не больше трех кнопок. Если элементы содержательной части являются кликабельными, можно вообще обойтись без кнопок (см. рис. 3).

Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 142 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 143 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

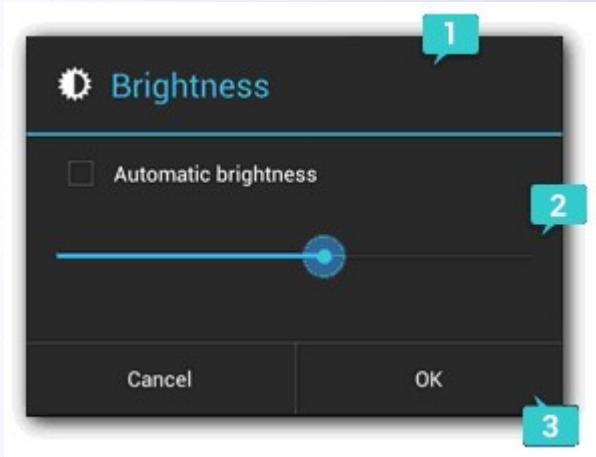


Рисунок 2. Компоновка диалогового окна (класс AlertDialog)

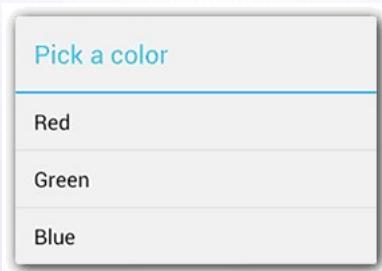


Рисунок 3. Можно выбрать один из трех перечисленных цветом непосредственно нажатием на его название

Возможности использования диалоговых окон будут рассмотрены в лабораторной работе более подробно.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 144 из 469

Назад

На весь экран

Закрыть

Уведомления

Уведомления являются неотъемлемой частью дизайна Android- приложений. На рис. 4 показаны уведомления в свернутом и развернутом виде.



Рисунок 4. Уведомления. Слева - информационная панель со свернутыми уведомлениями, справа эти же уведомления развернуты

Существует два варианта отображения уведомлений в развернутом виде - нормальный и расширенный (доступен начиная с Android 4.1).

Состав уведомления в нормальном виде представлен на рис. 5 Высота уведомления составляет 64 dp. Уведомление содержит следующие части:

1. Заголовок.
2. Большая иконка.
3. Текст сообщения.
4. Информация о сообщении.
5. Маленькая иконка приложения.
6. Время (или дата), когда было отправлено сообщение.

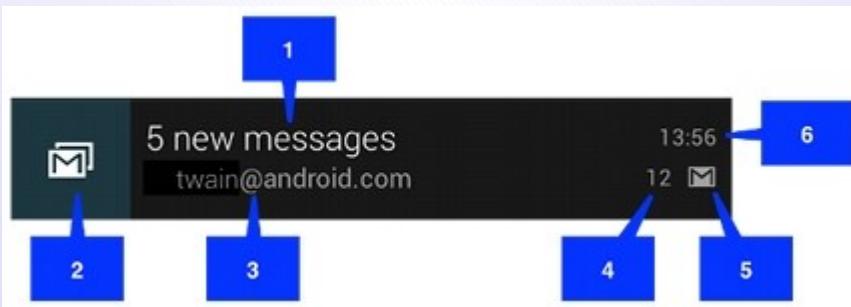


Рисунок 5. Стандартное уведомление



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶ ▶▶

Страница 145 из 469

Назад

На весь экран

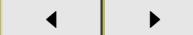
Закрыть



Кафедра ПМиИ

Начало

Содержание



Страница 146 из 469

Назад

На весь экран

Закрыть

Уведомление появляется в расширенном виде, только если оно находится вверху списка уведомлений, или же когда пользователь сделал жест с целью его увеличения. Расширенное уведомление (см. рис. 6) включает в себя те же пункты, что и обычное, но при этом дополнительно содержит детализированную область (на рисунке отмечено номером 7). Это может быть, например, картинка до 256 dp высотой, блок текстовой информации или что-то еще.

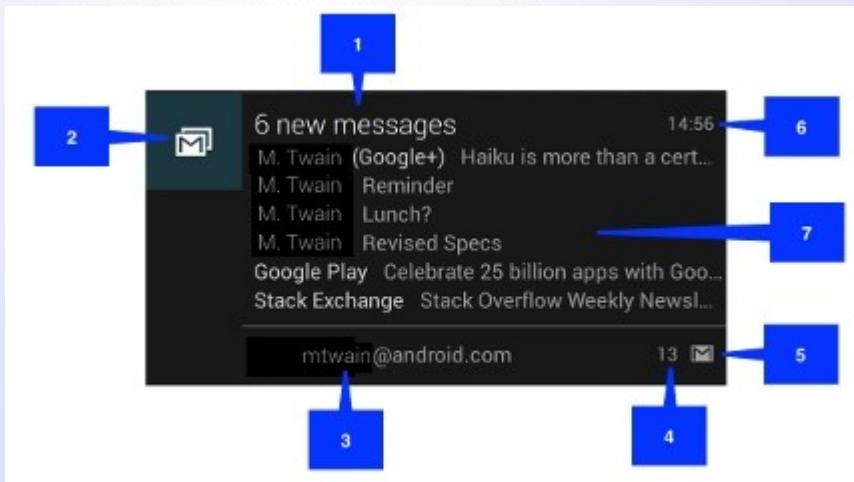


Рисунок 6. Расширенное уведомление

Всплывающие подсказки

Всплывающие подсказки помогают отобразить обратную связь с действиями пользователя. Они занимают минимум места на экране и быстро исчезают (см. рис. 7). Поэтому рекомендуется использовать их для простого уведомления пользователя, когда не требуется получить от него ответа. Всплывающие подсказки могут появляться в любом месте экрана, что позволяет делать их работу более эффективной.

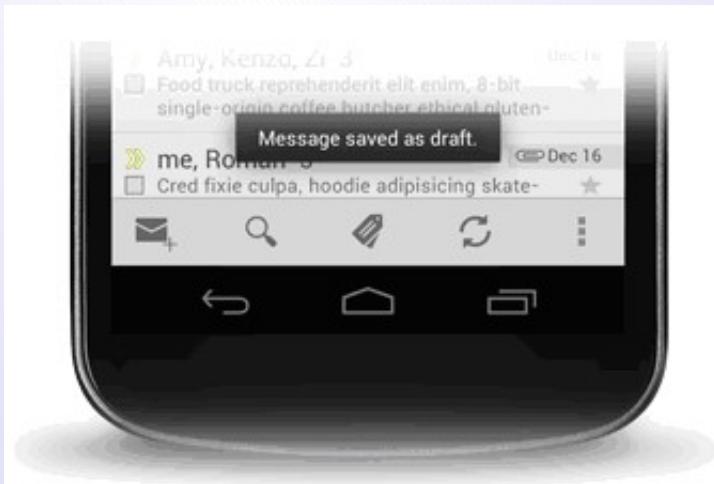


Рисунок 7. Всплывающие подсказки



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 147 из 469

Назад

На весь экран

Закрыть

14.3 Особенности разработки приложения, содержащего несколько активностей

Приложения, содержащие несколько активностей, используются в самых разных сферах. При проектировании такого приложения следует уделить большое внимание распределению его функционала по разным активностям. С одной стороны, не стоит перегружать экран информацией, а с другой - нужна ли *активность*, содержащая только одно поле для ввода? Может быть, стоит ее заменить диалоговым окном?

Существует два основных способа переключения между активностями:

- **При помощи кнопок и других элементов управления.** Не требует перестройки мышления у программистов, которые имеют большой опыт разработки десктопных приложений, а так же у пользователей, привыкших к действиям в стиле "нажал на кнопку, получил результат". Однако этот способ не является наиболее подходящим для сенсорных экранов и требует от опытного пользователя смартфона совершения лишних движений.
- **С использованием сенсорного экрана смартфона.** Основная идея состоит в том, что весь экран мобильного устройства можно использовать в качестве управляющего элемента, и, нажимая на отдельные его участки, пользователь может инициировать те или иные действия. Более подробно возможности жестового интерфейса



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

◀ ▶

◀◀ ▶▶

Страница 148 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 149 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

будут рассмотрены далее, в этой и следующей темах.

Какой из способов выбрать, зависит от конкретной задачи.

Существует ряд правил расположения интерфейсных элементов в зависимости от их важности. Так, кнопку, выполняющую важное действие (например, отправку письма), не стоит располагать в том месте, где она может быть случайно нажата. В то же время *управляющие элементы*, используемые наиболее часто, должны быть расположены наиболее удобным для нажатия образом. Скорее всего, перемещение между активностями будет использоваться не очень часто, поэтому рекомендуется располагать кнопки, *управляющие* этими действиями, в верхней части экрана. Одновременно с этим неплохо продублировать нажатия кнопок перелистыванием между активностями.

В любом случае для вызова другой активности необходимо вручную править *файл манифеста*. Для каждой новой активности необходимо занести информацию о ее имени и названии xml-файла, в котором она описана (см. [листинг 7.1](#)). Обратите внимание, что при загрузке приложения первой появляется *активность*, чье описание находится первым в манифесте! Если вы хотите изменить порядок загрузки активностей, необходимо поместить новую *активность* на первое место.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.myproject.screen"
```



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 150 из 469

Назад

На весь экран

Закрыть

```
android:versionCode="1"
android:versionName="1.0»  
  
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="17"/>  
  
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme»
<activity
    android:name="com.myproject.screen.MainActivity"
    android:label="@string/app_name»
<intent-filter>
    <action android:name="android.intent.action.MAIN"/>
    <category android:name="android.intent.category.LAUNCHER"/>
</intent-filter>
</activity>
<activity
    android:name="com.myproject.screen.AboutActivity"
    android:label="@string/about_title»
```



```
</activity>
<activity
    android:name="com.myproject.screen.SecondActivity"
    android:label="@string/title_activity_second">
</activity>
</application>
</manifest>
```

Листинг 7.1. Исправленный файл манифеста.

14.4 Перелистывание (Swipe)

Существует способ разместить на *активности* больше элементов, чем одновременно помещается на экран, иными словами, отображать по очереди несколько экранов, используя только одну *активность*. В этом случае не нужно править *файл манифеста* - *активность* только одна. Однако для каждого экрана необходимо сделать свой xml-*файл* с его описанием.

Такой способ размещения элементов удобен и программисту, и пользователю. Разработчик может организовать перемещение между частями активности и с помощью кнопок, и с помощью перелистывания. В качестве примера подобного интерфейса можно привести *приложение Twitter* (см. рис. 8).



Рисунок 8. Главный экран приложения Twitter. Можно переключаться между экранами, используя жест горизонтальной прокрутки или с помощью кнопок в верхней части



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 152 из 469

Назад

На весь экран

Закрыть

§15. Использование возможностей смартфона в приложениях



В этом курсе проделана уже немалая работа: установлена и настроена *среда разработки*, созданы первые приложения, хочется двигаться дальше. Разработка мобильных приложений под *Android* имеет ряд особенностей, часть из них мы уже рассмотрели, часть ожидают рассмотрения в ближайших темах. Данная тема, как видно из названия, посвящена возможностям смартфонов и их использованию в приложениях.

Особенностью большинства мобильных устройств является наличие сенсорного экрана и возможность управления пальцем (*touch-interface*), очевидно, что это необходимо учитывать и использовать при разработке приложений. Смартфон, если уж появляется у человека, становится его спутником всегда и везде, в связи с этим, довольно часто используется, как фотоаппарат или проигрыватель музыки, а также смартфоны все чаще становятся инструментами ориентирования на местности.

В данной теме предполагается рассмотрение вопросов разработки приложений, ориентированных на тач-интерфейс, работу со звуком, использование камеры и глобальных систем позиционирования.

Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 153 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

15.1 Отличительные особенности смартфонов

Пришло время поговорить о наиболее интересных возможностях смартфонов, которые можно использовать в приложениях. Ни для кого не секрет, что *смартфон* является "умным телефоном": предполагает обязательное наличие операционной системы и возможность установки дополнительных приложений, существенно расширяющих функционал устройства. С одной стороны, *смартфон* выполняет все привычные функции мобильного телефона и, благодаря компактным размерам, всегда под рукой. С другой стороны, благодаря наличию процессора и операционной системы, позволяет выполнять многие функции полноценного компьютера. Дополнительно ко всему, смартфоны обладают рядом интересных особенностей, не характерных для телефонов и компьютеров.

Для начала обратим внимание на экран смартфона. В современных смартфонах экран занимает практически всю *площадь* передней панели устройства, имеет высокое разрешение и является чувствительным к прикосновениям. Благодаря такой чувствительности, для взаимодействия с устройством и его приложениями можно использовать виртуальные *элементы управления*, чаще всего кнопки, отображаемые на экране. В связи с чем отпадает необходимость в физических кнопках. В смартфонах реализуется, так называемый, *touch-интерфейс* - интерфейс, основанный на виртуальных элементах управления, выбор которых выполняется простым касанием, а также на использовании жестов (*gestures*).



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 154 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 155 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Если точек касания несколько (т. е. используется несколько пальцев), такой *интерфейс*, уже называется multi-touch.

Еще одна особенность смартфонов состоит в том, что для большинства их владельцев не последнюю роль играет возможность использования этого "умного телефона" в качестве аудио или видеоплеера, поэтому современные устройства становятся все более и более мультимедийными. В первой лекции обсуждалось, что в состав платформы *Android* входит набор библиотек для обработки *мультимедиа* *Media Framework*, в котором реализована *поддержка* большинства общих медиа-форматов. В связи с чем, в приложения, разрабатываемые для смартфонов под управлением *Android*, можно интегрировать запись и воспроизведение аудио и видео, а также работу с изображениями.

Важной и часто используемой особенностью смартфонов является наличие камеры, которая позволяет снимать все самое интересное: от первых шагов ребенка до падения метеорита. Телефон всегда под рукой и готов к работе, в связи с этим количество фотографий и небольших видеороликов резко увеличилось, и любое интересное событие в жизни индивидуума может быть запечатлено и сохранено для потомков. С ростом возможностей получения фото и видео материалов увеличивается потребность в приложениях, способных работать с этими материалами. Платформа *Android* позволяет разрабатывать такие приложения, которые предоставляют пользователям возможности делать фотоснимки или записывать видео, каким-то образом обрабатывать полученные ма-



Кафедра ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 156 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

териали и использовать их далее.

Большинство смартфонов оснащены *GPS*-модулем, а некоторые даже комбинированным модулем *GPS/GЛОНАСС*, что позволяет использовать такое устройство в качестве инструмента для ориентирования на местности. Во многих случаях *смартфон* с установленным соответствующим программным обеспечением вполне может заменить *GPS* навигатор. В разрабатываемых приложениях иногда бывает очень полезно добавить возможность получения координат устройства и хозяина, если оба находятся в одном месте, и использовать эти *координаты* для каких-либо целей. Например, уже существуют приложения, которые позволяют отслеживать параметры человека (спортсмена) во время преодоления некоторых расстояний бегом, на велосипеде, на лыжах и т. д. Такое *приложение* работает во время тренировки (устройство должно перемещаться вместе со спортсменом), по окончанию можно получить полную статистику маршрута: *точное время* в пути, *расстояние*, подъемы/спуски, среднюю скорость, потраченные калории и т. д. Заметим, что большая часть информации опирается на данные, полученные со спутников *GPS*.

Рассмотрение особенностей смартфонов будет неполным, если оставить без внимания датчики и сенсоры, которыми оснащены большинство устройств. Эти микроустройства обеспечивают связь смартфона с окружающей средой и добавляют новые удивительные функции. С помощью датчика приближения, например, можно отключать подсветку



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 157 из 469](#)

[Назад](#)

[На весь экран](#)

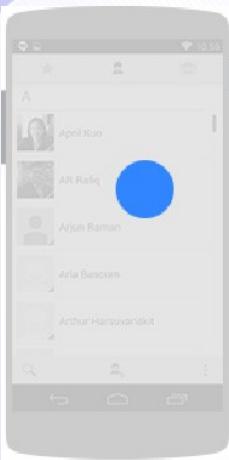
[Закрыть](#)

экрана при приближении телефона к уху пользователя во время разговора, блокировать экран, чтобы не было возможности случайно нажать на отбой. *Акселерометр* может использоваться для смены ориентации экрана, для управления в играх, особенно симуляторах, а также в качестве шагомера. Датчик освещенности позволяет регулировать яркость экрана. *Гироскоп* может применяться для определения более точного позиционирования устройства в пространстве.

Все рассмотренные особенности в совокупности увеличивают привлекательность смартфонов, позволяют разработчикам создавать приложения с разнообразными, полезными, интересными и иногда неожиданными функциями. Далее в лекции рассмотрим перечисленные возможности смартфонов более подробно и узнаем как можно их использовать при разработке приложений.

15.2 Сенсорное (touch) управление

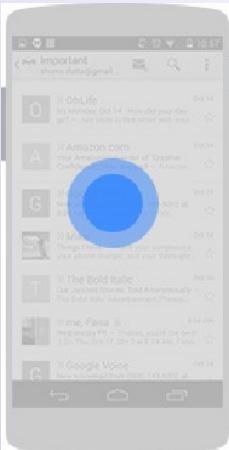
В этом разделе лекции рассмотрим возможности добавления сенсорного управления в мобильные приложения под *Android*. Сенсорное управление подразумевает использование сенсорных жестов для взаимодействия с приложением. Ниже представлен набор жестов, поддерживаемый системой *Android*.



Касание (touch).

Использование: Запуск действия по умолчанию для выбранного элемента.

Выполнение: нажать, отпустить.



Длинное касание (long touch).

Использование: Выбор элемента. Не стоит использовать этот жест для вызова контекстного меню.

Выполнение: нажать, ждать, отпустить.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 158 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 159 из 469

Назад

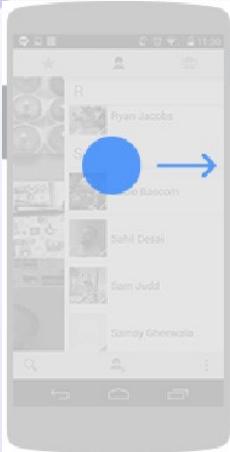
На весь экран

Закрыть

Скольжение или перетаскивание (swipe or drag).

Использование: Прокрутка содержимого или навигация между элементами интерфейса одного уровня иерархии.

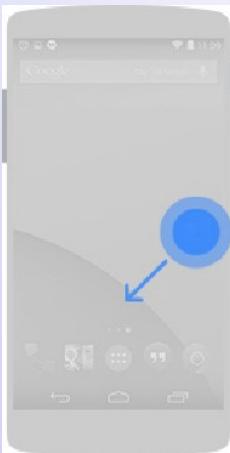
Выполнение: нажать, переместить, отпустить.

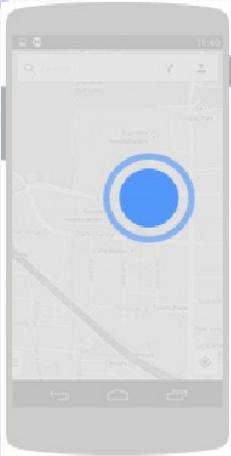


Скольжение после длинного касания (long press drag).

Использование: Перегруппировка данных или перемещение в контейнер.

Выполнение: длительное касание, переместить, отпустить.





Двойное касание (double touch).

Использование: Увеличение масштаба, выделение текста.

Выполнение: быстрая последовательность двух касаний.



Перетаскивание с двойным касанием (double touch drag).

Использование: Изменение размеров: расширение или сужение по отношению к центру жеста.

Выполнение: касание, следующее за двойным касанием со смещением вверх или вниз при этом:

- смещение вверх уменьшает размер содержимого;
- смещение вниз увеличивает размер содержимого.



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 160 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 161 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Сведение пальцев (pinch close).

Использование: уменьшение содержимого, сворачивание.

Выполнение: касание экрана двумя пальцами, свести, отпустить.



Разведение пальцев (pinch open).

Использование: увеличение содержимого, разворачивание.

Выполнение: касание экрана двумя пальцами, развести, отпустить.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 162 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

О возможности управлять приложением с помощью сенсорных жестов можно говорить в том случае, когда *приложение* способно распознать, что под набором касаний экрана скрывается некоторый жест и выполнить соответствующее действие. Процесс распознавания жеста обычно состоит из двух этапов: *сбор данных* и *распознавание* жеста. Рассмотрим эти этапы подробнее.

Сбор данных о сенсорных событиях

Основные действия, которые может произвести пользователь при взаимодействии с сенсорным экраном: коснуться экрана пальцем, переместить палец по экрану и отпустить. Эти действия распознаются системой Android, как сенсорные события (*touch-события*).

Каждый раз при появлении сенсорного события инициируется вызов метода `onTouchEvent()`. Обработка события станет возможной, если этот метод реализован в классе [активности](#) или некоторого компонента, иначе событие просто игнорируется.

Жест начинается, при первом касании экрана, продолжается пока система отслеживает положение пальцев пользователя и заканчивается получением финального события, состоящего в том, что ни один палец не касается экрана. Объект `MotionEvent`, передаваемый в метод `onTouchEvent()`, предоставляет детали каждого взаимодействия. Рассмотрим основные константы класса `MotionEvent`, определяющие сенсорные события:

- `MotionEvent.ACTION_DOWN` - касание экрана пальцем, является начальной точкой для любого сенсорного события или жеста;
- `MotionEvent.ACTION_MOVE` - перемещение пальца по экрану;
- `MotionEvent.ACTION_UP` - поднятие пальца от экрана.

Приложение может использовать предоставленные данные для распознавания жеста.

Можно реализовать свою собственную обработку событий для распознавания жеста, таким образом можно работать с произвольными жестами в приложении. Если же в приложении необходимо использовать стандартные жесты, описанные выше, можно воспользоваться классом `GestureDetector`. Этот класс позволяет распознать стандартные жесты без обработки отдельных сенсорных событий.

Распознавание жестов

Android предоставляет класс `GestureDetector` для распознавания стандартных жестов. Некоторые жесты, которые он поддерживает включают: `onDown()`, `onLongPress()`, `onFling()` и т. д. Можно использовать класс `GestureDetector` в связке с методом `onTouchEvent()`. Подробно распознавание поддерживаемых жестов рассмотрено в первой части лабораторной работы в этой теме.

Начиная с версии 1.6, Android предоставляет API для работы с жестами, который располагается в пакете `android.gesture` и позволяет сохра-



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 163 из 469

Назад

На весь экран

Закрыть



нять, загружать, создавать и распознавать жесты. Виртуальное устройство Android (AVD), начиная все с той же версии 1.6, содержит предустановленное приложение, которое называется Gesture Builder и позволяет создавать жесты. После создания жесты сохраняются на SD карте виртуального устройства и могут быть добавлены в приложение в виде бинарного ресурса.

Для распознавания жестов необходимо добавить компонент `GestureOverlayView`, в XML файл активности. Этот компонент может быть добавлен как обычный элемент графического интерфейса пользователя и встроен в компоновку, например `RelativeLayout`. С другой стороны он может быть использован, как прозрачный слой поверх других компонентов, в этом случае в XML файле активности он должен быть записан, как корневой элемент.

Кроме всего вышеперечисленного, для использования собственных жестов в приложении необходимо реализовать интерфейс `OnGesturePerformedListener` и его метод `onGesturePerformed()`. Подробно создание и использование собственных жестов рассмотрено во второй части лабораторной работы в этой теме.

15.3 Работа с мультимедиа

Мультимедиа библиотека *Android* включает поддержку воспроизведения МНОЖЕСТВА наиболее распространенных форматов, что позво-

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 164 из 469

Назад

На весь экран

Закрыть



ляет легко использовать в приложениях аудио, видео и изображения. Можно проигрывать аудио или видео из медиа файлов сохраненных как ресурсы приложения (*raw* ресурсы), из файлов, расположенных в файловой системе или из потока данных, получаемого через сетевое соединение, для всего этого используется *MediaPlayer API*.

Замечание: проигрывать аудиофайлы можно только на стандартном устройстве вывода, невозможно воспроизводить аудио во время звонка.

Актуальная *информация* о поддерживаемых форматах аудио и видео приводится по ссылке: [здесь](#).

Для воспроизведения аудио и видео *Android* предоставляет *класс MediaPlayer*. Причем при работе с аудиоконтентом этот *класс* позволяет воспроизводить необработанные данные, т. е. возможно проигрывание динамически генерируемого аудио.

Диаграмма жизненного цикла экземпляра класса *MediaPlayer* представлена на рис. 1. Овалы представляют состояния объекта *MediaPlayer*, дуги показывают вызовы каких методов необходимо выполнить, чтобы сменить состояние объекта *MediaPlayer*. Дуги с одной стрелкой представляют вызовы синхронных методов, с двумя стрелками - вызовы асинхронных методов.

В ходе жизненного цикла объект *MediaPlayer* проходит через несколько состояний:

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 165 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

- **бездействие (Idle)** - создан экземпляр класса `MediaPlayer` для создания может использоваться оператор `new` или вызов метода `reset()` (см. рис. 1);
(источник: [здесь](#).)
- **инициализирован (Initialized)** - задан источник медиа- информации, для задания источника используется метод `setDataSource()`;
- **ошибка (Error)** - появилась какая-то ошибка, например, не поддерживаемый аудио/видео формат, слишком высокое разрешение, чтобы вывести объект из этого состояния, необходимо вызвать метод `reset()`;
- **подготовка (Preparing)** - `MediaPlayer` занимается подготовкой медиаисточника к воспроизведению, подготовка инициируется методом `prepareAsync()`;

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 166 из 469

[Назад](#)

[На весь экран](#)

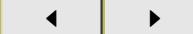
[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание



Страница 167 из 469

Назад

На весь экран

Закрыть

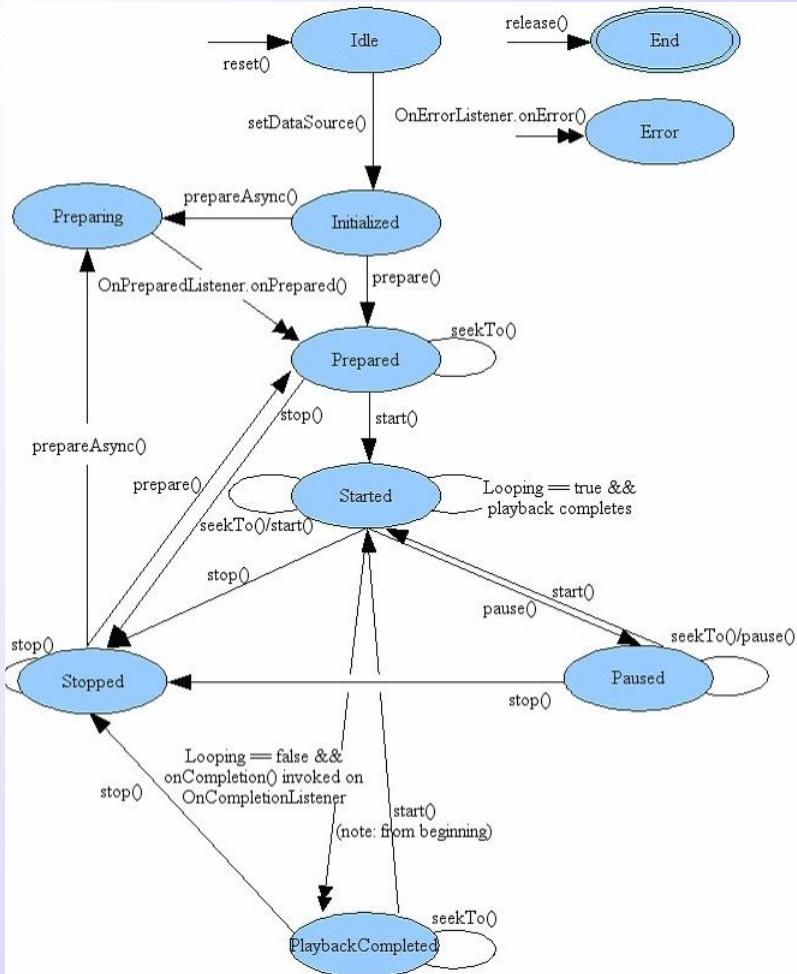


Рисунок 1. Жизненный цикл экземпляра класса MediaPlayer



Кафедра ПМиИ

- **готов (Prepared)** - состояние готовности к воспроизведению, может быть достигнуто двумя способами:
 - синхронный способ: вызов метода `prepare()`, который переводит объект в готовое состояние;
 - асинхронный способ: срабатывание метода `onPrepared()` интерфейса `OnPreparedListener()` в состоянии подготовки, как реакция на событие готовности;
- **запущен (Started)** - выполняется воспроизведение медиа-контента, в это состояние объект переходит после вызова метода `start()`;
- **приостановлен (Paused)** - воспроизведение приостановлено, `MediaPlayer` переходит в это состояние после вызова метода `pause()`;
- **остановлен (Stopped)** - воспроизведение остановлено, `MediaPlayer` переходит в это состояние после вызова метода `stop()`;
- **воспроизведение завершено (Playback Completed)** - достигнут конец воспроизводимого содержания, в это состояние объект переходит после срабатывания метода `onCompleted()` интерфейса-слушателя `OnComptionListener`, как реакции на конец воспроизводимого материала;

Замечание: из состояний `Paused`, `Playback Completed` можно вернуться к воспроизведению вызовом метода `start()`. Из состояния

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 168 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 169 из 469

Назад

На весь экран

Закрыть

Stopped прежде, чем вернуться в состояние воспроизведения, необходимо пройти через подготовку медиа-содержимого.

Вызов метода `seekTo()` позволяет поменять место воспроизведения.

- **конец (End)** - конец жизненного цикла MediaPlayer объекта, в это состояние объект переходит после вызова метода `release()`.

Для получения более детальной информации см. ссылки: [1](#) ; [2](#).

Для записи аудио и видео *Android* предоставляет класс `MediaRecorder`.

Диаграмма жизненного цикла экземпляра класса `MediaRecorder` представлена на рис. 2. Овалы представляют состояния объекта `MediaPlayer`, дуги показывают вызовы каких методов необходимо выполнить, чтобы сменить состояние объекта `MediaPlayer`. Дуги с одной стрелкой представляют вызовы синхронных методов, с двумя стрелками - вызовы асинхронных методов.

(источник: [здесь](#).)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 170 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

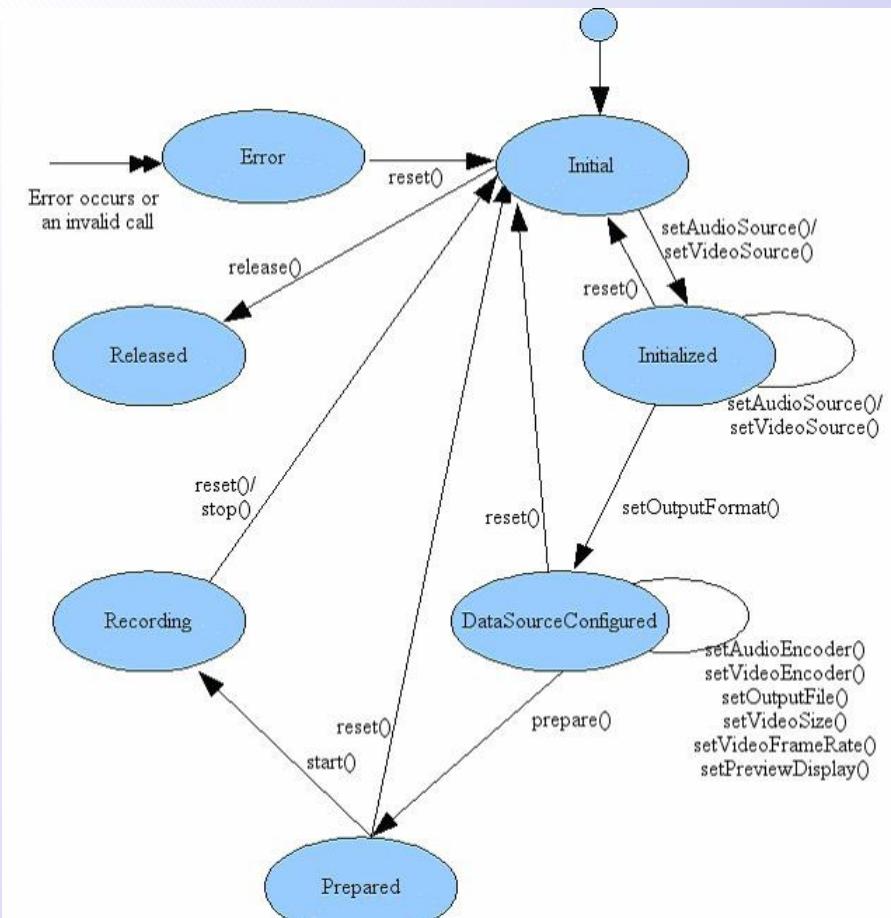


Рисунок 2. Жизненный цикл экземпляра класса `MediaRecorder`



В ходе жизненного цикла объект MediaRecorder проходит через несколько состояний:

**начальное
(Initial)**

- создан объект класса `MediaRecorder`, для создания может использоваться оператор `new` или вызов метода `reset()` (см. рис. 2);

**инициализирован
(Initialized)**

- объект `MediaRecorder` готов к использованию, в данное состояние объект переходит после вызова одного из методов `setAudioSource()` или `setVideoSource()`, которые задают источники аудио или видео для записи;

**сконфигурирован
приемник дан-
ных для записи
(Data Source
Configured)**

- задаются основные свойства приемника данных, состояние инициируется методом `setOutputFormat()`, для настройки свойств должны быть выполнены некоторые методы из списка: `setAudioEncoder()`, `setVideoEncoder()`, `setOutputFile()`, `setVideoSize()`, `setVideoFrameRate()`, `setPreviewDisplay()`;

готов (Prepared)

- состояние готовности к записи, инициируется методом `prepare()`;

**записывает
(Recording)**

- идет запись, инициируется вызовом метода `start()`;

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 171 из 469

Назад

На весь экран

Закрыть



освобожден (Released)

- запись завершена, все ресурсы освобождены.

Для получения более детальной информации см. ссылки: [1](#); [2](#).

15.4 Использование встроенной камеры

Платформа *Android* включает поддержку камеры, доступной на устройстве, позволяющей приложениям получать фотографии и записывать видео. Для решения этих задач, существует два способа:

1. непосредственное обращение к камере;
2. использование намерений (Intent) для вызова существующего приложения.

Рассмотрим основные относящиеся к делу классы:

Camera

- класс, реализующий управление камерами устройства. Этот класс используется для получения фотографий или записи видео при создании приложения, работающего с камерой.

SurfaceView

- класс, используемый для предоставления пользователю возможности предварительного просмотра.

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 172 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 173 из 469

Назад

На весь экран

Закрыть

MediaRecorder

- класс, используемый для записи видео с камеры.

Intent

- класс, содержащий абстрактное описание выполняемой операции, которое передается системе Android, а ОС сама находит и запускает необходимое приложение и возвращает результат его работы. Для работы с камерой используются два типа намерений:

- `MediaStore.ACTION_IMAGE_CAPTURE`
 - для запроса на выполнение фотоснимков;
- `MediaStore.ACTION_VIDEO_CAPTURE` - для запроса на запись видео.

Подробно процесс разработки приложения, позволяющего производить фото и видеосъемку рассмотрен в третьей части лабораторной работы к данной теме.

15.5 Взаимодействие с системами позиционирования

Системы позиционирования позволяют определить местоположение в некоторой системе координат, обычно определяются широта и долгота. Так как *смартфон* является мобильным телефоном, ему доступны



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 174 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

методы, обычно используемые мобильными телефонами для определения своего местоположения.

- Во-первых, смартфон постоянно связывается с сотовой вышкой, в зоне действия которой он находится. Каждая сотовая вышка в мире имеет уникальный идентификатор, называемый идентификатором соты (Cell ID), а также для нее точно известны широта и долгота ее расположения. В связи с этим, смартфон, зная идентификатор соты, в которой он находится, может получить географические координаты центра этой соты. Радиусы сот варьируются в зависимости от того, насколько активный сетевой трафик ожидается в конкретном районе. Разумеется, такой способ позиционирования дает очень приблизительные результаты, что называется: "плюс-минус трамвайная остановка".
- Во-вторых, чаще всего смартфон оказывается в зоне действия более, чем одной сотовой вышки. В современных мобильных технологиях, начиная с поколения 2G, сотовая вышка может определить, с какого направления приходит сигнал. В случае, когда телефон находится в зоне действия двух или трех сотовых вышек, они могут выполнять триангуляцию его местоположения. Телефон может запросить у сети информацию о том, где он находится. Такая техника определения местоположения может быть очень точной и не требует установки дополнительного оборудования.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 175 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Дополнительно к возможностям определения местоположения, доступным обычным мобильным телефонам, большинство смартфонов укомплектованы спутниковыми системами глобального позиционирования (*Global Positioning System, GPS*). В настоящее время наиболее распространенными в мире системами глобального спутникового позиционирования являются: GPS, разработанная и реализованная в США, и система ГЛОНАСС (Глобальная навигационная спутниковая система), советская, а позже российская спутниковая система навигации. Многие смартфоны могут использовать сигналы сразу от двух навигационных систем, что позволяет серьезно увеличить надежность и точность определения координат, прежде всего, в городских условиях.

В дополнение к вышеперечисленным методам позиционирования, добавляется возможность использования сигналов WiFi, Bluetooth и NFC, а также внутреннего сенсора для более точной геолокации, особенно внутри помещений.

В этом разделе нас, в первую очередь, будет интересовать возможность добавления в приложения способностей определять координаты устройства и работать с картами. При создании приложений, учитывающих текущее местоположение, под *Android* можно воспользоваться *GPS* и определением местоположения в сети (с помощью *Network Location Provider*). Несмотря на то, что *GPS* дает более точные результаты, он не очень хорошо работает в помещениях (чаще не работает), он сильно расходует заряд батареи и скорость определения координат не всегда



соответствует ожиданиям пользователя. *Network Location Provider* определяет *координаты*, используя сигналы сотовых вышек и WiFi, может работать как на улице, так и внутри помещений, более экономно расходует заряд батареи и работает быстрее по сравнению с *GPS*. Для получения координат в приложении можно использовать оба способа или один из них на выбор.

Android предоставляет приложениям *доступ* к геолокационным возможностям мобильного устройства, через классы пакета *android.location*. Центральным классом этого пакета является *класс LocationManager*, который предоставляет *доступ* к системным *сервисам* для определения координат устройства.

В приложения можно добавлять карты, используя Google Maps *Android API*, которое автоматически управляет доступом к серверам Google Maps, загрузкой данных, отображением карт и сенсорными жестами на карте. Также можно использовать вызовы *API* для добавления маркеров, многоугольников и внешних прозрачных слоев, а также для изменения пользовательского представления отдельных участков карты.

Ключевым классом в Google Maps *Android API* является *класс MapView*, который отображает карту с данными полученными из сервиса Google Maps. Когда MapView имеет фокус, он может перехватывать нажатия клавиш и сенсорные жесты для выполнения автоматического перемещения и изменения масштаба карты, а также может управлять сетевыми запросами для получения дополнительных фрагментов кар-

Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 176 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 177 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

ты. Этот класс так же предоставляет все элементы пользовательского интерфейса, необходимые для управления картой.

Google Maps *Android API* не является частью платформы *Android*, но доступен на любом устройстве с Google *Play Store*, работающем, начиная с *Android 2.2*, через Google *Play services*. Чтобы обеспечить возможность интеграции Google Maps в приложения, в *Android SDK* необходимо установить библиотеку Google *Play services*.

Подробнее вопросы добавления в приложения геолокационных возможностей и использование карт (Google Maps) рассмотрены в четвертой части лабораторной работы к данной теме.

15.6 Другие сенсоры и датчики

Большинство устройств, работающих под управлением *Android*, укомплектованы встроенными сенсорами, которые предоставляют исходные данные высокой точности. Сенсоры могут быть полезны в том случае, если необходимо зарегистрировать положение и перемещения, повороты устройства в трехмерном пространстве, а также изменения параметров окружающей среды.

Платформа *Android* поддерживает три категории сенсоров:



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 178 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Датчики движе- ния

Эти сенсоры измеряют силы ускорения и вра-
щательные силы по трем осям. Эта категория
включает акселерометры, гироскопы, датчи-
ки вектора вращения и сенсоры силы тяже-
сти.

Датчики окру- жающей среды

Эти сенсоры измеряют различные парамет-
ры окружающей среды, такие как температу-
ра воздуха и давление, освещенность и влаж-
ность. Эта категория включает барометры,
термометры и датчики освещенности.

Датчики поло- жения

Эти сенсоры измеряют физическое положение
устройства. Эта категория включает магни-
тотометры и датчики ориентации устройства в
пространстве.

Сенсоры могут быть реализованы аппаратно или программно.

Аппаратно-реализованные датчики являются физическими элементами
встроенным в *мобильное устройство*, они получают данные путем
прямых измерений некоторых свойств, таких как ускорение, сила гео-
магнитного поля или изменение углов. Программно-реализованные дат-
чики получают свои данные с одного или нескольких физических дат-
чиков и вычисляют значение, которое от них ожидается.

Какие типы датчиков поддерживаются *Android* можно узнать по ссыл-

ке: [здесь](#).

Android предоставляет набор классов и интерфейсов для работы с сенсорами. Эти классы и интерфейсы являются частью пакета *android.hardware* и позволяют выполнять следующие задачи:

- определять какие сенсоры доступны на устройстве;
- определять индивидуальные возможности сенсоров, такие как максимальное значение, производитель, требования к потребляемой энергии и разрешения;
- собирать данные с сенсоров и определять минимальную частоту, с которой выполняется сбор данных;
- подключать и отключать слушателей событий от датчиков, события состоят в изменении значений датчиков.

Для работы с датчиками *Android* предоставляет следующие классы и интерфейсы:



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 179 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 180 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

SensorManager

-Этот класс может использоваться для создания экземпляра сервиса, связанного с сенсором. Также он предоставляет различные методы для доступа и составления списка сенсоров, подключения и отключения слушателей событий от сенсоров, сбора информации. Этот класс содержит константы, которые используются для задания точности сенсора, частоты получения данных и настройки датчиков.

Sensor

-Этот класс используется для создания экземпляра датчика, предоставляет методы, позволяющие определить свойства сенсора.

SensorEvent

-Система использует этот класс для создания объекта, соответствующего событию датчика и предоставляющего следующую информацию: данные сенсора; тип сенсора, который породил событие, точность данных и время появления события.

SensorEventListener

-Данный интерфейс может использоваться для реализации двух методов, получающих уведомления (события датчиков), когда меняется значение сенсора или когда меняется точность сенсора.

Использование в приложении полученных от сенсоров данных будет рассмотрено в лабораторной работе темы 7. Подробнее об использовании сенсоров можно узнать по ссылке: [здесь](#).



Кафедра ПМиИ

[Начало](#)

[Содержание](#)



Страница 181 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

§16. Использование библиотек

Аннотация: Прежде чем браться за решение какой-то вспомогательной задачи, следует сначала выяснить, не была ли она решена кем-то ранее. Повторное использование кода позволяет сберечь ресурсы на выполнение проекта. Такие возможности предоставляют подключаемые библиотеки, рассмотрению возможностей которых посвящена данная тема. В лекции приведена классификация библиотек по их назначению и возможности их подключения. Рассматриваются некоторые популярные подключаемые библиотеки, как официальные, так и альтернативные. Затрагиваются вопросы безопасности использования библиотек. Лекция может быть использована как в рамках изучения данного курса, так и отдельно от него, если читатель желает подробнее ознакомиться с возможностью работы с подключаемыми библиотеками.

Скриншоты приложений взяты из [магазина приложений Google Play](#) или сделаны самостоятельно, в том числе с использованием смартфона Мегафон SP-A20i Mint на платформе Intel Medfield. Некоторые иллюстрации взяты с официальных сайтов.

16.1 Библиотеки

Использование библиотек

Библиотека (от англ. *library*) в программировании - сборник под-



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 182 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



программ или объектов, используемых для разработки программного обеспечения (ПО). Для ОС Android существует большое количество подключаемых библиотек. Их можно классифицировать в зависимости от их предназначения. Выделим следующие группы:

- **Библиотеки совместимости.** Они позволяют использовать возможности, появившиеся в какой-то версии ОС Android, на более ранних версиях платформы. Дело в том, что новые версии API выходят гораздо быстрее, чем в широком использовании оказываются устройства, поддерживающие эту версию. Разработчик с одной стороны должен ориентироваться на новые возможности и уметь их использовать, а с другой - стараться сделать так, чтобы приложение работало на максимальном количестве устройств. Библиотеки совместимости позволяют сделать это противоречие менее жестким.
- **Библиотеки специального назначения.** Используются для разработки игр, работы с социальными сетями, сбора статистики и в других случаях.
- **Библиотеки, предоставляющие дополнительные возможности.** В эту категорию можно отнести большое количество самых разных библиотек. Сюда можно отнести библиотеки рисования графиков, работы с изображениями, модифицированные элементы управления и многое другое.

Подключение библиотек

Кафедра
ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 183 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Библиотеки могут поставляться как в собранном и уже готовом к использованию виде (jar-файлы), так и в исходниках. Подключить библиотеку (файл *.jar) очень просто. Достаточно создать папку **libs** в проекте (на том же уровне, что и папки **src** и **res**) и копировать туда файл библиотеки (можно просто перетащить). Дальше необходимо добавить ее в проект через меню **Project -> Properties**.

Если библиотека представлена в виде исходного кода, необходимо ее предварительно собрать. Необходимо щелкнуть правой кнопкой по корневой папке проекта - > **Export: -> Java -> Runnable JAR file -> Указать класс для запуска -> Указать место сборки -> Finish** (см. рис. 1).

Кафедра ПМиИ

[Начало](#)

[Содержание](#)



Страница 184 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 185 из 469

Назад

На весь экран

Закрыть

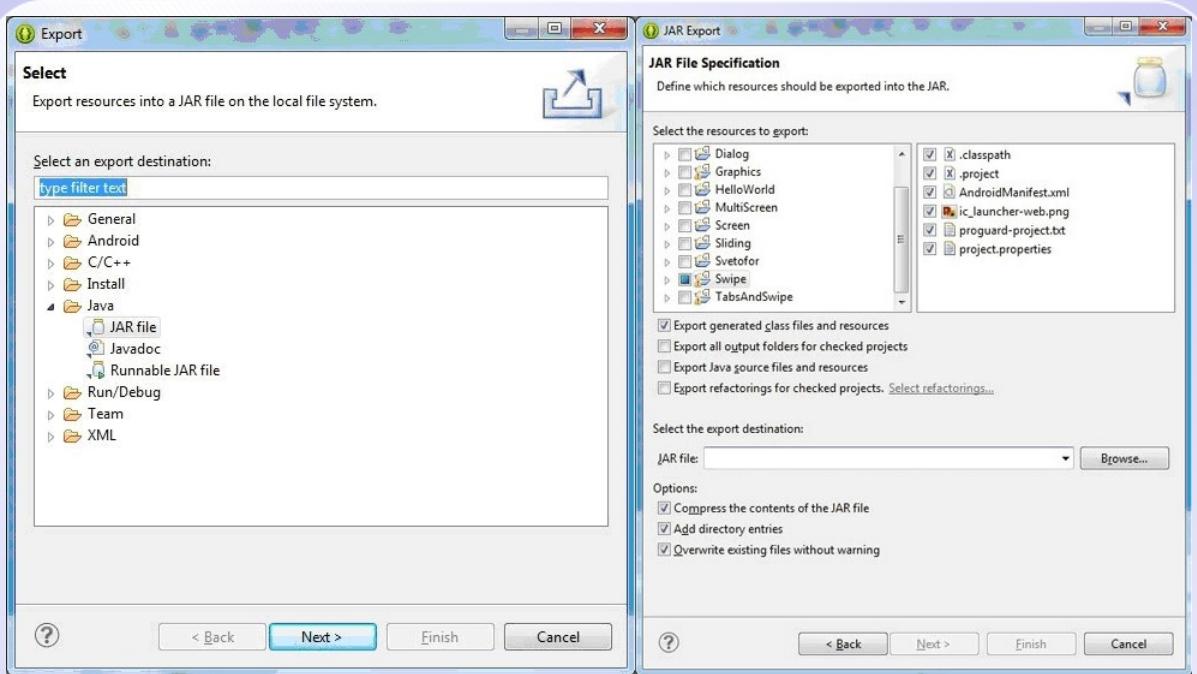


Рисунок 1. Сборка библиотеки из исходного кода

16.2 Обзор популярных библиотек

Android Support Library

Android Support Library - это набор библиотек, которые обеспечивают обратную совместимость новых API на более старых версиях платформы. Каждая библиотека из этого набора обладает обратной совместимостью к конкретному уровню Android API. Это означает, что в-



ши приложения смогут использовать возможности библиотеки и быть запущены на устройствах Android 1.6 (API level 4) и выше.

Подключение библиотек поддержки в Android является хорошим тоном в разработке приложений, зависящих от версии и возможностей платформы. Использование возможностей Support Library поможет вам распространить ваше приложение для большего числа пользователей. Если вы используете примеры Android-приложений, вы можете заметить, что все они содержат по умолчанию одну или несколько библиотек поддержки.

О возможностях различных версий [*Android Support Library*](#) можно узнать на [официальном сайте](#). Скачать и установить эти библиотеки можно с помощью Android [*SDK Manager*](#), выбрав в разделе Extras нужные пункты (см. рис. 2).

Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 186 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание



Страница 187 из 469

Назад

На весь экран

Закрыть

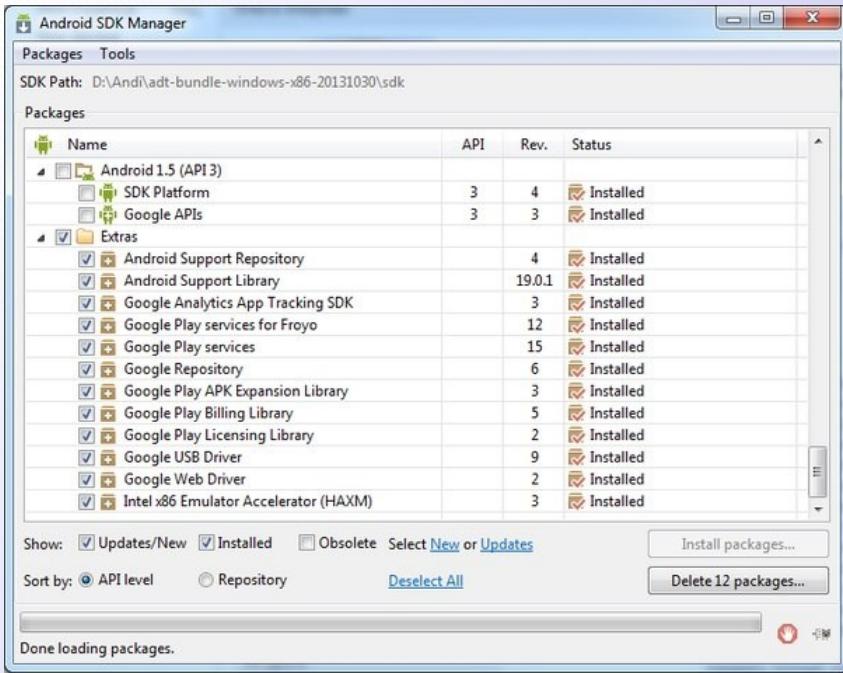


Рисунок 2. Подключение Android Support Library

При настройке обратной совместимости необходимо отредактировать файл манифеста, указав в нем минимальную версию Android SDK, которая необходима для запуска приложения, и основную (целевую) версию:

```
<uses-sdk  
    android:minSdkVersion="7"  
    android:targetSdkVersion="17"/>
```

Сторонние библиотеки

Помимо официальных и поддержанных Google библиотек совместимости, существуют аналогичные решения от сторонних разработчиков. Вы можете их использовать на свой страх и риск, но в некоторых случаях они предпочтительнее, так как обладают некоторыми дополнительными возможностями.

NineOldAndroids - один из примеров таких библиотек. Она предназначена для использования анимации, которая стала доступна только с версии Honeycomb (Android 3.0), на всех более ранних платформах. Она поддерживает различные возможности анимации и очень удобна в использовании (см. рис. 3). Главным преимуществом этой библиотеки является то, что она работает для всех версий Android, начиная с 1.0.

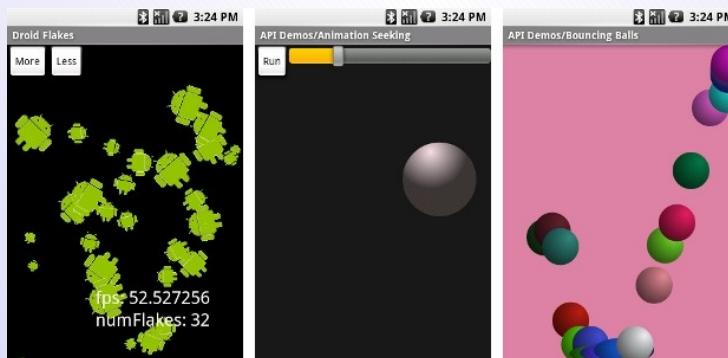


Рисунок 3. Использование библиотеки NineOldAndroids



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 188 из 469

Назад

На весь экран

Закрыть

Другим примером подобного решения является автономная библиотека **ActionBarSherlock**. С ее помощью можно использовать нативный компонент ActionBar, появившийся только в версии Android 4.0, в более ранних (2.x и выше). Ее можно загрузить с [официального сайта](#). Там же содержатся подробные указания по работе с этой библиотекой, имеются примеры. На рис. 4 представлена работа приложения, использующего библиотеку ActionBarSherlock, на устройствах со старыми версиями.

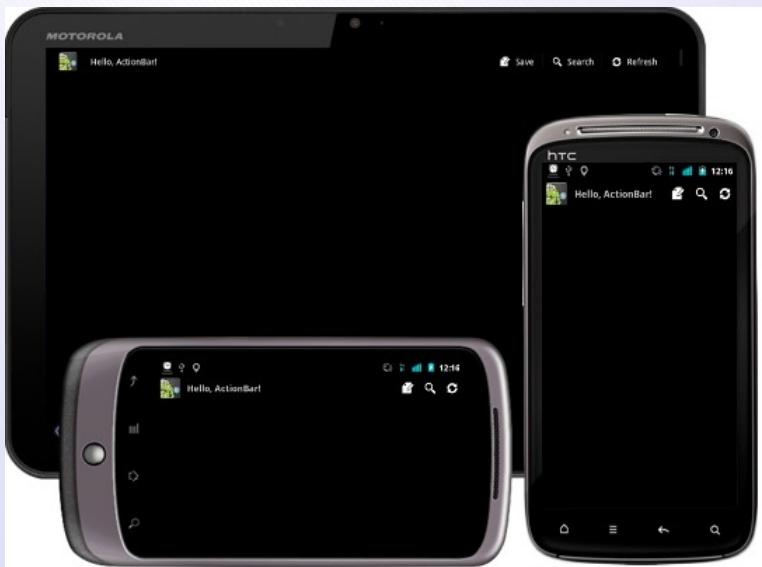


Рисунок 4. Использование библиотеки ActionBarSherlock

Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#)

[▶](#)

[◀◀](#)

[▶▶](#)

[Страница 189 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

[Закрыть](#)



Библиотеки специального назначения

Yandex.Metrica for Apps - набор библиотек для сбора статистики использования мобильного приложения. Метрика показывает актуальную статистику об использовании приложения. [сервис](#) позволяет отвечать на вопросы об аудитории и выделять любые ее сегменты. Инструменты помогают понять, как люди пользуются приложением. SDK позволяет отслеживать следующие данные:

- информация об устройстве;
- информация о сессиях;
- информация об источнике перехода пользователя на страницу скачивания приложения;
- действия, выполненные пользователем в приложении;
- местоположение пользователя;
- ошибки, возникающие во время использования приложения;
- собственные события;
- другие данные (например, количество пользователей, установивших приложение).

Подробные инструкции по добавлению в приложение Яндекс.Метрики и работе с ней есть на [официальном сайте](#).

Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 190 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Facebook SDK for Android - официальная библиотека Facebook для Android. Позволяет писать сообщения на стену, читать и менять статусы, смотреть ленту друзей и многое другое. **Официальный сайт** содержит большое количество примеров и указаний по разработке приложений (см. рис. 5).

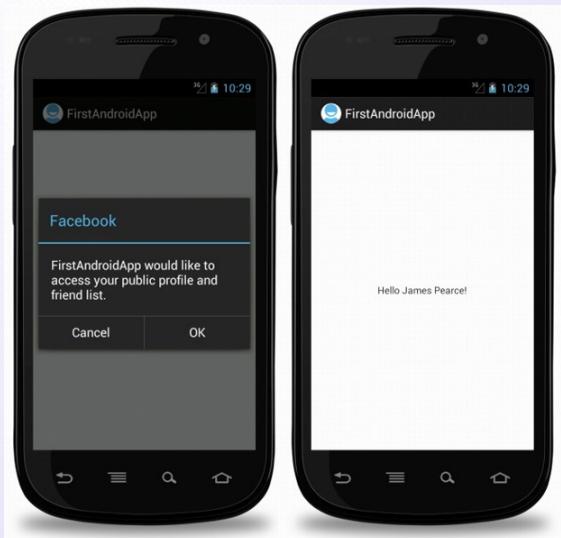


Рисунок 5. Пример приложения, подготовленного с использованием Facebook SDK for Android



*Кафедра
ПМиИ*

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 191 из 469

Назад

На весь экран

Закрыть

Прикладные библиотеки

К этой категории можно отнести различные библиотеки, предоставляющие дополнительные возможности.

Universal Image Loader for Android - мощная и гибкая библиотека, предназначенная для загрузки, кеширования и отображения картинок в Android. Подробности на [сайте](#).

Возможности:

- Многопоточная загрузка изображений.
- Широкие возможности настройки и конфигурирования.
- Кеширование загруженных изображений как в оперативной памяти, так и на карте.
- Поддержка виджетов.
- Поддерживает Android 2.0 и выше.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 192 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 193 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Рисунок 6. Пример работы с библиотекой Universal Image Loader for Android

jsoup: Java HTML Parser предназначена для парсинга HTML-страниц. Предоставляет очень удобный API для извлечения данных и манипуляции с ними, используя DOM, CSS и методы в стиле jQuery. Поддерживает спецификации HTML5 и позволяет парсить страницы так же, как это делают современные браузеры.

Возможности:

- Может принимать в качестве параметра URL, файл или строку.
- Находит и извлекает данные, используя DOM и селекторы CSS.
- Позволяет манипулировать HTML-элементами, атрибутами и текстом.

- Выводит чистый HTML.

Примеры ее использования есть на [официальном сайте](#).

Android Holo ColorPicker - удобная библиотека, позволяющая выбирать цвет с использованием цветового колеса, выполненная в официально рекомендованном стиле Holo. [Сайт](#) библиотеки содержит описание работы с ней и необходимые ссылки.

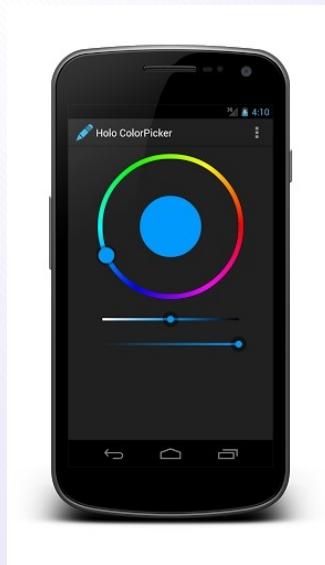


Рисунок 7. Пример использования Android Holo ColorPicker



*Кафедра
ПМиИ*

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 194 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Библиотека **MapNavigator** предназначена для работы с картами Google Maps. Позволяет определять направления и отображать маршрут на карте. Работает только с Google Maps v2. Скачать можно на [официальном сайте](#).

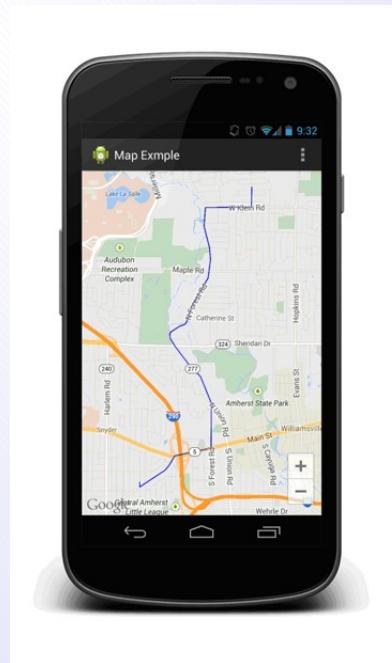


Рисунок 8. Пример использования библиотеки MapNavigator



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 195 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

AChartEngine - библиотека, предназначенная для построения графиков. Позволяет строить графики различных типов:

- Линии графиков функций.
- Поточечные графики.
- Гистограммы.
- Круговые диаграммы.
- Пузырьковые диаграммы.
- Комбинированные диаграммы.
- Другие виды диаграмм и графиков.

Все типы диаграмм поддерживают несколько рядов данных.

[Сайт разработчика](#) содержит подробную документацию, оформленную в стиле Javadoc pages, примеры использования библиотеки, а также ее исходный код.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 196 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 197 из 469

Назад

На весь экран

Закрыть

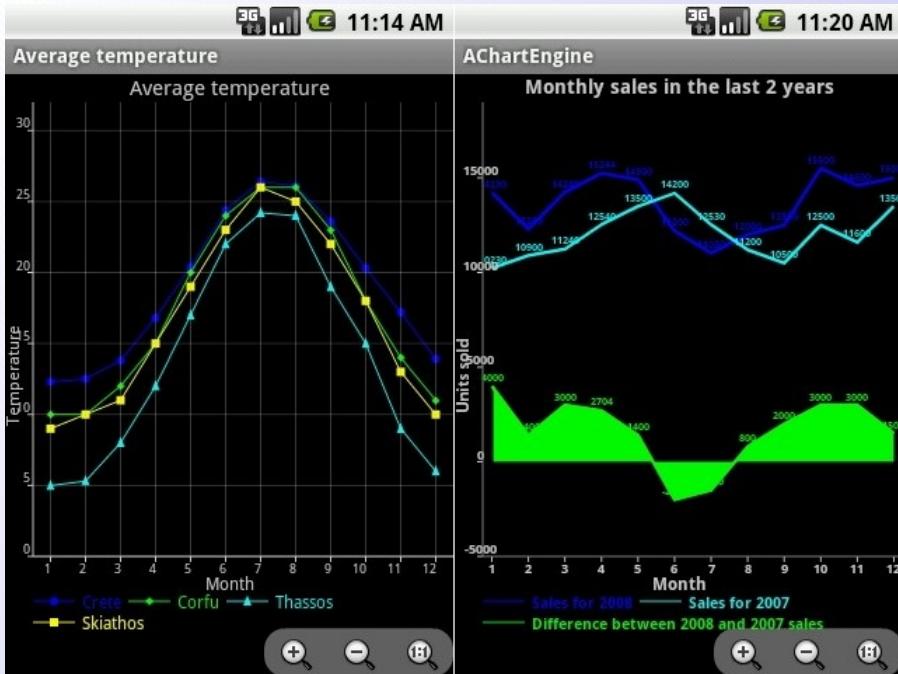


Рисунок 9. Пример использования библиотеки AChartEngine

Разумеется, мы рассмотрели лишь малую долю существующих библиотек. Обзор не претендует на полноту, но это не так важно, так как его целью было представить многообразие возможностей, открывающихся перед разработчиком. Кроме того, большое количество разнообразных библиотек описаны на [сайте](#).

16.3 Безопасность использования подключаемых библиотек

Подключаемые *библиотеки* являются очень удобным инструментом, облегчающим труд программиста. Однако разработчики приложений, использующие сторонние библиотеки подобного рода, часто не подозревают об их проблемах с безопасностью. Библиотека может содержать возможности, которые могут использоваться злоумышленниками в преступных целях.

Например, в октябре 2013 года была опубликована статья с результатами исследования, согласно которому популярная у разработчиков библиотека, предоставляющая возможность отображения рекламы в приложениях, может использоваться для сбора информации и запуска вредоносного кода. Исследователи не раскрыли истинного названия библиотеки, зато описали возможный вред. Например, она может запускать на устройстве произвольный код, извлекать текстовые сообщения, список контактов и вызовов, передавать секретную информацию пользователя в виде простого текста по протоколу *HTTP*, использовать камеру без ведома пользователя, запускать вредоносные java-скрипты. Злоумышленник может превратить эту библиотеку в ботнет, перехватывая ее трафик и отправляя вредоносные команды и код [37].

При выборе библиотеки следует соблюдать осторожность. Если библиотека поставляется в виде исходников и о ней мало информации, не



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 198 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

лишним будет просмотреть ее код в поисках странных функций, которые не нужны для ее функционирования. Но для уже собранной библиотеки *анализ* исходников может стать непростой задачей. Поэтому при выборе библиотеки следует соблюдать ряд правил:

- Не использовать скомпрометированные библиотеки. Если о какой-то библиотеке появляются сведения, что она может содержать вредоносный код, следует отказаться от ее использования в новых проектах и по возможности пересмотреть ее применение в уже существующих.
- С осторожностью использовать библиотеки из сомнительных источников.
- Обязательно ознакомиться с форумами и сайтами, где могут обсуждаться библиотеки. Кроме того, это может помочь вам подобрать наиболее подходящее решение для вашей конкретной задачи.
- По возможности просмотреть исходники.
- Применять другие правила информационной безопасности, которые могут иметь значение в каждом конкретном случае.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 199 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

§17. База данных и мультимедия в Android

Современное *программирование* трудно представить без использования баз данных, рано или поздно в процессе развития приложения появляется осознание необходимости долговременного хранения и обработки структурированной информации. Данная лекция посвящена рассмотрению вопросов, связанных с использованием баз данных SQLite в приложениях, разрабатываемых под *Android*. *Базы данных SQLite* являются основой построения рабочей и функциональной программы, в которой необходимо работать с большими объемами структурированной информации.

Далее в лекции перейдем к рассмотрению таких интересных тем, как создание графических изображений и анимации, а также работа с этими элементами. Платформа *Android* предоставляет разнообразные способы для добавления в приложения и использования графики и анимации.

Очень часто мобильные устройства помогают "скротать время" в очередях, в ожидании транспорта и многих других ситуациях, часто возникающих в современной жизни. Проще всего в такие моменты занять себя несложной игрой, в связи с этим тема разработки игр для мобильных устройств стала довольно популярна в последнее время. Разумеется, разработка игр дело серьезное, но даже отдельному разработчику по силам создать игру, способную увлечь пользователя. В данной теме рассмотрим основные принципы создания игр для смартфонов, в лабораторной



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 200 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

работе рассмотрим процесс создания несложной игры.

17.1 Основы работы с базами данных, SQLite

SQLite - небольшая и при этом мощная система управления базами данных. Эта система создана в 2000 году, ее разработчик доктор Ричард Хипп (Dr. Richard Hipp). В настоящее время является одной из самых распространенных *SQL*-систем управления базами данных в мире. Можно выделить несколько причин такой популярности SQLite: она бесплатная; она маленькая, примерно 150 Кбайт; не требует установки и администрирования. Подробнее см. .

База данных SQLite - это обычный *файл*, его можно перемещать и копировать на другую систему (например, с телефона на рабочий *компьютер*) и она будет отлично работать. *Android* хранит *файл базы данных* приложения в папке (см. рис. 1):

`data/data/packagename/databases/`,

где **packagename** - *имя пакета*, в котором расположено *приложение*.

Для доступа к этому файлу необходимо запускать команды *SQL*, *Android* с помощью вспомогательных классов и удобных методов скрывает часть деталей, но все-таки необходимо иметь хотя бы минимальные знания об *SQL*, чтобы пользоваться этими инструментами.



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 201 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Name	Size	Date	Time	Permissions	Info
com.example.sqllite		2014-02-22	07:10	drwxr-x--x	
cache		2014-02-22	07:10	drwxrwx--x	
databases		2014-02-22	07:10	drwxrwx--x	
myDB	20480	2014-02-22	07:17	-rw-rw----	

Рисунок 1. Расположение файла базы данных SQLite

Обращения к базе данных *SQL* выполняются посредством запросов, существует три основных вида *SQL* запросов: *DDL*, *Modifcation* и *Query*.

- **DDL запросы.** Такие запросы используются для создания таблиц.

Каждая таблица характеризуется именем и описанием столбцов, которое содержит имя столбца и тип данных. В файле базы данных может быть несколько таблиц.

Пример запроса для создания таблицы:

```
create Table_Name (
    _id integer primary key autoincrement,
    field_name_1 text,
    field_name_2 text);
```

Первый столбец обозначен, как **primary key** (первичный ключ), т.е. уникальное число, которое однозначно идентифицирует строку. Слово **autoincrement** указывает, что база данных будет автоматически увеличивать значение ключа при добавлении каждой записи, что и

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 202 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 203 из 469

Назад

На весь экран

Закрыть

обеспечивает его уникальность. Существует договоренность первый столбец всегда называть `_id`, это не жесткое требование SQLite, однако может понадобиться при использовании контент-провайдера в Android.

Стоит иметь в виду, что в SQLite, в отличие от многих других баз данных, типы данных столбцов являются лишь подсказкой, т. е. не вызовет никаких нареканий попытка записать строку в столбец, предназначенный для хранения целых чисел или наоборот. Этот факт можно рассматривать, как особенность базы данных, а не как ошибку, на это обращают внимание авторы SQLite.

- **Modification запросы.** Такие запросы используются для добавления, изменения или удаления записей.

Пример запроса на добавление строки:

`insert into Table _Name values(null, value1, value2);`

В этом случае значения разместятся в соответствующие столбцы таблицы, первое значение задается для поля `_id` и равно `null`, т. к. SQLite вычисляет значение этого поля самостоятельно.

При добавлении можно указывать столбцы, в которые будут размещаться значения, остальные столбцы заполняются значениями по умолчанию, в этом случае можно добавлять элементы в измененном порядке. Пример такого запроса:



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 204 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

```
insert into Table_Name(field_name_2, field_name_1)  
values(value2, value1);
```

В этом случае добавляются значения только в поля `field_name_1` и `field_name_2`, причем изменен порядок следования полей, а вместе с этим и порядок следования значений, иногда это бывает удобно.

Примеры запросов на изменение строки:

```
update Table_Name set Field_Name_1 = value;
```

поменяет значение столбца `Field_Name_1` на `value` во всей таблице;

```
update Table_Name set Field_Name_1 = value where _id = smth;
```

поменяет значение столбца `Field_Name_1` только в той строке, `_id` которой равен `smth`.

Примеры запросов на удаление строк:

```
delete from Table_Name;
```

```
delete from Table_Name where Field_Name_1 = smth;
```

первый запрос удаляет таблицу целиком, второй - только те строки, в которых столбец `Field_Name_1` имеет значение `smth`.

- **Query запросы.** Такие запросы позволяют получать выборки из таблицы по различным критериям. Пример запроса:

```
select from Table_Name where (_id = smth);
```

```
select Field_Name_1,
```



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 205 из 469

Назад

На весь экран

Закрыть

Field_Name_2 from Table_Name
Field_Name_1 = smth);

Первый запрос выводит строку с `_id` равным `smth`, второй - выводит два элемента `Field_Name_1` и `Field_Name_2` строк, в которых `Field_Name_1` равен `smth`.

Вернемся к рассмотрению вопросов, связанных с использованием базы данных SQLite в приложениях под *Android*. Любая база данных, созданная в приложении доступна любому классу приложения, но недоступна из вне. Чтобы открыть доступ к базе данных другим приложениям необходимо использовать контент-провайдеры (*Content Providers*).

Для создания и обновления базы данных в *Android* предусмотрен класс `SQLiteOpenHelper`. При разработке приложения, работающего с базами данных, необходимо создать класс-наследник от `SQLiteOpenHelper`, в котором обязательно реализовать методы:

- `onCreate()` - вызывается при первом создании базы данных;
- `onUpgrade()` - вызывается, когда необходимо обновить базу данных.

По желанию можно реализовать метод:

- `onOpen()` - вызывается при открытии базы данных.

В этом же классе имеет смысл объявить строковые константы, в ко-



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 206 из 469

Назад

На весь экран

Закрыть

торых определить названия таблиц и столбцов. Полученный *класс* позаботится об открытии *базы данных*, если она существует, или о создании ее в противном случае, а так же об обновлении *базы данных* в случае необходимости.

В *Android* предусмотрен *класс* для работы с базой данных *SQLite* напрямую, этот *класс* называется **SQLiteDatabase** и содержит методы:

- openDatabase()** - позволяет открыть базу данных;
- update()** - позволяет обновить строки таблицы базы данных;
- insert()** - позволяет добавлять строки в таблицу базы данных;
- delete()** - позволяет удалять строки из таблицы базы данных;
- query()** - позволяет составлять запросы к базе данных;
- execSQL()** - позволяет выполнять запросы к базе данных.

Для добавления новых строк в таблицу используется *класс* **ContentValues**, каждый *объект* этого класса представляет собой одну строку таблицы и выглядит как ассоциативный *массив* с именами столбцов и значениями, которые им соответствуют.

Для получения результатов запросов к базе данных используется *класс* **Cursor**, объекты этого класса ссылаются на результирующий набор данных, позволяют управлять текущей позицией в возвращаемом при запросе наборе данных.

Для предоставления доступа к данным для других приложений мож-



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 207 из 469

Назад

На весь экран

Закрыть

но использовать контент-провайдеры (ContentProvider). Любая информация, управляемая контент-провайдером адресуется посредством *URI*:
content://authority/path/id

где:

- content://* - стандартный требуемый префикс;
- authority* - имя провайдера, рекомендуется использовать полное квалификационное имя пакета для избежания конфликта имен;
- path* - виртуальная папка внутри провайдера, которая определяет вид запрашиваемых данных;
- id* - первичный ключ отдельной запрошенной записи, для запроса всех записей определенного типа этот параметр не указывается.

Контент-провайдеры поддерживают стандартный *синтаксис* запросов для чтения, изменения, вставки и удаления данных.

Подробнее работу с SQLite базами данных в приложениях под *Android* рассмотрим в первой части лабораторной работы в этой теме.

17.2 Анимация

Android предоставляет мощные *API* для анимации элементов пользовательского интерфейса и построения *2D* и *3D* изображений.



Платформа *Android* предоставляет две системы анимации: *анимация* свойств, появившаяся в *Android* 3.0, и *анимация* компонентов пользовательского интерфейса (наследников класса *View*). Рассмотрим подробнее обе эти системы.

Анимация свойств (Property Animation). Система анимации свойств позволяет определить анимацию для изменения любого свойства объекта, независимо от того изображается оно на экране или нет. Используя эту систему, можно задать следующие характеристики анимации:

- **Продолжительность** предполагает задание длительности временного промежутка выполнения анимации, по умолчанию это значение равно 300 мс.
- **Временная интерполяция** предполагает вычисление значения свойства в каждый момент времени, как функции от промежутка времени, прошедшего с начала анимации.
- **Количество повторов и поведение** определяет необходимость повторения анимации при достижении конца заданного временного промежутка, а также количество повторов в случае необходимости. Эта же характеристика позволяет задать возможность воспроизведения в обратном порядке, если эта возможность выбрана, то анимация прокручивается вперед-назад заданное число раз.
- **Группа анимаций** позволяет организовать анимации в некоторое

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 208 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 209 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

множество и задать режим исполнения: одновременно, последовательно непрерывно или с некоторыми задержками.

- **Частота обновления кадров** определяет, как часто будет происходить смена кадров анимации. По умолчанию обновление происходит каждые 10 мс, однако скорость, с которой приложение сможет обновлять кадры, в конечном итоге, зависит от загруженности системы.

Большая часть *API* системы анимации свойств находится в пакете **android.animation**. Также можно использовать блоки интерполяции, определенные в пакете **android.view.animation**.

Класс **Animator** предоставляет базовую структуру для создания анимации. Напрямую этот класс обычно не используется, так как обеспечивает минимальную функциональность, поэтому чаще всего используются классы-наследники, расширяющие возможности класса **Animator**. Рассмотрим основные классы, используемые для создания анимации свойств.

- **ValueAnimator** (потомок класса **Animator**). Этот класс является главным обработчиком распределения времени для анимации свойств, а также рассчитывает значения свойства, предназначенного для анимации. Он обеспечивает всю основную функциональность: рассчитывает значения анимации и содержит распределенные во времени детали каждой анимации; содержит информацию о необходи-

мости повторений анимации; содержит слушателей, получающих уведомления о событиях обновления; предоставляет возможность задавать пользовательские типы для вычисления. В процессе **анимации свойств** можно выделить две части: вычисление значения свойства, для которого определяется анимация, и присвоение полученного значения соответствующему полю объекта.

ValueAnimator не выполняет вторую часть, поэтому необходимо следить за обновлениями значений, вычисляемых в классе **ValueAnimator**, и изменять объекты, подверженные анимации. Наглядно рассмотренные части анимации с использованием класса **ValueAnimator** представлены на рис. 2

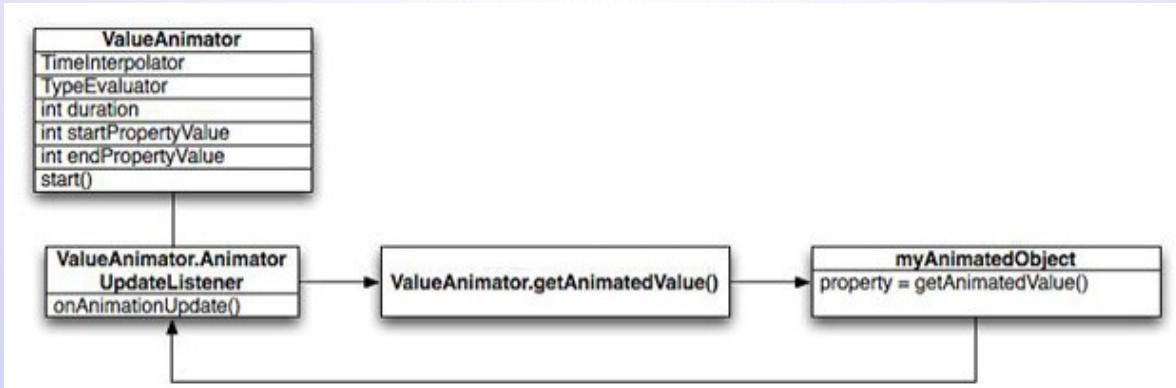


Рисунок 2. Процесс анимации свойств с использованием класса ValueAnimator



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 210 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



- **AnimatorSet** (потомок класса **Animator**). Предоставляет механизмы группировки анимаций, таким образом, что они выполняются некоторым образом относительно друг друга. Можно определять выполнение анимаций одновременно, последовательно и с временными задержками.

Классы-вычислители определяют как вычислять значения заданных свойств. Они получают: данные о *распределение времени*, предоставляемые классом **Animator**, начальное и конечное значения свойства, после чего на основе этих данных вычисляют значения свойства, для которого выполняется *анимация*. В системе **анимации свойств** существуют следующие вычислители:

- **IntEvaluator** для вычисления целочисленных значений свойств;
- **FloatEvaluator** для вычисления вещественных значений свойств;
- **ArgbEvaluator** для вычисления значений цвета в шестнадцатеричном представлении;
- **TypeEvaluator** - интерфейс, позволяющий создавать собственных вычислителей.

Интерполяторы определяют с помощью каких функций от времени, вычисляются значения свойств, для которых задается *анимация*. Интерполяторы определены в пакете *android.view.animation*. Если ни один

Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 211 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



из существующих интерполяторов не подходит, можно создать собственный, реализовав *интерфейс TimeInterpolator*.

Подробнее с системой анимации свойств можно познакомиться по ссылке: [здесь](#).

Анимация компонентов пользовательского интерфейса. Эта система может быть использована для реализации анимации преобразований над наследниками класса *View*. Для расчета анимации преобразований используется следующая *информация*: начальная точка, конечная точка, размер, поворот и другие общие аспекты анимации. *Анимация* преобразований может выполнять серии простых изменений содержимого экземпляра класса *View*. Например, для текстового поля можно перемещать, вращать, растягивать, сжимать текст, если определено фоновое изображение, оно должно изменяться вместе с текстом. Пакет **android.view.animation** предоставляет все классы, необходимые для реализации анимации преобразований.

Для задания последовательности инструкций анимации преобразований можно использовать или *XML*, или *Android* код. Более предпочтительным является *определение* анимации в *XML* файлах, расположаться эти файлы должны в папке **res/anim/** проекта. *XML* файл должен иметь единственный *корневой элемент*, это может быть любой из отдельных элементов: **<alpha>**, **<scale>**, **<translate>**, **<rotate>**, **интерполятор**, или же элемент **<set>**, который содержит группы этих элементов, в том числе может содержать другие элементы **<set>**. *По*

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 212 из 469

Назад

На весь экран

Закрыть

умолчанию инструкции анимации выполняются одновременно, чтобы задать последовательное *исполнение* необходимо определить *атрибут startOffset*.

Подробнее с системой анимации преобразований можно познакомиться по ссылке: [здесь](#).

Дополнительно к рассмотренным системам анимации может использоваться, кадровая *анимация*, которая реализуется быстрой сменой кадров, каждый *кадр* является графическим ресурсом и располагается в папке **res/drawable/** проекта.

Подробнее с кадровой анимацией можно познакомиться по ссылке: [здесь](#).

17.3 2D и 3D графика

При разработке приложения важно четко понимать требования к графике в этом приложении. Для разных графических задач необходимы разные техники их решения. Далее в лекции рассмотрим несколько способов изображения графических объектов в *Android*.

Холсты и графические объекты. Платформа *Android* предоставляет API для изображения 2D графики, который позволяет изображать на холсте свои графические объекты или изменять существующие. Для отображения 2D графики существуют два пути:

1. Изобразить графику или анимацию в элементе пользовательского



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 213 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 214 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

интерфейса. В этом случае графика управляется процессом отображения иерархии элементов интерфейса. Подходит, когда необходимо отобразить простую графику, не требующую динамических изменений.

2. Изображать графику напрямую на холсте (класс `Canvas`). В этом случае необходимо позаботиться о вызове метода `onDraw()`, передавая его в класс `Canvas`, или же о вызове одного из `draw...()` методов класса `Canvas` (например, `drawPicture()`). Действуя таким образом, можно управлять анимацией. Этот путь подходит, когда необходимо постоянно перерисовывать окно приложения, например, для видео игр.

Аппаратное ускорение. Начиная с *Android* 3.0 (*API* уровень 11), конвейер изображения 2D графики в *Android* поддерживает аппаратное ускорение. Это означает, что все *операции* рисования на холсте исполняются с использованием *GPU*. В связи с увеличением требований к ресурсам *приложение* будет потреблять больше *RAM*. Аппаратное ускорение доступно *по умолчанию*, если целевой уровень *API* больше или равен 14, но может быть включено явно. Если в приложении используются только стандартные представления и *графика*, включение аппаратного ускорения не должно привести к каким-либо нежелательным графическим эффектам. Однако из-за того, что аппаратное ускорение поддерживается не всеми операциями 2D графики, его включение может нарушать неко-



торые пользовательские изображения или вызовы рисования. Проблемы обычно проявляются в невидимости некоторых элементов, появлении исключений или неверно изображенных пикселях. Чтобы исправить это, *Android* позволяет включать или выключать аппаратное ускорение на разных уровнях: уровень приложения, уровень активности, уровень окна, уровень элемента интерфейса.

OpenGL. *Android* поддерживает высокопроизводительную 2D и 3D графику с использованием открытой графической библиотеки OpenGL, точнее OpenGL ES API. Библиотека OpenGL является кросс-платформенным API, который определяет стандартный программный интерфейс для аппаратного обеспечения, занимающегося обработкой 3D графики. OpenGL ES является разновидностью OpenGL, предназначеннной для встроенных устройств. *Android* поддерживает несколько версий OpenGL ES API:

- OpenGL ES 1.0 и 1.1 поддерживается *Android* 1.0 и выше;
- OpenGL ES 2.0 поддерживается *Android* 2.2 (API уровень 8) и выше;
- OpenGL ES 3.0 поддерживается *Android* 4.3 (API уровень 18) и выше.

Поддержка OpenGL ES 3.0 на реальном устройстве требует реализации графического конвейера, предоставленной производителем. Поэтому устройство с *Android* 4.3 и выше может не поддерживать OpenGL ES 3.0.

Подробнее с графикой в *Android* можно познакомиться по ссылкам:
1; 2; 3.

17.4 Основные принципы разработки игровых приложений для смартфонов

Разработка игр дело обычно благодарное т. к. в игры люди играли, играют и будут играть. Даже если результат работы не принесет особой прибыли, в любом случае он способен доставить радость детям и друзьям, да и о себе забывать не стоит. При этом сам процесс разработки способен серьезно повысить уровень мастерства особенно начинающего разработчика.

Если возникло острое желание создать именно игровое *приложение*, необходимо иметь в виду некоторые особенности: практически любая *игра* предполагает наличие сюжета, игры обычно отличаются эффектным графическим оформлением и обеспечивают определенный игровой процесс (геймплей). И эти моменты стоит хорошо продумать прежде, чем начинать *программирование*.

Сюжет игры состоит из последовательности событий. Необходимость сюжета больше всего зависит от жанра игры: в некоторых жанрах можно обойтись совсем без сюжета. Не стоит, как недооценивать, так и переоценивать важность сюжета, т. к. он является лишь одной из составляющих успеха игры. Решение о том нужен или ненужен сюжет в игре,



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 216 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 217 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

если нужен, то в какой мере и каким образом он будет выстраиваться, необходимо принимать взвешенно и до начала разработки.

При выборе способа графического оформления игры стоит иметь в виду, что использование *3D* графики серьезно усложнит процесс разработки, даже несложная *3D игра* отнимет очень много времени. В большинстве игр для мобильных устройств достаточно 2D графики, особенно в случае начинающего разработчика или команды таковых.

Следует учитывать ограниченные возможности мобильных устройств: сравнительно невысокая вычислительная *мощность*; ограниченный объем оперативной и дисковой памяти; небольшой размер и невысокое разрешение экрана; возможные проблемы, связанные с организацией передачи данных; ограниченный заряд аккумуляторных батарей.



ГЛАВА 3

Лабораторный практикум

§18. Лабораторная работа №1

Цель лабораторной работы:

Установка и настройка среды программирования ADT Bundle

Задачи лабораторной работы:

- Установить и настроить среду программирования ADT
- Создать первое приложение **Hello, world!**
- Научиться запускать приложение на эмуляторе мобильного устройства

18.1 Введение

Лабораторная работа посвящена описанию работы в среде *ADT Bundle (Android IDE)*. В работе рассказывается о программном обеспечении, которое необходимо скачать с официальных сайтов, установить и настроить, создании и запуске простейшего приложения на эмуляторе и мобильном устройстве. В связи с тем, что *отладка* на устройствах со-пряжена с рядом трудностей, приводится подробная *инструкция* по настройке устройств и операционной системы *Windows* (версия не ниже 7).

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 218 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

18.2 Установка среды

Большинство приложений для OS *Android* написано на *Java*. Одной из самых популярных сред разработки является *Eclipse* (для неё также необходим *JDK*) с установленным плагином *ADT* и *Android SDK*. Раньше приходилось ставить все компоненты отдельно. Сейчас появилась версия среды *Eclipse* с уже настроенными дополнениями - *ADT Bundle*. Здесь есть *минимум* инструментов, необходимый для разработки приложений. С этой версией мы и будем работать. Однако в ней есть далеко не всё, поэтому, если при разработке какого-либо проекта вам потребуются инструменты, не входящие в *ADT Bundle*, вы можете скачать их с сайта разработчиков и дополнить свою среду.

Скачать среду можно с сайта для разработчиков *Android* (<http://developer.android.com/sdk/index.html>).



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 219 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

The screenshot shows the 'Get the Android SDK' section of the developer.android.com website. On the left, there's a sidebar with 'Developer Tools' expanded, showing 'Download' selected. Under 'Download', there are links for 'Setting Up the ADT Bundle', 'Setting Up an Existing IDE', 'Android Studio', 'Exploring the SDK', and 'Download the NDK'. Below these are sections for 'Workflow', 'Support Library', 'Tools Help', 'Revisions', 'Samples', and 'ADK'. A callout box highlights the 'Android Studio Early Access Preview' section, which mentions the availability of a new development environment based on IntelliJ IDEA. It also provides links for 'USE AN EXISTING IDE', 'SYSTEM REQUIREMENTS', and 'DOWNLOAD FOR OTHER PLATFORMS'. At the bottom of the page, there are links for 'Except as noted, this content is licensed under Creative Commons Attribution 2.5. For details and restrictions, see the Content License.', 'About Android', 'Legal', and 'Support'. A large blue button at the bottom right says 'Download the SDK ADT Bundle for Windows'.

Рисунок 1.Сайт разработчика

Для того, чтобы скачать среду необходимо принять условия лицензионного соглашения и выбрать вашу версию *Windows* (32-bit или 64-bit).



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 220 из 469

Назад

На весь экран

Закрыть

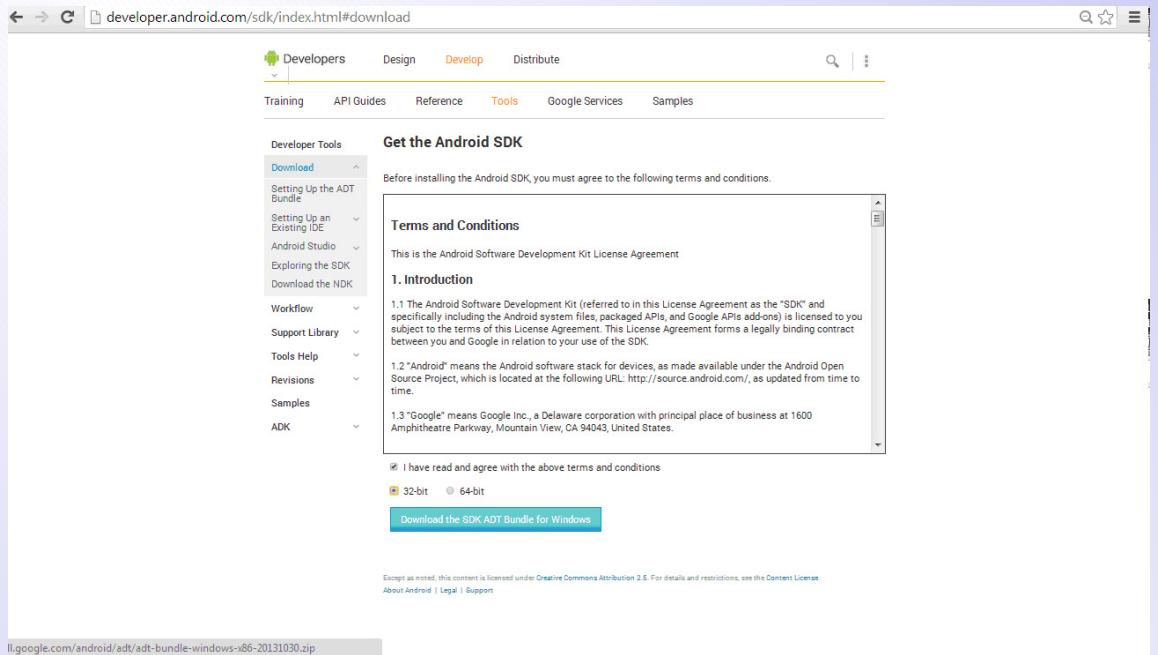


Рисунок 2. Скачивание среды

После скачивания распакуйте *архив* в ту папку, где собираетесь работать (среда не требует специальной установки). После распаковки зайдите в папку и запустите Eclipse. Здесь возможна небольшая проблема: если у вас не установлен *JDK*, среда не запустится и потребует указать *путь* к папке с *JDK* или установить его. Скачать *JDK* можно с сайта *Oracle* ()

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 221 из 469

Назад

На весь экран

Закрыть

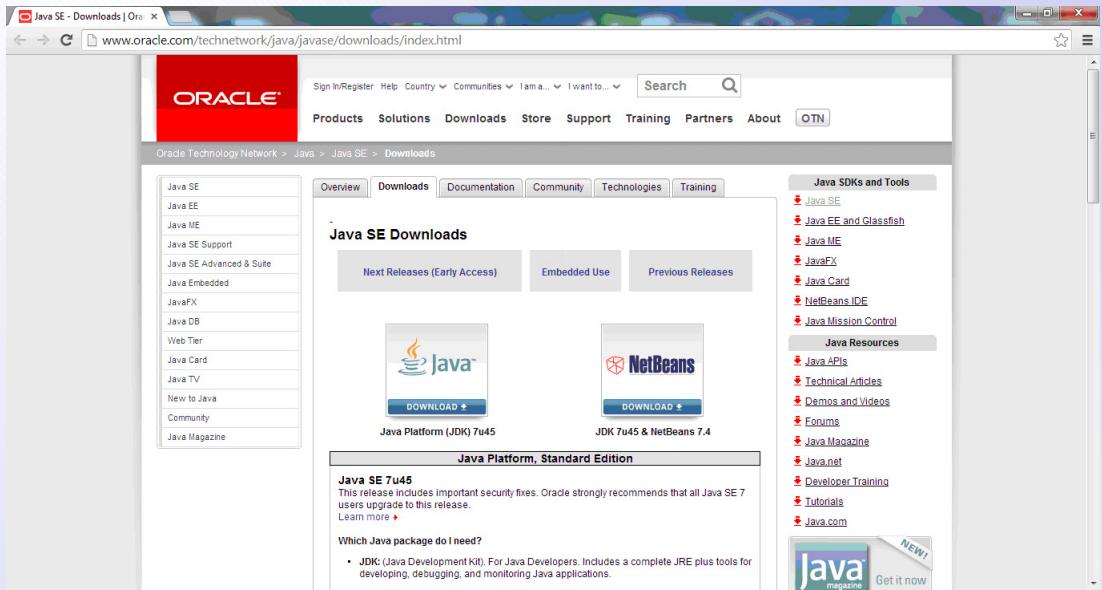


Рисунок 3.Сайт компании Oracle

Чтобы скачать *JDK* нужно сначала принять условия лицензионного соглашения, а затем выбрать нужную версию.

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 222 из 469

Назад

На весь экран

Закрыть





Рисунок 4. Скачивание JDK

После скачивания запустите *setup-файл* и установите *JDK*.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 223 из 469

Назад

На весь экран

Закрыть



Рисунок 5. Установка JDK

После установки *JDK* среда должна запуститься.

Далее вам необходимо выбрать (или создать новое) рабочее *пространство*, т.е. *место*, где будут находиться ваши проекты. Если поставить галочку, то это рабочее *пространство* будет выбираться по умолчанию, а противном случае это окно будет появляться при каждом запуске Eclipse.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 224 из 469

Назад

На весь экран

Закрыть

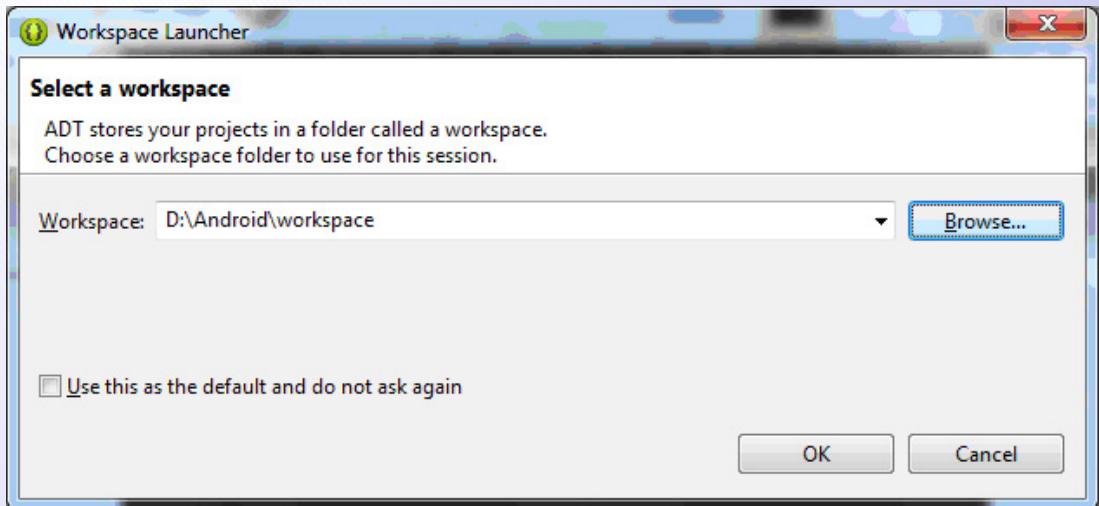


Рисунок 6. Выбор рабочего пространства

Затем появляется окно, в котором разработчики предлагают отправлять статистику для дальнейшего улучшения *SDK*. Вы можете согласиться или отказатьься.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶ ▶▶

Страница 225 из 469

Назад

На весь экран

Закрыть

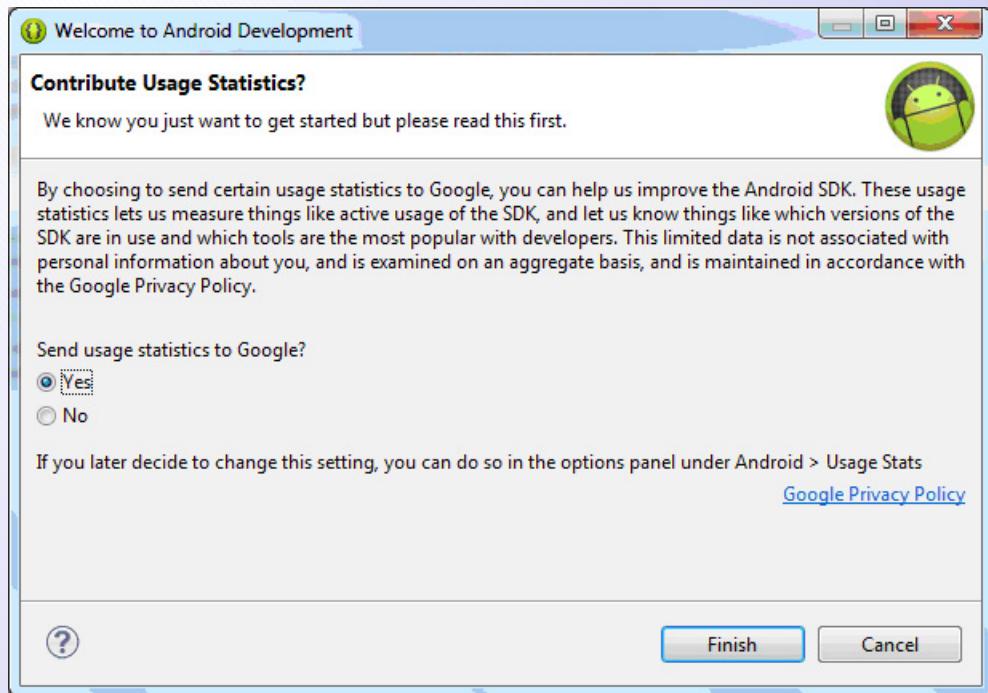


Рисунок 7.Отправка статистики

Обратите внимание на значок *Android SDK Manager*, находящийся на панели инструментов (его также можно найти в меню Window). С его помощью вы сможете добавлять в свою среду новые инструменты.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 226 из 469

Назад

На весь экран

Закрыть

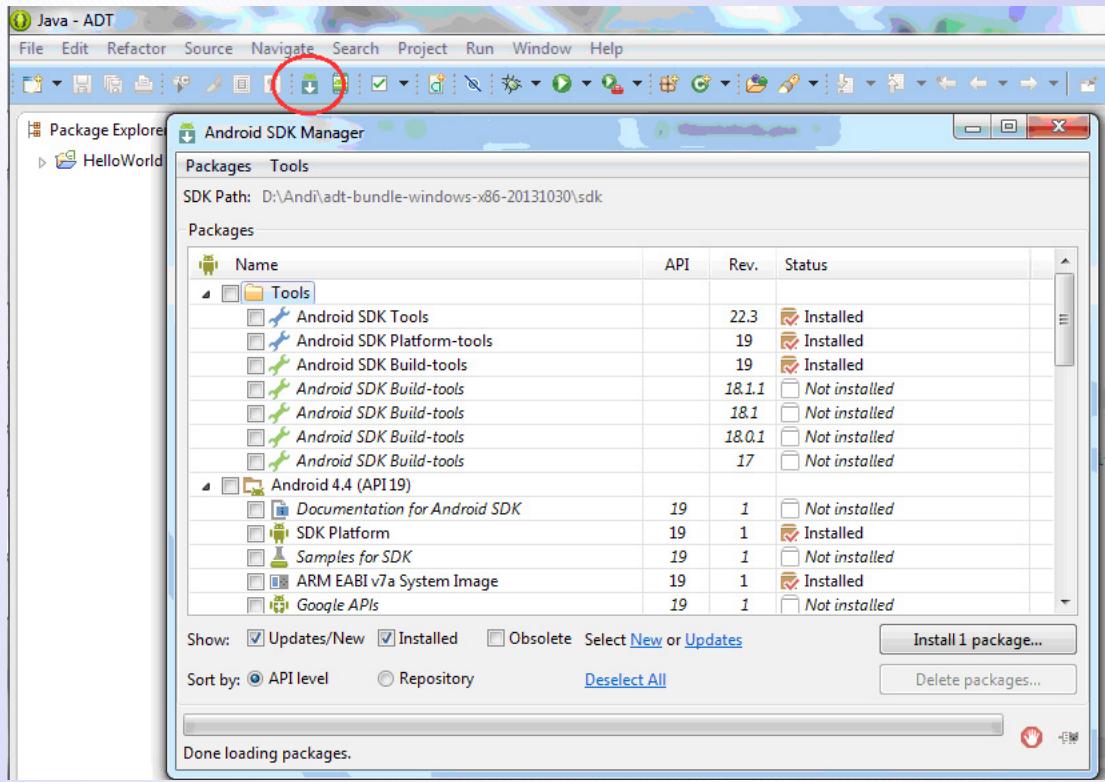


Рисунок 8. Android SDK Manager

18.3 Создание проекта

Итак, наконец, мы подошли к самому главному - созданию проекта.

Чтобы создать проект, зайдите в меню

File->New->Android Application Project.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 227 из 469

Назад

На весь экран

Закрыть

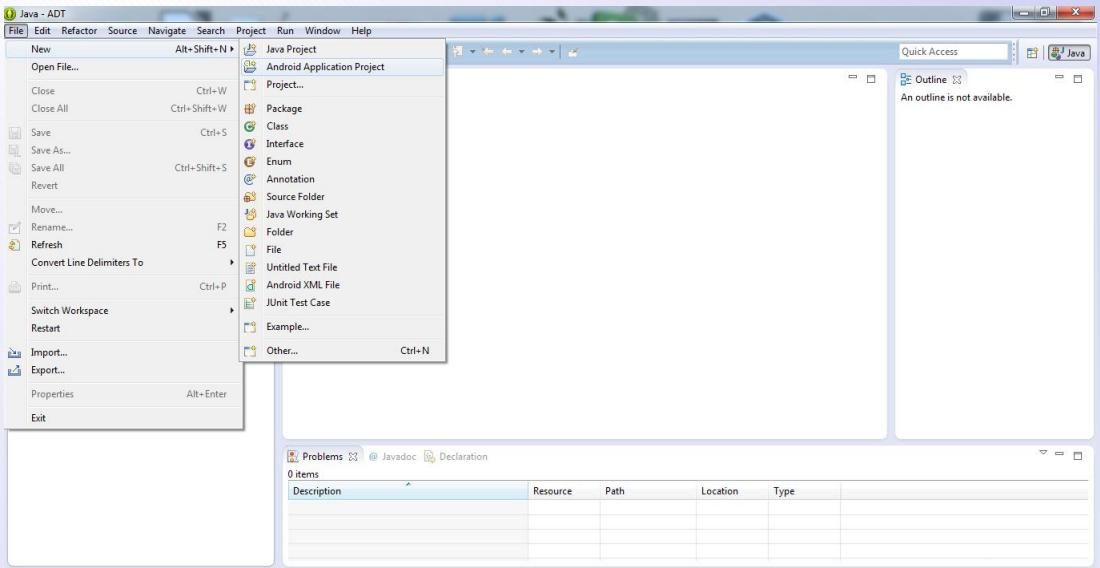


Рисунок 9. Создание проекта

В появившемся окне обязательно нужно прописать имя приложения, имя проекта, а также *имя пакета (package)*. Лучше не оставлять его именем *example*, т.к. пакет с таким именем нельзя разместить в Google Play. Конечно, учебные приложения туда не загружают, однако, следует иметь это в виду на будущее.

Minimum Required SDK - минимальная версия *Android*, которую будет поддерживать *приложение*. Чаще всего по умолчанию указывается версия 2.2, чтобы поддерживать как можно больше устройств. Если



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 228 из 469

Назад

На весь экран

Закрыть



определенная *функция* вашего приложения работает только на более новых версиях *Android*, и это не является критическим для основного набора функций приложения, вы можете включить ее в качестве опции на версиях, которые поддерживают его.

Target SDK - версия *Android*, под которую будет написано ваше *приложение*; определяет максимальную версию *Android*, на которой вы тестировали *приложение*. Это нужно для режимов совместимости.

Compile With определяет, возможности какой версии *Android* будет использовать *приложение*.

Оставьте пока установки, заданные по умолчанию в качестве значений для этого проекта.

Кафедра ПМиИ

[Начало](#)

[Содержание](#)



Страница 229 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

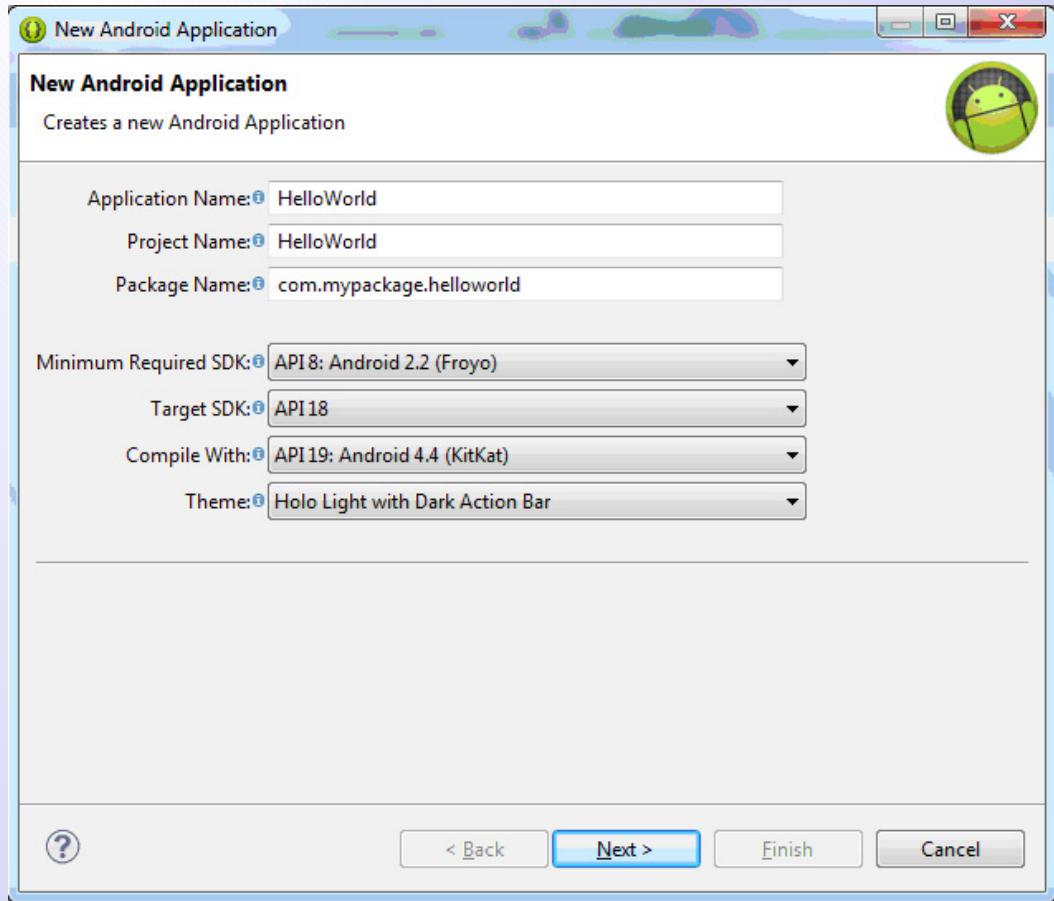


Рисунок 10.Наименование проекта

Следующее окно можно пропустить без изменений.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 230 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 231 из 469

Назад

На весь экран

Закрыть

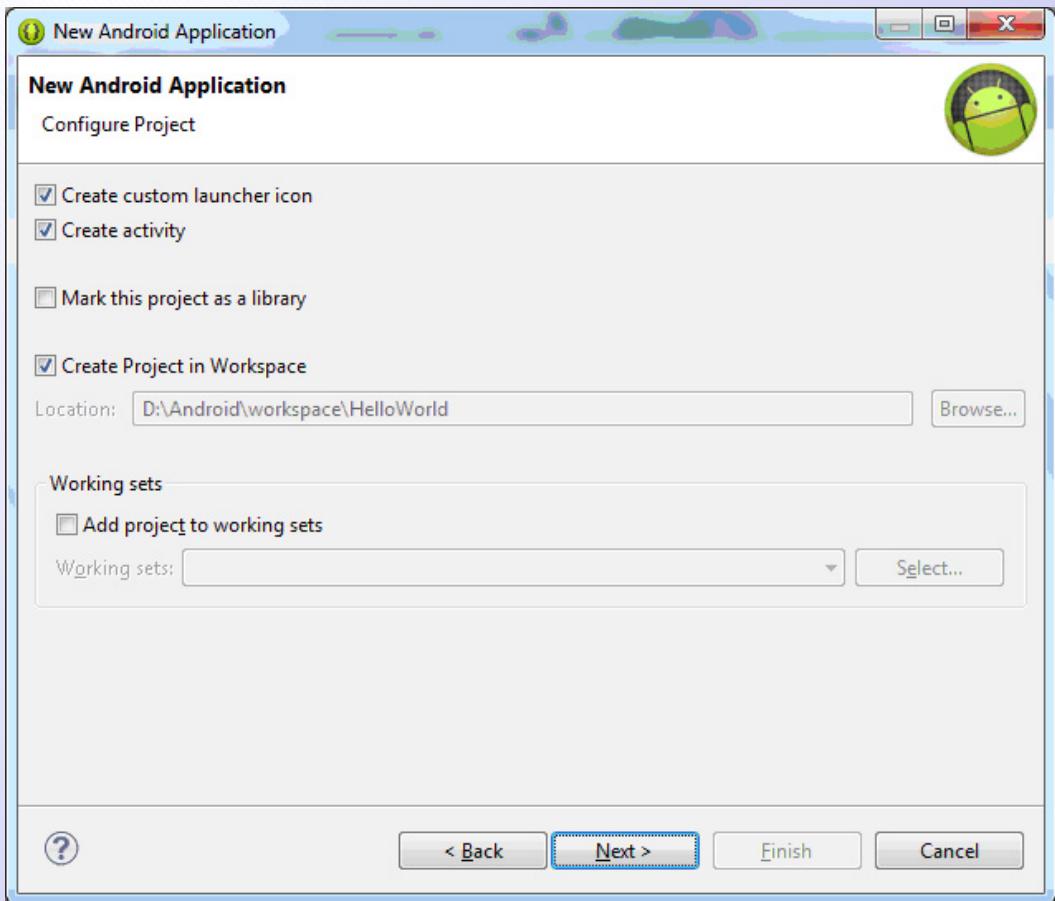


Рисунок 11. Конфигурация проекта

Create custom launcher icon - создать значок приложения.

Create activity - создать *Activity* (*активность, деятельность*).

Mark this project as library - создать проект, как библиотеку. Сейчас в этом нет необходимости, наше *приложение* в других проектах использовать не будет.

Create Project in Workspace - создать проект в папке *Workspace*.

В этой папке будут храниться все наши проекты.

Следующий этап - создание иконки. Можете оставить стандартную или создать свою собственную. В нашем примере изменена цветовая гамма, форма, а также выбрана фигурка из клипарта.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#)

[▶](#)

[◀◀](#)

[▶▶](#)

Страница 232 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

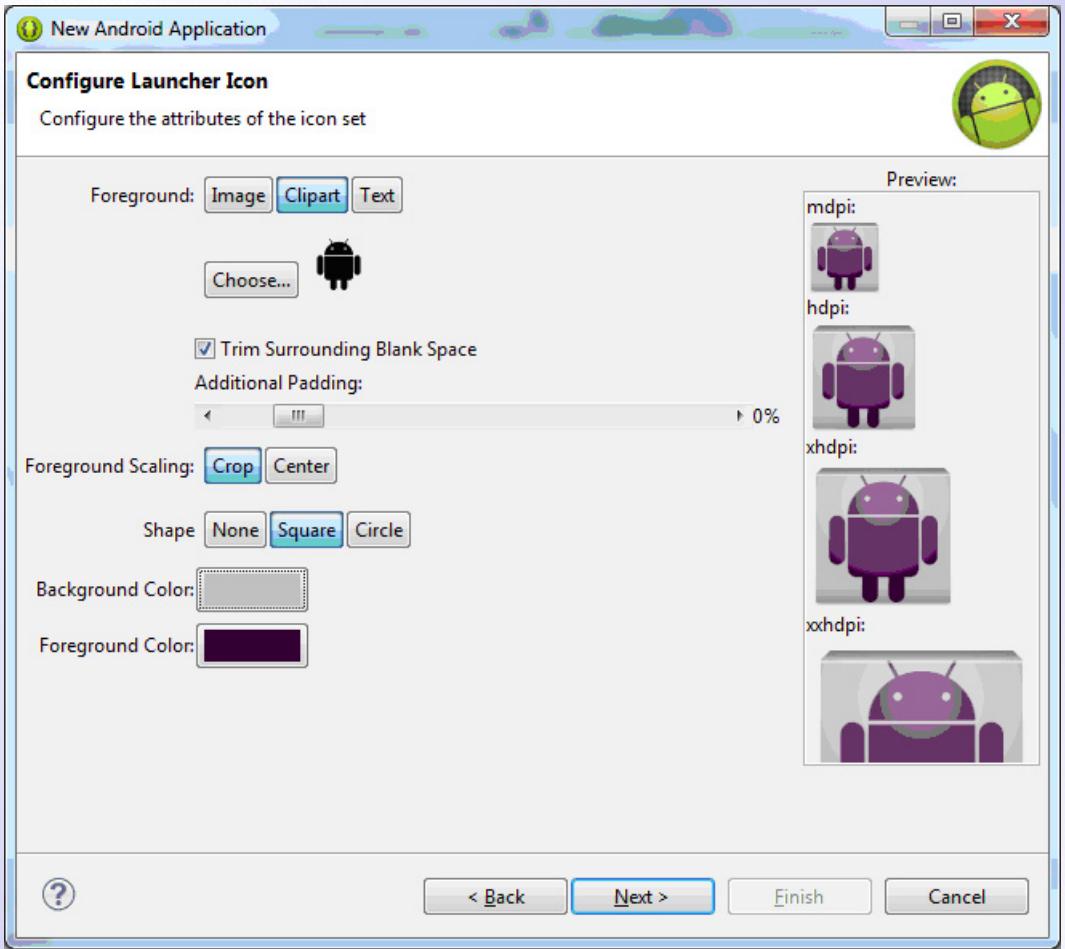


Рисунок 12. Создание иконки приложения



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 233 из 469

Назад

На весь экран

Закрыть

Большинство приложений на *Android* имеют свой экран (форму, окно), которое называется активностью или деятельностью (*Activity*).

Следующее два окна создают пустую *активность*. В первом ничего пока менять не нужно. Во втором вы можете переименовать свою *активность* (в приложении их может быть несколько).



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

◀

▶

◀◀

▶▶

Страница 234 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

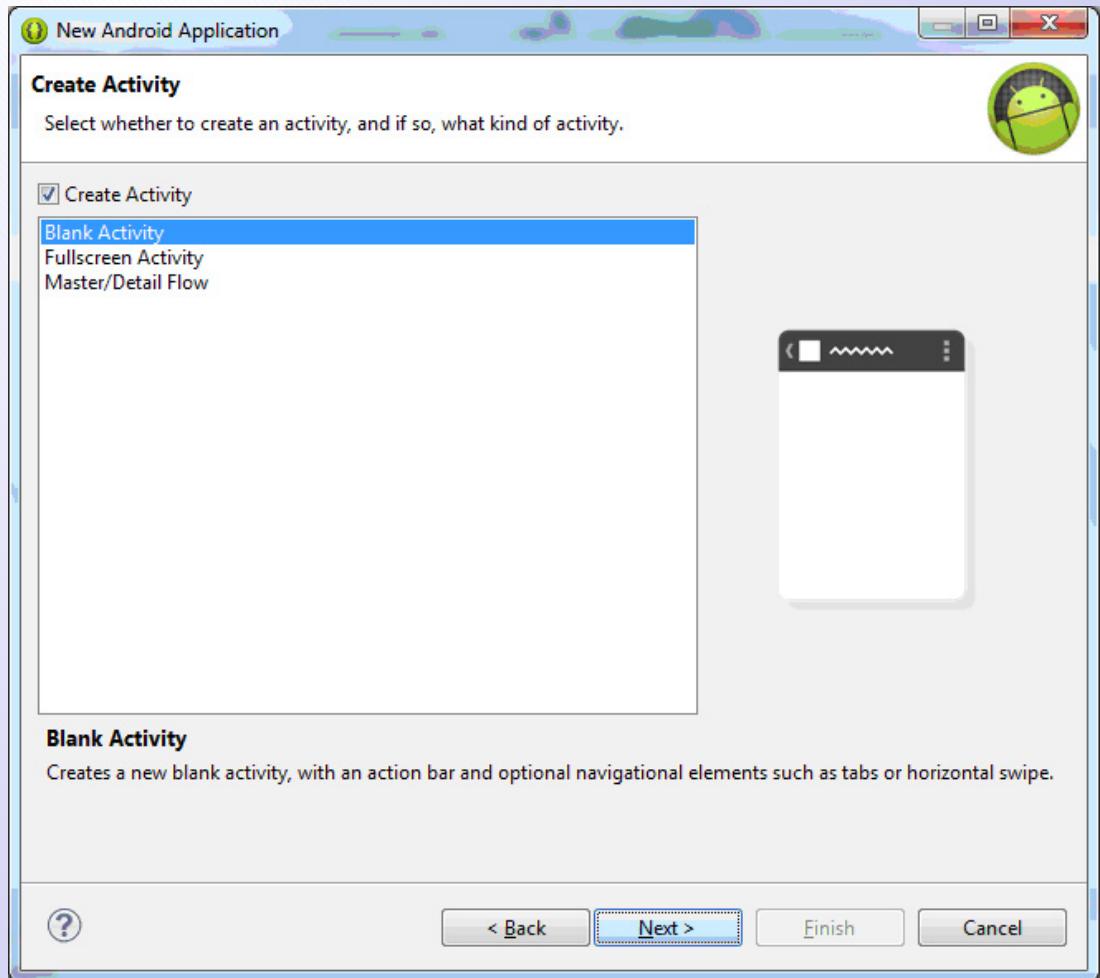


Рисунок 13.Создание активности



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶ ▶▶

Страница 235 из 469

Назад

На весь экран

Закрыть

Blank Activity - шаблон, предназначенный для мобильных телефонов.

Fullscreen Activity - шаблон, позволяющий растянуть приложение на весь экран (без навигационной панели и статус-бара).

Master/Detail Flow - шаблон, предназначенный для планшетных компьютеров.



Кафедра ПМиИ

Начало

Содержание



Страница 236 из 469

Назад

На весь экран

Закрыть

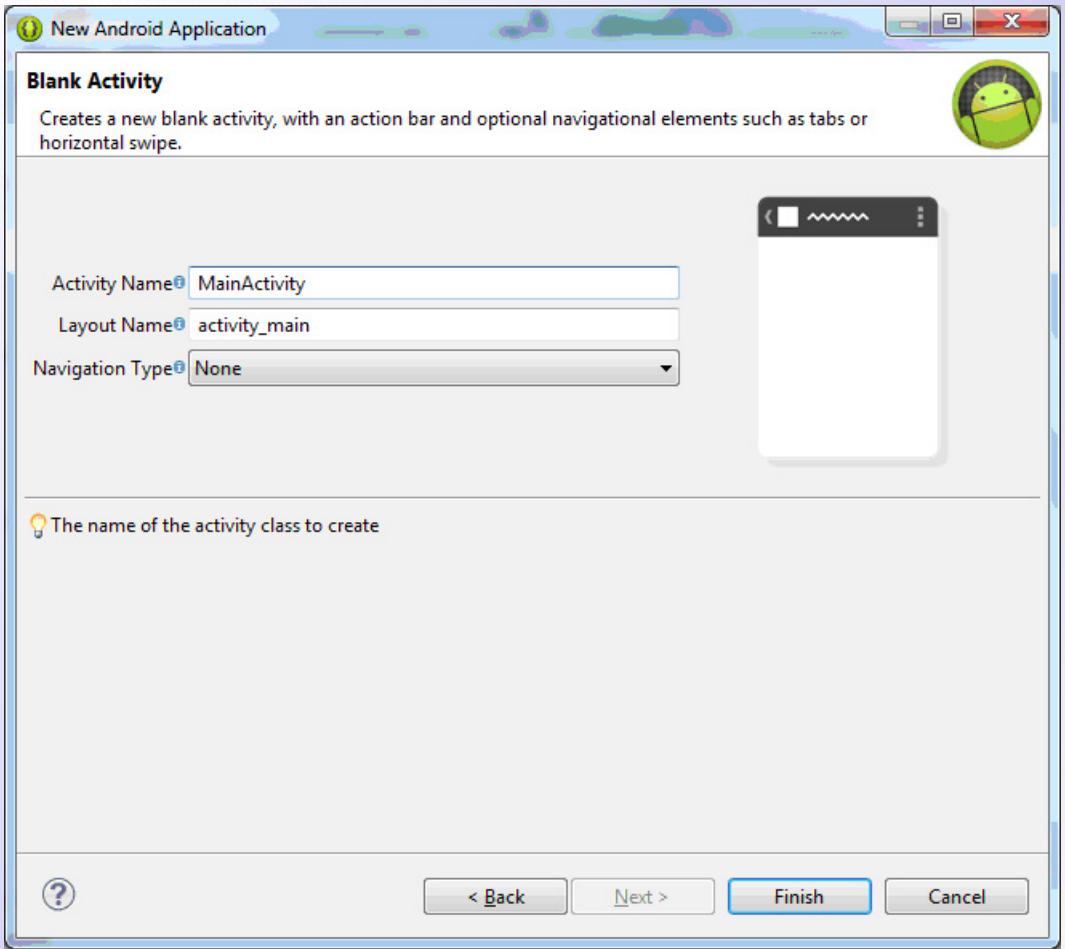


Рисунок 14.Переименование активности



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 237 из 469

Назад

На весь экран

Закрыть

Итак, вы создали свой первый проект. Конечно, это всего лишь встроенное в среду *приложение для проверки корректной установки инструментария*, однако множество приложений создаются именно из него.

Посмотрим на его структуру. Она показана в области слева.

В первую очередь нас интересует *файл* активности. Он находится в папке **src** в вашем пакете. Он имеет расширение *.java*.

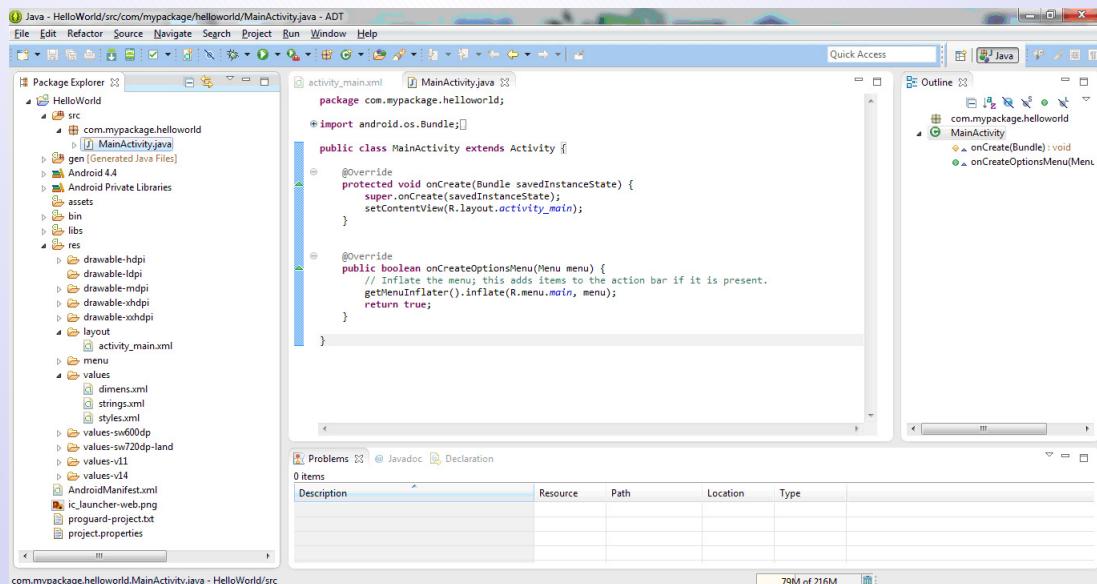


Рисунок 15. Активность



Кафедра
ПМиИ

Начало

Содержание



Страница 238 из 469

[Назад](#)

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание



Страница 239 из 469

Назад

На весь экран

Закрыть

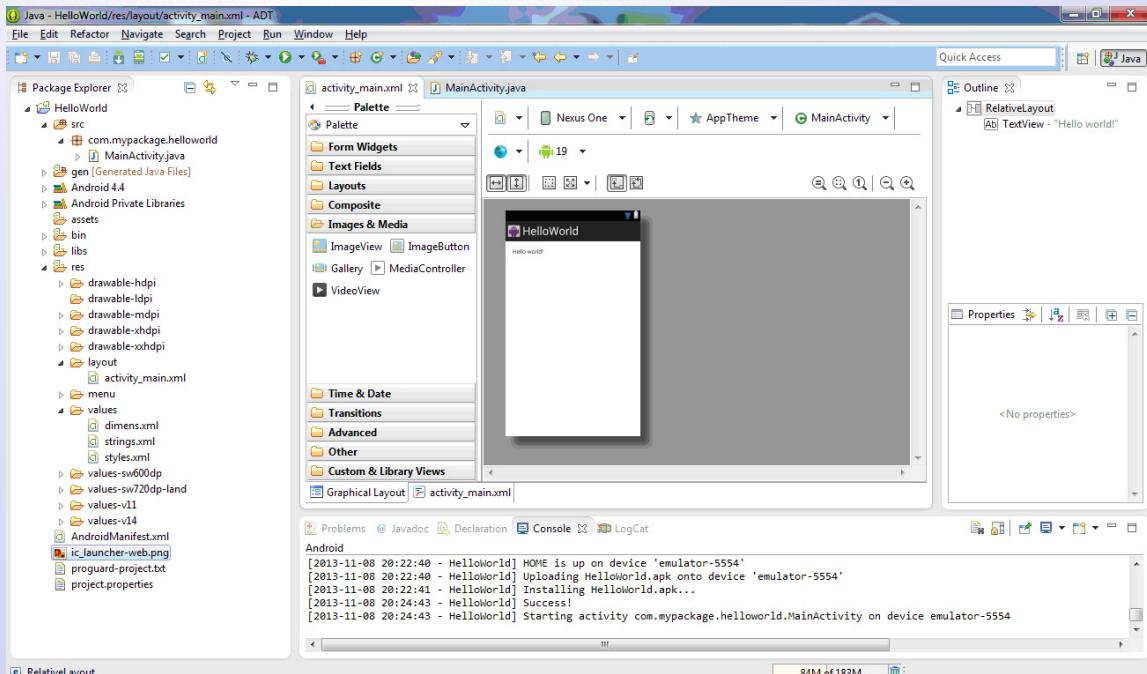


Рисунок 16.Xml-файл. Графический редактор



Кафедра
ПМиИ

Начало

Содержание



Страница 240 из 469

Назад

На весь экран

Закрыть

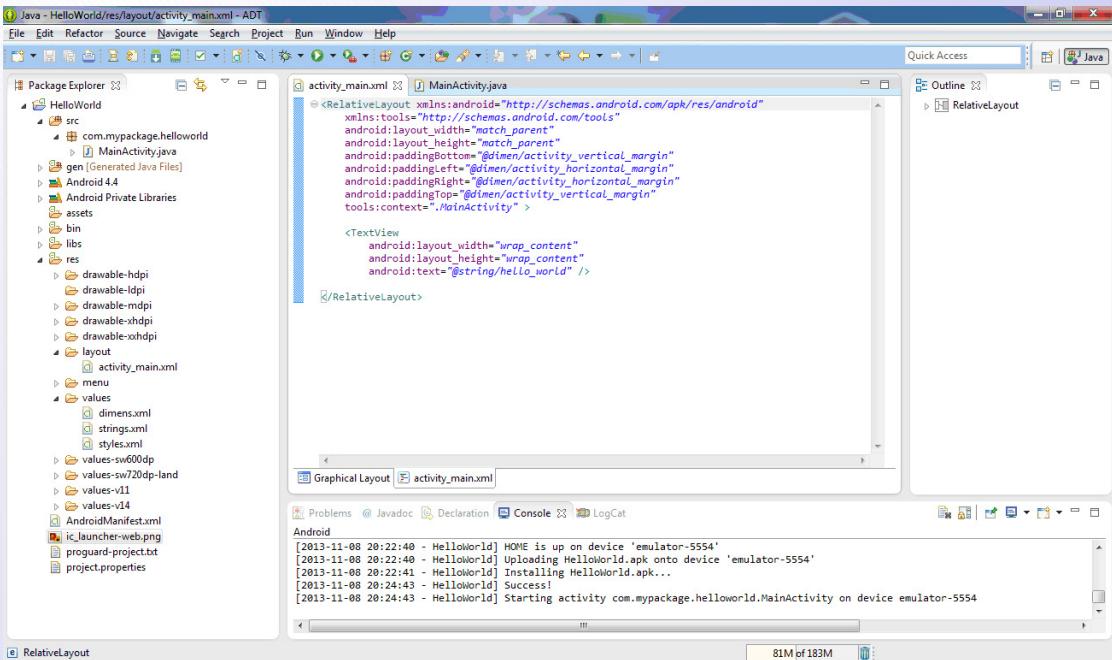


Рисунок 17.Xml-файл

18.4 Запуск проекта на эмуляторе устройства

В первую очередь нужно создать эмулятор устройства. Это можно сделать, нажав на кнопку на панели инструментов, изображающую смартфон. Если кнопки нет на панели, ее можно найти в меню **Window**.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 241 из 469

Назад

На весь экран

Закрыть

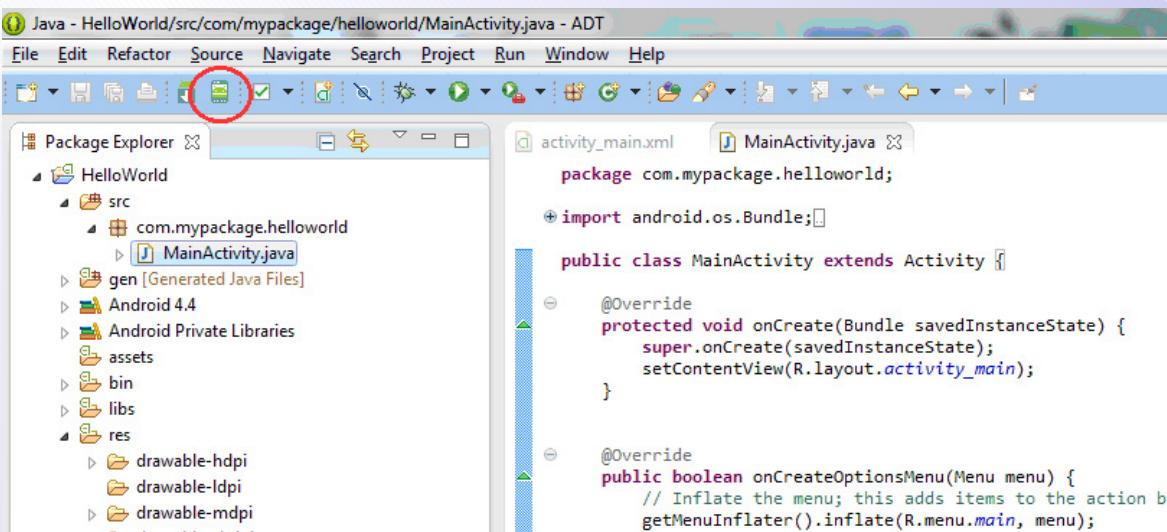


Рисунок 18. Запуск проекта

Откроется *Android Virtual Device Manager*. Пока в нем нет ни одного виртуального устройства.



Кафедра ПМиИ

Начало

Содержание



Страница 242 из 469

Назад

На весь экран

Закрыть

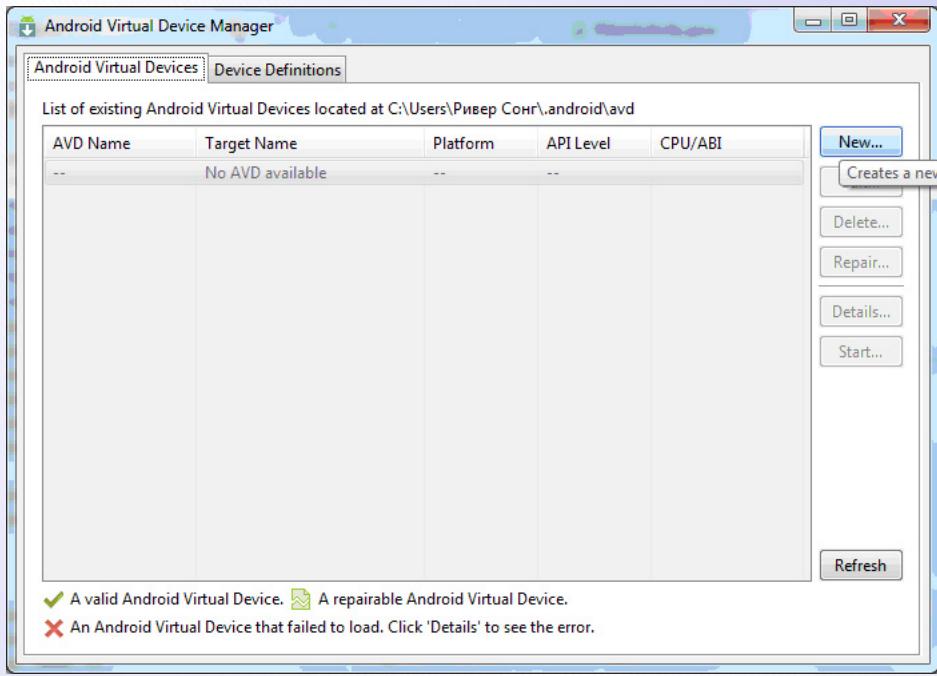


Рисунок 19. Android Virtual Device Manager

Чтобы создать *виртуальное устройство*, нажмите кнопку **New**. Появится окно создания. Вам нужно назвать устройство и выбрать обязательные характеристики: **Device** - модель вашего устройства, и **Target** - версия *Android*. Также можно изменять дополнительные параметры: размер sd-карты, встроенной памяти и т.п.



Кафедра ПМиИ

Начало

Содержание



Страница 243 из 469

Назад

На весь экран

Закрыть

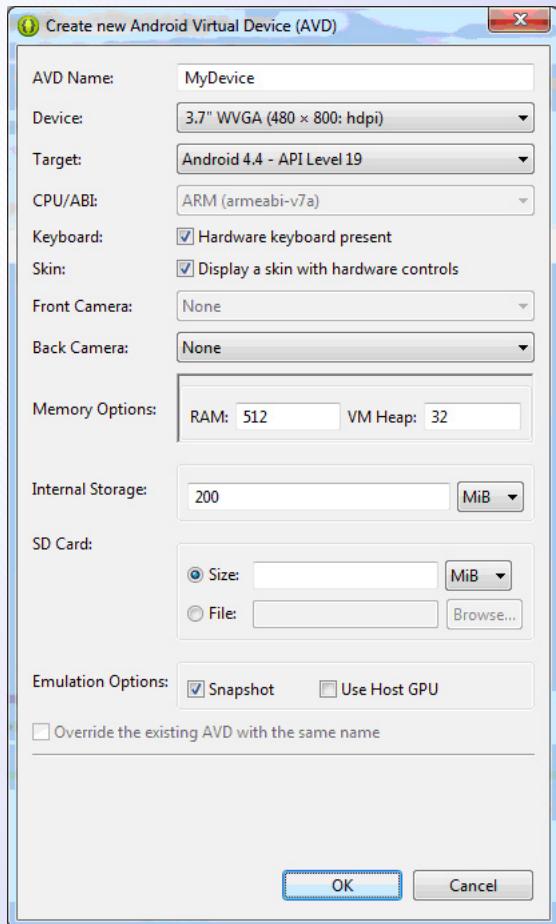


Рисунок 20. Создание AVD



Теперь можно запускать *приложение*. Для этого нужно нажать на кнопку **Run** (белый треугольник в зеленом кружочке) на панели инструментов. Проблемы с запуском можно отследить в консоли.

Если *приложение* не запускается, попробуйте нажать на черный треугольник справа от кнопки **Run**, выбрать **Run Configurations**, затем во вкладке **Target** выбрать созданное устройство и запустить проект снова.

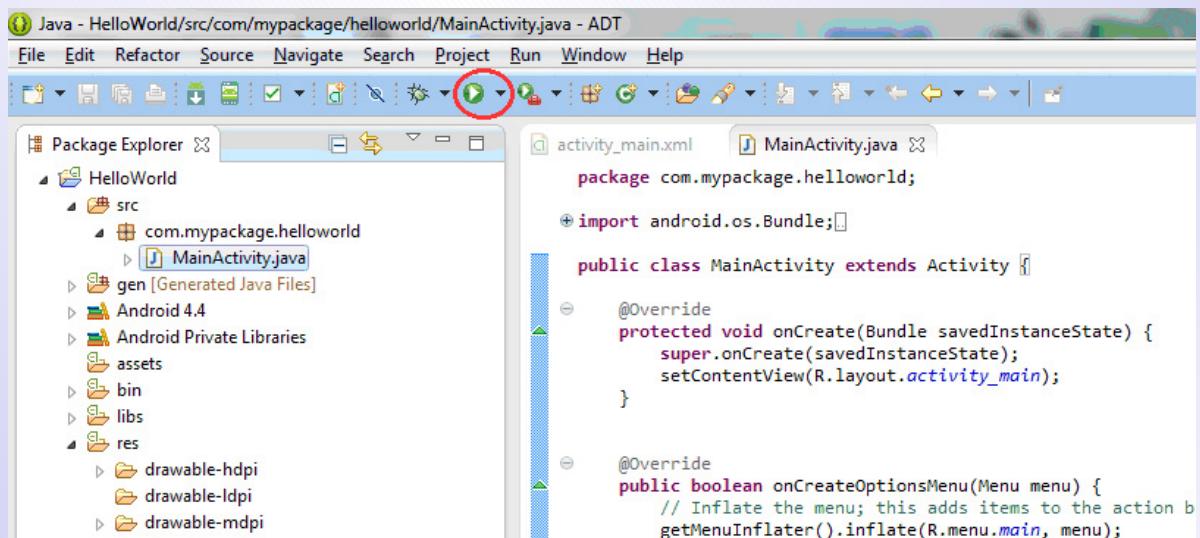


Рисунок 21.Запуск приложения

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 244 из 469

Назад

На весь экран

Закрыть

Если все сделано правильно, должен запуститься эмулятор. Время

запуска зависит от размера оперативной памяти на вашем компьютере. В дальнейшем эмулятор можно не закрывать, приложения будут запускаться в работающем.



Рисунок 22.Запуск эмулятора



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 245 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 246 из 469

Назад

На весь экран

Закрыть



Рисунок 23. Запущенный эмулятор

Если ваше приложение сразу не запустилось, его можно найти в меню приложений устройства.



Кафедра
ПМиИ

Начало

Содержание



Страница 247 из 469

Назад

На весь экран

Закрыть

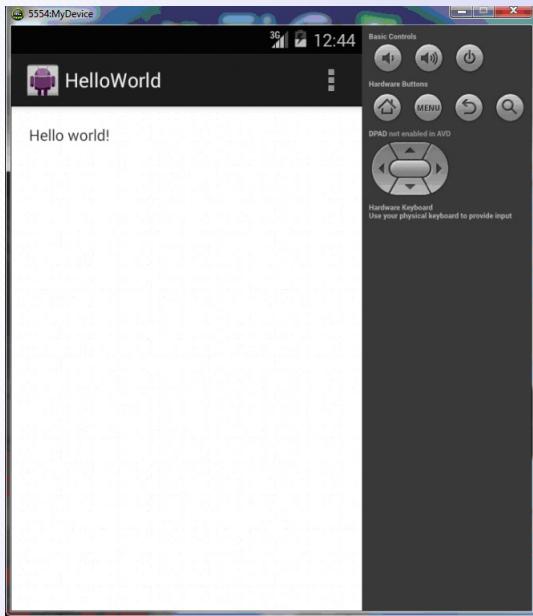


Рисунок 24.Приложение Hello, world!

18.5 Запуск проекта на устройстве

Прежде чем запустить проект на реальном устройстве, необходимо:

- Настроить устройство
- Настроить компьютер
- Настроить среду



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 248 из 469

Назад

На весь экран

Закрыть

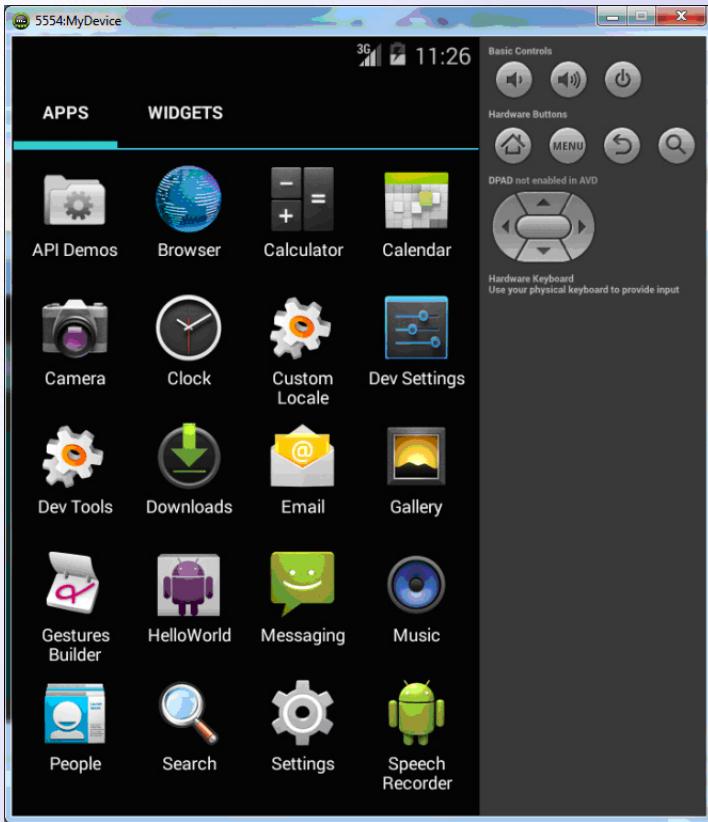


Рисунок 25.Меню приложений

Настройка устройства

Настройка устройства состоит в следующем:

1. Включить режим отладки по USB.
2. Для запуска файлов с расширением *.apk, полученных не из магазина приложений Google Play, необходимо разрешить установку приложений из альтернативных источников.

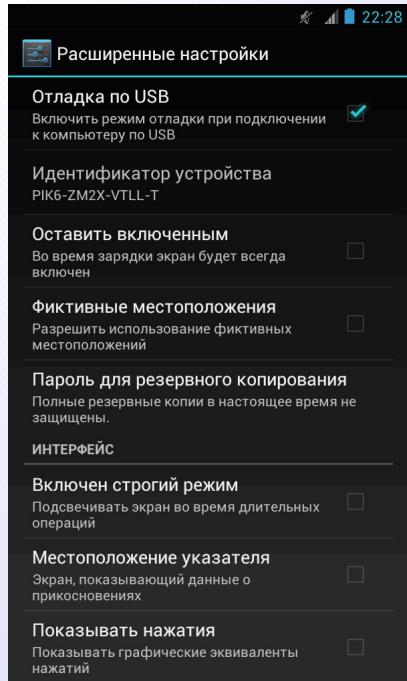


Рисунок 26. Настройка устройства



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 249 из 469

Назад

На весь экран

Закрыть



Настройка компьютера

Чтобы настроить компьютер необходимо выполнить следующие действия:

1. Зайти в Диспетчер устройств (**Пуск->Панель управления->Система и безопасность->Система->Диспетчер устройств**) и найти ваше устройство

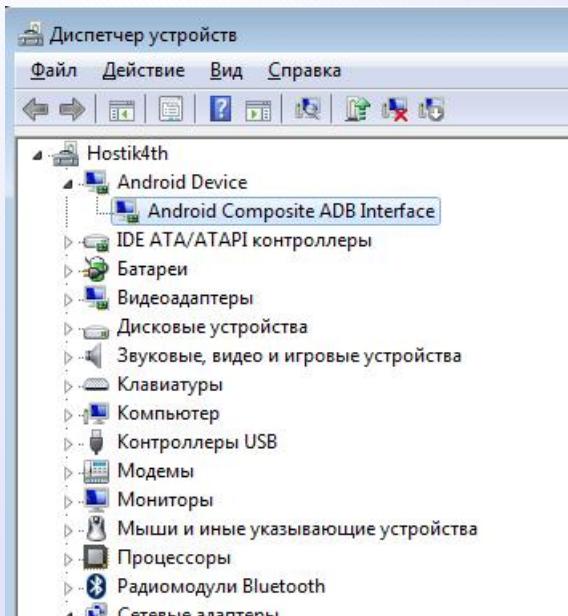


Рисунок 27. Диспетчер устройств

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 250 из 469

Назад

На весь экран

Закрыть

2. Щелкнуть по нему правой кнопкой мыши и вызвать меню Свойства.

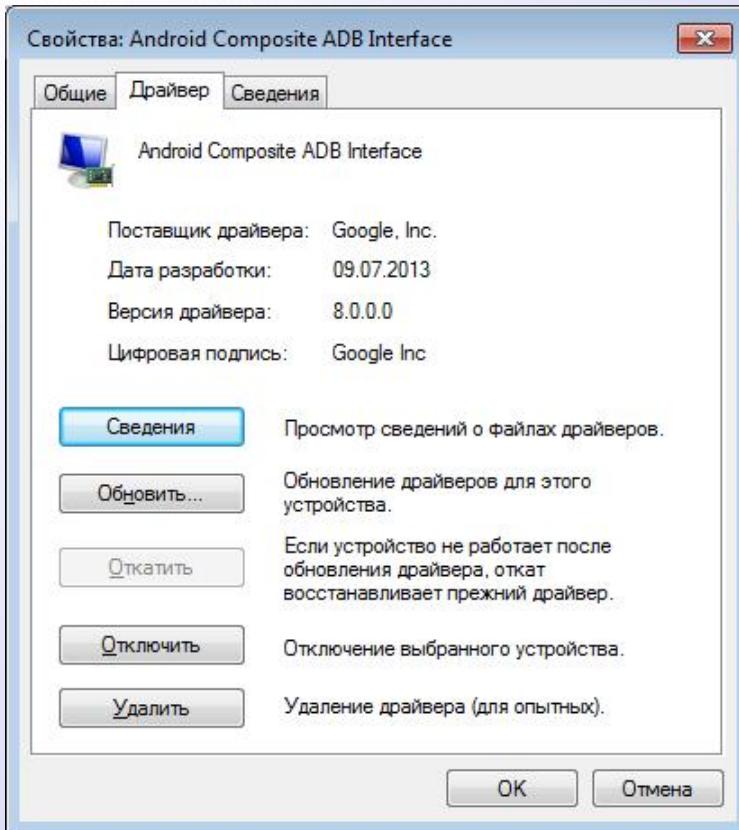


Рисунок 28.Меню Свойства



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶ ▶▶

Страница 251 из 469

Назад

На весь экран

Закрыть

3. Во вкладке **Драйвер** нажать на кнопку **Обновить** драйвер.

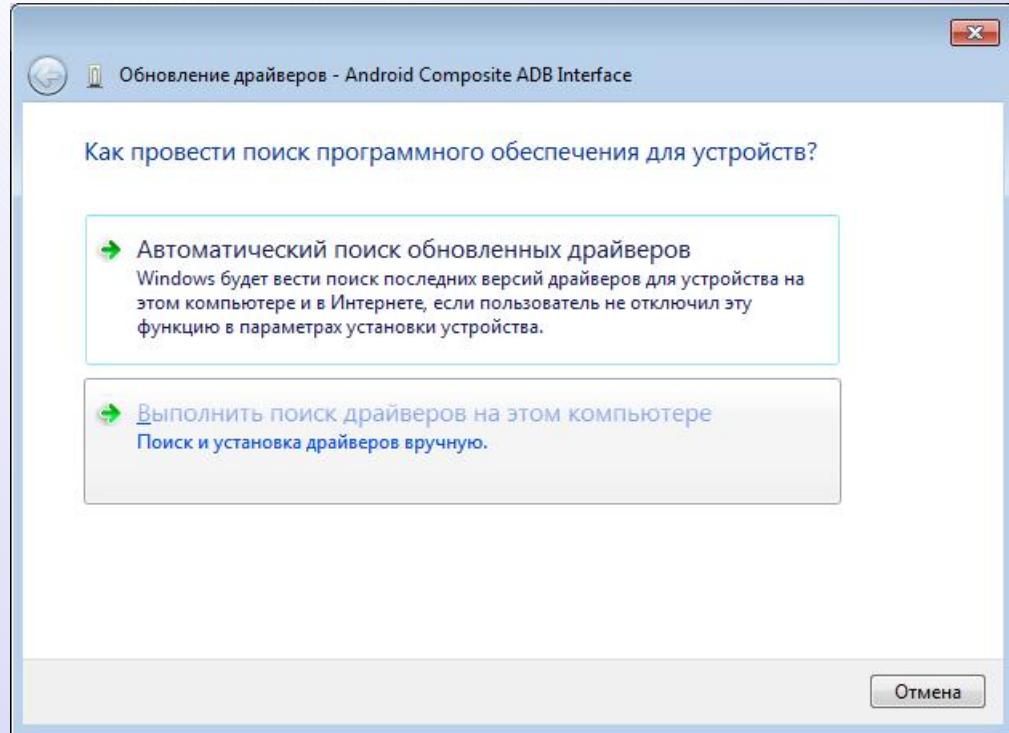


Рисунок 29. Обновление драйвера

4. В появившемся окне выбрать **Выполнить поиск драйвера на этом компьютере**.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 252 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 253 из 469

Назад

На весь экран

Закрыть

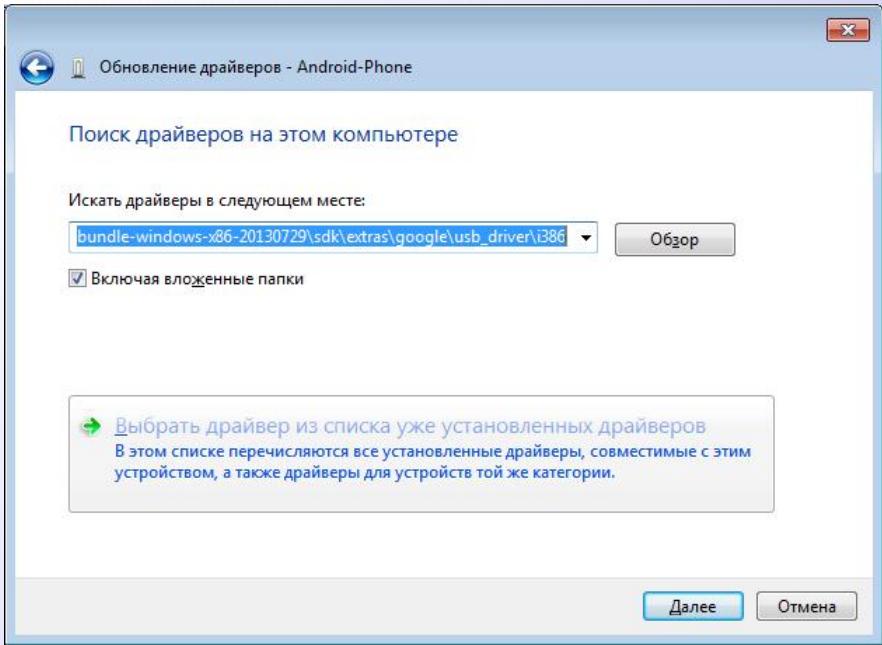


Рисунок 30.Поиск драйверов на компьютере

5. В следующем окне нужно прописать путь, откуда будет установлен драйвер.

В нашем случае путь следующий:

**adt-bundle-windows-x86-20130729\ sdk\extras\ google
\ usb_driver\ i386.**

Или можно отыскать драйвер следующим образом:

- В следующем окне нажмите **Установить с диска**

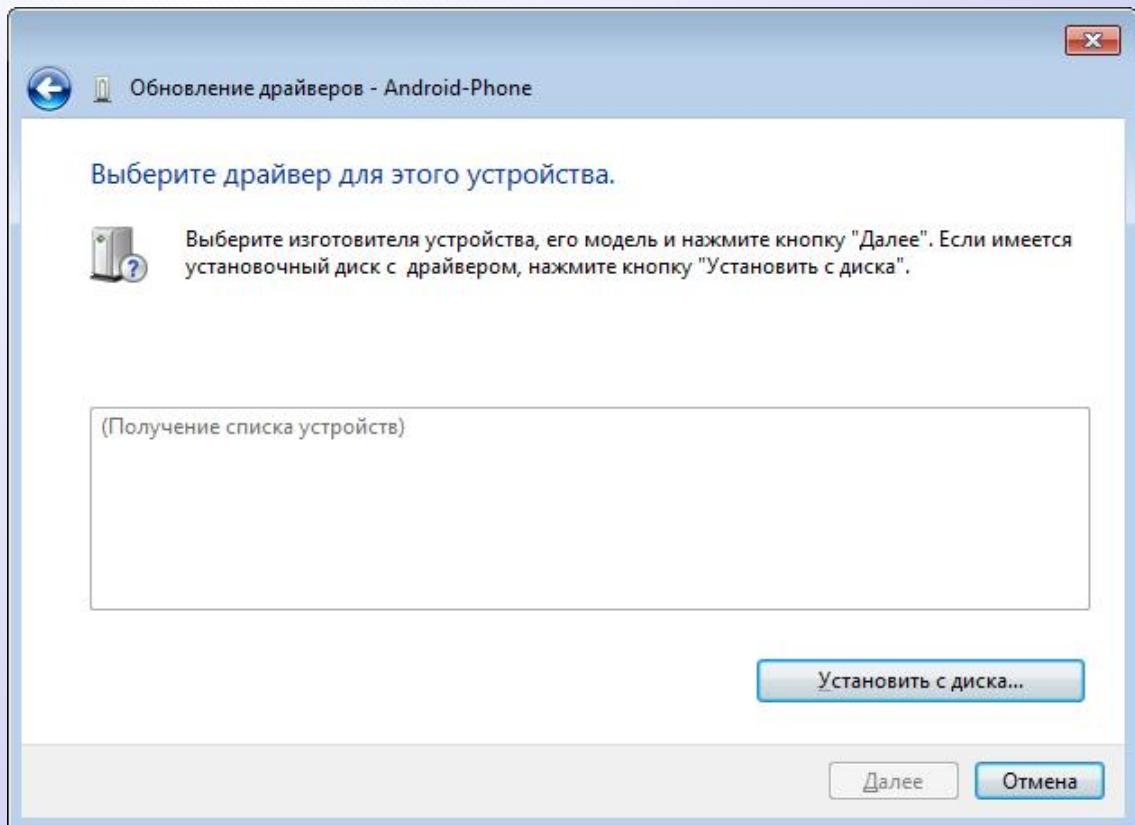


Рисунок 31. Выбор драйвера устройства



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 254 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 255 из 469

Назад

На весь экран

Закрыть

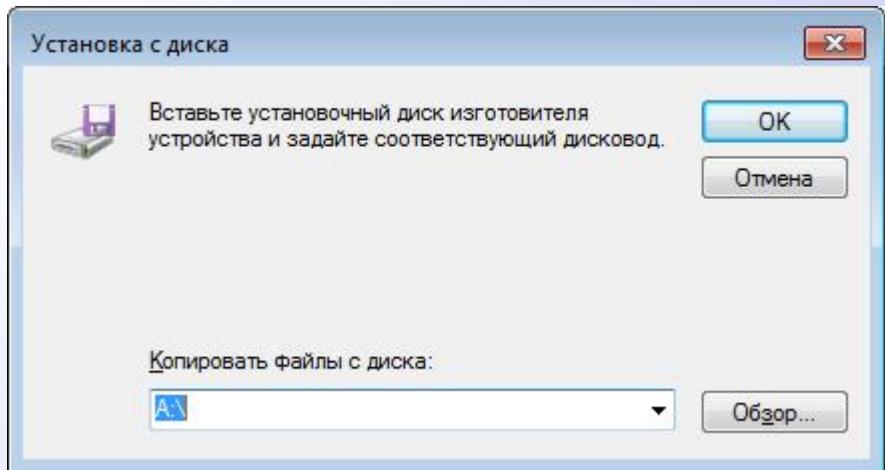


Рисунок 32.Поиск драйвера

- Далее выберите файл android_winusb.inf



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 256 из 469

Назад

На весь экран

Закрыть

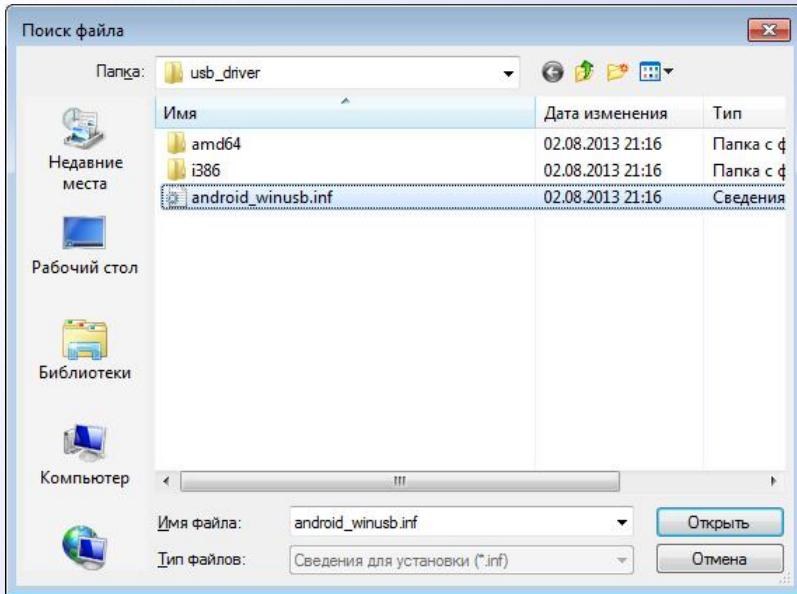


Рисунок 33. Сведения для установки

- Нажмите **Далее** в этом и следующем окнах.
- Из предложенного списка выберите **Android ADB Interface** и нажмите **Далее** и **Да**.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 257 из 469

Назад

На весь экран

Закрыть

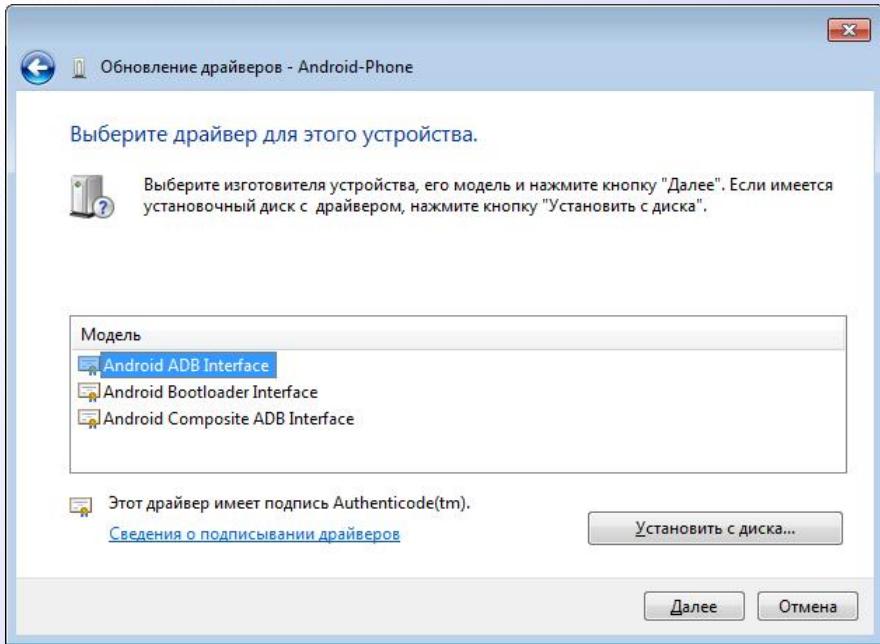


Рисунок 34. Завершение установки драйвера

Настройка среды

1. В ADT зайдите в меню **RunConfigurations** и перейдите на вкладку **Target**. Поставьте флажок напротив **Always prompt to pick device**.



Кафедра
ПМиИ

Начало

Содержание



Страница 258 из 469

Назад

На весь экран

Закрыть

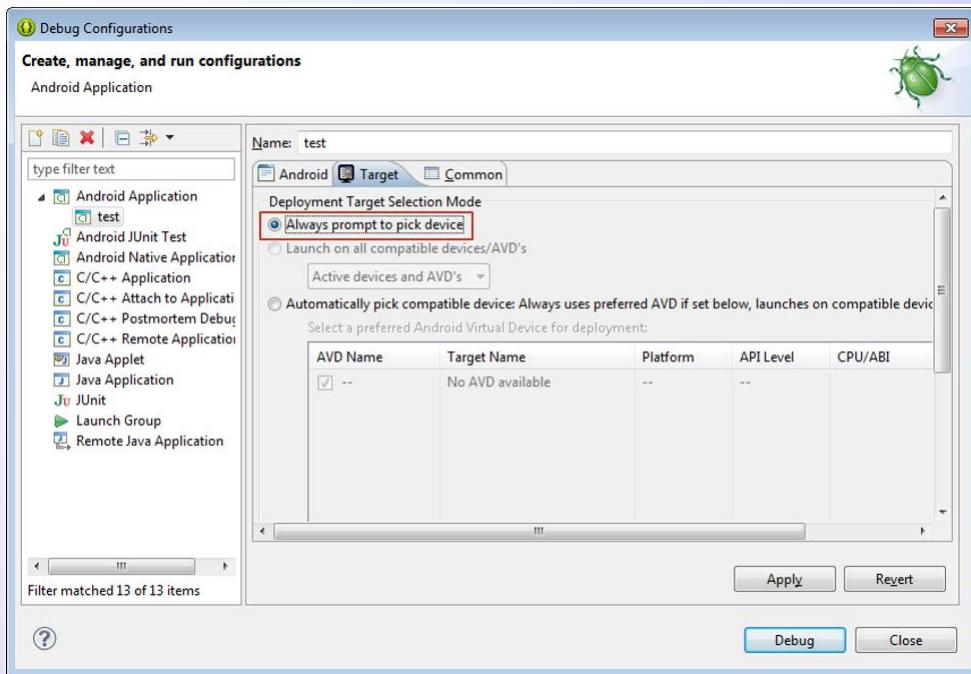


Рисунок 35.Настройка среды

2. В открывшемся окне выберите подключенное устройство, поставив флажок напротив **Choose a running Android Device**.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 259 из 469

Назад

На весь экран

Закрыть

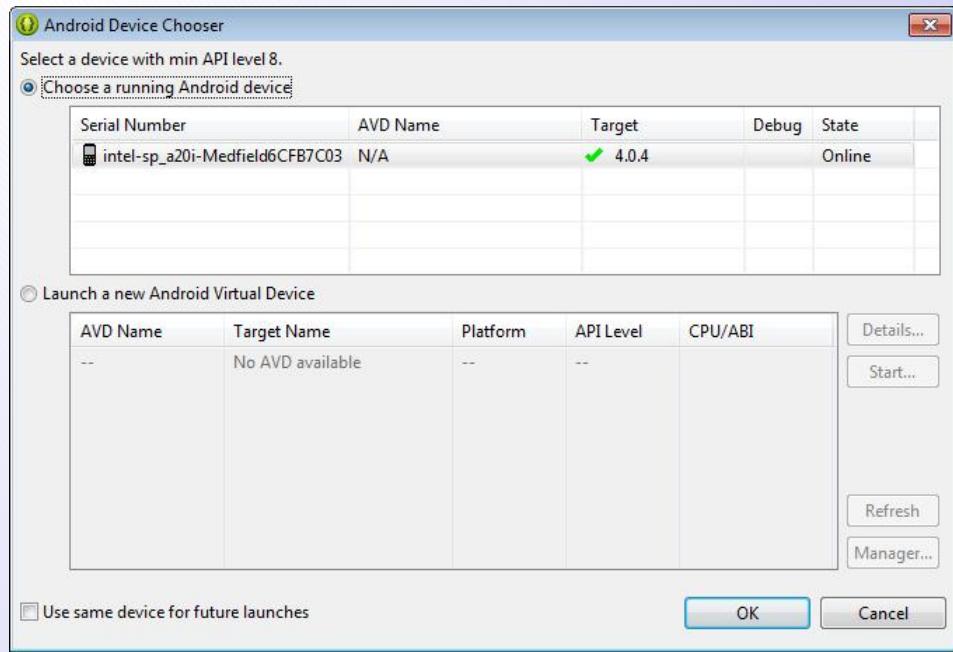


Рисунок 36. Выбор устройства

3. Теперь можно запустить приложение.

§19. Лабораторная работа №2



Цель лабораторной работы:

Разработка простого приложения, помогающего понять структуру приложения, освоить основные *операторы*, привыкнуть к среде разработки.

Задачи лабораторной работы:

- создать новое приложение и изучить его структуру;
- настроить интерфейс приложения;
- реализовать логику приложения.

19.1 Введение

Для достижения поставленной цели в лабораторной работе создадим *приложение* в среде разработки *Android IDE* (*Eclipse* и *ADT*), подробно рассмотрим структуру полученного проекта и разберем назначение основных его элементов.

Чтобы дальнейшие действия приобрели некоторый смысл, сформулируем задачу, которую будет решать наше *приложение*, назовем его "Угадай число". Суть приложения в том, что *программа* случайным образом "загадывает" число от 0 до 100, а *пользователь* должен угадать это число. При каждом вводе числа, *программа* сообщает пользователю результат: введенное число больше загаданного, меньше или же "Ура, победа!" число угадано.

Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 260 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 261 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Разрабатываемое *приложение* выполняет свои функции только когда видимо на экране, когда оно не видимо его работа приостанавливается, т. е. имеем дело с приложением переднего плана. Для выполнения всей работы достаточно определить одну *активность* в приложении, фоновые процессы не предусмотрены.

Далее в работе рассмотрим простейшие элементы интерфейса пользователя и добавим их в *приложение*, а также рассмотрим вопросы, связанные непосредственно с программированием: научимся обрабатывать события, возникающие при взаимодействии приложения с пользователем; реализуем логику проверки числа на совпадение с загаданным.

19.2 Создание приложения и изучение его структуры

Создайте новый проект в среде *Android IDE* (Eclipse с ADT). Процесс создания нового проекта и описание основных настроек подробно рассмотрен в лабораторной работе к первой лекции.

В процессе создания проекта, мы назвали его ProjectN, *среда разработки* подготавливает необходимые папки и файлы. Полный иерархический список обязательных элементов проекта можно увидеть на вкладке *Package Explorer* (аналогичную информацию предоставляет вкладка *Project Explorer*), иерархия полученных папок и файлов для нашего проекта изображена на рис 1.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 262 из 469

Назад

На весь экран

Закрыть

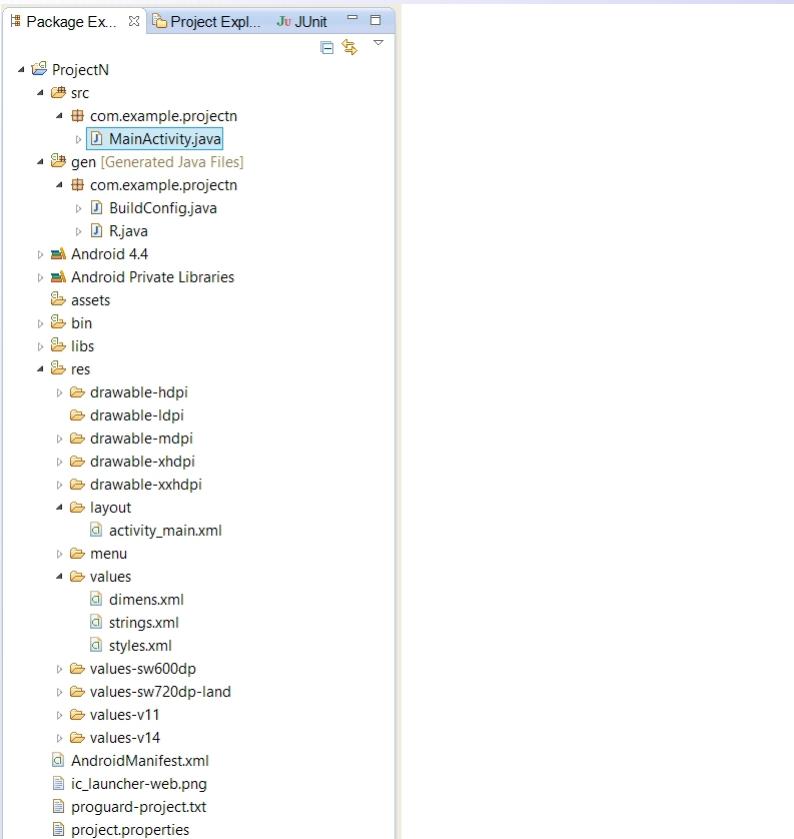


Рисунок 1. Структура проекта ProjectN

В настоящее время нас будет интересовать назначение нескольких файлов и папок.

Рассмотрим папки:

- папка **src** - содержит файлы с исходным кодом на языке Java. Именно в этой папке размещаются все классы, создаваемые в процессе разработки приложения. Сейчас в этой папке в пакете *com.example.projectn* размещается единственный класс **FullscreenActivity.java**. Этот класс определяет главную и единственную активность в этом приложении.

Комментарий 1: Имя пакету присваивается в процессе создания приложения в *поле Package Name*, использовать *com.example* не рекомендуется, т. к. пакет с таким именем нельзя загрузить в Google Play. Часто рекомендуют использовать в качестве имени пакета название сайта программиста, записанное в обратном порядке, можно просто использовать свои имя и фамилию. Последнее слово в имени пакета формируется автоматически и совпадает с именем проекта.

Комментарий 2: Имя файлу присваивается в процессе создания приложения на этапе настройки активности. Имя определяется в *поле Activity Name*.

Комментарий 3: *Package Explorer* отображает структуру папок, которая создается в каталоге, выбранном в качестве рабочего (Workspace) при запуске Eclipse. Например, рабочий каталог называется *workspaceADT*, в нем для нашего проекта появилась папка с именем *ProjectN*, в ней есть папка *src*, в ней *com*, в ней *example*, в ней *projectn* (заметьте, что название пакета распалось на три пап-



Кафедра ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 263 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 264 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

ки, каждое слово, отделенное точкой определило новую папку). И уже в папке projectn находится файл *MainActivity.java* и будут размещаться другие *java*-файлы проекта.

- папка **gen** - содержит java-файлы, которые не требуется изменять и лучше вообще не трогать. Эти файлы генерируются автоматически. Нас может заинтересовать файл R.java он содержит идентификаторы (ID) для всех ресурсов приложения.
- папка **res** - содержит структуру папок ресурсов приложения, рассмотрим некоторые из них:
 - **layout** - в данной папке содержатся xml-файлы, которые описывают внешний вид форм и их элементов, пока там находится только activity_main.xml;
 - **values** - содержит XML файлы, которые определяют простые значения, таких ресурсов как, строки, числа, цвета, темы, стили, которые можно использовать в данном проекте;
 - **menu** - содержит XML файлы, которые определяют все меню приложения.

Рассмотрим файл **AndroidManifest.xml** - файл в формате *xml*, который описывает основные свойства проекта, разрешение на использование ресурсов устройства и др. Сразу после создания приложения файл *AndroidManifest.xml* выглядит так, как показано на рис 2.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 265 из 469

Назад

На весь экран

Закрыть

The screenshot shows the Android Studio interface with the manifest file open. The left pane displays the XML code of the `AndroidManifest.xml` file, which defines a package named `com.example.projectn`, sets the min SDK version to 8, target to 18, and includes an activity named `MainActivity` with a launcher intent filter. The right pane shows the project outline with nodes for manifest, uses-sdk, application, activity, intent-filter, action, and category.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.projectn"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Manifest Application Permissions Instrumentation AndroidManifest.xml

Рисунок 2.Файл AndroidManifest.xml только созданного проекта

Рассмотрим подробно *файл* манифеста.

Первый обязательный элемент `<manifest>` является корневым элементом файла, должен содержать обязательный элемент `<application>` и все остальные элементы по необходимости. Рассмотрим основные атрибуты этого элемента:

`xmlns:android`

- определяет пространство имен Android, всегда должен иметь значение:

`"http://schemas.android.com/apk/res/android"`.

Обязательный атрибут.

package

- полное имя пакета, в котором располагается приложение. Обязательный атрибут. Имя должно быть уникальным, может содержать заглавные и строчные латинские буквы, числа и символ подчеркивания. Однако начинаться должно только с буквы. Для избежания конфликтов с другими разработчиками рекомендуется использовать имя вашего сайта (если он есть) записанное в обратном порядке. В нашем случае пакет имеет имя "com.example.projectn" и наше приложение не удастся разместить в Google Play (но мы на это и не претендуем). **Внимание:** если Вы опубликовали свое приложение, Вы **не можете менять имя пакета**, т.к. имя пакета служит уникальным идентификатором для приложения и в случае его смены приложение будет рассматриваться, как совсем другое и пользователи предыдущей версии не смогут его обновить.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 266 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 267 из 469

Назад

На весь экран

Закрыть

android:versionCode

- внутренний номер версии приложения не виден пользователю. Этот номер используется только для определения является ли одна версия более современной по сравнению с другой, больший номер показывает более позднюю версию.

android:versionNumber

- номер версии, является строкой и используется только для того, чтобы показать пользователю номер версии приложения.

android:shareUserID,
android:sharedUserLabel,
android:installLocation

- эти атрибуты в нашем файле манифеста не представлены, про их назначение можно почитать по ссылке: <http://developer.android.com/guide/topics/manifest/manifest-element.html>.

Рассмотрим элемент `<uses-sdk>`, который показывает совместимость приложения с версиями *Android*. Основные атрибуты:



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 268 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

android:minSdkVersion

- указывает значение минимального уровня API, необходимого для работы приложения. Система Android не позволит установить приложение, если уровень API ниже, чем уровень, указанный в этом атрибуте. **Внимание:** если этот атрибут не указан, система установит значение по умолчанию равным "1"; которое означает, что приложение совместимо со всеми версиями Android. И в случае, если приложение не совместимо со всеми версиями, установка пройдет на любую версию Android, а во время работы приложение сломается, когда попытается обратиться к недоступным элементам API. Поэтому необходимо всегда указывать значение этого атрибута.

- указывает уровень API целевой платформы Android приложения, если этот атрибут пропущен, по умолчанию принимается значение android:minSdkVersion.

android:targetSdkVersion



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 269 из 469

Назад

На весь экран

Закрыть

android:maxSdkVersion

- указывает максимальное значение уровня API, под который разрабатывалось приложение. Если значение этого атрибута ниже, чем уровень API соответствующий версии Android, на которую устанавливается приложение, то система не позволит установить такое приложение.

Если внимательно посмотреть на *файл* манифеста рис 2, можно заметить, что данный *атрибут* в нем отсутствует. На самом деле разработчики не рекомендуются задавать *значение* этого атрибута. Во-первых, это *значение* будет препятствовать использованию приложений на новых версиях *Android* при их появлении, несмотря на то, что все новые версии полностью обратно-совместимы. Во-вторых, стоит иметь ввиду, что задание этого атрибута может привести к тому, что *приложение* будет удалено с устройства после обновления *Android* до более высокого уровня *API*.

Подробнее с элементом `<uses-sdk>` и его атрибутами можно ознакомиться по ссылке:<http://developer.android.com/guide/topics/manifest/uses-sdk-element.html>.

Рассмотрим элемент `<application>`, который является обязательным элементом манифеста, полностью определяет состав приложения. Представляет собой *контейнер* для элементов `<activity>`, `<service>`,



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 270 из 469

Назад

На весь экран

Закрыть

<receiver>, <provider> (и не только), каждый из которых определяет соответствующий компонент приложения. Содержит набор атрибутов, действие которых распространяется на все компоненты приложения. Рассмотрим атрибуты элемента <application>, представленные в манифесте на рис 2:

android:allowBackup

- определяет разрешение для приложения участвовать в резервном копировании и восстановлении. Если значение этого атрибута **false**, то для приложения никогда не может быть создана резервная копия, даже если проводится резервное копирование всей системы целиком. По умолчанию значение этого атрибута равно **true**.

android:icon

- определяет иконку для приложения целиком, а также иконку по умолчанию для компонентов приложения, которая может быть переопределена атрибутом android:icon каждого компонента. Задается как ссылка на графический ресурс, содержащий изображение, в нашем случае значение этого атрибута равно "@drawable/ic_launcher".



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 271 из 469

Назад

На весь экран

Закрыть

android:label

- определяет видимый для пользователя заголовок приложения целиком, а также заголовок по умолчанию для компонентов приложения, который может быть переопределен атрибутом `android:label` каждого компонента. Задается как ссылка на строковый ресурс, в нашем случае значение атрибута равно `"@string/app_name"`.

android:theme

- определяет тему по умолчанию для всех активностей приложения, может быть переопределен атрибутом `android:theme` каждой активности. Задается как ссылка на стилевой ресурс, в нашем случае значение атрибута равно `"style/AppTheme"`.

На самом деле у элемента `<application>` гораздо больше атрибутов, чем нам удалось рассмотреть, найти полный список атрибутов с описаниями можно по ссылке: <http://developer.android.com/guide/topics/manifest/application-element.html>.

В приложении всего одна *активность*, других компонентов нет, в связи с этим элемент `<application>` в манифесте содержит ровно один элемент `<activity>` и больше никаких других элементов не содержит. Рассмотрим элемент `<activity>`, который определяет *активность*. Для



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 272 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

каждой активности обязательно необходим свой элемент `<activity>` в манифесте. Рассмотрим атрибуты элемента `<activity>`, представленные в манифесте на рис 3:

android:name

- определяет имя класса, который задает активность. Значение атрибута должно полностью определять имя класса с указанием пакета, в котором располагается класс. В нашем случае атрибут имеет значение: "`com.example.projectn.MainActivity`". Можно использовать сокращенную запись `".MainActivity"`; в этом случае добавляется имя пакета, определенное соответствующим атрибутом элемента `<manifest>`.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 273 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

android:configChanges

- перечисляет изменения конфигурации, которыми может управлять активность. Если конфигурация меняется во время работы, то по умолчанию активность останавливается и перезапускается. Если же изменение конфигурации указано в этом атрибуте, то при появлении этого изменения активность не перезапускается, вместо этого она продолжает работать и вызывает метод `onConfigurationChanged()`. В нашем случае атрибут имеет значение "orientation|keyboardHidden|screenSize"; т. е. при смене ориентации экрана, смене размера экрана и изменении доступности клавиатуры не произойдет перезапуск активности.

- определяет видимый пользователю заголовок активности, если он отличается от общего заголовка приложения. Задается как ссылка на строковый ресурс, в нашем случае значение атрибута равно "`@string/app_name`" (т.е. можно было и не задавать).

android:label



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 274 из 469

Назад

На весь экран

Закрыть

android:theme

- определяет тему активности, если она отличается от общей темы приложения, заданной соответствующим атрибутом элемента `<application>`. Задается как ссылка на стилевой ресурс, в нашем случае значение атрибута равно "`@style/FullscreenTheme`".

На самом деле у элемента `<activity>` гораздо больше атрибутов, чем нам удалось рассмотреть, найти полный список атрибутов с описаниями можно по ссылке: <http://developer.android.com/guide/topics/manifest/application-element.html>.

В манифесте для нашего приложения элемент `<activity>` содержит ровно один элемент: `<intent-filter>`, определяющий типы намерений, которые может принимать активность. Этот элемент содержит два элемента: `<action>` и `<category>`.

Первый элемент определяет действия, которые проходят в фильтр намерений, при этом `<intent-filter>` должен содержать хотя бы один элемент `<action>`, в противном случае ни один объект-намерение не сможет пройти через фильтр и активность не возможно будет запустить. Элемент `<action>` имеет единственный атрибут `android:name="android.intent.action.MAIN"`.

Второй элемент определяет имя категории в фильтре намерений. Име-

ет единственный атрибут `android:name="android.intent.category.LAUNCHER"`.

На этом разбор манифеста приложения закончим, подробно с описанием всех элементов этого файла можно познакомиться по ссылке: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>.

Чаще всего, при создании приложения приходится иметь дело с папками `src`, `res/layout` и `res/values`, т.к. там находятся основные файлы проекта.

19.3 Настройка интерфейса приложения

До того, как начнем формировать *интерфейс*, имеет смысл подготовить возможность проверки разрабатываемого приложения на ошибки. Чтобы не загружать каждый раз *приложение* на реальное устройство, в *Android SDK* предусмотрена возможность использования виртуального устройства (*AVD* или *Android virtual device*), эмулирующего работу реального смартфона. Процесс создания виртуального устройства или эмулятора подробно расписан в "[Установка и настройка среды программирования ADT Bundle](#)".

Пришло время задуматься о внешнем виде приложения. Для начала необходимо определить какие элементы графического интерфейса нам нужны, как эти элементы будут располагаться на форме и каким образом будет реализовано взаимодействие с пользователем.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 275 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 276 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

Так как *приложение* очень простое, то и *интерфейс* особой сложностью отличаться не будет. Нам потребуется *поле* для ввода чисел (**TextEdit**), текстовая *метка* для вывода информации (**TextView**) и кнопка для подтверждения введенного числа (**Button**). Располагать элементы интерфейса будем друг под другом, сверху информационная часть, ниже *поле ввода*, кнопку разместим в самом низу приложения. Взаимодействие приложения с пользователем организуется очень просто: *пользователь* вводит число в *поле* для ввода и нажимает кнопку, читает результат в информационном *поле* и, либо радуется победе, либо вводит новое число.

Если *пользователь* вводит правильное число, то *приложение* предлагает ему сыграть снова при этом кнопка будет играть роль подтверждения, а в информационное *поле* будет выведено приглашение к повторной игре.

Схематично *интерфейс* приложения изображен на рис 3.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 277 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

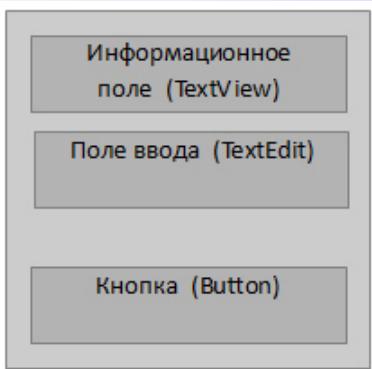


Рисунок 3. Схема интерфейса приложения "Угадай число" @

Нам необходимо добавить на форму три элемента: информационное поле (**TextView**), поле ввода (**TextEdit**) и кнопку (**Button**).

Android IDE поддерживает два способа для выполнения действий по формированию интерфейса приложения: первый основан на *XML*-разметке, второй относится к визуальному программированию и позволяет перетаскивать объекты интерфейса и размещать их на форме с помощью мыши. Считается, что визуальный способ подходит для новичков, а более продвинутые разработчики могут писать код вручную, однако чаще всего используется комбинированный подход.

Для формирования интерфейса будем работать с файлом `textbfres/layout/activity_main.xml`. На рис 4 можно увидеть редактор, соответствующий визуальному способу формирования интерфейса, это-

му режиму соответствует вкладка **Graphical Layout**.

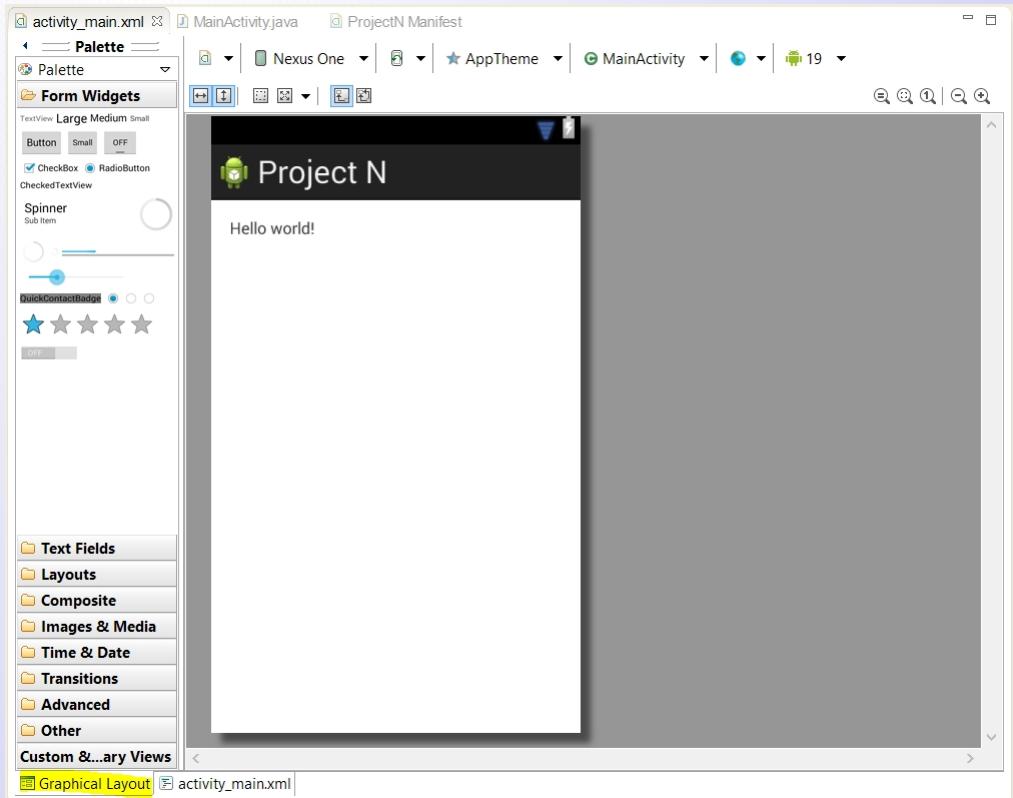


Рисунок 4.Графическое изображение активности приложения

На рис 4 рядом с вкладкой **Graphical Layout** расположена вкладка **activity_fullscreen.xml**. Она соответствует режиму редактирования



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

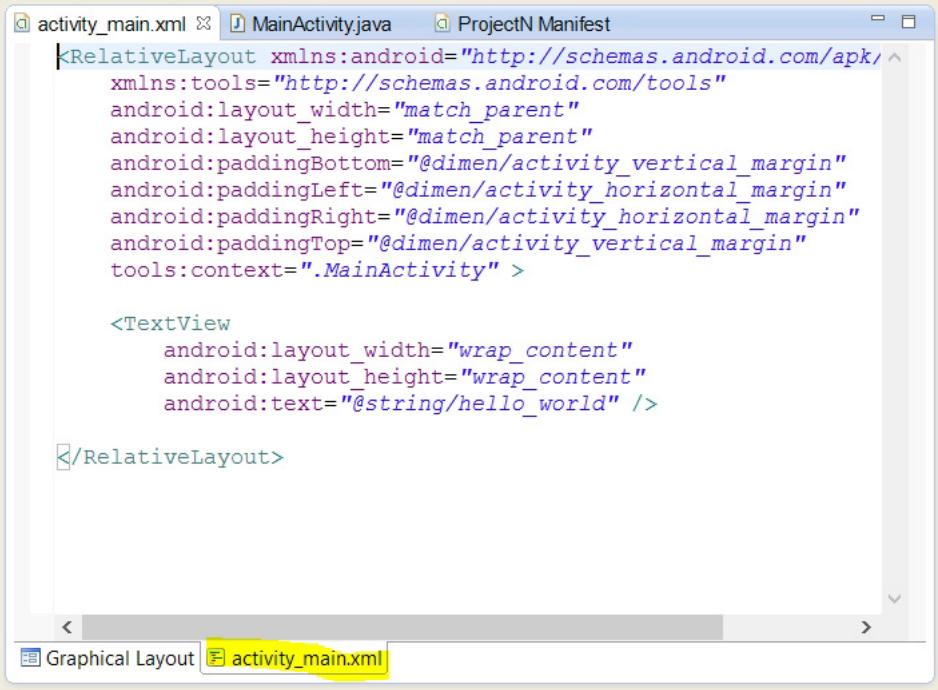
Страница 278 из 469

Назад

На весь экран

Закрыть

интерфейса путем формирования *XML* файла. На рис 5 можно увидеть редактор *XML* файла.



The screenshot shows the Android Studio XML Editor. The title bar has tabs for 'activity_main.xml' (selected), 'MainActivity.java', and 'ProjectN Manifest'. The main area displays the XML code for the activity:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/^\n    xmlns:tools="http://schemas.android.com/tools"\n    android:layout_width="match_parent"\n    android:layout_height="match_parent"\n    android:paddingBottom="@dimen/activity_vertical_margin"\n    android:paddingLeft="@dimen/activity_horizontal_margin"\n    android:paddingRight="@dimen/activity_horizontal_margin"\n    android:paddingTop="@dimen/activity_vertical_margin"\n    tools:context=".MainActivity" >\n\n    <TextView\n        android:layout_width="wrap_content"\n        android:layout_height="wrap_content"\n        android:text="@string/hello_world" />\n\n</RelativeLayout>
```

At the bottom, there are tabs for 'Graphical Layout' (disabled) and 'activity_main.xml' (selected). A yellow box highlights the 'activity_main.xml' tab.

Рисунок 5.Описание активности в XML формате

Зададим табличное расположение компонентов на форме, для этого выберем вкладку **Layouts**, найдем там **TableLayout** и добавим его на форму. На рис 6 можно увидеть результат этих действий.

Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 279 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 280 из 469

Назад

На весь экран

Закрыть

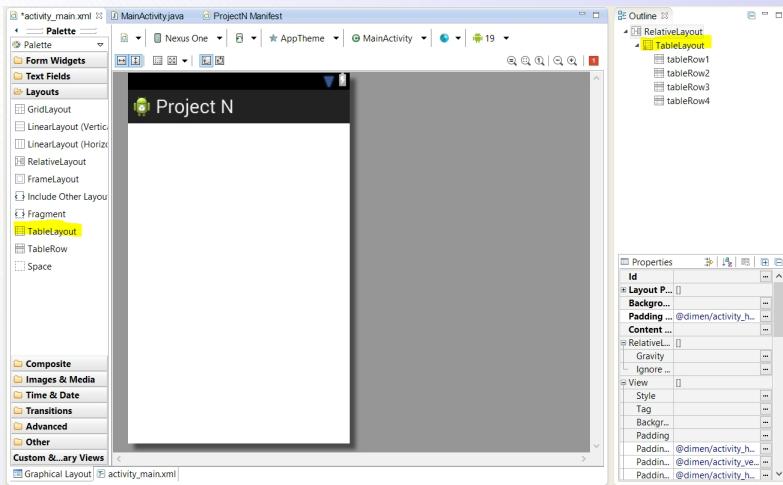


Рисунок 6. Настройка интерфейса, добавление TableLayout

Теперь начнем добавлять элементы интерфейса, будем использовать *графический режим* правки.

Во-первых, нам необходимо добавить информационное *поле*. Для этого на панели =Palette= выбираем вкладку **Form Widgets**, на этой вкладке найдем поле **TextView**, перенесем в окно приложения, разместим в первой строке таблицы (**TableLayout**).

Во-вторых, нам потребуется поле ввода информации, на вкладке **Text Fields** найдем текстовое поле **Number** и разместим во второй строке таблицы.

В-третьих, вернемся на вкладку **Form Widgets**, выберем там эле-

мент **Button** и добавим в третью строку таблицы. Не нужную четвертую строку таблицы удалим, получим следующий вид приложения, см. рис 7.

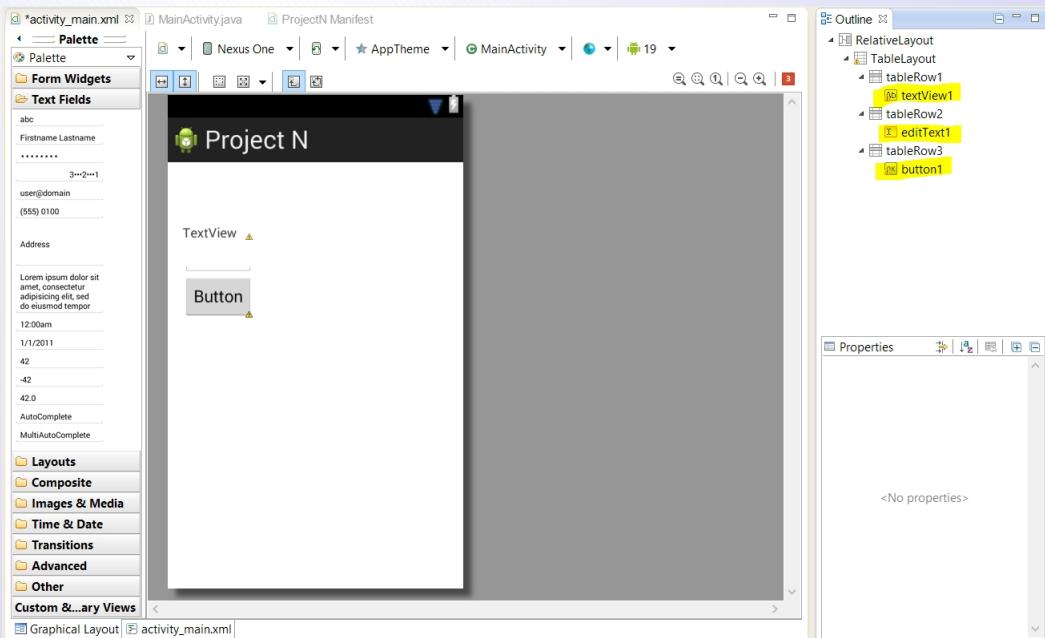


Рисунок 7.Интерфейс приложения

После настройки интерфейса можно заглянуть в *файл activity_main.xml*, в этом файле прописано, что используется



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 281 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание



Страница 282 из 469

Назад

На весь экран

Закрыть

```
*activity_main.xml>MainActivity.java>ProjectN Manifest
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TableLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="52dp" >

        <TableRow
            android:id="@+id/tableRow1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" >

            <TextView
                android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="TextView" />

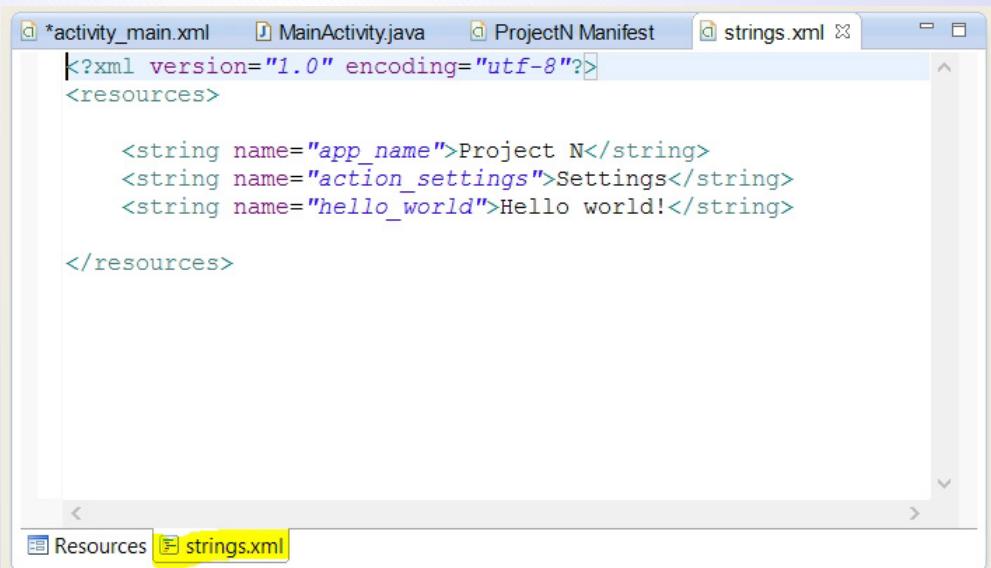
        </TableRow>
    

```

Рисунок 8.Фрагмент файла activity_fullscreen.xml, описание строки в TableLayout

Теперь необходимо наполнить наши элементы интерфейса смыслом, нам понадобится текст для общения с пользователем, при программировании под *Android* существует практика разделять ресурсы и код прило-

жения. Для хранения любых строк, которые могут понадобиться приложению, используется *файл strings.xml*. Хранение всех строковых ресурсов в этом файле серьезно облегчает локализацию приложения на другие языки. Этот *файл* можно найти в *Package Explorer* в папке **res/values**. Откроем его и посмотрим, что там есть, см. рис 9.



The screenshot shows the Android Studio interface with the Project N Manifest tab selected. Below it, the strings.xml file is open, displaying the following XML code:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Project N</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
</resources>
```

The strings.xml file is located in the Resources folder, which is highlighted with a yellow box in the bottom navigation bar.

Рисунок 9. Файл string.xml

Уберем лишние строки и добавим новые, результат можно посмотреть на рис 10.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 283 из 469

Назад

На весь экран

Закрыть



```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Угадай число!</string>
    <string name="behind">Недолёт!</string>
    <string name="ahead">Перелёт!</string>
    <string name="hit">В точку!</string>
    <string name="input_value">Ввести значение</string>
    <string name="play_more">Сыграть ещё</string>
    <string name="try_to_guess">Попробуйте угадать число (1 &#8211; 100)</string>
    <string name="error">Неверный ввод!</string>

</resources>
```

Рисунок 10.Отредактированный файл string.xml

Данные переменные будут выполнять следующие задачи:

- **app_name** установит "видимое" название приложения;
- **behind, ahead, hit** оповестят пользователя об его успехах в игре;
- **play_more** и **try_to_guess** установят название кнопки, которое объяснит её функции;
- **input_value** пригласит пользователя к вводу числа;
- **error** сообщит о неверном вводе.

После изменения **strings.xml**, при переходе на другую вкладку, не забудьте сохранить изменения (самый быстрый способ - нажать **Ctrl+S**).

Настроим текст в информационном *поле*. Для этого на вкладке **Properties** в правой части окна выберем элемент **textView1** (это и есть

Начало

Содержание

◀ ▶

◀▶ ▶▶

Страница 284 из 469

Назад

На весь экран

Закрыть



наше информационное *поле*, имеет смысл придумать ему более осмысленное имя). Найдем свойство **Text**, подставим в него значение строки с именем **try_to_guess**, см. рис 11.

Аналогично можно настроить текст, которым нас будет приветствовать кнопка, только в этом случае надо работать с элементом **button1**.

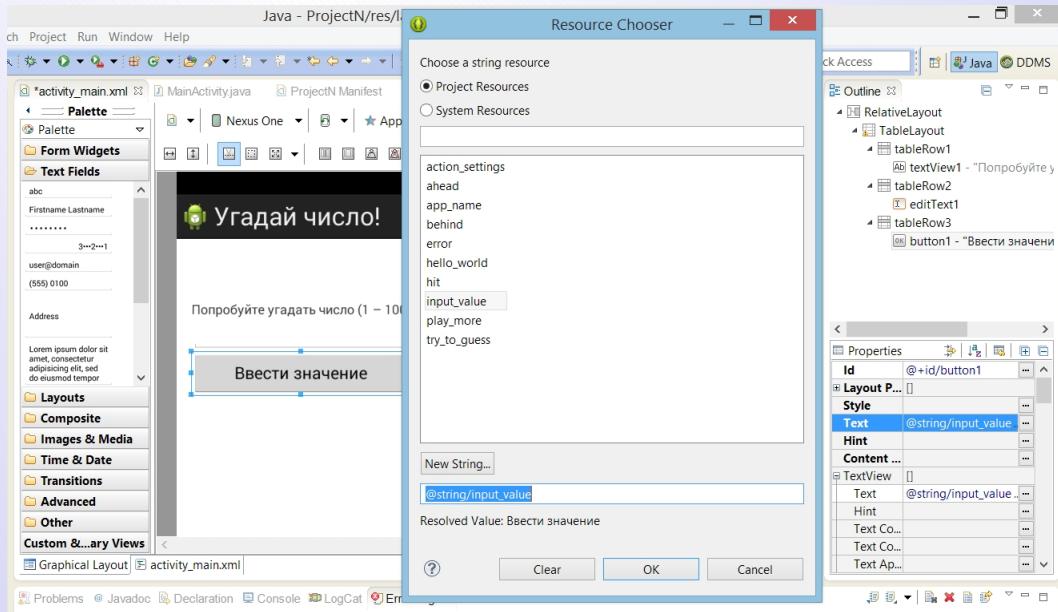


Рисунок 11.Настройка текста для кнопки button1

Пришло время вспомнить о виртуальном устройстве, если оно работает, уже можно запустить проект и посмотреть, как *приложение* будет

Кафедра
ПМиИ

Начало

Содержание



Страница 285 из 469

Назад

На весь экран

Закрыть

выглядеть на экране устройства, а выглядеть оно может как показано на рис 12.

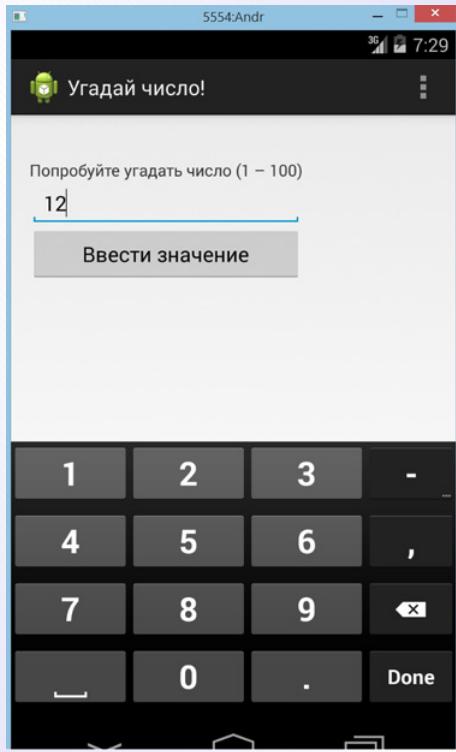


Рисунок 12.Запуск приложения на виртуальном устройстве

Приложение выглядит довольно просто, но мы на многое и не рассчитывали. Главное, что нас интересует, это наличие всех элементов на



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 286 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

экране, верный текст в каждом элементе, где он предусмотрен и возможность вводить числа в поле ввода. На рис 12 видно, что все требования выполнены. *Приложение* есть, его можно запустить на виртуальном или реальном устройстве, но оно ничего не делает. Следующим шагом будет реализация логики приложения, т. е. обработка события нажатия на кнопку, как было прописано в задании.

19.4 Реализация логики приложения

Приступим непосредственно к программированию, работать будем с файлом **src/com.example.projectn/MainActivity.java**. Найдем этот файл в *Package Explorer* см. рис 13, откроем и начнем редактировать.

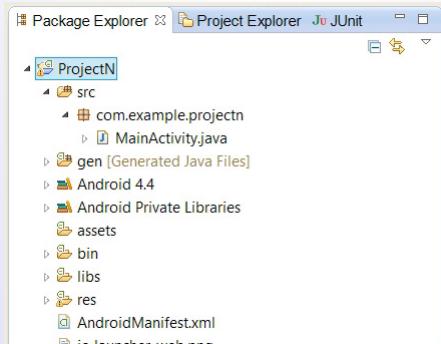


Рисунок 13.Файл MainActivity.java в Package Explorer



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 287 из 469

Назад

На весь экран

Закрыть

Пока файл выглядит следующим образом, см. рис 14.



```
activity_main.xml MainActivity.java ProjectN Manifest strings.xml

package com.example.projectn;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

Рисунок 14.Файл MainActivity.java после создания приложения

Можно заметить, что *класс MainActivity* является наследником класса *Activity* и в нем уже реализован метод *onCreate()*, который запускается при первоначальном создании активности, нам потребуется его дополнить, но об этом чуть позже.

Мы предполагаем программно менять информацию в *поле TextView*, получать *значение* из поля *EditText* и обрабатывать события нажатия на кнопку *Button*, поэтому необходимо объявить соответствующие переменные, как поля *класса MainActivity*:



```
TextView tvInfo;  
EditText etInput;  
Button bControl;
```

Чтобы не было ошибок, необходимо импортировать пакет `android.widget`, который содержит все элементы графического интерфейса:

```
import android.widget.*;
```

На самом деле *среда разработки* подскажет, что делать.

Теперь необходимо связать эти переменные с элементами интерфейса, уже добавленными нами в **activity_main.xml**, сделать это необходимо в методе `onCreate()`, а для получения уже созданного элемента интерфейса воспользуемся методом `findViewById()`. Итак в метод `onCreate()` добавим следующие строки:

```
tvInfo = (TextView) findViewById(R.id.textView1);  
etInput = (EditText) findViewById(R.id.editText1);  
bControl = (Button) findViewById(R.id.button1);
```

Метод `findViewById()` возвращает *объект* класса `View`, который является общим предком для всех компонентов пользовательского интерфейса, для того чтобы избежать возможных ошибок в скобках перед вызовом метода указываем до какого конкретно компонента необходимо сузить возможности объекта `View`.

Пришло время выполнить обработку нажатия на кнопку. Вернемся к файлу **activity_main.xml** в *графический режим* редактирования,

Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 289 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание



Страница 290 из 469

Назад

На весь экран

Закрыть

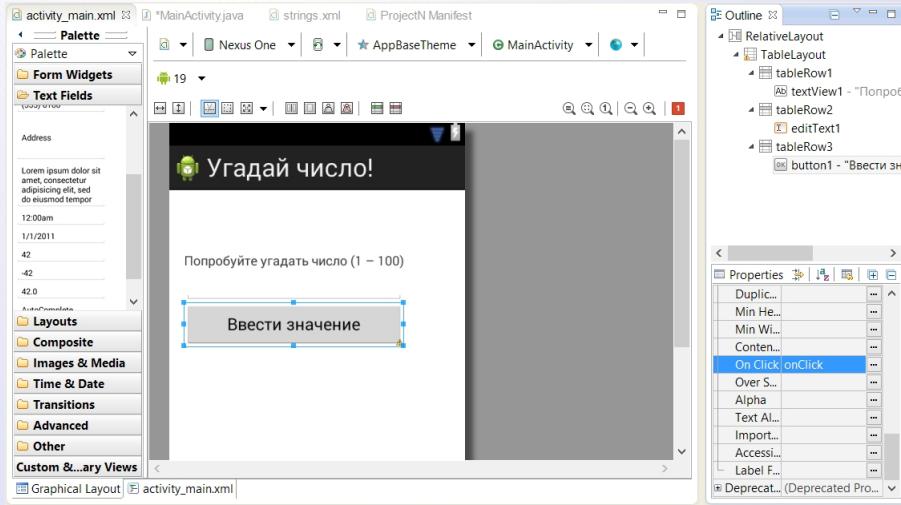


Рисунок 15.Настройка свойства On Click для кнопки

выберем элемент **Button** и на вкладке со свойствами элемента найдем свойство **On Click** и запишем в него **onClick** - имя метода, который будет обрабатывать нажатие на кнопку. Как это выглядит показывает рис 15.

Эти же действия можно выполнить в файле **activity_main.xml**, достаточно дописать выделенную на рис 16 строку:



```
<Button  
    android:id="@+id/button1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:onClick="onClick"  
    android:text="@string/input_value" />
```

Рисунок 16.Настройка свойствами On Click для кнопки в файле XML

Для настройки свойств элементов интерфейса достаточно использовать любой способ: графический или редактирование *XML* файла.

Вернемся в *файл MainActivity.java*, в *класс* активности необходимо добавить метод:

```
public void onClick(View v){...}
```

Имя метода не обязательно должно быть *onClick()*, главное, чтобы оно совпадало с именем, указанным в свойстве **On Click**. В этом методе и будет происходить все наше *программирование* в этой лабораторной работе.

Нам потребуются две переменные:

- целочисленная для хранения загаданного числа (случайное число от 1 до 100);
- логическая для хранения состояния закончена игра или нет.

Обе эти переменные имеет смысл объявить как *поля класса* активности, первоначальные значения присвоить в методе *onCreate*.



Получить целочисленное значение из поля ввода, можно с помощью следующей конструкции:

`Integer.parseInt(etInput.getText().toString())`

в данном случае в информационном поле появится значение строкового ресурса с именем `ahead`.

Осталось реализовать логику приложения в методе `onClick()`. Предлагаем написать код этого метода самостоятельно, для контроля в приложении предложен листинг, который содержит один из вариантов кода описанного приложения.

19.5 Задания для самостоятельной работы:

Разумеется предложенная в приложении реализация не идеальна, требует доработок:

- Что произойдет, если кнопка будет нажата до ввода какого-либо числа? Скорей всего приложение будет остановлено, необходимо как-то отслеживать этот вариант развития событий и адекватно реагировать. Предлагаем поработать самостоятельно над этой проблемой.
- Что произойдет, если пользователь введет число меньшее нуля или большее 100? Скорей всего приложение обработает этот ввод, но было бы лучше, если бы появилось сообщение о том, что введенное число не удовлетворяет условиям задачи.

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 292 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

- Как завершить приложение? И надо ли это делать?
- Предлагаем подумать, как еще можно улучшить приложение и реализовать эти улучшения.

19.6 Немного о работе с эмулятором

При выполнении данной работы мы использовали эмулятор для проверки работоспособности приложения. Рассмотрим некоторые возможности, облегчающие и ускоряющие взаимодействие с виртуальным устройством.

Во-первых, существует набор полезных комбинаций клавиш для управления виртуальным устройством:

- **Alt+Enter** - разворачивает эмулятор до размеров экрана;
- **Ctrl+F11** - меняет ориентацию эмулятора с портретной на альбомную и обратно;
- **F8** - включает/выключает сеть.

Полный список комбинаций клавиш для работы с эмулятором можно найти по ссылке:<http://developer.android.com/tools/help/emulator.html>.

Во-вторых, кто бы ни работал с эмулятором, тот на себе прочувствовал насколько терпеливым надо быть, чтобы взаимодействовать с ним, так медленно он работает. Существует решение для ускорения работы



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 293 из 469

Назад

На весь экран

Закрыть

эмулятора *Android* и этим решением является Intel *Hardware Accelerated Execution Manager* (Intel® HAXM).

Intel Hardware Accelerated Execution Manager (Intel® HAXM)

- это *приложение* с поддержкой аппаратной виртуализации (гипервизор), которое использует технологию виртуализации Intel для ускорения эмуляции приложений *Android* на компьютере для разработки. (<http://software.intel.com/ru-ru/android/articles/intel-hardware-accelerated-execution-manager>)

Intel HAXM способен ускорить работу эмулятора для *x86* устройств. При этом эмулятор будет работать со скоростью, приближенной к скорости работы реального устройства, что поможет сократить время на запуск и отладку приложения. Подробно познакомиться с установкой Intel HAXM и настройкой эмулятора на работу с ускорителем можно в статье по ссылке:<http://habrahabr.ru/company/intel/blog/146114/>.

В-третьих, не стоит прерывать процесс запуска виртуального устройства, наберитесь терпения, на старых компьютерах первый запуск *AVD* может занять до 10 минут, на современных - от одной до трех минут. После того, как создали и запустили *виртуальное* устройство имеет смысл оставить его открытym, при всех последующих запусках приложения будет использоваться уже открытое *виртуальное* устройство, что позволит сэкономить время.



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 294 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

19.7 Заключение

В лабораторной работе рассмотрен процесс разработки простого приложения переднего плана. Описано создание активности, настройка интерфейса и реализация логики приложения. Других компонентов в приложении не предусмотрено. В последующих работах будут рассматриваться приложения, содержащие несколько активностей. Смешанные приложения, работающие на переднем плане и при этом поддерживающие сервисы, работающие в фоновом режиме.

Приложение 1.

```
package com.example.projectn;  
import android.os.Bundle;  
import android.app.Activity;  
import android.view.Menu;  
import android.view.View;  
import android.widget.*;  
public class MainActivity extends Activity {  
    TextView tvInfo;  
    EditText etInput;  
    Button bControl;  
    int guess;  
    boolean gameFinished;  
    @Override
```



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 295 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    tvInfo = (TextView)findViewById(R.id.textView1);  
    etInput = (EditText)findViewById(R.id.editText1);  
    bControl = (Button)findViewById(R.id.button1);  
    guess = (int)(Math.random()*100);  
    gameFinished = false;  
}  
  
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    getMenuInflater().inflate(R.menu.main, menu);  
    return true;  
}  
  
public void onClick(View v){  
    if (!gameFinished){  
        int inp=Integer.parseInt(etInput.getText().toString());  
        if (inp > guess)  
            tvInfo.setText(getResources().getString(R.string.ahead));  
        if (inp < guess)  
            tvInfo.setText(getResources().getString(R.string.behind));  
        if (inp == guess)
```

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 296 из 469

Назад

На весь экран

Закрыть

```
{  
    tvInfo.setText(getResources().getString(R.string.hit));  
    bControl.setText(getResources().getString(R.string.play_more));  
    gameFinished = true;  
}  
}  
}  
else  
{  
    guess = (int)(Math.random()*100);  
    bControl.setText(getResources().getString(R.string.input_value));  
    tvInfo.setText(getResources().getString(R.string.try_to_guess));  
    gameFinished = false;  
}  
etInput.setText(  
}  
}
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 297 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

§20. Лабораторная работа №3



Цель лабораторной работы:

Изучение основ разработки интерфейсов мобильных приложений

Задачи лабораторной работы:

- Изучить элементы интерфейса
- Практическим путём научиться размещать элементы и менять их свойства
- Разработать прототип интерфейса собственного приложения

20.1 Введение

Лабораторная работа посвящена разработке интерфейсов мобильных приложений. Работа содержит подробное описание построения гармоничного понятного пользовательского интерфейса для главной активности приложения и описание основных элементов интерфейса. Лабораторная работа поможет выбрать концепцию своего приложения и начать разработку его интерфейса.

Более подробную информацию о разработке интерфейсов мобильных приложений под *Android* можно узнать на сайте [Design \[Android Developers\]](#).

Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 298 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

20.2 Создание прототипа интерфейса

Рассмотрим пример разработки интерфейса приложения, которое ищет в сети *Интернет* изображения по запросу пользователя, позволяет оценивать их, скачивать, и посещать *интернет*-страницы сайтов, на которых было найдено изображение.

Создание заготовки для приложения

Выглядеть главное окно будет примерно так:

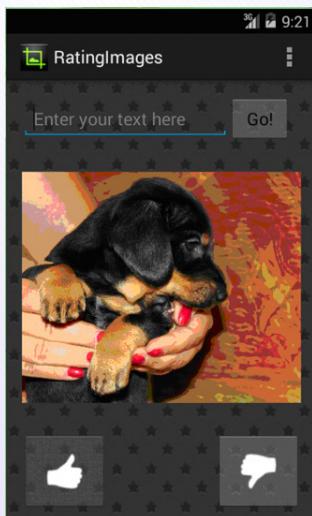


Рисунок 1.Интерфейс главной активности



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 299 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 300 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

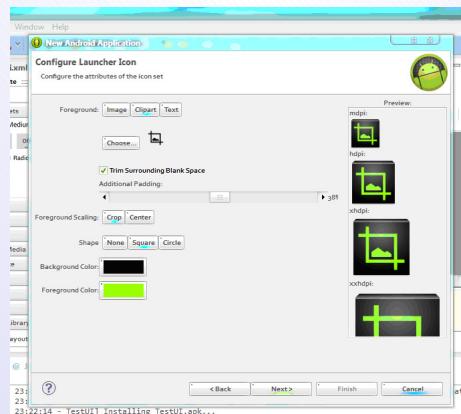


Рисунок 2. Создание иконки

На нём присутствуют поле ввода текста для запроса пользователя и кнопка, начинаящая поиск изображений. Внизу экрана две кнопки: "like" и "dislike" , с их помощью пользователь сможет оценить изображение. После того, как пользователь сделает оценку изображения, текущее изображение закрывается и загружается следующее.

Итак, начнём с создания нового проекта. Назовём его "**RatingImages**" ("Рейтинг изображений").

На данном этапе изучения программирования под Android не обязательно менять иконку запуска, но эта лабораторная работа направлена на разработку дизайна интерфейса, и потому, приступим.

Создайте иконку на свой вкус.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 301 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

После создания проекта откройте **activity_main.xml** из каталога **res/layout/**.

Когда вы откроете файл **activity_main.xml**, вы увидите графический редактор макета. Благодаря этому редактору создание интерфейсов стало ещё интереснее, поскольку добавить элемент на форму можно при помощи перетаскивания мышью, к тому же, благодаря графическому редактору, не обязательно запускать эмулятор, чтобы увидеть результат своих трудов.

Теперь щелкните по вкладке **activity_main.xml** в нижней части экрана. Открылся XML-редактор кода. Этот способ редактирования стандартный, но все изменения, вносимые в этот документ, можно так же ощутить визуально, перейдя на графический редактор.

Вернёмся на вкладку с графическим редактором. Во-первых, подготовим документ к началу работы, для этого удалите <TextView>.

Результат выглядит так:



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 302 из 469

Назад

На весь экран

Закрыть

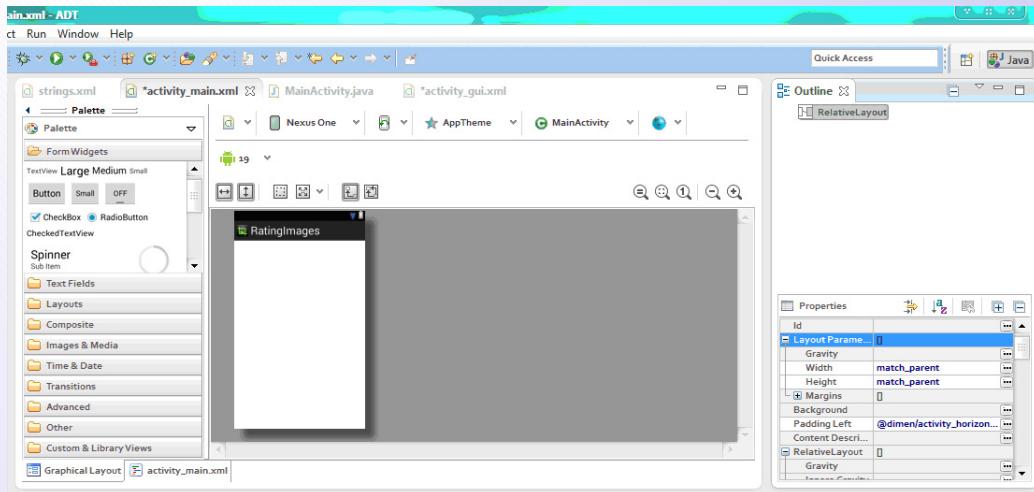


Рисунок 3.Проект, готовый к началу разработки

На рабочей области экрана остался один элемент. Это макет `<RelativeLayout>`. В нём позиция дочерних элементов может быть описана по отношению друг к другу или к родителю. Подробнее о макетах можно узнать [здесь](#).

Два атрибута, ширина и высота (`android:layout_width` и `android:layout_height`), требуются для всех элементов для того, чтобы указать их размер.

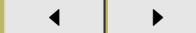
Так как `<RelativeLayout>` - это корень в макете, то нужно, чтобы он заполнял всю область экрана. Это достигается при помощи установки параметра "`match_parent`" для ширины и высоты. Это значение ука-



Кафедра ПМиИ

Начало

Содержание



Страница 303 из 469

Назад

На весь экран

Закрыть

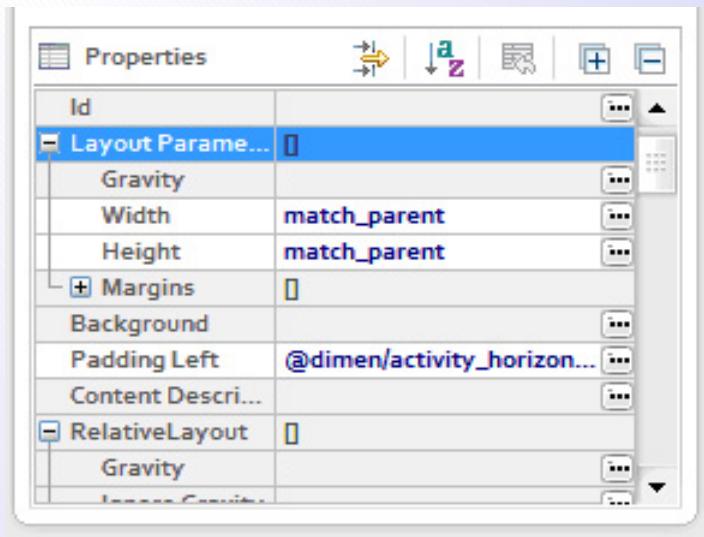


Рисунок 4. Свойства <RelativeLayout> элемента

Как установить это значение? Будем разбираться.

Однократным щелчком левой кнопкой мыши по надписи "**Width**" активируйте строку с параметрами ширины:



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 304 из 469

Назад

На весь экран

Закрыть

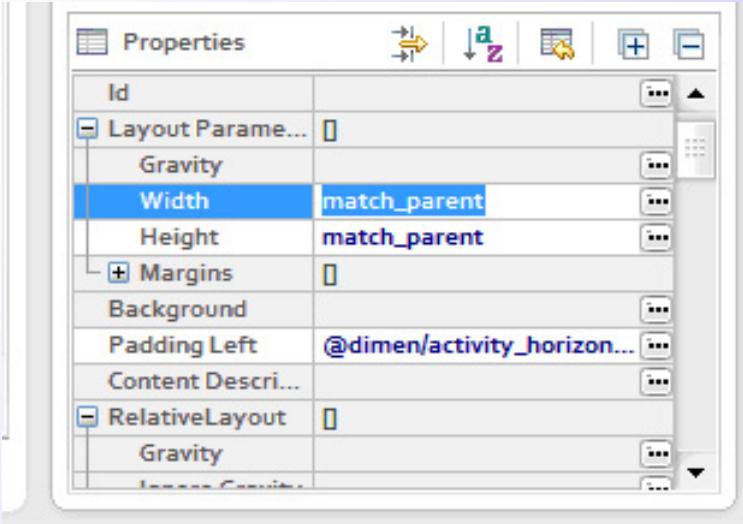


Рисунок 5. Свойства <RelativeLayout> элемента, выбран атрибут android:layout_width

Щелчком левой кнопки мыши по области ввода вызовите диалоговое окно, и двойным щелчком сделайте выбор параметра:



Кафедра ПМиИ

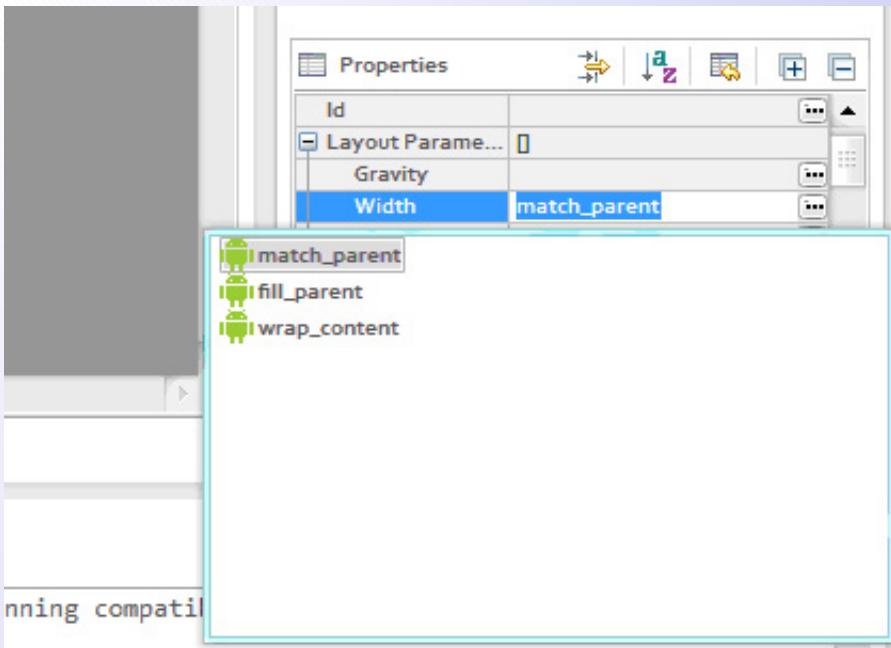


Рисунок 6. Свойства <RelativeLayout> элемента, выбран параметр "match_parent"

Начало

Содержание

◀ ▶

◀ ▶

Страница 305 из 469

Назад

На весь экран

Закрыть

Или щелчком правой кнопки мыши по <RelativeLayout> в Outline:

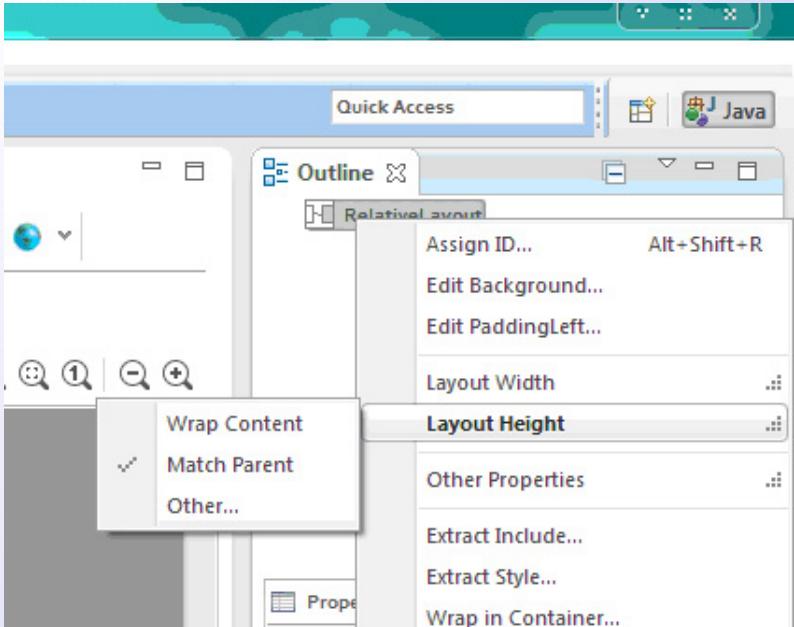


Рисунок 7.Контекстное меню <RelativeLayout> элемента

При выполнении первого способа вы увидели еще два возможных параметра: "fill_parent" и "wrap_content".

На самом деле, `match_parent = fill_parent`, но "fill_parent" считается устаревшим, и к использованию в новых проектах предлагается "match_parent".

Кафедра
ПМиИ

Начало

Содержание



◀ ▶

Страница 306 из 469

Назад

На весь экран

Закрыть





Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 307 из 469

Назад

На весь экран

Закрыть

Параметр "wrap_content" указывает, что представление будет увеличиваться при необходимости, чтобы поддерживать соответствие содержанию экрана.

Добавление текстового поля

Для начала добавьте элемент <LinearLayout> с горизонтальной ориентацией в <RelativeLayout>, и укажите для ширины и высоты параметр "wrap_content". Теперь, для создания пользовательского редактируемого текстового поля, добавьте элемент <EditText> с параметром "wrap_content" для ширины и высоты в <LinearLayout>.

Сейчас должно получиться примерно следующее:

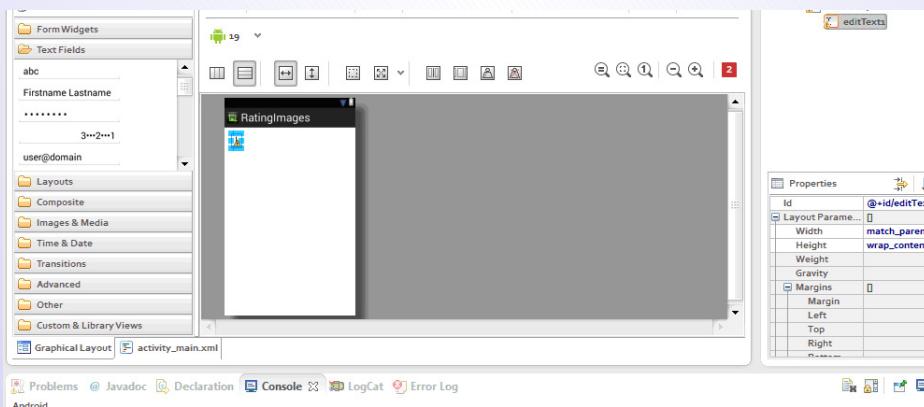


Рисунок 8. Добавление текстового поля



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 308 из 469

Назад

На весь экран

Закрыть

Возможно, появился желтый предупреждающий знак, но сейчас это не важно, со временем он исчезнет. Наличие таких предупреждений никак не влияет на компилируемость проекта.

Теперь переходим к настройке добавленных нами элементов.

Для многих элементов нужно назначать **id**, он обеспечивает уникальный идентификатор, который можно использовать как ссылку на объект из кода вашего приложения для управления им.

Откроем редактор кода XML-файла и обратим внимание на элемент

<EditText>:

<EditText

android:id="@+id/editText1"

android:layout_width="wrap_content"

android:layout_height="wrap_content" >

<requestFocus/>

</EditText>

Строка **<requestFocus/>** появляется добавлением элемента **requestFocus** (папка **Advanced**), и позволяет установить фокус на нужном компоненте. Важно использовать этот элемент, когда у вас имеется, к примеру, три текстовых поля, и нужно, чтобы фокус был на втором из них.

При указании **id**, знак (@) требуется в том случае, если вы имеете в виду любой ресурс объекта из XML-файла. За ним следуют тип ресурса (в данном случае **ID**), косая черта (слеш) и имя ресурса (**editText1**).



Знак плюс (+) перед типом ресурсов необходим только тогда, когда вы впервые определяете идентификатор ресурса.

По сути, **id**, который создается автоматически, уже уникален, но грамотнее переименовывать **id** в соответствии со назначением элемента.

Зададим **id** для текстового поля. Для этого прямо в коде строку `android:id="@+id/editText1"` заменяем на `android:id="@+id/edit_message"`, жмём **CTRL+S** и открываем графический редактор. Если всё хорошо, то в свойствах текстового поля в графе **id** будет следующее:

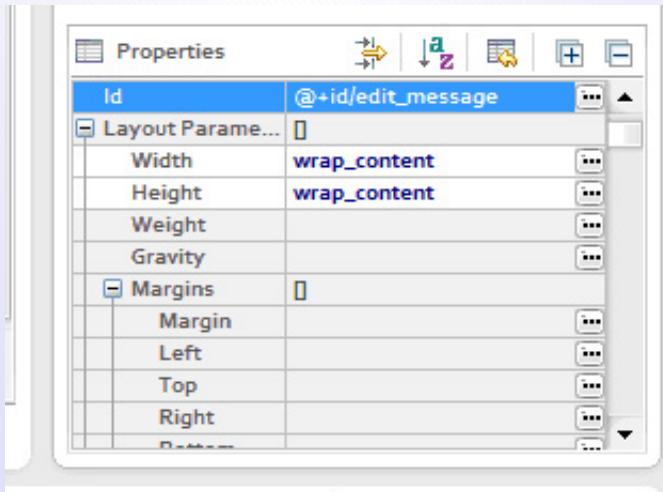


Рисунок 9.Идентификатор текстового поля

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 309 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 310 из 469

Назад

На весь экран

Закрыть

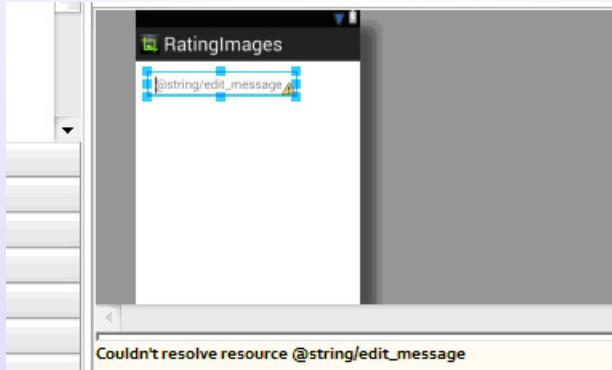


Рисунок 10. Предупреждение: не найден ресурс, на который прописана ссылка



Кафедра ПМиИ

Начало

Содержание

◀ ▶

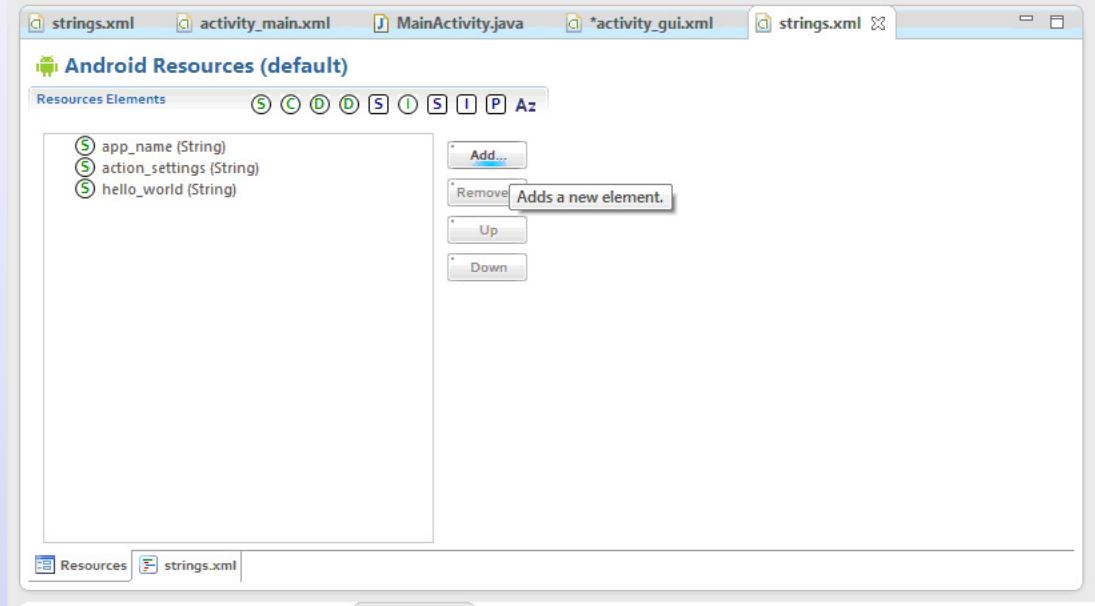
◀ ▶

Страница 311 из 469

Назад

На весь экран

Закрыть



Для того, чтобы ссылка на ресурс начала работать, нужно этот ресурс создать.

Откройте файл **res/values/strings.xml**. Очевидно, что его тоже можно редактировать двумя способами: графически и вручную.

Выбирайте тот способ, который больше по душе.

Рисунок 11. Редактирование файла ресурсов графическим способом; добавление нового ресурса

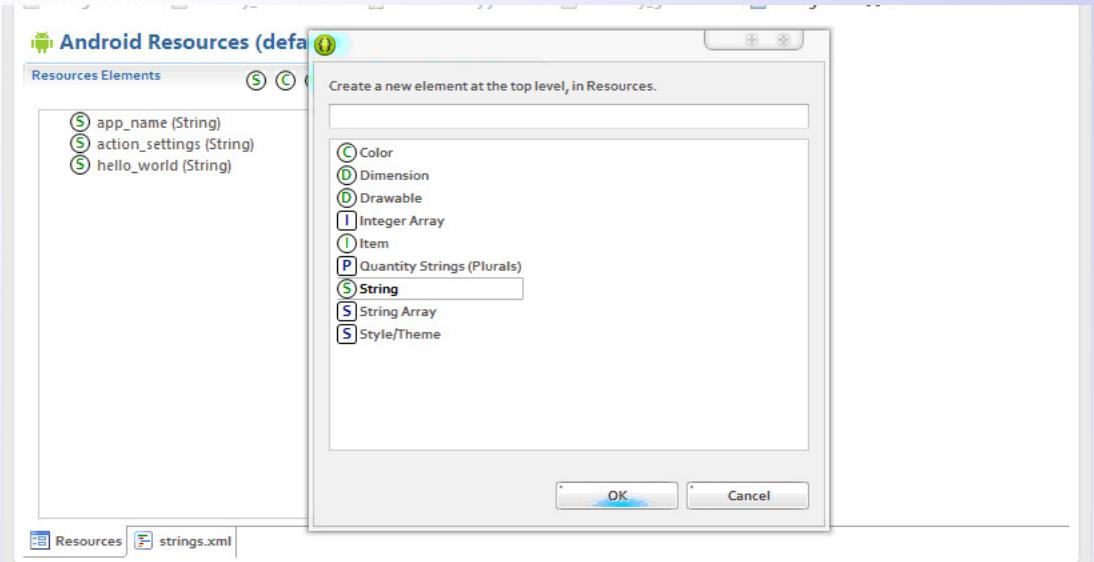


Рисунок 12. Редактирование файла ресурсов графическим способом; выбор типа ресурса, выбран тип String

Заполняем поля "Имя" и "Значение":



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶ ▶▶

Страница 312 из 469

Назад

На весь экран

Закрыть

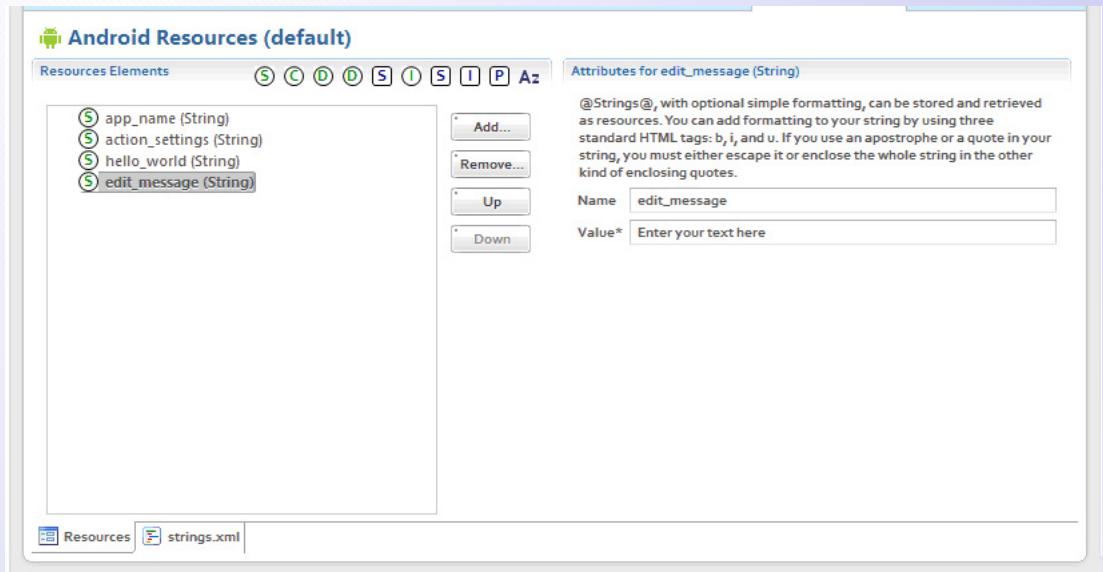


Рисунок 13.Редактирование файла ресурсов графическим способом

Сохраняем и любуемся результатом:



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 313 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 314 из 469

Назад

На весь экран

Закрыть



Рисунок 14. Отображение текста "По умолчанию" в поле ввода

Подробнее о ресурсах - [здесь](#)

Аналогично создадим id для <LinearLayout>

android:id="@+id/linear1"

Сохраните изменения.

Добавление кнопки

Теперь добавьте <Button> в макет после элемента <EditText>:



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 315 из 469

Назад

На весь экран

Закрыть

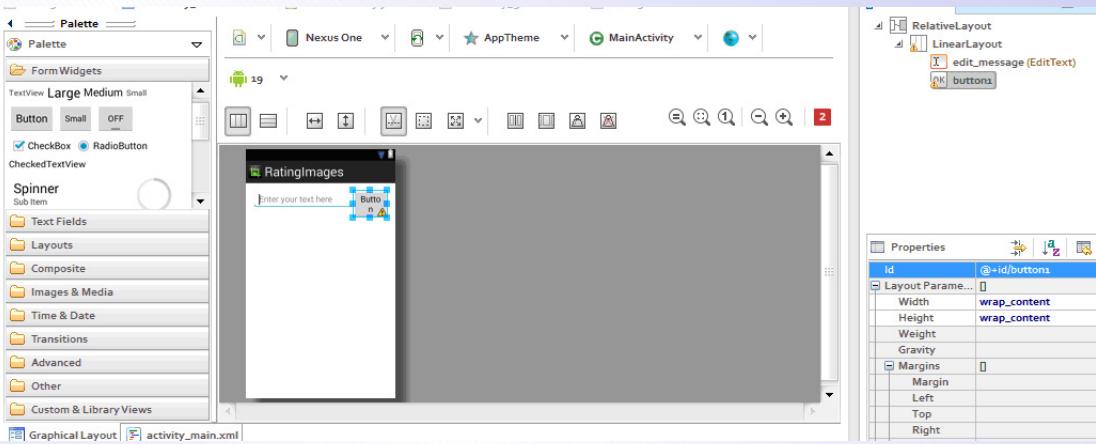


Рисунок 15.Новая кнопка

Чтобы кнопка трансформировалась в соответствии с текстом кнопки, ширина и высота должны быть установлены во "`wrap_content`" .

Теперь поменяем надпись на кнопке на "**Go!**" с помощью ссылки на ресурс в XML-коде главной активности и добавления одного ресурса в файл **strings.xml**:



```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">RatingImages</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="edit_message">Enter your text here</string>
    <string name="button_send">Go!</string>

</resources>
```

Рисунок 16.Редактор XML-кода файла strings.xml

Сохраните.

На графическом редакторе главной активности будут изменения:

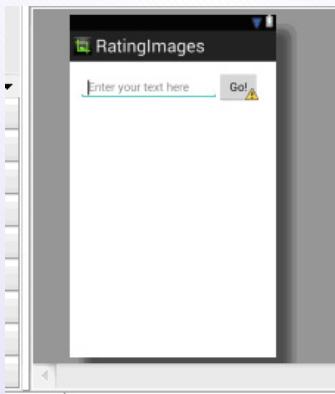


Рисунок 17.Кнопка приняла новую форму, в соответствии с надписью на ней

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 316 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 317 из 469

Назад

На весь экран

Закрыть

Теперь, когда мы поместили два главных представления на `<LinearLayout>` элемент, настало время добавить ещё два параметра для этого элемента.

Речь идет о "приращении" правого и левого краёв лейаута к правому и левому краям `<RelativeLayout>` элемента соответственно. А сделать это проще простого - просто потяните мышкой один край к другому!

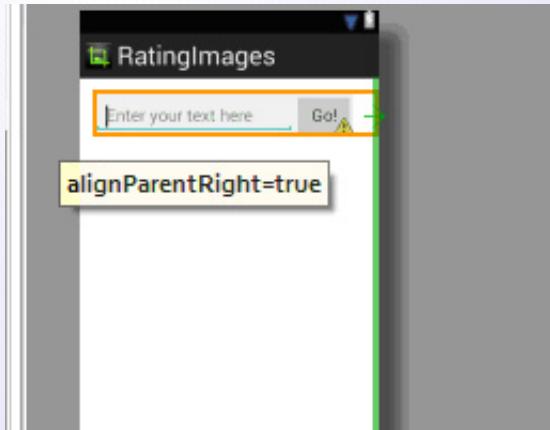


Рисунок 18."Приращение" правого края

Зелёные стрелки по краям дают понять, к какому элементу удалось прицепить край:



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 318 из 469

Назад

На весь экран

Закрыть

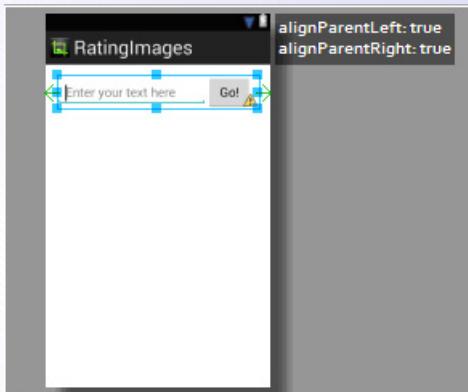


Рисунок 19.<LinearLayout> после выравнивания

И, конечно, это отобразится в свойствах:

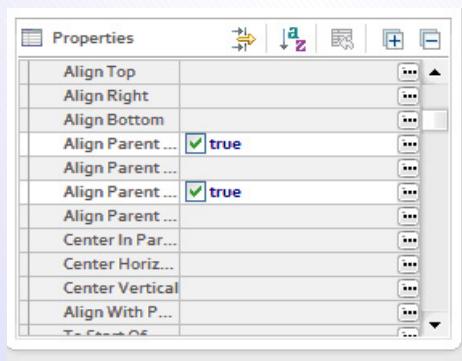


Рисунок 20.Свойства <LinearLayout>



Кафедра ПМиИ

Начало

Содержание



Страница 319 из 469

Назад

На весь экран

Закрыть

Смена фона

Идём дальше - попробуем поменять фон.

Чтобы изменить цвет фона на чёрный, нужно в XML-коде главной активности написать одну строку в блоке `<RelativeLayout>` элемента:
`android:background="#000000".`

Сохраните и проверьте результат, открыв графический редактор.

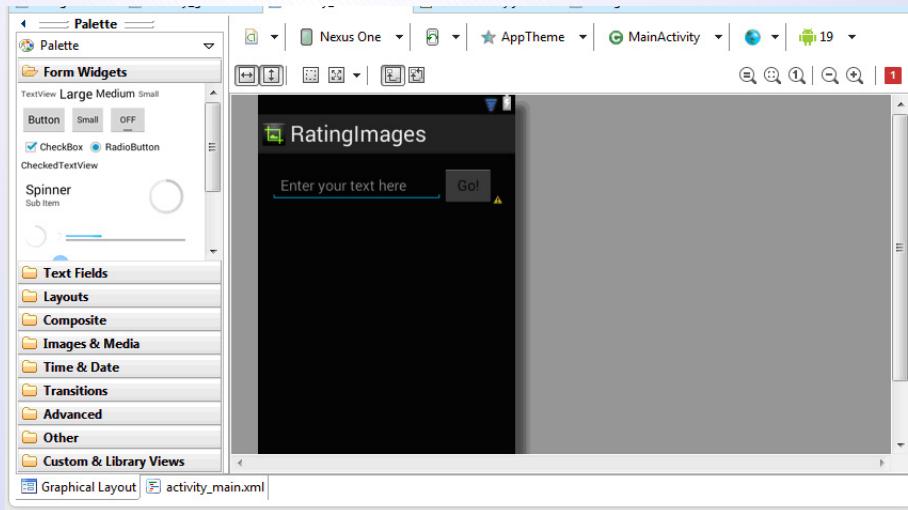


Рисунок 21.Фон стал чёрным

Красиво, но скучно. Как сделать фон ещё интереснее? Поместить на него рисунок!

Для этого сначала в папке **res/** создадим папку **drawable/**

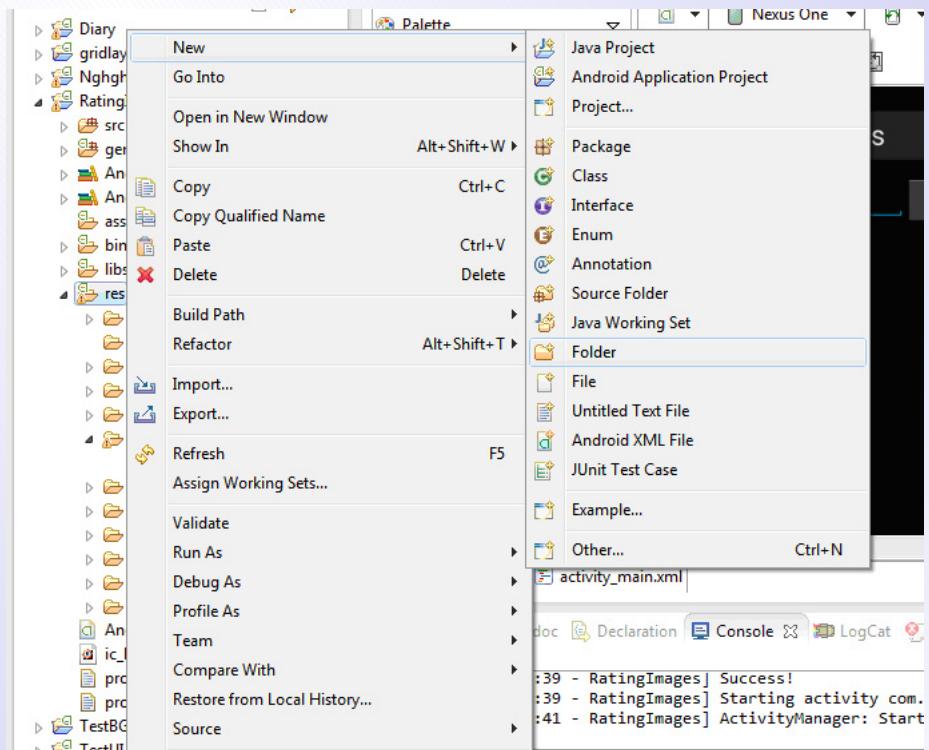


Рисунок 22.Создание папки

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 320 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание



Страница 321 из 469

Назад

На весь экран

Закрыть

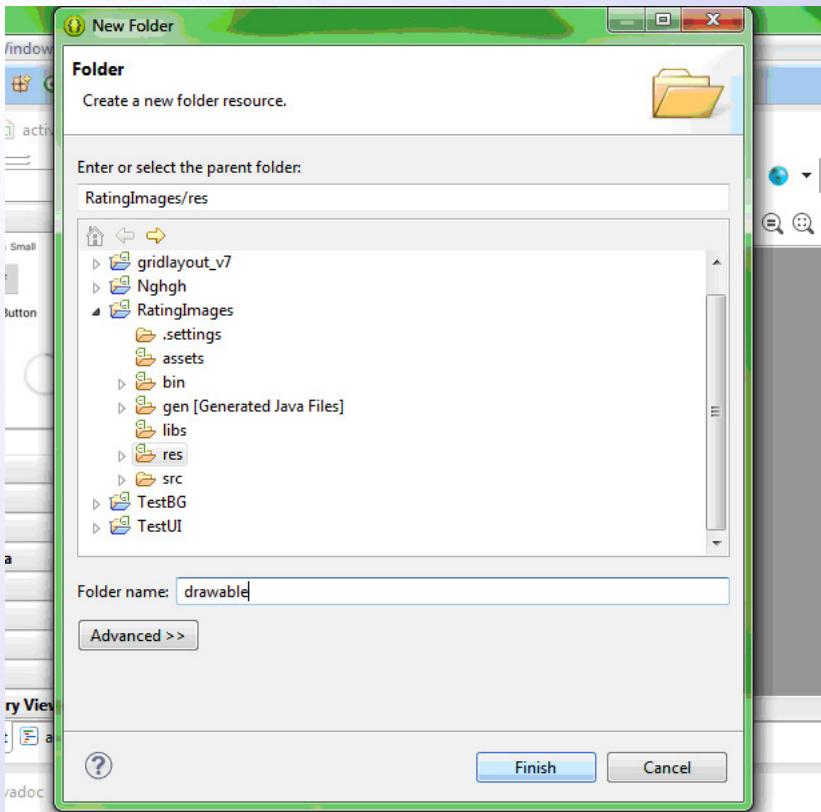


Рисунок 23. Создание папки: имя папки

После того, как папка создана, нужно положить в эту папку изображение - картинка называется **got.png**:



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 322 из 469

Назад

На весь экран

Закрыть

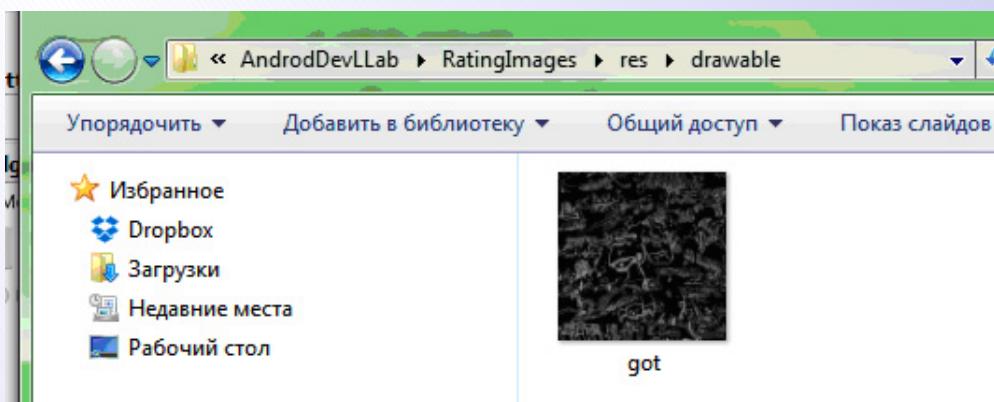


Рисунок 24. Изображение в папке drawable/

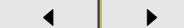
После этого в папке **drawable/** нужно создать файл **background.xml**, важно при создании выбрать параметр **bitmap**.



Кафедра ПМиИ

Начало

Содержание



Страница 323 из 469

Назад

На весь экран

Закрыть

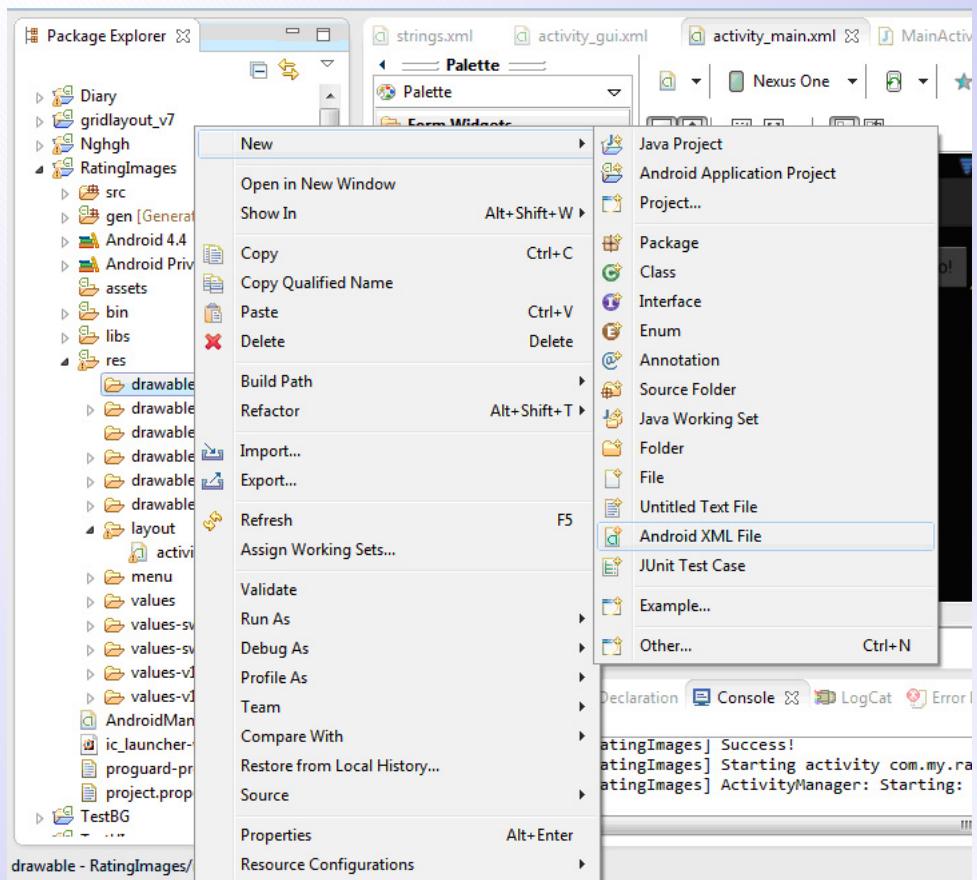


Рисунок 25. Создание нового XML-файла



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 324 из 469

Назад

На весь экран

Закрыть

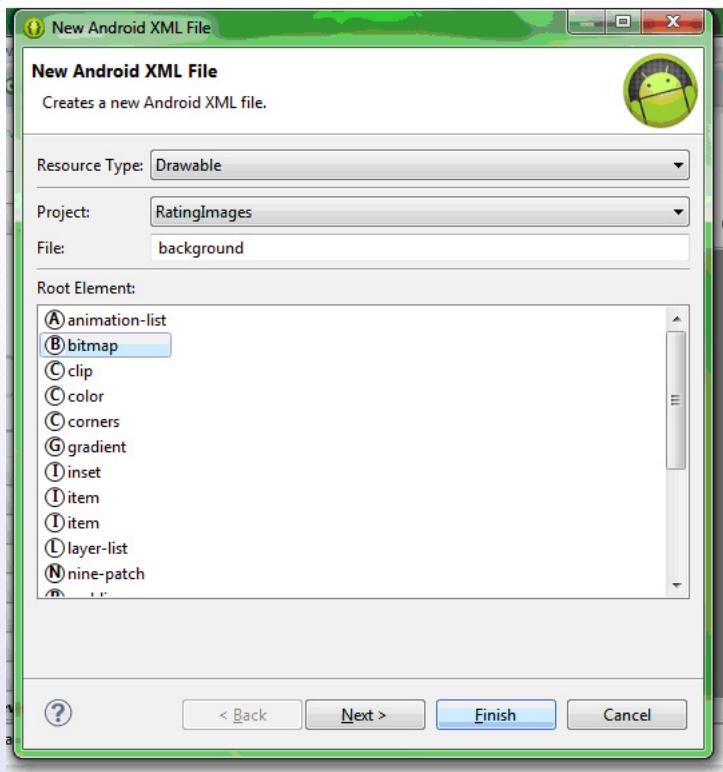


Рисунок 26. Создание нового XML-файла

Как только новый файл открылся, пропишем в него одну строчку, с указанием на то, откуда и какой файл использовать:

```
<?xml version="1.0" encoding="utf-8"?>  
<bitmap xmlns:android="http://schemas.android.com/apk/res/android"
```

```
        android:src="@drawable/got"  
    
```

Вернемся в редактор XML-кода, туда, где прописывали цвет фона. Вместо строки `android:background="#000000"` напишем ссылку на XML-файл `android:background="@drawable/background"`.

Сохраняем и видим результат:

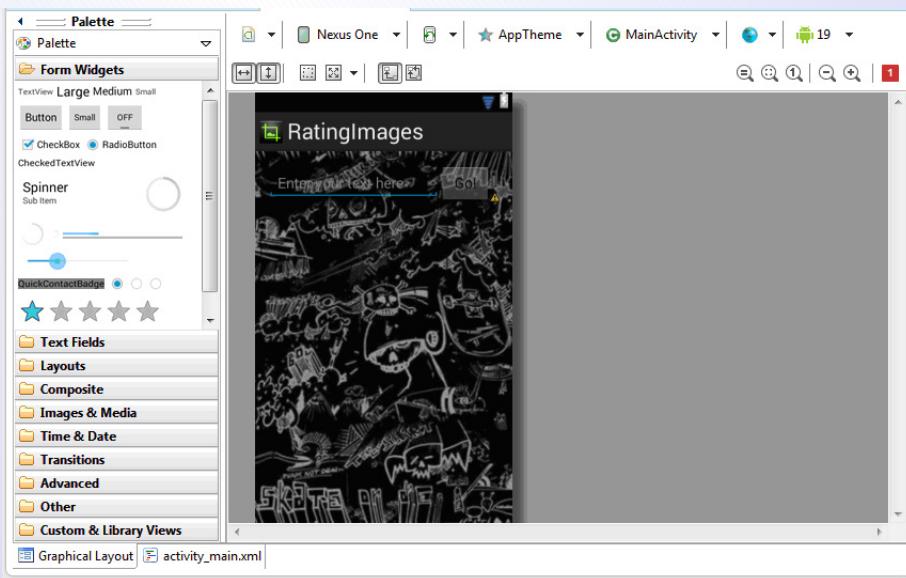


Рисунок 27.Новый фон

Несомненно, фон смотрится хорошо, но очевидно, что кнопка и поле ввода просто затерялись, а это значит, что для этого приложения такой



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 325 из 469

Назад

На весь экран

Закрыть

фон не подходит. Можно продолжить подбирать изображения на фон, но лучше создать черепичную заливку небольшим изображением. На **этом** сайте можно найти узор на любой вкус!

Когда вы выбрали узор и скачали его, скопируйте изображение в папку **drawable/**.

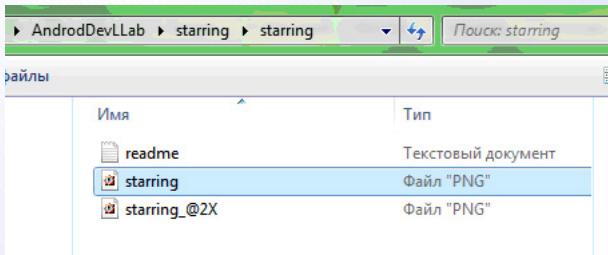


Рисунок 28.Копирование изображения

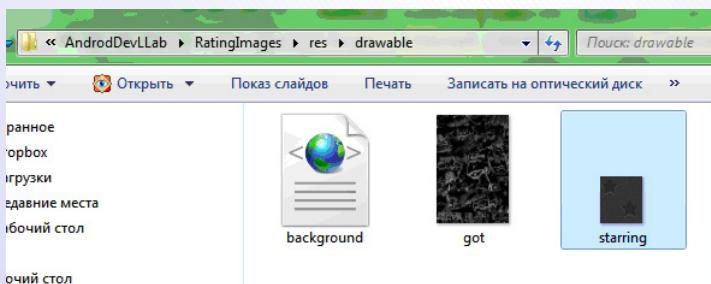


Рисунок 29.Скопированное изображение



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 326 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 327 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

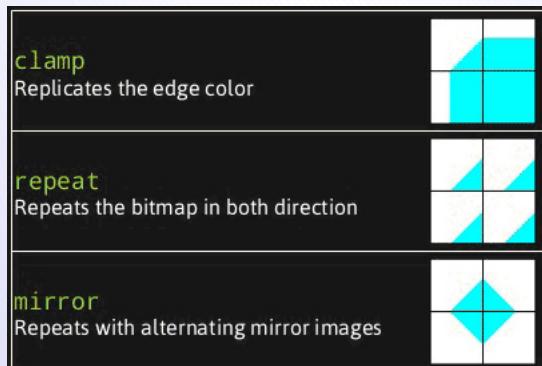


Рисунок 30. Варианты заполнения

Настало время посмотреть, что из этого получилось:

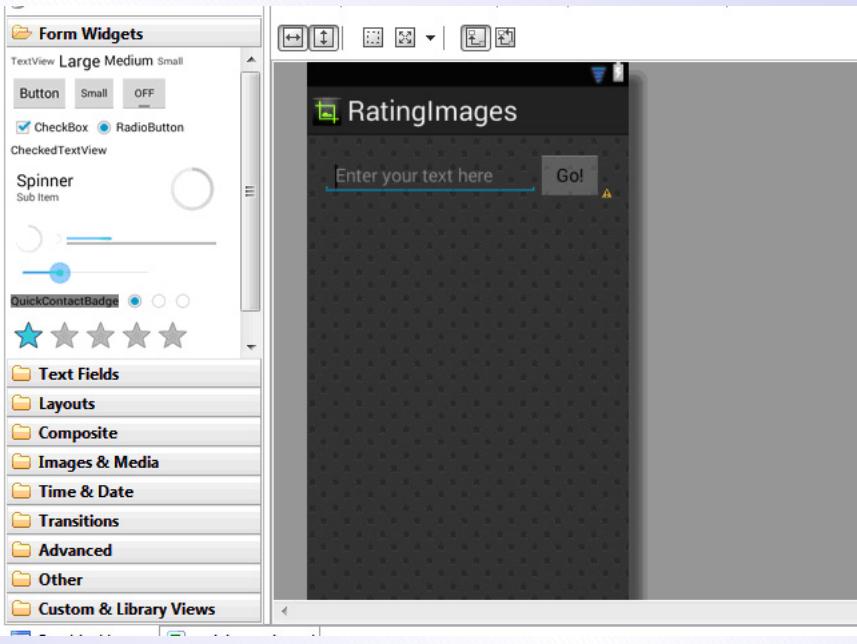


Рисунок 31.Фон из звездочек

В `<RelativeLayout>` стоит добавлять `android:background` только в том случае, если вы хотите неподвижный фон, а в `<ScrollView>` чтобы фон прокручивался вместе с контентом.

Если вы выбрали тёмный фон, то стоит поменять цвет текста, вводимого в поле ввода, например на белый.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 328 из 469

Назад

На весь экран

Закрыть

Для этого в блок <EditText> добавим строчку
android:textColor="#fffff"

Область просмотра изображений

Теперь займемся созданием области отображения изображений, которые пользователь будет оценивать.

Добавьте "рамку" <FrameLayout> в <RelativeLayout>:

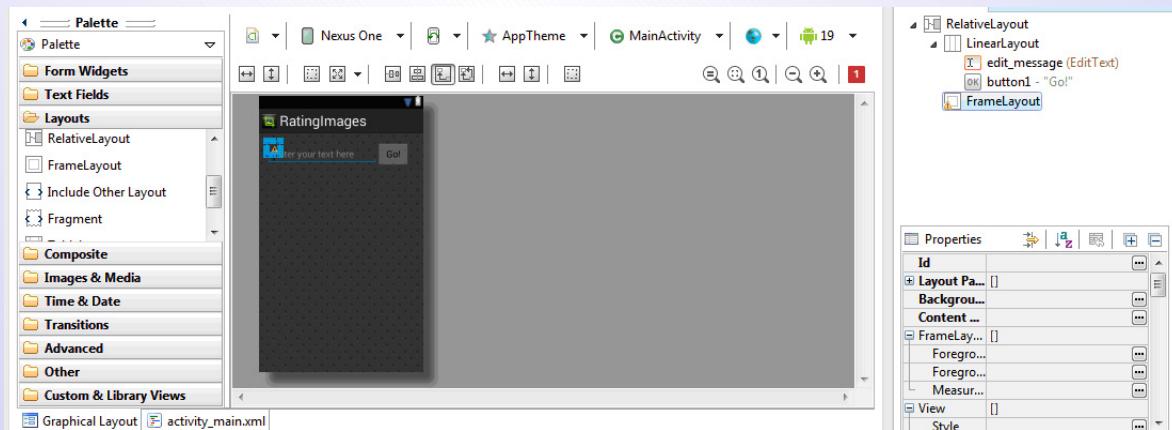


Рисунок 32.Добавление "рамки"

Обратите внимание, что предупреждающий жёлтый треугольник исчез, но появился у нового элемента, и это значит, что всё идёт по плану.



Кафедра
ПМиИ

Начало

Содержание



Страница 329 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание



Страница 330 из 469

Назад

На весь экран

Закрыть

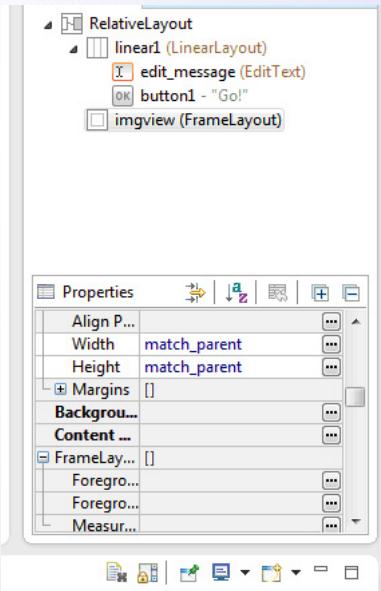


Рисунок 33. Свойства "рамки"

Укажем этому элементу ширину, высоту и id. Предупреждение должно пропасть.

Сейчас **FrameLayout** "заезжает" на поле ввода и кнопку. Чтобы исправить это, нужно задать для "рамки" параметр, который будет следить, чтобы "рамка" находилась ниже, чем поле ввода и кнопка.

Выберите из списка **Outline** "рамку" и кликом правой кнопки мыши



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 331 из 469

Назад

На весь экран

Закрыть

вызовите контекстное меню: Other Properties -> Layout Parametrs
-> Layout Below...

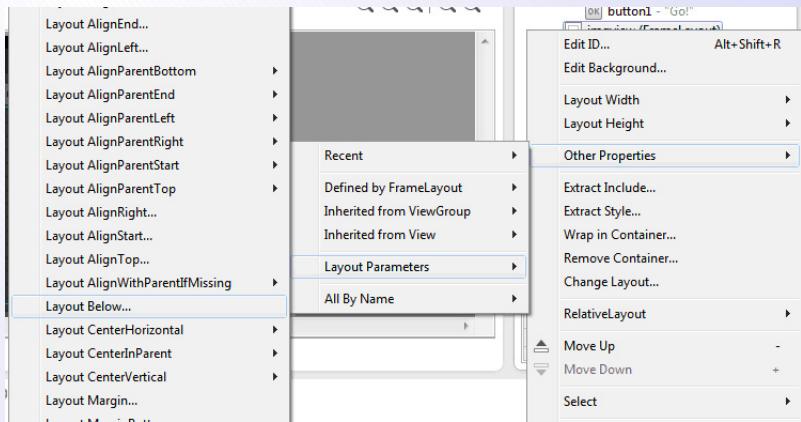


Рисунок 34.Контекстное меню "рамки"

Появится окно. В нём выбираем из списка "ID" имя лейаута, на котором находятся поле ввода и кнопка:



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 332 из 469

Назад

На весь экран

Закрыть

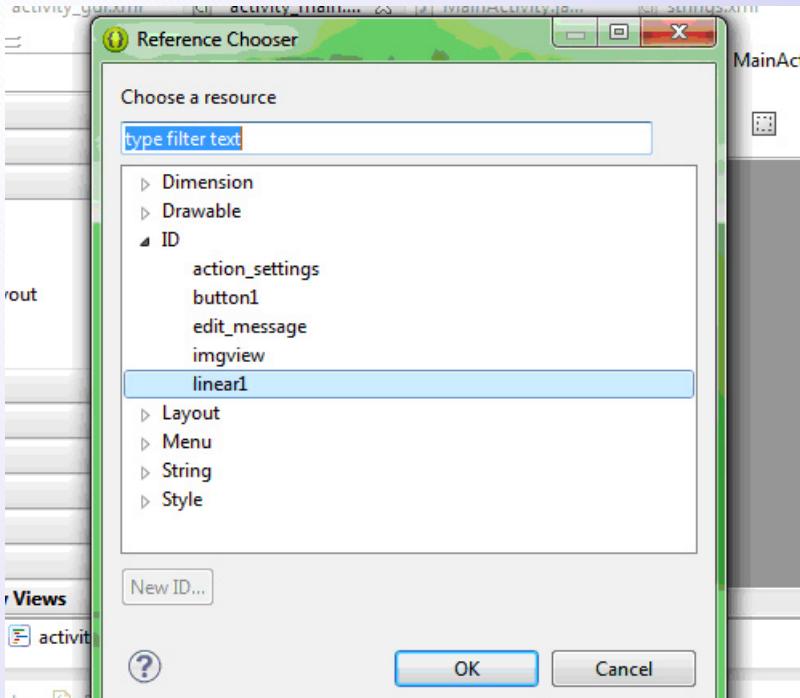


Рисунок 35. Выбор элемента, ниже которого должна быть "рамка"

Жмём **OK** и "рамка" примет вид:

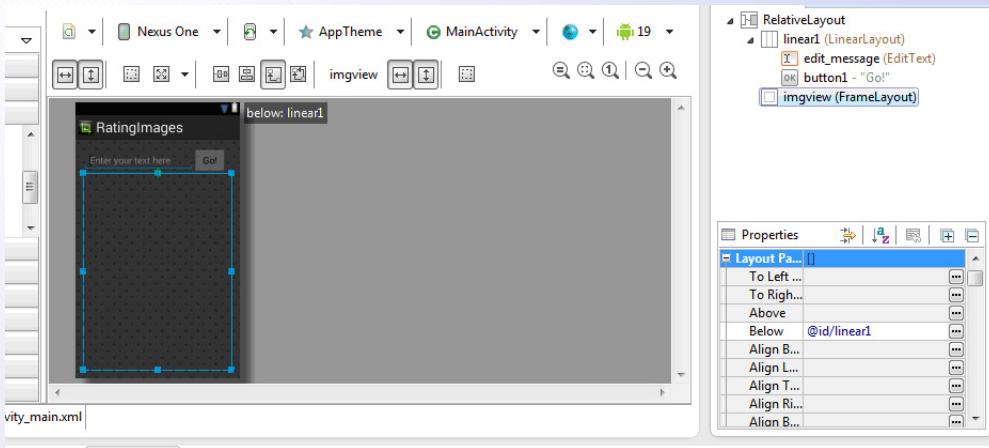


Рисунок 36.Новое расположение "рамки"

Теперь для наглядности поместим туда изображение.
Сначала поместим само изображение в папку **res/drawable/** и обновим её.
Это нужно для того, чтобы новые данные загрузились в проект.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 333 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

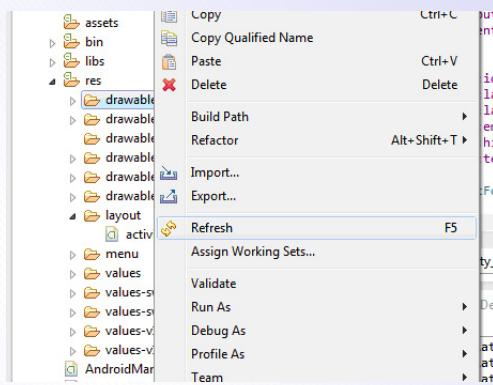


Рисунок 37.Обновление папки

Теперь поместим внутрь "рамки" <ImageView>

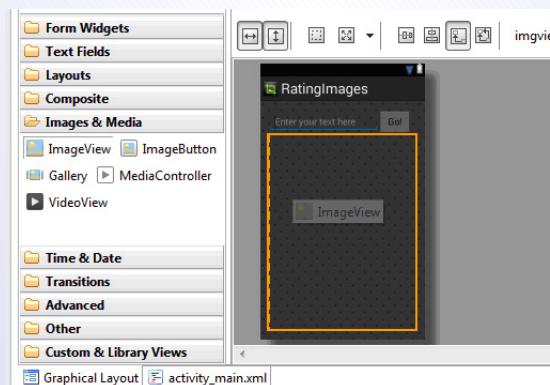


Рисунок 38.Перемещение <ImageView>

Начало

Содержание

◀ ▶

◀ ▶

Страница 334 из 469

Назад

На весь экран

Закрыть

В появившемся окне выбираем нужное нам изображение и жмём **OK**

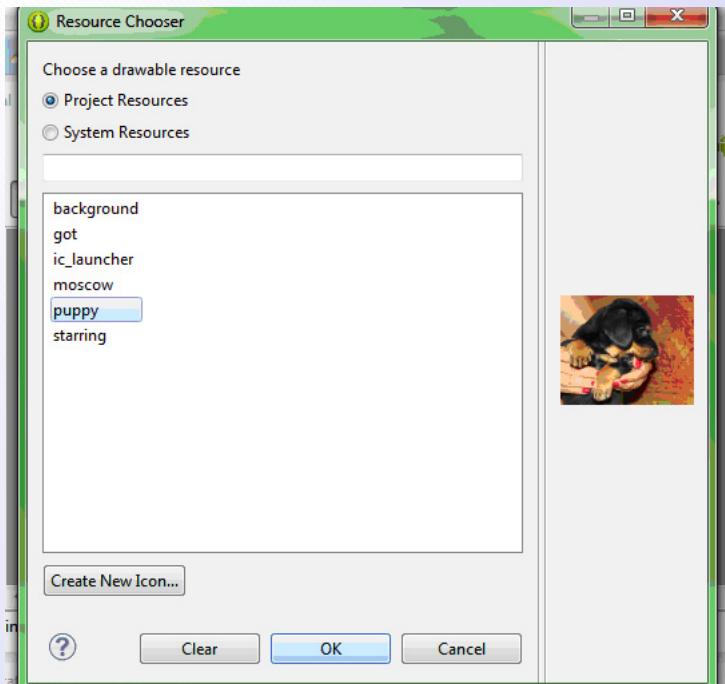


Рисунок 39. Выбор файла

Изображение должно появится на активности:



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 335 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание



Страница 336 из 469

Назад

На весь экран

Закрыть

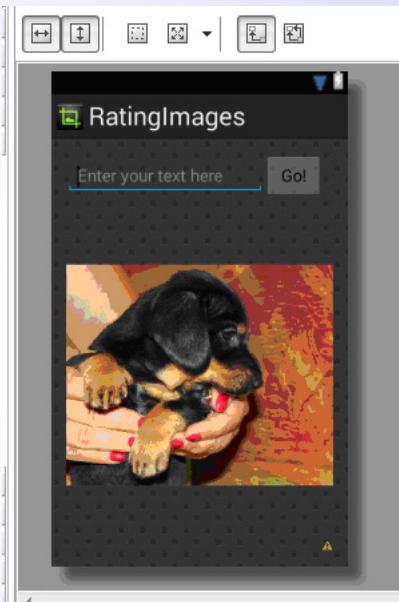


Рисунок 40. Визуализация изображения

Вместе с изображением появился новый предупреждающий знак.

В описании предупреждения сказано: "**[Accessibility] Missing contentDescription attribute on image**". Давайте разберёмся, что это означает.

Атрибут **android:contentDescription** используется в программах, которые поддерживают специальные возможности для людей с нарушениями зрения и слуха. То есть, текст, прописанный в этом атрибуте, не



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 337 из 469

Назад

На весь экран

Закрыть

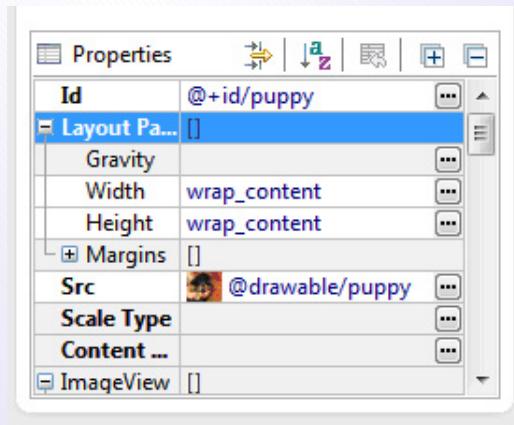


Рисунок 41. Свойства изображения

будет показан на экране, но при включенной в "Специальных возможностях" услуге "звуковые подсказки" этот текст будет проговариваться, когда пользователь переходит к этому элементу управления.

Этот атрибут можно задать для **ImageButton**, **ImageView** и **CheckBox**, и задавать его или нет - дело каждого.

Свойства у изображения должны быть следующими:

Кнопки "like" и "dislike"

Пришло время создать кнопки оценивания.



Для этого добавьте на форму <RelativeLayout>, задайте для него ширину и высоту wrap_content, и укажите id.

Снова в папку res/drawable/ нужно добавить файлы. Найдите изображения "Палец вверх" и "Палец вниз" , и поместите их в эту папку, после чего обновите её.

Изображения и другие полезные файлы можно скачать [здесь](#).

Добавьте <ImageButton>, выберите изображение "Палец вверх" , и переместите <RelativeLayout> в такое положение:

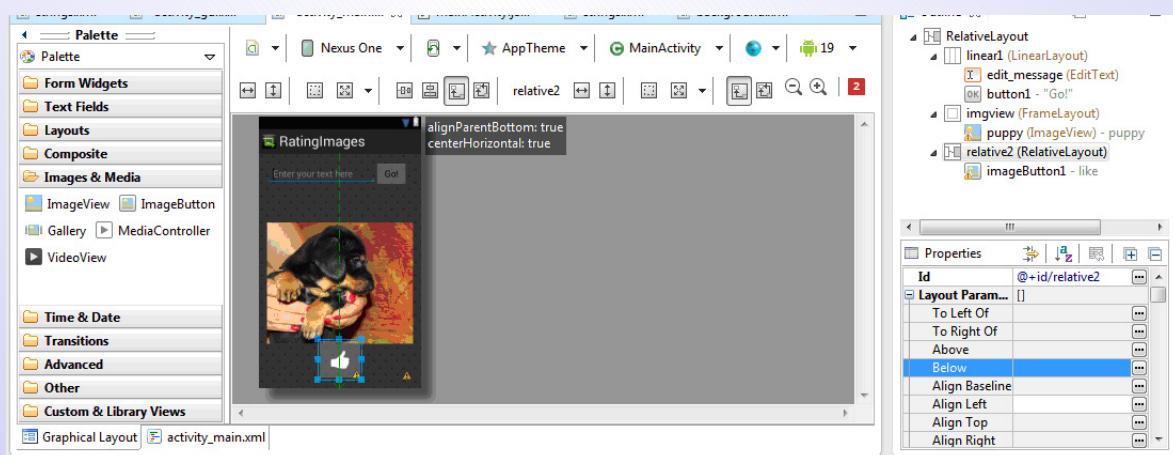


Рисунок 42.Новая кнопка

Укажите "рамке" быть выше, чем поле с кнопкой оценки:

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 338 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание



Страница 339 из 469

Назад

На весь экран

Закрыть

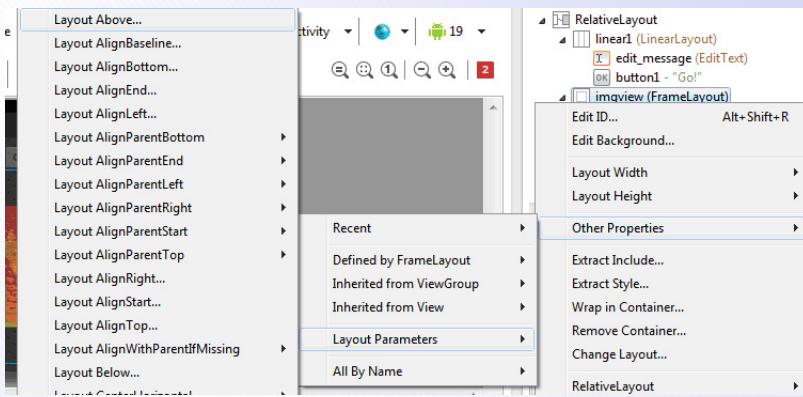


Рисунок 43.Контекстное меню "рамки"

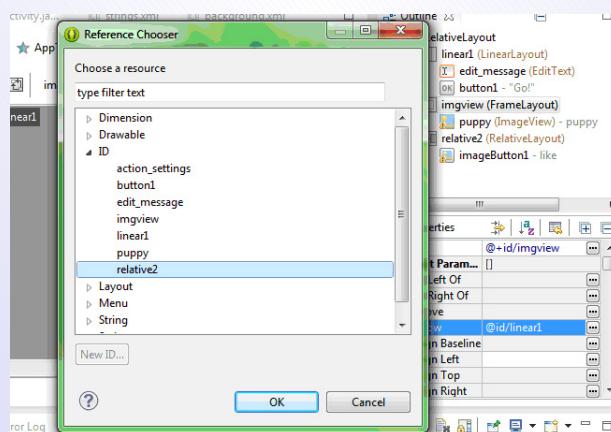


Рисунок 44.Выбор элемента, выше которого должна быть "рамка"



Кафедра
ПМиИ

Начало

Содержание



Страница 340 из 469

Назад

На весь экран

Закрыть

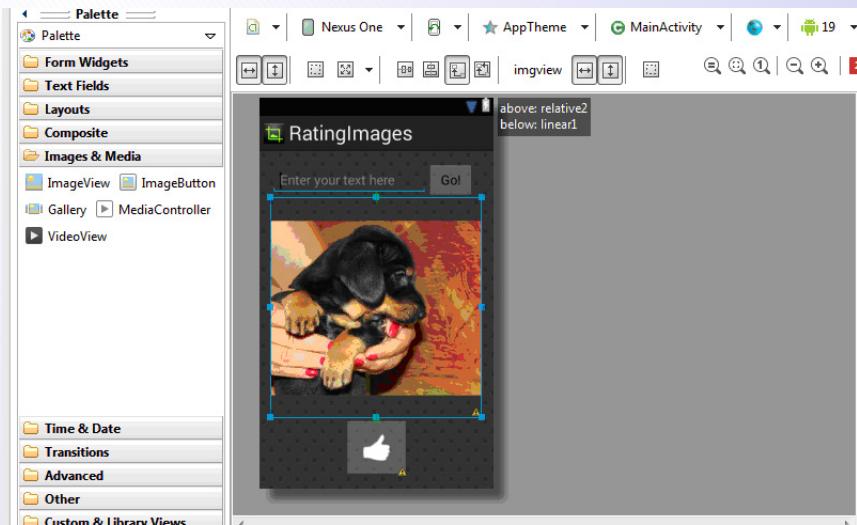


Рисунок 45.Новое расположение элементов

Добавьте еще одну кнопку - кнопку "**Палец вниз**". Она "наложилась" на первую кнопку. Чтобы это исправить, проделайте с `<RelativeLayout>` то же самое, что и с `<LinearLayout>`: растяните элемент влево и вправо, до получения такого результата:

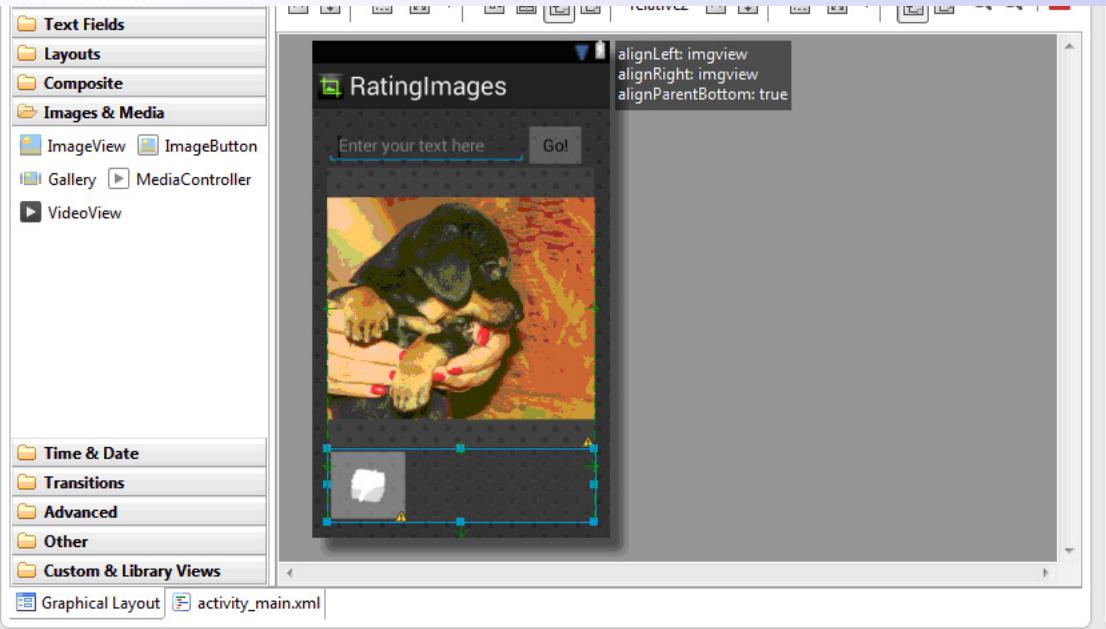


Рисунок 46.Выравнивание области с кнопками

Теперь расставьте кнопки по краям так, чтобы они "прикрепились" к краям.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 341 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶ ▶▶

Страница 342 из 469

Назад

На весь экран

Закрыть

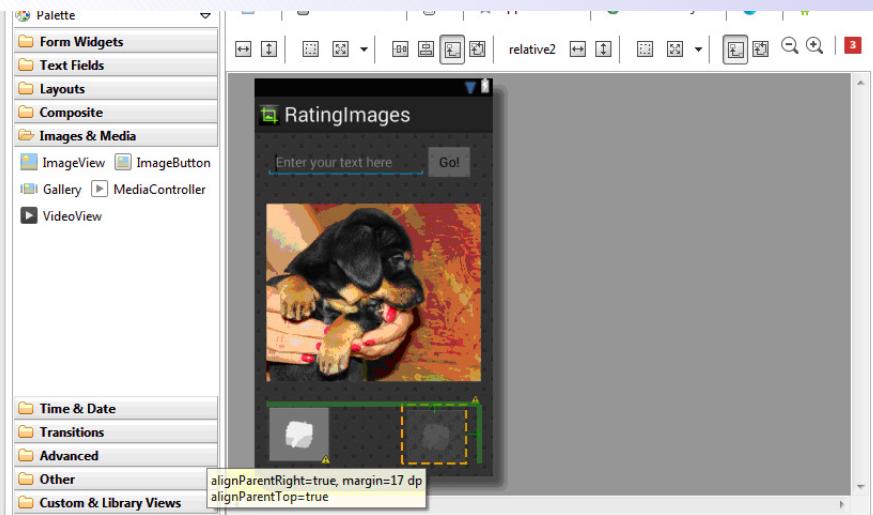


Рисунок 47.Выравнивание кнопок

Готово! Теперь можно запустить эмулятор и посмотреть, что получилось.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 343 из 469

Назад

На весь экран

Закрыть

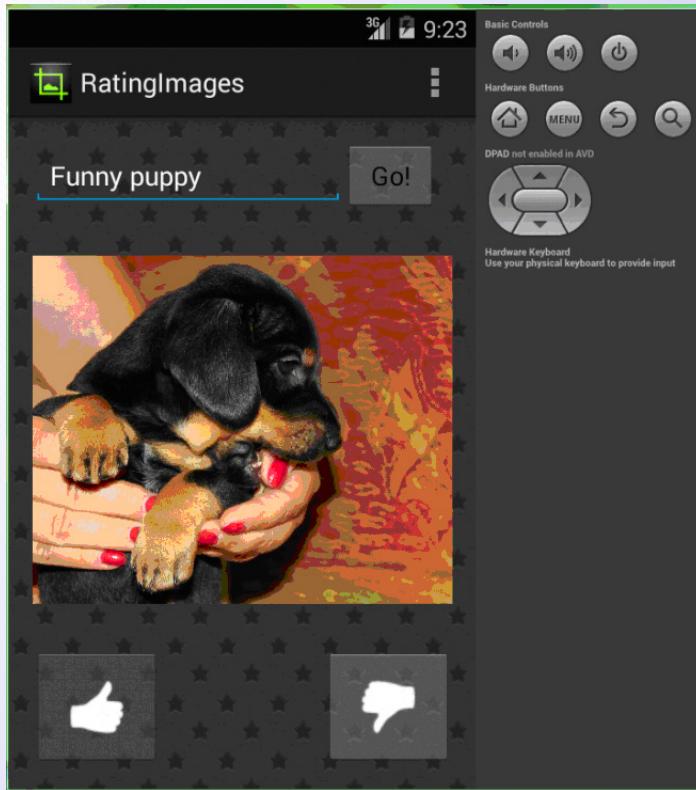


Рисунок 48.Главная активность

Листинги

1)

```
<RelativeLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:paddingBottom="@dimen/activity_vertical_margin"  
    android:paddingLeft="@dimen/activity_horizontal_margin"  
    android:paddingRight="@dimen/activity_horizontal_margin"  
    android:paddingTop="@dimen/activity_vertical_margin"  
    tools:context=".MainActivity"  
    android:background="@drawable/background" >  
    <LinearLayout  
        android:id="@+id/linear1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignParentLeft="true"  
        android:layout_alignParentRight="true"  
        android:orientation="horizontal" >  
        <EditText  
            android:id="@+id/edit_message"
```



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 344 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ems="10"
    android:hint="@string/edit_message"
    android:textColor="#fffff" >
<requestFocus />
</EditText>
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
/>
</LinearLayout>
<FrameLayout
    android:id="@+id/imgview"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_above="@+id/relative2"
    android:layout_below="@+id/linear1" >
<ImageView
    android:id="@+id/puppy"
    android:layout_width="wrap_content"
```

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 345 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

```
    android:layout_height="match_parent"
    android:src="@drawable/puppy"
/>
</FrameLayout>
<RelativeLayout
    android:id="@+id/relative2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/imgview"
    android:layout_alignParentBottom="true"
    android:layout_alignRight="@+id/imgview" >
<ImageButton
    android:id="@+id/imageButton2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:src="@drawable/dislike"
/>
<ImageButton
    android:id="@+id/imageButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 346 из 469

Назад

На весь экран

Закрыть



```
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:src="@drawable/like"
/>
</RelativeLayout>
</RelativeLayout>
```

2)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name" >RatingImages</string>
    <string name="action_settings" >Settings</string>
    <string name="hello_world" >Hello world!</string>
    <string name="edit_message" >Enter your text here</string>
    <string name="button_send" >Go!</string>
</resources>
```

3)

```
<?xml version="1.0" encoding="utf-8"?>
<bitmap
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:src="@drawable/starring"
    android:tileMode="repeat" >
</bitmap>
```

Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 347 из 469

Назад

На весь экран

Закрыть

20.3 BuildingBlocks или элементы для построения интерфейса

Среда разработки "AndroidIDE" открывает широкий выбор готовых к использованию элементов для создания выдающихся приложений.

Кнопка (Button) состоит из текста и/или изображения, которые дают понять пользователю, что произойдёт, если нажать на эту кнопку. Важно помнить, что человек по своей природе привык взаимодействовать с объектами, поэтому фон у кнопки абсолютно не обязателен.



Рисунок 49.Кнопка-изображение



Рисунок 50.Кнопка-текст



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 348 из 469

Назад

На весь экран

Закрыть



Текстовые поля (TextFields)

Текстовые поля позволяют пользователю вводить текст в приложения. Они могут быть однострочными или многострочными. Одно касание текстового поля помещает курсор на поле ввода и автоматически выводит на экран клавиатуру. В дополнение к набору текста на клавиатуре текстовые поля позволяют выделять текст (вырезать, копировать, вставить). Поиск вариантов завершения слова помогает правописанию слов и упрощает поиск контактов в списке.

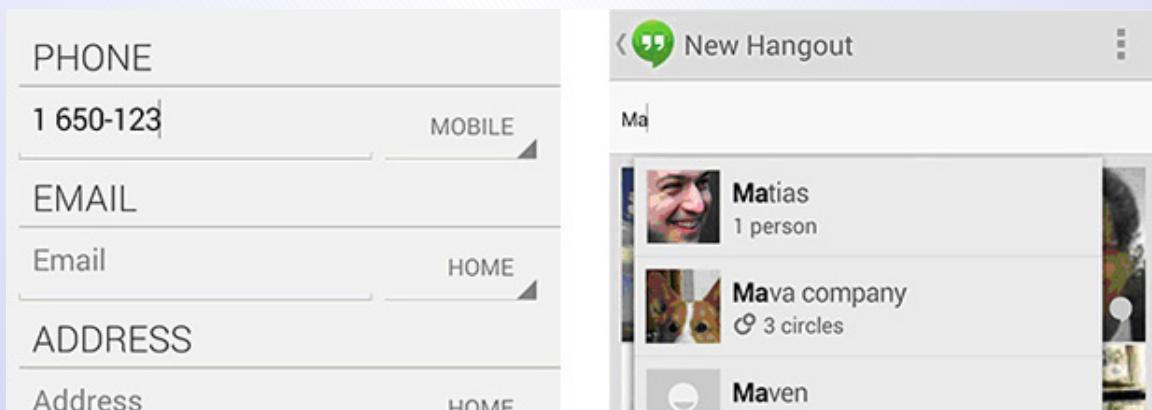


Рисунок 51. Текстовые поля, поиск контактов

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 349 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 350 из 469

Назад

На весь экран

Закрыть

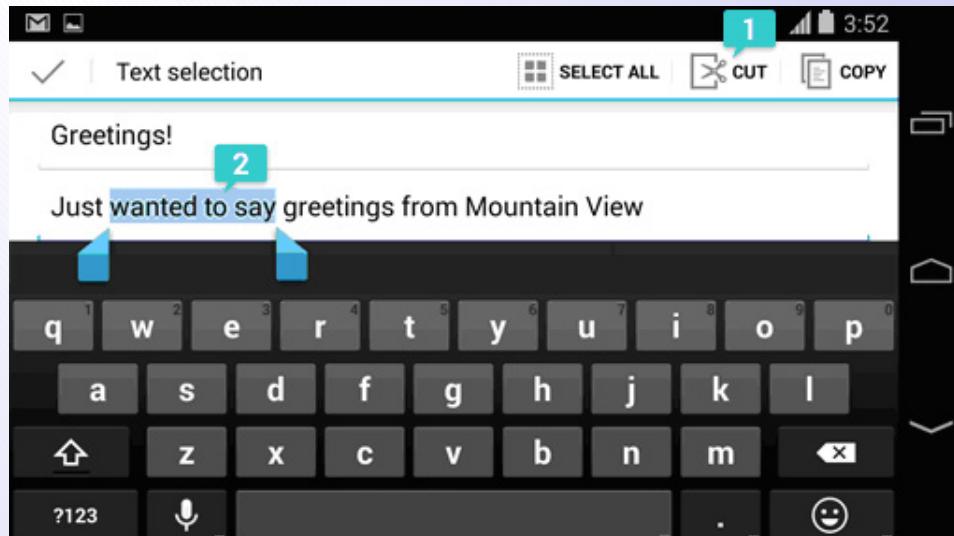


Рисунок 52.Выделение текста

Ползунки и слайдеры (Seek Bars and Sliders)

Интерактивные ползунки позволяют выбирать значение из непрерывного или дискретного диапазона значений путем перемещения ползунка. Наименьшее значение находится слева, наибольшее справа.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 351 из 469

Назад

На весь экран

Закрыть



Рисунок 53. Ползунки в светлой и темной темах

Интерактивный характер слайдера делает его удобным для настроек, которые отражают уровни интенсивности, такие как громкость, яркость, или насыщенность цвета.

Так, например, уровень громкости можно регулировать кнопками на корпусе устройства или при помощи жеста пальцем по экрану.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 352 из 469

Назад

На весь экран

Закрыть

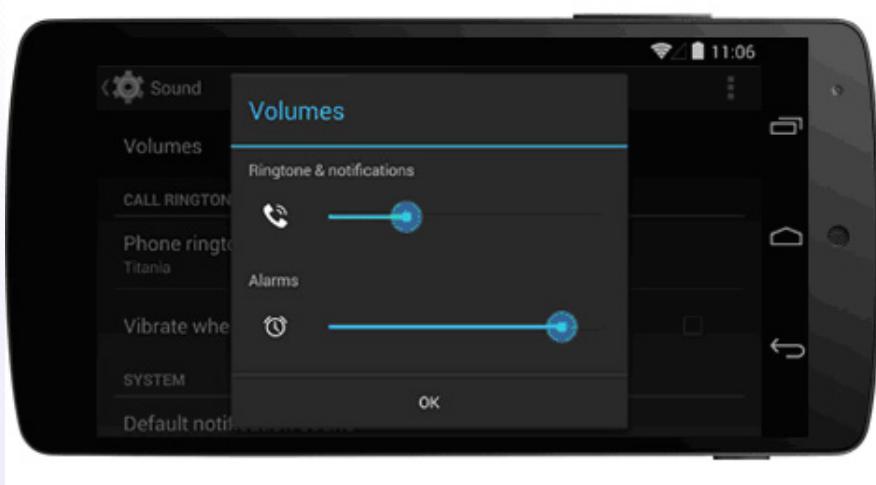


Рисунок 54. Настройка звука

Прогресс-бары и активности (ProgressDialog)

Прогресс-бары и показатели активности сигнализируют пользователям о происходящем в данный момент времени длительном действии, что означает для пользователя подождать завершения процесса некоторое время.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 353 из 469

Назад

На весь экран

Закрыть

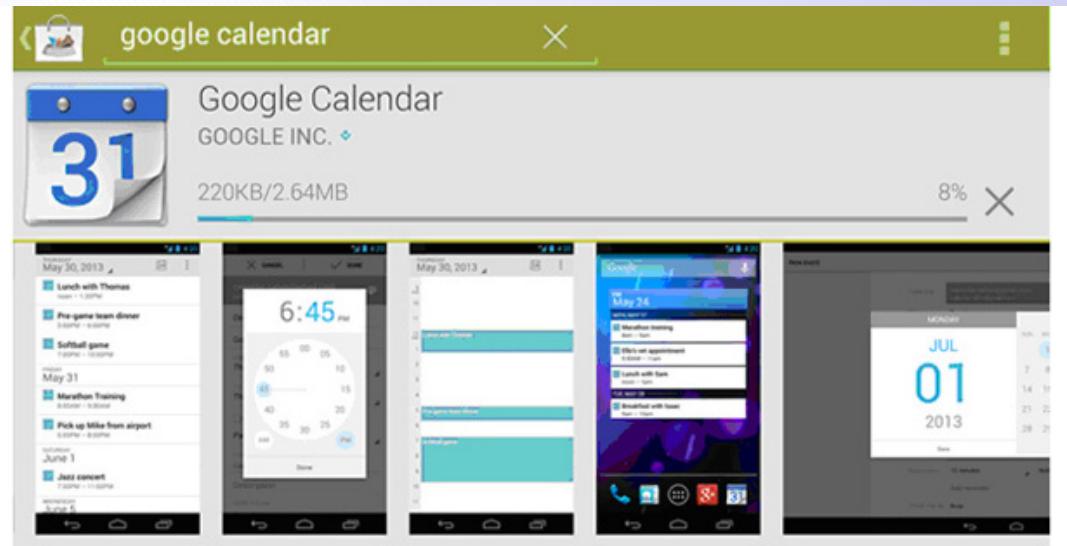


Рисунок 55.Процесс загрузки приложения

Переключатели (Switches) позволяют пользователю выбирать параметры.

Есть три вида переключателей: флагки, радио-кнопки, и включение/выключение выключателей.

Флагки используются в том случае, если из предлагаемого списка можно выбрать одновременно несколько вариантов.

Радио-кнопки позволяют выбрать только один вариант из списка. Радио-кнопки формируются в группы.



Кафедра ПМиИ

	NORMAL	FOCUSSED	PRESSED	DISABLED	DISABLE FOCUSSED
UNCHECKED	<input type="checkbox"/>				
CHECKED	<input checked="" type="checkbox"/>				
UNCHECKED	<input type="checkbox"/>				
CHECKED	<input checked="" type="checkbox"/>				

Рисунок 56.Флажки

	NORMAL	FOCUSSED	PRESSED	DISABLED	DISABLE FOCUSSED
UNCHECKED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
CHECKED	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
UNCHECKED	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
CHECKED	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

Рисунок 57.Радио-кнопки

Выключатели дают возможность сделать флагок более наглядным, применив в качестве основы кнопку-значок, которая может фиксироваться в нажатом состоянии.

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 354 из 469

Назад

На весь экран

Закрыть

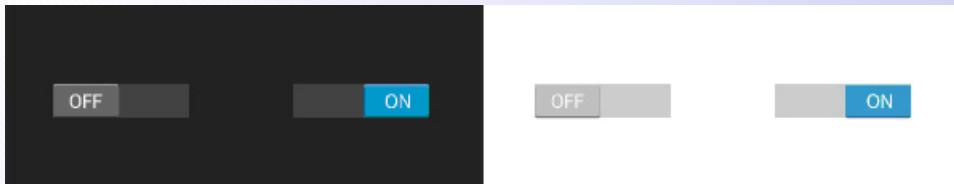


Рисунок 58.Выключатели

Диалоговые окна (Dialogs) помогают приложению взаимодействовать с пользователем. Это может быть как банальный вопрос с двумя вариантами ответа (OK и Отмена), так и сложные окна с множеством настраиваемых пользователем параметров.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 355 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 356 из 469

Назад

На весь экран

Закрыть

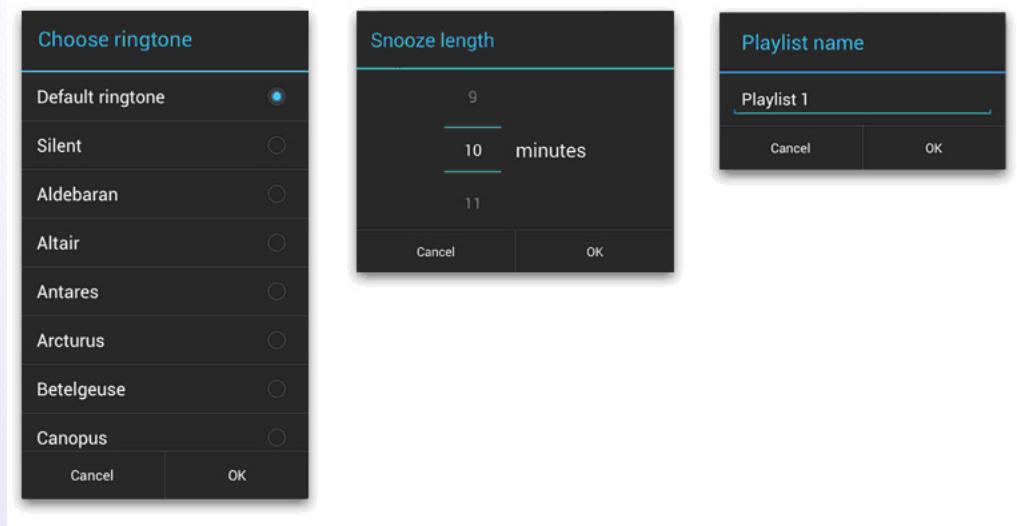


Рисунок 59.Примеры диалоговых окон

Средство выбора (Pickers) - это простой способ выбрать одно значение из набора путём последовательного перебора. Удобный перебор предлагаемых значений при помощи клика по стрелочкам или прокрутки, так же можно ввести значение с клавиатуры.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 357 из 469

Назад

На весь экран

Закрыть

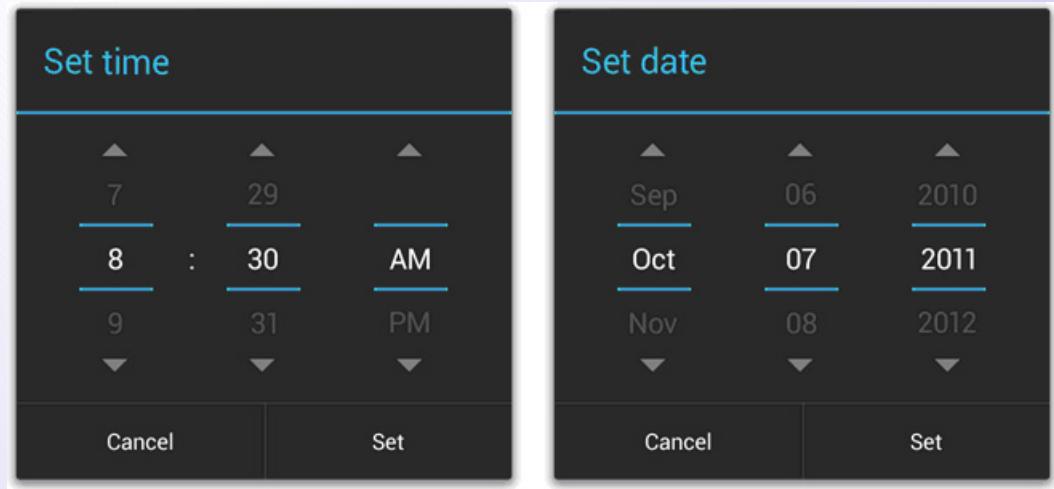


Рисунок 60.Примеры пикеров

Прокрутка (Scrolling), как интуитивно ожидаемый элемент, уже давно плотно сидит в головах у разработчиков и пользователей. Плавное или быстрое перемещение по содержимому экрана и контенту, который "вылез" за рамки экрана, осуществляется прокруткой, достаточно лишь провести пальцем по экрану в нужную сторону с желаемой скоростью.

Для того чтобы *индикатор* прокрутки не занимал *место* у рабочей области экрана, после приостановки пользования прокруткой, *индикатор* исчезает, и появляется снова, стоит только начать "прокручивать" пальцем. Для удобства поиска чего-либо в алфавитном списке, рядом с *индикатором* прокрутки возникает *индекс* с указанием на то, в каком разделе

находится индикатор.

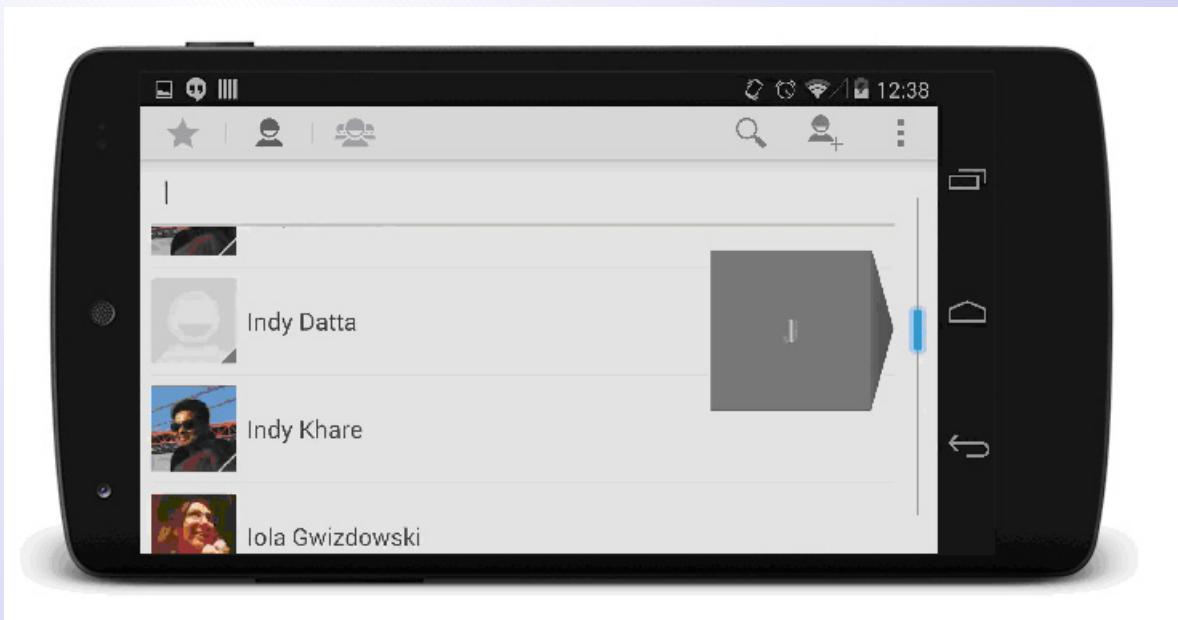


Рисунок 61.Индикатор и индекс прокрутки

Выпадающий список (Spinner) обеспечивает удобный способ выбора одного значения из набора.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 358 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 359 из 469

Назад

На весь экран

Закрыть

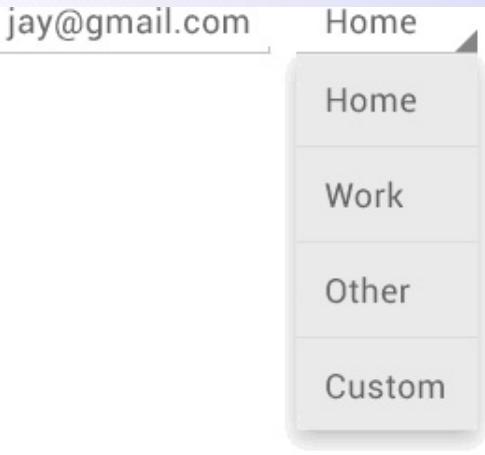


Рисунок 62. Спиннер для выбора типа e-mail адреса

Вкладки (Tabs) позволяют легко просматривать и переключаться между различными окнами или функциональными элементами приложения. Вкладки, прокручиваемые движением пальца (ScrollableTabs), делают этот процесс еще проще и приятнее.

Основные вкладки (FixedTabs) удобны при отображении трех и менее вкладок, поскольку ширина вкладок фиксируется по самому длинному названию вкладки.

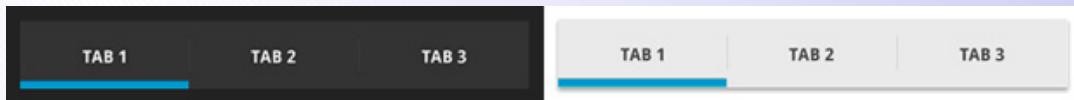


Рисунок 63. Вкладки

Списки (Lists) удобны для представления нескольких позиций информации в вертикальном расположении.

The screenshot displays three distinct list components:

- SINGLE LINE LIST**: A vertical list of three items:
 - List item number one
 - Second list item
 - This is the third item
- 2 LINE LIST**: A vertical list of two items:
 - 2-Line List
Austin mixtape Cosby sweater butcher. Fixie ad vice, Brooklyn...
 - Second list item
Assumenda commodo laborum accusamuA small thumbnail image of a man and a woman is shown next to the second item.
- 3 LINE LIST**: A vertical list of two items:
 - Three line list title
Put a bird on it qui fanny pack, portland irony nisi fap irure.
Donec hendrerit elit nec ligula dapibus
 - Second row in list
Vinyl laboris lo-fi ethical, adipiscing assumenda beard.
Curabitur gravida quam id orci sodales

Рисунок 64. Список



Кафедра
ПМиИ

- Начало
- Содержание
- ◀ ▶
- ◀◀ ▶▶
- Страница 360 из 469
- Назад
- На весь экран
- Закрыть

Сетки (GridLists) - это альтернатива стандартного списка. С помощью сетки можно создавать более удобное представление образных данных, нежели с помощью списка. Ещё одним плюсом сетки является то, что прокрутку можно осуществлять не только в вертикальном, но и в горизонтальном направлении.

Однако нужно стараться избегать создания сетки с возможностью прокрутки в обоих измерениях сразу.

Использование ярлычков для всех панелей сетки и полупрозрачных " занавесок" для неактивных панелей помогает пользователю быстрее понять, что скрывается под той или иной панелью.

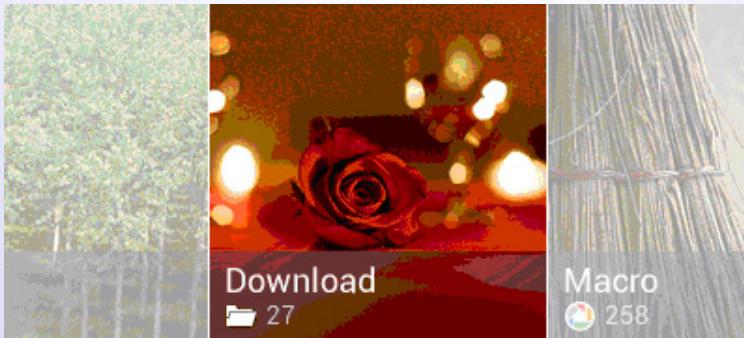


Рисунок 65. Сетка с альбомами



Кафедра ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 361 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

20.4 Задание

1. Создайте в приложении на главной активности строку, в которой будет выводиться адрес сайта, откуда загружено изображение, и кнопку для перехода на этот сайт.

2. Подумайте над интерфейсом собственного приложения. Какие элементы управления оно может содержать? Попробуйте воплотить его интерфейс на практике.

§21. Лабораторная работа №4

Аннотация: Научиться создавать приложения, состоящие из нескольких активностей, и диалоговые окна, а также познакомиться с элементами тач-интерфейса.

Цель лабораторной работы:

Научиться создавать приложения, состоящие из нескольких активностей, и диалоговые окна, а также познакомиться с элементами тач-интерфейса.

Задачи лабораторной работы:

- Научиться создавать многоэкранные приложения
- Научиться создавать диалоговые окна и всплывающие подсказки
- Научиться писать приложения со слайдингом



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)



Страница 362 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

21.1 Введение

Данная лабораторная работа посвящена различным способам разработки многоэкраных приложений. Предлагается разработать три приложения, каждое из которых посвящено отдельному аспекту этой тематики. По завершению работы над приложениями слушатели смогут разработать собственные программы с использованием изученных компонентов и технологий.

21.2 Создание многоэкранного приложения со списком

1. Создайте проект **MultiScreen**. В *java*-файле замените **Activity** на **ListActivity**. Класс **ListActivity** разработан таким образом, что на экране есть только прокручиваемый *список* и ему не нужна дополнительная разметка. Поэтому *файл activity_main.xml* можно удалить. Также следует удалить следующую строчку из класса **MultiScreen**:

```
setContentView(R.layout.activity_main);
```

т.к. *layout-файл* мы только что удалили.

2. Теперь нам нужен посредник, который свяжет *список* и названия элементов этого списка. Для этого в *Android* есть *интерфейс Adapter*. Нам потребуется наследник этого класса **ArrayAdapter**:

```
new ArrayAdapter(Context context, int textViewResourceId, String[] objects)
```

В качестве аргументов **ArrayAdapter** потребует текущий *контекст*,



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 363 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 364 из 469

Назад

На весь экран

Закрыть

идентификатор ресурса с разметкой для каждой строки, массив строк.

Мы можем указать в качестве текущего контекста **ListActivity** (можно использовать ключевое слово **this**), готовый системный идентификатор ресурса (**android.R.layout.simple_list_item_1**) и созданный массив строк. А выглядеть это будет так:

```
private ArrayAdapter<String> adapter;  
adapter = new ArrayAdapter<String>(this,  
        android.R.layout.simple_list_item_1, islands);
```

Обратите внимание на строчку **android.R.layout.simple_list_item_1**. В ней уже содержится необходимая разметка для элементов списка. Если вас не устраивает системная разметка, то можете создать собственную разметку в xml-файле и подключить её.

Осталось только подключить *адаптер*:

```
setListAdapter(adapter);
```

3. Теперь нам нужно подключить обработку нажатия. Для этого необходимо знать, на какой пункт списка осуществляется нажатие. Существует специальный интерфейс **OnItemClickListener** с методом **onItemClick()**. Набираем все в том же **onCreate**

```
OnItemClickListener itemListener = new OnItemClickListener(){ }
```

Если подчеркнётся первое слово, то импортируем нужный класс. Далее подведём курсор ко второму слову **new OnItemClickListener**. Нам будет предложено добавить метод (**Add unimplemented method**). Соглашаемся и получаем заготовку:



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 365 из 469

Назад

На весь экран

Закрыть

```
OnItemClickListener itemListener = new OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,  
    long arg3) {  
        // TODO Auto-generated method stub  
    }  
};
```

Меняем имена переменных *arg* на более привычные и понятные

```
public void onItemClick(AdapterView<?> parent, View v, int position,  
long id);
```

4. Осталось описать, что будет происходить при нажатии на элемент.

По нашей задумке каждый элемент будет открывать новое окно с соответствующим содержимым.

Для начала следует создать еще 4 класса: **Canari**, **Curili**, **Maldivi**, **Philippini**, и 4 xml-оболочки к ним. Можно просто создать и копировать *по одному* файлу в обеих директориях, меняя только название и содержимое.

Например, создадим файлы **Canari.java** и **canari.xml**. Обратите внимание, что как только мы создали ещё один *java-файл*, нужно записать его в *файл манифеста*, иначе *приложение* не будет видеть этот новый *класс*. *Файл AndroidManifest.xml* находится сразу под папкой **res**. Добавьте код с именем класса в тегах **<activity></activity>**, сразу под главной активностью.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 366 из 469

Назад

На весь экран

Закрыть

```
<activity android:name="com.mypackage.multiscreen.Canari"  
        android:label="@string/can" >  
</activity>
```

Строчку **can** нужно создавать в файле **strings.xml**, также как и другие строки.

Перейдем в *файл canari.xml* и создадим на экране **TextView**. Экраны будут отличаться друг от друга картинками. Возьмите 4 любые картинки с вашего компьютера (можете также найти их в интернете) и перетащите из проводника *Windows* в папку **res**. Теперь вы можете поместить элемент на экран и, выбрав нужную картинку из ресурсов проекта, подключить ее.

Остальные экраны создаются аналогично.

5. Теперь перейдем в главный *класс* и опишем обработку события **onItemClick()**. Создадим *переключатель*, который зависит от номера элемента.

```
switch (position) {  
    case 0:  
        break;  
    case 1:  
        break;  
    case 2:  
        break;  
    case 3:  
        break;
```

```
break;
```

```
}
```

Для запуска нового экрана необходимо создать экземпляр класса `Intent` и указать `класс`, на который будем переходить (у нас их 4, поэтому для каждого случая выбираем свою). После этого вызывается метод `startActivity()`, который и запускает новый экран.

```
Intent intent = new Intent(MultiScreenMainActivity.this, Canari.class);
startActivity(intent);
```

6. Теперь осталось добавить всплывающее окно `Toast`, которое будет показывать, какой элемент мы выбрали. Этот виджет можно импортировать так же, как и предыдущие. Нам потребуется метод `makeText()`, у которого есть три параметра: `контекст` приложения, текстовое сообщение и продолжительность времени показа уведомления.

```
Toast.makeText(getApplicationContext(), "Вы выбрали " +
parent.getItemAtPosition(position).toString(),
Toast.LENGTH_SHORT).show();
```

Полные листинги файлов проекта, в которых были сделаны изменения, см. ниже.

```
package com.mypackage.multiscreen;
import android.os.Bundle;
import android.view.View;
import android.app.ListActivity;
import android.content.Intent;
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 367 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

```
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Toast;
public class MultiScreenMainActivity extends ListActivity{
String[] islands = { "Канары" "Курилы" "Мальдивы" , "Филиппины"
};
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, islands);
setListAdapter(adapter);
OnItemClickListener itemListener = new OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> parent, View v, int position,
long id) {
switch (position) {
case 0:
Intent intent = new Intent(MultiScreenMainActivity.this, Canari.class);
startActivity(intent);
break;
case 1:
```

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 368 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

```
Intent intent1 = new Intent(MultiScreenMainActivity.this, Curili.class);
startActivity(intent1);
break;
case 2:
Intent intent2 = new Intent(MultiScreenMainActivity.this, Maldivi.class);
startActivity(intent2);
break;
case 3:
Intent intent3 = new Intent(MultiScreenMainActivity.this, Philippini.class);
startActivity(intent3);
break;
}
Toast.makeText(getApplicationContext(), "Вы выбрали " +
parent.getItemAtPosition(position).toString(),
Toast.LENGTH_SHORT).show();
}
};
getListView().setOnItemClickListener(itemListener);
}
}
```

Листинг 8.1. Файл MultiScreenMainActivity.java

```
package commypackage.multiscreen;  
import androidapp.Activity;
```

Начало

Содержание



Страница 369 из 469

[Назад](#)

На весь экран

Закрыть

```
import android.os.Bundle;
public class Canari extends Activity
{
@Override
protected void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.canari);
}
}
```

Листинг 8.2. Файл Canari.java

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
android:id="@+id/RelativeLayout1"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:padding="10dip" >
<ImageView
android:id="@+id/imageView1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentTop="true"
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 370 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 371 из 469

Назад

На весь экран

Закрыть

```
    android:layout_centerHorizontal="true"
    android:layout_marginTop="186dp"
    android:src="@drawable/canari" />
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="80dp"
    android:text="@string/enjoy"
    android:textAppearance="?android:attr/textAppearanceLarge" />
</RelativeLayout>
```

Листинг 8.3. Файл canari.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name" >Куда бы поехать в отпуск?</string>
    <string name="action_settings" >Settings</string>
    <string name="enjoy" >Enjoy yourself!</string>
    <string name="can" >Канары</string>
    <string name="phil" >Филиппины</string>
    <string name="cur" >Курилы</string>
    <string name="mal" >Мальдивы</string>
```

</resources>

Листинг 8.4. Файл strings.xml

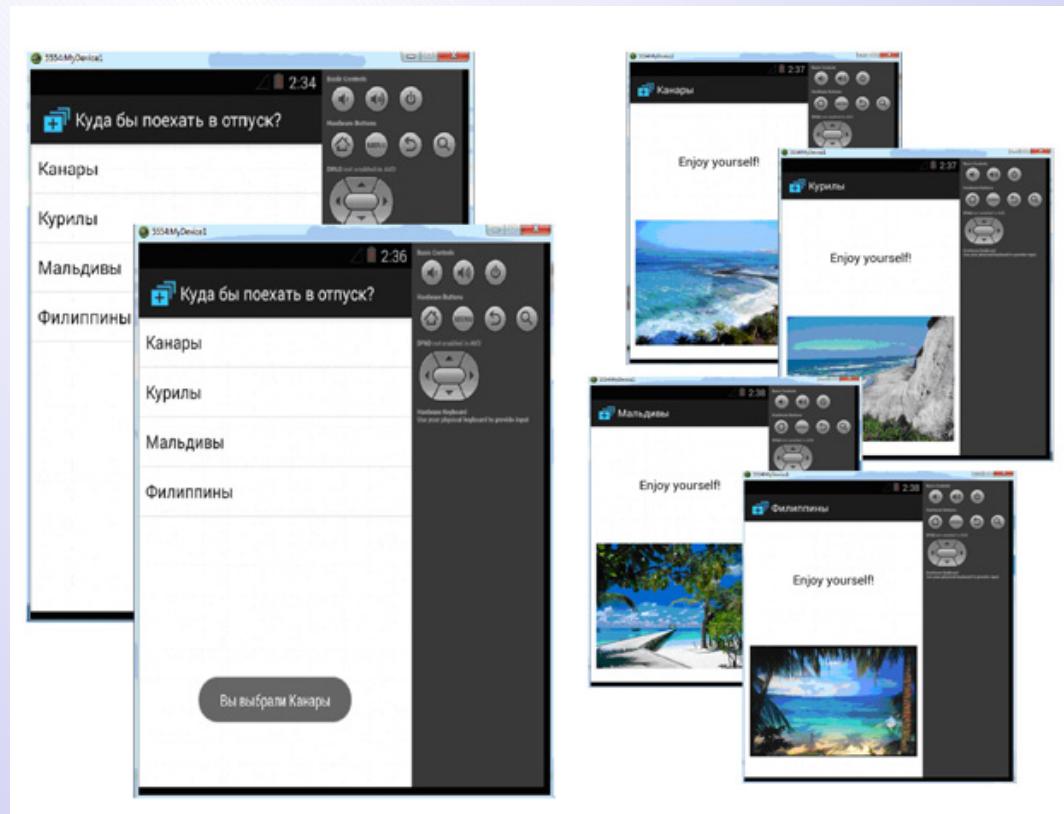


Рисунок 1.Приложение "Куда бы поехать в отпуск?" запущенное на эмуляторе



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 372 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 373 из 469

Назад

На весь экран

Закрыть

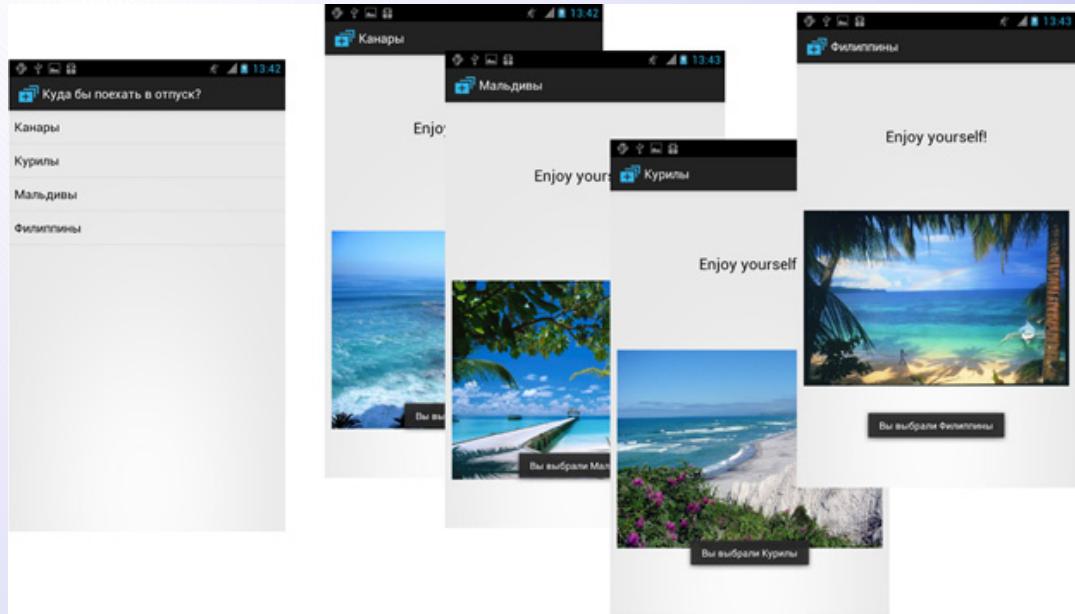


Рисунок 2.Приложение "Куда бы поехать в отпуск?" запущенное на устройстве

21.3 Создание диалогового окна

1. Создайте новый проект **Dialog**
2. Создайте кнопку на вашей активности и назовите ее "**Выбрать фон**"(для этого нужно в файле **strings** создать строку соответствующего содержания). Присвойте активности и кнопке **id**.

```
<RelativeLayout xmlns:android= "http://schemas.android.com/apk/res/android"
```



Кафедра ПМиИ

```
    android:id="@+id/relativelayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<Button
    android:id="@+id/background_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="174dp"
    android:text="@string/bg" />
</RelativeLayout>
```

Листинг 8.5. Код файла activity_main.xml

3. Наше приложение меняет фон на выбранный. Значит, нам нужно создать цвета и их названия в файле **strings.xml**. Как вы помните, этот файл находится в папке **values**, которая в свою очередь находится в папке **res**. Также создадим строку **messages**, которая нам понадобится для диалогового окна.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name" >Dialog</string>
<string name="action_settings" >Settings</string>
<string name="bg" >Выбрать фон</string>
```

Начало

Содержание

◀ ▶

◀ ▶

Страница 374 из 469

Назад

На весь экран

Закрыть



```
<string name="message" >Хотите поменять фон?</string>
<color name="redColor" >#FFFF0000</color>
<color name="yellowColor" >#FFFF00</color>
<color name="greenColor" >#FF00FF00</color>
<string name="red" >Красный</string>
<string name="yellow" >Жёлтый</string>
<string name="green" >Зелёный</string>
</resources>
```

Листинг 8.6. Файл strings.xml

4. Перейдем в файл **MainActivity.java**. Создайте следующие переменные:

```
private Button bgButton;
public RelativeLayout relativeLayout;
Context context;
```

Если компилятор подчеркивает тип и сообщает об ошибке, например, подчеркивает **Context**, наведите курсор на подчеркнутое слово: должно появиться контекстное меню, предлагающее варианты, как можно исправить ошибку. Выберете **Import 'Context'**, чтобы импортировать библиотеку.

5. Теперь нужно описать, что будет происходить при нажатии на нашу кнопку.

Для начала свяжем объекты из **activity_main.xml** и переменные в **MainActivity.java** через **id** (в **onCreate**):

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

« « » »

Страница 375 из 469

Назад

На весь экран

Закрыть



bgButton = (Button) findViewById(R.id.background_button);
relativeLayout = (RelativeLayout) findViewById(R.id.relativelayout);
Context - это объект, который предоставляет доступ к базовым функциям приложения.

Добавляем в код

```
context = MainActivity.this;
```

6. Теперь нужно добавить событие `onClick` и навесить на кнопку `OnClickListener`. Добавляем в заголовок класса `MainActivity implements OnClickListener`. Чтобы связать кнопку и `Listener` в `onCreate` пишем

```
bgButton.setOnClickListener(this);
```

Создадим теперь событие `onClick`

```
@Override
```

```
public void onClick(View v){  
}
```

В этом объекте создаем собственно диалог и называем его:

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);  
builder.setTitle(R.string.message);  
AlertDialog alert = builder.create();  
alert.show();
```

Мы будем создавать *диалоговое окно*, предоставляющее пользователю выбор из списка. Для этого потребуется ещё одна *переменная*, которая сформирует *список* из имеющихся строк.

```
final CharSequence[] items = {
```

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 376 из 469

Назад

На весь экран

Закрыть

```
getText(R.string.red), getText(R.string.yellow), getText(R.string.green)  
};
```

7. Сформируем собственно наш *список* и зададим еще один *Listener*, который будет менять *цвет фона* на выбранный:

```
builder.setItems(items, new DialogInterface.OnClickListener() {  
    public void onClick(DialogInterface dialog, int item) {  
        switch (item) {  
            case 0: { relativeLayout.setBackgroundResource(R.color.redColor); break; }  
            case 1:  
                relativeLayout.setBackgroundResource(R.color.yellowColor); break;  
            case 2:  
                relativeLayout.setBackgroundResource(R.color.greenColor); break;  
        }  
    }  
}
```

8. Осталось добавить в каждый *case* всплывающие окна *Toast*, и *приложение* полностью готово!

```
Toast.makeText(context, R.string.green, Toast.LENGTH_LONG).show();  
package com.mypackage.dialog;  
import android.app.Activity;  
import android.app.AlertDialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.os.Bundle;  
import android.view.View;
```



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 377 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

```
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.Toast;
public class MainActivity extends Activity implements OnClickListener
{
    private Button bgButton;
    public RelativeLayout relativeLayout;
    Context context;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bgButton = (Button) findViewById(R.id.background_button);
        bgButton.setOnClickListener(this);
        context = MainActivity.this;
        relativeLayout = (RelativeLayout)findViewById(R.id.relativelayout);
    }
    @Override
    public void onClick(View v) {
        final CharSequence[] items = getText(R.string.red) ,
        getText(R.string.yellow),getText(R.string.green)
    };
}
```

Начало

Содержание

◀ ▶

◀▶

Страница 378 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle(R.string.message);
builder.setItems(items, new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int item) {
        switch (item) {
            case 0: { relativeLayout.setBackgroundResource(R.color.redColor);
                Toast.makeText(context, R.string.red, Toast.LENGTH_LONG).show();
                break; } case 1:
            {relativeLayout.setBackgroundResource(R.color.yellowColor);
                Toast.makeText(context, R.string.yellow, Toast.LENGTH_LONG).show();
                break; }
            case 2: {relativeLayout.setBackgroundResource(R.color.greenColor);
                Toast.makeText(context, R.string.green, Toast.LENGTH_LONG).show();
                break; }
        }
    }
});
});
```

```
AlertDialog alert = builder.create();
alert.show();
}
}
```

Листинг 8.7. Код файла MainActivity.java

Скриншоты работающего приложения

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 379 из 469

Назад

На весь экран

Закрыть

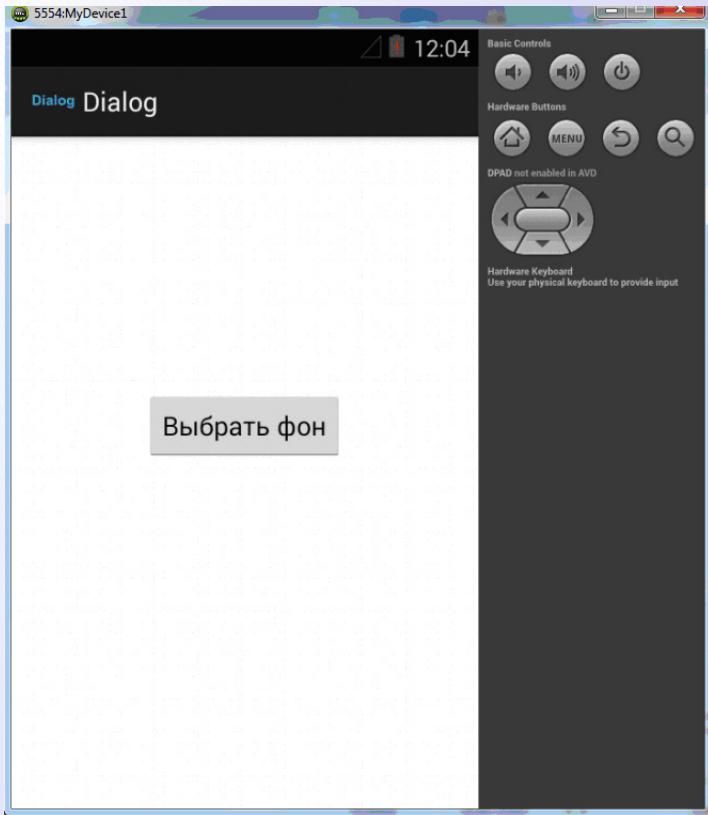


Рисунок 3.Приложение "Dialog" запущенное на эмуляторе



*Кафедра
ПМиИ*

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 380 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 381 из 469

Назад

На весь экран

Закрыть

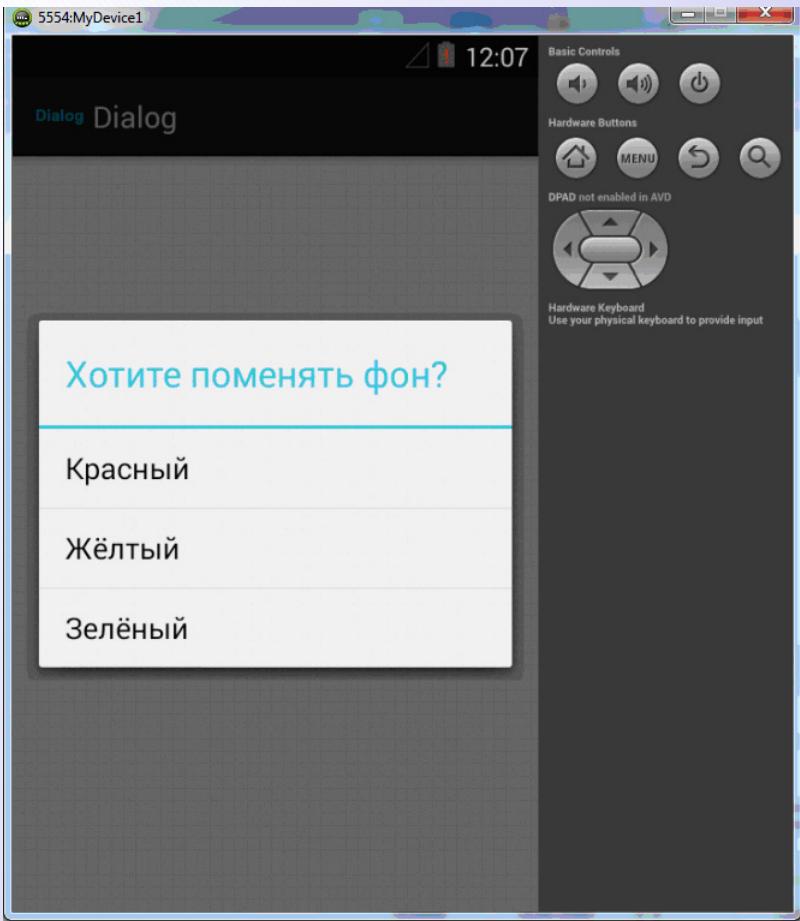


Рисунок 4. Диалоговое окно



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 382 из 469

Назад

На весь экран

Закрыть

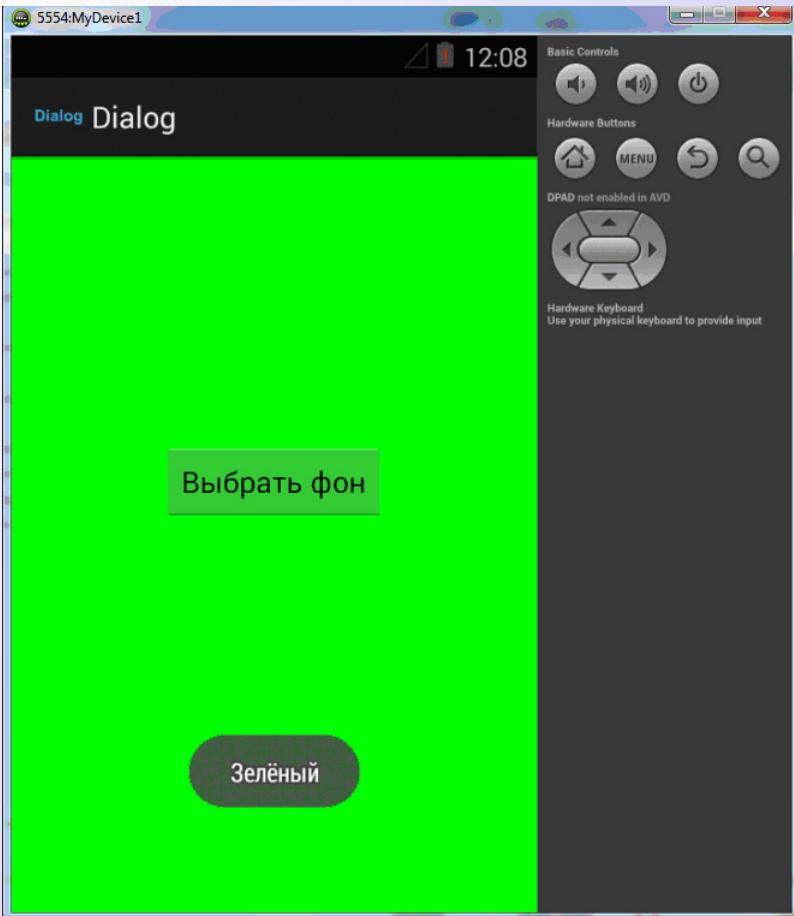


Рисунок 5. Выбран зеленый фон



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 383 из 469

Назад

На весь экран

Закрыть

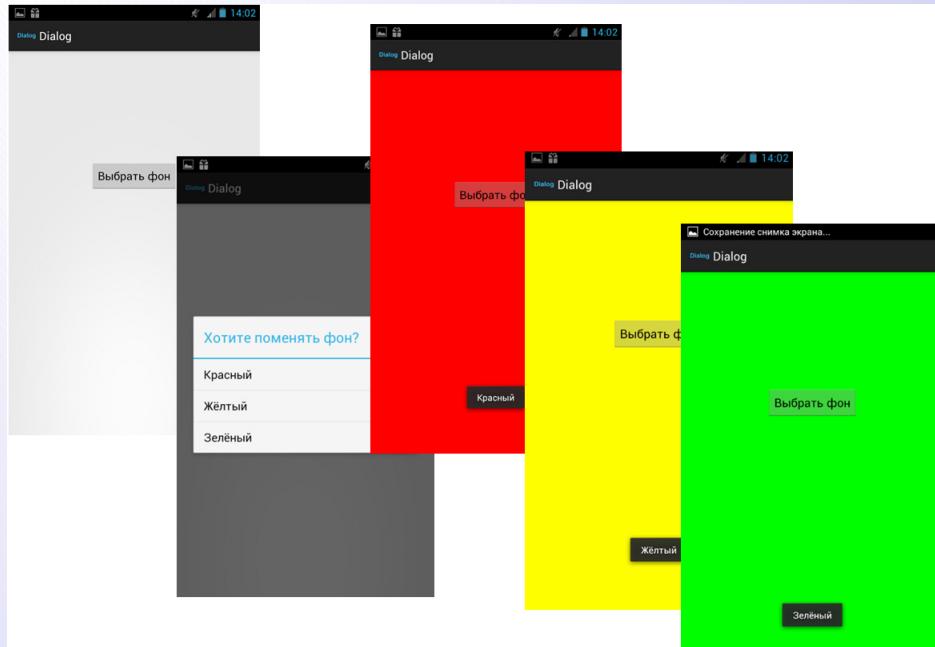


Рисунок 6.Приложение "Dialog" запущенное на устройстве

21.4 Создание приложения со слайдингом из шаблона

1. Создайте проект **TabsAndSwipe**. Обратите внимание: чтобы использовать стандартный шаблон активности **Fixed Tabs + Swipe**, вам необходимо при создании проекта указать **Minimum Required SDK** не меньше, чем **API11**, т.к. в более ранних версиях этот шаблон не под-

держивается.

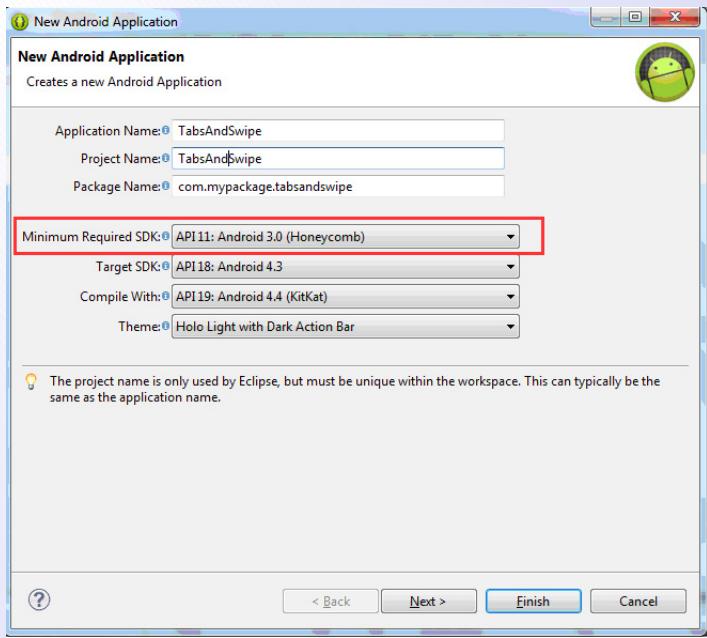


Рисунок 7. Создание приложения TabsAndSwipe

Следующие три окна оставляем без изменений.

Далее в окне **Blank Activity** выбираем в графе **Navigation Type**.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 384 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 385 из 469

Назад

На весь экран

Закрыть

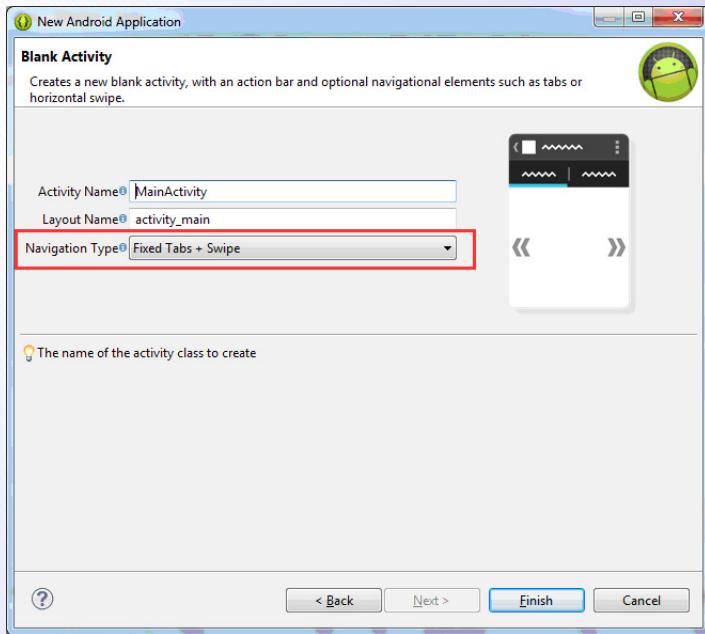


Рисунок 8. Выбор шаблона Navigation Type

2. Посмотрите на структуру проекта: у вас появились два xml-файла в папке **res- activity_main.xml** и **fragment_main_dummy.xml**.

3. Запустите *приложение*, чтобы убедиться, что все компилируется правильно.

4. Создайте три копии файла **fragment_main_dummy.xml**. Назовите их соответственно **first_tab**, **second_tab** и **third_tab**. Присвойте элементу **TextView** в каждом файле уникальный **id**. Можете поме-



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 386 из 469

Назад

На весь экран

Закрыть

стить на экраны какие-нибудь элементы, например картинки или надписи.

5. Теперь переходим к файлу **MainActivity.java**. Нас интересует класс

```
public static class DummySectionFragment extends Fragment
```

Делаем три копии данного класса со всем содержимым, называя их соответственно **FirstActivity**, **SecondActivity**, **ThirdActivity**.

6. Поменяйте в них следующие строки

```
View rootView = inflater.inflate(R.layout.fragment_main_dummy,  
container, false);
```

```
TextView dummyTextView = (TextView) rootView.findViewById  
(R.id.section_label);
```

Замените `R.layout.fragment_main_dummy` на `R.layout.first` и `R.id.section_label` на `R.id.section_label1` соответственно.

7. Поменяйте строки-названия секций в файле **strings.xml**.

```
<string name="title_section1" >Лента</string>  
<string name="title_section2" >Фото</string>  
<string name="title_section3" >Карта</string>
```

8. Теперь переходим к классу **SectionsPagerAdapter()**. Нас интересует его метод **Fragment getItem()**, именно его мы будем изменять. Чтобы при перелистывании менялось не только содержимое **TextView**, но и всего фрагмента, замените код этого метода на следующий:

```
public Fragment getItem(int position) {
```

```
// getItem is called to instantiate the fragment for the given page.  
// Return a DummySectionFragment (defined as a static inner class  
// below) with the page number as its lone argument.  
Fragment fragment=null;  
Bundle args;  
switch (position) {  
    case 0:  
        fragment = new FirstFragment();  
        args = new Bundle();  
        args.putInt(FirstFragment.ARG_SECTION_NUMBER, position + 1);  
        fragment.setArguments(args);  
        break;  
    case 1:  
        fragment = new SecondFragment();  
        args = new Bundle();  
        args.putInt(SecondFragment.ARG_SECTION_NUMBER, position + 1);  
        fragment.setArguments(args);  
        break;  
    case 2:  
        fragment = new ThirdFragment();  
        args = new Bundle();  
        args.putInt(ThirdFragment.ARG_SECTION_NUMBER, position + 1);  
        fragment.setArguments(args);
```



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 387 из 469

Назад

На весь экран

Закрыть

```
break;  
}  
return fragment;  
}
```

9. Приложение готово, можно запускать.

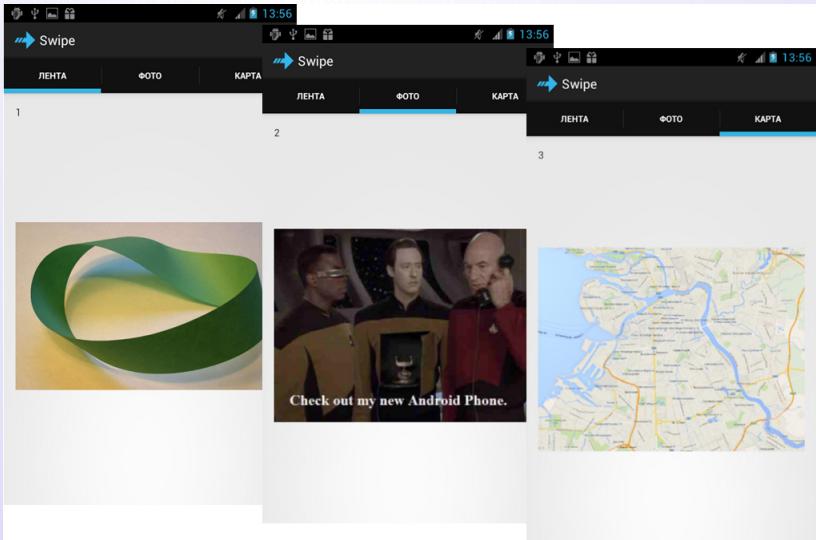


Рисунок 9.Приложение TabsAndSwipe, запущенное на устройстве



Кафедра
ПМИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 388 из 469

Назад

На весь экран

Закрыть

21.5 Задание для самостоятельного выполнения

Подумайте над собственным приложением, сочетающим различные возможности проектирования многооконных приложений, рассмотренные выше. Создайте прототип этого приложения и настройте его пользовательский *интерфейс*.

§22. Лабораторная работа №5

Демонстрации распознавания стандартных жестов

Аннотация: Разработать простейшие приложения для демонстрации распознавания стандартных жестов.

Цель лабораторной работы:

разработать простейшие приложения для демонстрации распознавания стандартных жестов.

Задачи лабораторной работы:

- рассмотреть распознавание всех поддерживаемых жестов;
- рассмотреть распознавание только части поддерживаемых жестов.

22.1 Введение

Для работы со стандартными жестами *Android* предоставляет класс *GestureDetector*. Этот класс содержит два вложенных



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 389 из 469

Назад

На весь экран

Закрыть



интерфейса-слушателя: `OnGestureListener` и `OnDoubleTapListener`, эти интерфейсы задают методы, отслеживающие стандартные жесты. А также `GestureDetector` содержит вложенный класс `SimpleOnGestureListener`, который содержит пустые реализации, возвращающие значение `false`, где это необходимо, всех методов интерфейсов: `OnGestureListener` и `OnDoubleTapListener`.

В лабораторной работе рассмотрим две возможности распознавания жестов:

- случай распознавания всех поддерживаемых жестов, для этого реализуем в классе активности оба интерфейса;
- случай распознавания только некоторого набора поддерживаемых жестов, для этого в классе активности объявим внутренний класс-наследник класса `GestureDetector.SimpleOnGestureListener`.

22.2 Распознавание всех поддерживаемых жестов

Разработаем *приложение*, в котором продемонстрируем *распознавание* всех поддерживаемых жестов. *Приложение* содержит одну активность, одно информационное поле для вывода информации о распознанном жесте. *Приложение* работает следующим образом: пользователь выполняет один из поддерживаемых сенсорных жестов, в информационном поле отображается *информация* о распознанном жесте.

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 390 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 391 из 469

Назад

На весь экран

Закрыть

1. Создадим простое приложение и добавим на форму TextView для вывода информации.

2. Настроим логику приложения. В java класс, соответствующий активности внесем следующие дополнения.

- Класс активности должен реализовывать интерфейсы:
`GestureDetector.OnGestureListener` и
`GestureDetector.OnDoubleTapListener`, для этого в объявление класса добавим конструкцию:

– implements `GestureDetector.OnGestureListener`,
`GestureDetector.OnDoubleTapListener`

- Нам понадобится экземпляр класса `GestureDetectorCompat` поэтому в качестве поля класса активности объявим следующую переменную:

– `GestureDetectorCompat mDetector;`

В методе `onCreate()` класса активности, создадим экземпляр класса `GestureDetectorCompat` и присвоим его переменной `mDetector`:

`mDetector = new GestureDetectorCompat(this, this);`

одним из параметров конструктора является *класс*, который реализует *интерфейс* `.OnGestureListener`, в нашем случае использовано слово `this`, т. е. параметром является сам *класс* активности. Этот *интерфейс*



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 392 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

уведомляет пользователей когда появляется определенное сенсорное событие.

В методе `OnCreate()` класса активности, следующая строка:

```
mDetector.setOnDoubleTapListener(this);
```

устанавливает слушатель событий, связанных с двойным касанием, это должен быть *класс*, реализующий *интерфейс*

`GestureDetector.OnDoubleTapListener`. В нашем случае использовано слово `this`, т.е. слушателем будет опять сам *класс* активности.

1. Чтобы позволить вашему объекту `GestureDetector` получать события, необходимо переопределить метод `onTouchEvent()` для активности или элемента GUI. И передавать в экземпляр детектора все обнаруженные события.

```
2. public boolean onTouchEvent(MotionEvent event){  
3.     this.mDetector.onTouchEvent(event);  
4.     // Be sure to call the superclass implementation  
5.     return super.onTouchEvent(event);  
6. }
```

7. После проведенной подготовки пришло время реализовать все методы, объявленные в интерфейсах, отвечающих за прослушивание сенсорных событий.

Методы интерфейса `GestureDetector.OnGestureListener`:



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 393 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

[onDown\(\)](#)

- отслеживает появление касания, т. е. палец прижат к экрану;

[onFling\(\)](#)

- отслеживает появление жеста смахивания;

[onLongPress\(\)](#)

- отслеживает удерживание пальца прижатым к экрану длительное время;

[onScroll\(\)](#)

- отслеживает появление жеста прокрутки (пролистывания);

[onShowPress\(\)](#)

- отслеживает, что произошло событие касания и больше никаких событий не происходит короткое время;

[onSingleTapUp\(\)](#)

- отслеживает появление жеста одиночного нажатия (клик).

Методы интерфейса [GestureDetector.OnDoubleTapListener](#):

[onDoubleTap\(\)](#)

- отслеживает появление жеста двойного нажатия ("двойной клик");

[onDoubleTapEvent\(\)](#)

- отслеживает появление события во время выполнения жеста двойного нажатия, включая касание, перемещение, подъем пальца.

[onSingleTapConfirmed\(\)](#)

- отслеживает появление жеста одиночного нажатия (клик).

В [листе 10.1](#) представлен код приложения, в котором распозна-



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 394 из 469

Назад

На весь экран

Закрыть

ются все поддерживаемые жесты, *информация* о появившемся и распознанном жесте выдается в информационное *поле* (*TextView*).

В качестве практики предлагается воспроизвести данное *приложение* и проверить, как система распознает тот или иной жест. Очень полезно для понимания, как выполняются основные жесты.

22.3 Распознавание только части поддерживаемых жестов

Разработаем *приложение*, в котором продемонстрируем *распознавание* только некоторой части поддерживаемых жестов по выбору программиста. Мы рассмотрим *распознавание* жеста смахивания (fling). *Приложение* содержит одну *активность*, одно информационное *поле* для вывода информации о распознанном жесте. *Приложение* работает следующим образом: *пользователь* выполняет один из поддерживаемых сенсорных жестов, в информационном *поле* отображается *информация* о распознанном жесте.

1. Создадим простое приложение и добавим на форму *TextView* для вывода информации.

2. Настроим логику приложения. В java класс, соответствующий активности внесем следующие дополнения.

Нам понадобится экземпляр класса *GestureDetectorCompat* поэтому в качестве поля класса активности объявим следующую переменную:

GestureDetectorCompat mDetector;



В методе `onCreate()` класса активности, создадим экземпляр класса `GestureDetectorCompat` и присвоим его переменной `mDetector`:

```
mDetector=new GestureDetectorCompat(this, new MyGestListener());
```

в конструкторе аргументом, отвечающим за отслеживание сенсорных событий, служит экземпляр класса `MyGestListener()` - внутренний класс, который является наследником класса

`GestureDetector.SimpleOnGestureListener`.

Имеет смысл немного рассмотреть класс

`GestureDetector.SimpleOnGestureListener`. Этот класс реализует интерфейсы `GestureDetector.OnGestureListener` и

`GestureDetector.OnDoubleTapListener`, все методы заявленные в интерфейсах в этом классе имеют пустую реализацию и те, которые должны возвращать значение, возвращают `false`. Поэтому для распознавания какого-то события или некоторого подмножества событий достаточно написать реализацию соответствующих методов, в классе наследнике.

В листинге 10.2 представлен код приложения, в котором распознается только жест смахивания, т. е. реализован метод `onFling()`, информация о появившемся и распознанном жесте выдается в информационное поле (`TextView`). В качестве слушателя используется экземпляр класса `MyGestListener()`, являющийся наследником класса

```
GestureDetector.SimpleOnGestureListener
```

Получить больше информации о распознавании жестов можно по ссылке: <http://developer.android.com/training/gestures/index.html>

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 395 из 469

Назад

На весь экран

Закрыть

```
package com.example.lab5_1_gestall;
import android.os.Bundle;
import android.app.Activity;
import android.support.v4.view.GestureDetectorCompat;
import android.view.*;
import android.widget.*;
public class MainActivity extends Activity implements
GestureDetector.OnGestureListener,
GestureDetector.OnDoubleTapListener
{
    TextView tvOutput;
    GestureDetectorCompat mDetector;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tvOutput = (TextView)findViewById(R.id.textView1);
        mDetector = new GestureDetectorCompat(this,this);
        mDetector.setOnDoubleTapListener(this);
    }
    public boolean onTouchEvent(MotionEvent event){
        this.mDetector.onTouchEvent(event);
        // Be sure to call the superclass implementation
    }
}
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 396 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

```
return super.onTouchEvent(event);
}
@Override
public boolean onDown(MotionEvent event) {
    tvOutput.setText("onDown: " + event.toString());
    return false;
}
@Override
public boolean onFling(MotionEvent event1, MotionEvent event2,
    float velocityX, float velocityY) {
    tvOutput.setText("onFling: " + event1.toString() + event2.toString());
    return true;
}
@Override
public void onLongPress(MotionEvent event) {
    tvOutput.setText("onLongPress: " + event.toString());
}
@Override
public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,
    float distanceY) {
    tvOutput.setText("onScroll: " + e1.toString() + e2.toString());
    return true;
}
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 397 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

```
@Override
public void onShowPress(MotionEvent event) {
    tvOutput.setText("onShowPress: " + event.toString());
}

@Override
public boolean onSingleTapUp(MotionEvent event) {
    tvOutput.setText("onSingleTapUp: " + event.toString());
    return true;
}

@Override
public boolean onDoubleTap(MotionEvent event) {
    tvOutput.setText("onDoubleTap: " + event.toString());
    return true;
}

@Override
public boolean onDoubleTapEvent(MotionEvent event) {
    tvOutput.setText("onDoubleTapEvent: " + event.toString());
    return true;
}

@Override
public boolean onSingleTapConfirmed(MotionEvent event) {
    tvOutput.setText("onSingleTapConfirmed: " + event.toString());
    return true;
}
```

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 398 из 469

Назад

На весь экран

Закрыть

```
}
```

Листинг 10.1. Распознавание поддерживаемых жестов с помощью реализации интерфейсов

```
package com.example.lab5_1_gestsubset;
import android.os.Bundle;
import android.app.Activity;
import android.support.v4.view.*;
import android.view.*;
import android.widget.*;
public class SubsetGestActivity extends Activity {
    private GestureDetectorCompat mDetector;
    private TextView tvOut;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_subset_gest);
        mDetector = new GestureDetectorCompat(this, new MyGestListener());
        tvOut = (TextView)findViewById(R.id.textView1);
    }
    @Override
    public boolean onTouchEvent(MotionEvent event){
        this.mDetector.onTouchEvent(event);
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 399 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

```
return super.onTouchEvent(event);
}
class MyGestListener extends GestureDetector.SimpleOnGestureListener
{
    @Override
    public boolean onFling(MotionEvent event1, MotionEvent event2,
    float velocityX, float velocityY) {
        tvOut.setText("onFling: " + event1.toString()+event2.toString());
        return true;
    }
}
```

Листинг 10.2. Распознавание жестов с использованием класса GestureDetector.SimpleOnGestureListener.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 400 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

§23. Лабораторная работа №6



Принципы работы с жестами вводимыми пользователями

Аннотация: Разработка приложения, помогающего понять принципы работы с жестами вводимыми пользователями.

Цель лабораторной работы:

- создать набор жестов
- использовать созданные жесты в приложении

23.1 Введение

Начиная с версии 1.6, *Android* предоставляет *API* для работы с жестами, который располагается в пакете *android.gesture* и позволяет сохранять, загружать, создавать и распознавать жесты.

В данной работе рассмотрим процесс создания набора жестов, для создания жестов будем использовать *приложение GestureBuilder*, которое предустановлено на виртуальные устройства *Android (AVD)*, начиная с версии 1.6.

После этого разработаем *приложение*, в котором предполагается *распознавание* и использование созданных жестов.

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 401 из 469

Назад

На весь экран

Закрыть

23.2 Создание набора жестов

Для начала создадим новое *приложение*.

Далее запустим эмулятор и используем *приложение Gesture Builder* для создания жестов "1" "2" и "S". Жест всегда связан с именем, но имя не обязательно должно быть уникальным, в действительности, для повышения точности в распознавании жеста рекомендуется сохранять несколько жестов с одним и тем же именем.

На рис 1 можно увидеть *приложение Gesture Builder* в работе, чтобы добавить жест, необходимо нажать на кнопку **Add gesture**, в свободном пространстве изобразить жест (обычно он рисуется желтым цветом), в *поле ввода* задать имя жеста. Результат последовательного добавления жестов можно увидеть на рис 2.

Жесты сохраняются на *SD* карте эмулятора, чтобы использовать их в приложении необходимо импортировать *файл* жестов в проект.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 402 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 403 из 469

Назад

На весь экран

Закрыть

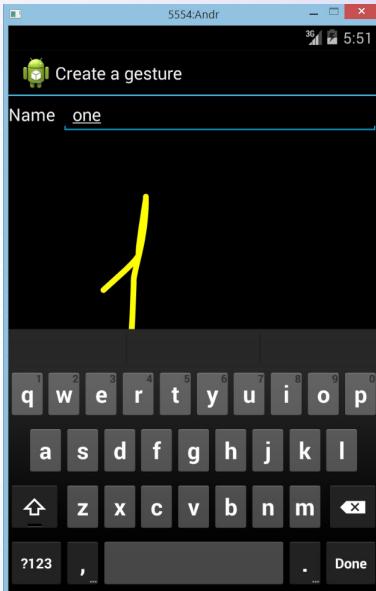


Рисунок 1. Создание жестов с помощью приложения Gesture Builder

Комментарий: *Gesture Builder* может сообщить, что ему некуда сохранять жесты, в этом случае необходимо запустить эмулятор с образом *SD* карты.

Сначала образ нужно создать с помощью утилиты **mksdcard** (расположена в папке <AndrSDK>/sdk/tools, где <AndrSDK> - путь, куда установлен *Android SDK*). В командной строке напишем:

mksdcard -l mySdCard 64M gesture.img



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 404 из 469

Назад

На весь экран

Закрыть

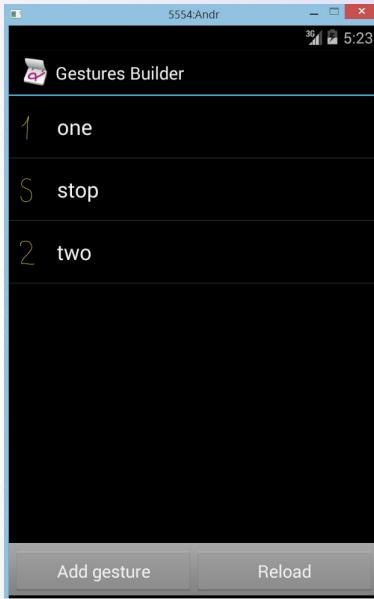


Рисунок 2. Набор жестов, созданных в приложении Gesture Builder

Следующим шагом будет создать и запустить эмулятор с образом *gesture.img*, для этого зайдем в папку <AndrSDK>/*sdk/tools* и выполним команду:

`emulator -avd nameEmulator -sdcard gestures.img`

nameEmulator - имя, которое присвоено эмулятору при создании.

Интересная статья на эту тему по ссылке:

<http://habrahabr.ru/post/120016/>

Каждый раз, когда мы создаем или редактируем жесты с помощью



Кафедра ПМиИ

Начало

Содержание



Страница 405 из 469

Назад

На весь экран

Закрыть

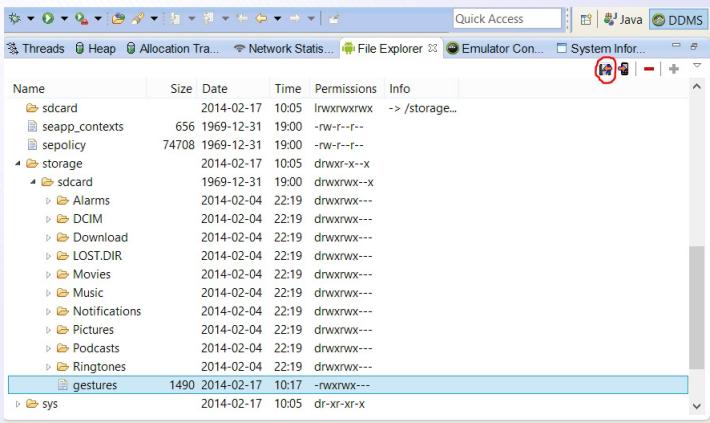


Рисунок 3. Расположение файла gestures

Gesture Builder, создается файл *gestures* на *SD* карте эмулятора. Необходимо импортировать этот файл в директорию **res/raw/**, созданного проекта, в котором планируем использовать жесты.

Самый простой способ импортировать жесты в проект заключается использовании вкладки **File Explorer** в компоновке (*perspective*) **DDMS**. (Если компоновки **DDMS** нет найти ее можно следующим образом: **Window->Open Perspective->Other...->DDMS**. Если вкладки **File Explorer** нет, можно добавить: **Window-> Show View-> File Explorer**). На вкладке **File Explorer** найти директорию **sdcard/** (в нашем случае



Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 406 из 469

Назад

На весь экран

Закрыть

оказалась директория `storage/sdcard/`, имеет смысл при создании жестов обратить внимание в какую директорию *Gesture Builder* их сохраняет). На рис 3 показана вкладка **File Explorer** в компоновке *DDMS*.

Чтобы скопировать файл жестов с эмулятора в проект, необходимо выбрать его и нажать кнопку "**Pull a file from the device**" выделенную на рис 3 красным подобием окружности. Откроется диалог с предложением выбрать папку, в которую необходимо скопировать жесты, здесь надо найти папку проекта, в ней папку `res/raw/` (если папки `raw/` нет, ее необходимо создать) и нажать кнопку **Сохранить**. Теперь жесты есть в нашем проекте и их можно использовать.

23.3 Использование созданных жестов в приложении

Для распознавания жестов необходимо добавить элемент `GestureOverlayView` в XML файл активности. И этот файл может выглядеть, например, как показано на рис 4:



```
activity_gestures.xml  Gestures.java  FullscreenActivity.java
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Gestures" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="15dp"
        android:text="TextView" />

    <@+id/gestureOverlayView
        android:id="@+id/gestureOverlayView1"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_below="@+id/textView1" >

    </android.gesture.GestureOverlayView>

</RelativeLayout>
```

Graphical Layout activity_gestures.xml

Рисунок 4. XML файл активности приложения, элемент GestureOverlayView обычный компонент интерфейса пользователя

Можно добавить элемент **GestureOverlayView** поверх всех компонентов, как прозрачный слой, в этом случае *XML файл* активности может выглядеть так, как показано на рис 5.

Кафедра ПМиИ

Начало

Содержание

◀ ▶

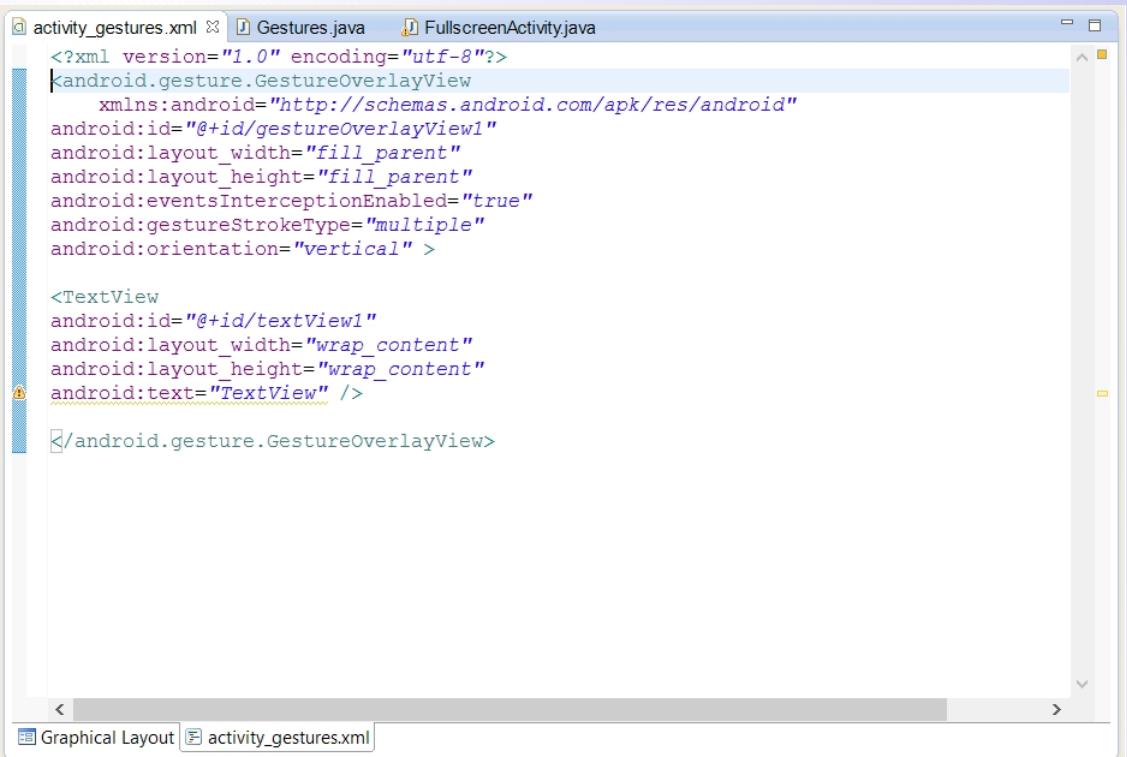
◀ ▶

Страница 407 из 469

Назад

На весь экран

Закрыть



The screenshot shows the Android Studio interface with the XML file 'activity_gestures.xml' open. The code defines a GestureOverlayView that contains a TextView. The GestureOverlayView is set to intercept events and support multiple gestures. The TextView is a simple text view with wrap_content dimensions.

```
<?xml version="1.0" encoding="utf-8"?>
<kandroid.gesture.GestureOverlayView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gestureOverlayView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:eventsInterceptionEnabled="true"
    android:gestureStrokeType="multiple"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView" />

</kandroid.gesture.GestureOverlayView>
```

Рисунок 5. XML файл активности приложения, элемент GestureOverlayView поверх всех компонентов интерфейса пользователя

Далее необходимо обработать ввод жеста пользователя, сравнить с загруженными жестами, и либо определить жест, либо сообщить пользователю, что такого жеста нет. Теперь вся работа будет выполняться в

Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 408 из 469

Назад

На весь экран

Закрыть

java файле, описываящем главную (и единственную) *активность* приложения. Внесем в этот *класс* следующие дополнения:

- Класс активности должен реализовывать интерфейс `OnGesturePerformedListener`, для этого в объявление класса добавим конструкцию:
 - implements `OnGesturePerformedListener`;
- Нам понадобятся экземпляры классов `GestureLibrary` и `GestureOverlayView`, поэтому в качестве полей класса активности объявим следующие переменные:
 - `GestureLibrary gLib;`
 - `GestureOverlayView gestures;`
- В методе `onCreate()` выполним следующие действия:
 - `gLib = GestureLibraries.fromRawResource(this, R.raw.gestures);`
 - `if (!gLib.load()) {`
 - `finish();`
 - `}`

В первой строке выполнена инициализация переменной `gLib` жестами, загруженными из файла `gestures` папки `res/raw/`.

Оператор `if` выполняет проверку загружены ли жесты, если нет, выполняется выход из приложения.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 409 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



- Добавим в метод `onCreate()` еще две строчки:

- ```
gestures = (GestureOverlayView) findViewById(R.id.gestureOverlayView1);
gestures.addOnGesturePerformedListener(this);
```

Для инициализации переменной `gesture` и подключения к ней слушателя событий появления жеста.

- И наконец напишем реализацию метода `OnGesturePerformed()`, который и будет вызываться при появлении события, соответствующего какому-либо жесту.

- ```
public void onGesturePerformed(GestureOverlayView overlay,
Gesture gesture) {
    //Создаёт ArrayList с загруженными из gestures жестами
    ArrayList<Prediction> predictions = gLib.recognize(gesture);
    if (predictions.size() > 0)
        //если загружен хотябы один жест из gestures
        Prediction prediction = predictions.get(0);
        if (prediction.score > 1.0) {
            if (prediction.name.equals("one"))
                tvOut.setText("1");
            else if (prediction.name.equals("stop"))
```

Начало

Содержание

◀ ▶

◀▶ ▶▶

Страница 410 из 469

Назад

На весь экран

Закрыть



```
tvOut.setText("stop");
else if (prediction.name.equals("two")
tvOut.setText("2");
} else{
tvOut.setText("Жест неизвестен";
}
}
}
```

В приложении всего лишь распознаются жесты и в информационное поле выводится *информация* о том, что за жест был использован. В листинге 6.1 представлен возможный код приложения.

23.4 Заключение

В качестве практического задания предлагаем реализовать жестами ввод чисел в приложении "Угадайка" разработанном в лабораторной работе второй темы. Создать жесты "0" "1" "2" "3" "4" "5" "6" "7" "8" "9" для ввода цифр и жест "S" для остановки ввода числа. В приложение добавить распознавание этих жестов, преобразование их в число и сравнение полученного числа с загаданным.

Еще варианты для самостоятельной работы:

1. разработать простой калькулятор с жестовым вводом чисел и операций;

2. разработать блокнотик для заметок с рукописным вводом текста.

```
package com.example.lab5_2_gestures;
```

```
import java.util.ArrayList;
```

```
import android.os.Bundle;
```

```
import android.app.Activity;
```

```
import android.gesture.Gesture;
```

```
import android.gesture.GestureLibraries;
```

```
import android.gesture.GestureLibrary;
```

```
import android.gesture.GestureOverlayView;
```

```
import android.gesture.GestureOverlayView.OnGesturePerformedListener;
```

```
import android.gesture.Prediction;
```

```
import android.view.Menu;
```

```
import android.widget.TextView;
```

```
public class Gestures extends Activity implements  
OnGesturePerformedListener{
```

```
    GestureLibrary gLib;
```

```
    GestureOverlayView gestures;
```

```
    TextView tvOut;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)



Страница 412 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

```
setContentView(R.layout.activity_gestures);
tvOut=(TextView)findViewById(R.id.textView1);
//Загрузка жестов (gestures) из res/raw/gestures
gLib = GestureLibraries.fromRawResource(this, R.raw.gestures);
if (!gLib.load()) {
    //Если жесты не загружены, то выход из приложения
    finish();
}
gestures = (GestureOverlayView) findViewById
(R.id.gestureOverlayView);
gestures.addOnGesturePerformedListener(this);
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.gestures, menu);
    return true;
}
public void onGesturePerformed(GestureOverlayView overlay,
Gesture gesture) {
    //Создаёт ArrayList с загруженными из gestures жестами
    ArrayList<Prediction> predictions = gLib.recognize(gesture);
    if (predictions.size() > 0) {
```

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 413 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

```
//если загружен хотя бы один жест из gestures
Prediction prediction = predictions.get(0);
if (prediction.score > 1.0) {
    if (prediction.name.equals("one") tvOut.setText("1");
    else if (prediction.name.equals("stop") tvOut.setText("stop");
    else if (prediction.name.equals("two"))
        tvOut.setText("2");
    }else{
        tvOut.setText("Жест неизвестен";
    }
}
}

}
```

Листинг 6.1. Распознавание жестов загруженных в файл
res/raw/gestures

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 414 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

§24. Лабораторная работа №7



Многооконное приложение

Аннотация: Разработка многооконного приложения, предоставляющего возможности: воспроизведения аудио и видео файлов, создания и отображения фотоснимков.

Цель лабораторной работы:

Разработка многооконного приложения, предоставляющего возможности: воспроизведения аудио и видео файлов, создания и отображения фотоснимков.

Задачи лабораторной работы:

- настроить интерфейс и реализовать логику активности для работы с камерой;
- настроить интерфейс и реализовать логику активности для воспроизведения аудио и видео;
- настроить интерфейс и реализовать логику активности для просмотра изображений;
- настроить интерфейс и реализовать логику главной активности приложения.

Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 415 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

24.1 Введение

Для достижения цели, поставленной в лабораторной работе, сформулируем требования к разрабатываемому приложению, назовем его "*Media*".

Приложение предоставляет пользователю возможность выбора рода деятельности:

- работа с камерой для создания снимков;
- воспроизведение аудио и видео;
- просмотр изображений.

В приложении предполагается реализовать четыре активности:

- главная активность, предназначена для выбора рода деятельности, содержит три кнопки, нажатие на каждую кнопку вызывает к жизни соответствующую активность;
- активность для работы с камерой и создания снимков;
- активность для воспроизведения аудио и видео;
- активность для просмотра изображений.

24.2 Создание приложения

Создадим новое *приложение*, *активность*, полученную при создании проекта, оставим с именем **MainActivity**, она будет главной ак-



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 416 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



тивностью приложения. Добавим в *приложение* еще три активности: **New->Other...->Android Activity**.

В нашем приложении созданные активности имеют следующие имена:

CameraActivity для работы с камерой и создания снимков;

MediaActivity для воспроизведения видео и аудио;

GalleryActivity для просмотра изображений.

Далее продолжим работу с этими активностями, настроим *интерфейс* и реализуем логику для каждой из них. Начнем с активностей, отвечающих за тот или иной вид деятельности, главную *активность* приведем в порядок в самом конце работы.

24.3 Настройка интерфейса и реализация логики активности для работы с камерой

Настроим *интерфейс* активности для работы с камерой. Нам понадобится окно предварительного просмотра, добавим в окно активности элемент **SurfaceView**. А так же нам понадобится кнопка для выполнения снимков, добавим в окно активности элемент **ImageButton**. Далее предлагаем настроить *интерфейс* самостоятельно. В нашем случае *активность* выглядит так, как показано на рис 1.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 418 из 469

Назад

На весь экран

Закрыть

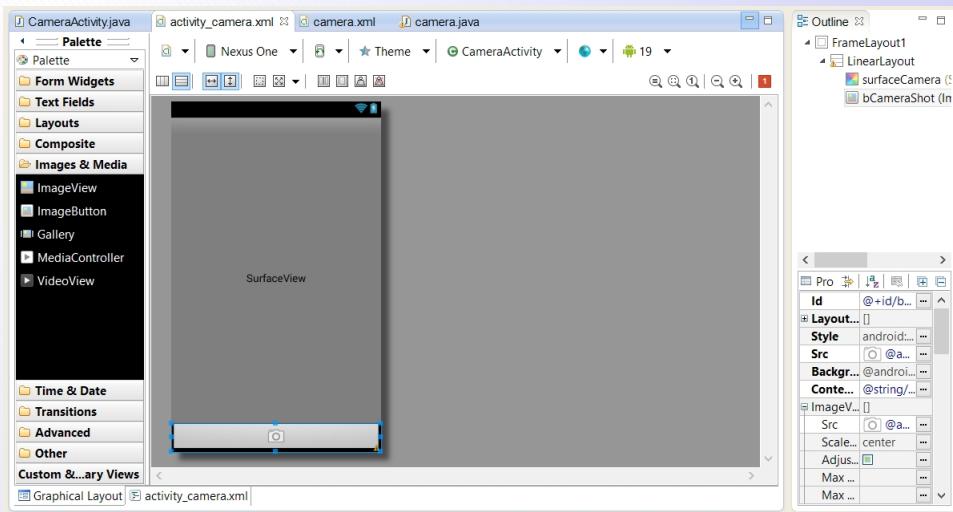


Рисунок 1.Интерфейс активности для работы с камерой

Реализуем логику активности, в данном случае необходимо при нажатии на кнопку выполнить снимок. Работаем с *java* файлом, описывающим соответствующий класс активности, **CameraActivity.java**.

Объявим следующие поля класса активности:

```
private Camera camera; //для проведения всех операций с камерой
private SurfaceHolder surfaceHolder; //для задания preview
private SurfaceView preview; //для отображения окна предпросмотра
private View shotBtn; //для выполнения снимка (кнопка)
```

Для начала работы с камерой необходимо ее инициализировать, сде-



лучше в методе `onResume()` класса активности, для инициализации используется следующая конструкция:

```
camera = Camera.open();
```

После завершения работы с камерой, необходимо ее освободить для других приложений, сделать это лучше в методе `onPause()` класса активности, для освобождения камеры используется следующая конструкция:

```
Camera.release();
```

Для активности, работающей с камерой, имеет смысл сразу задать расположение экрана следующим образом

```
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
```

в противном случае, приходится, например, в методе `surfaceCreated()` проверять расположение экрана и поворачивать окно предварительного просмотра, что не очень удобно, так как поворот экрана занимает некоторое время.

Выполним настройку окна предварительного просмотра:

```
preview = (SurfaceView) findViewById(R.id.surfaceCamera);
surfaceHolder = preview.getHolder();
surfaceHolder.addCallback(new MyCallback());
```

метод `addCallback()` используется для отслеживания изменений окна предпросмотра. Параметром этого метода служит экземпляр класса реализующего *интерфейс* `SurfaceHolder.Callback`, в нашем случае для реализации этого интерфейса создан внутренний *класс* активности

Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 419 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 420 из 469

Назад

На весь экран

Закрыть

MyCallback.

В этом классе необходимо реализовать методы:

```
public void surfaceCreated(SurfaceHolder holder);  
public void surfaceChanged(SurfaceHolder holder, int format,  
int width, int height);  
public void surfaceDestroyed(SurfaceHolder holder);
```

Мы реализуем метод `surfaceCreated()`, оставшиеся два метода запишем с пустой реализацией.

В методе `surfaceCreated()` зададим окно предварительного просмотра:
`camera.setPreviewDisplay(holder);`

Выполним необходимые настройки окна предпросмотра (см. [Листинг 12.1](#)).

Запустим *отображение* окна предварительного просмотра:

```
camera.startPreview();
```

Пришло время обсудить и реализовать выполнение фотосъемки. Для того, чтобы сделать снимок необходимо вызвать метод `takePicture()`. Вызов этого метода можно выполнить из метода-обработчика события нажатия кнопки, тогда при нажатии на кнопку сразу будет получена фотография, а можно "растянуть удовольствие" и воспользоваться предварительной автофокусировкой.

Сначала инициализируем кнопку и добавим слушателя события нажатия в методе `onCreate()` активности:

```
shotBtn = findViewById(R.id.bCameraShot);
```



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 421 из 469

Назад

На весь экран

Закрыть

```
shotBtn.setOnClickListener(new MyViewListener());
```

В данном случае слушателем является экземпляр внутреннего класса **MyViewListener**, который реализует *интерфейс* **View.OnClickListener**.

```
class MyViewListener implements View.OnClickListener{  
    @Override  
    public void onClick(View v){  
        if (v == shotBtn){  
            camera.autoFocus(new MyAutoFocusCallback());  
        }  
    }  
}
```

В методе **onClick()** вызываем обработчик автофокуса, слушателем события автофокусировки в данном случае является экземпляр внутреннего класса **MyAutoFocusCallback**, который реализует *интерфейс* **Camera.AutoFocusCallback**.

```
class MyAutoFocusCallback implements Camera.AutoFocusCallback{  
    @Override  
    public void onAutoFocus(boolean paramBoolean, Camera paramCamera){  
        if (paramBoolean){  
            // если удалось сфокусироваться, делаем снимок  
            paramCamera.takePicture(null, null, null, new MyPictureCallback());  
        }  
    }  
}
```

```
}
```

Метод `takePicture()` имеет четыре параметра:

`Camera.ShutterCallback` - вызывается в момент получения изображения с матрицы;

`Camera.PictureCallback raw` - программе передаются для обработки raw данные (если поддерживается аппаратно);

`Camera.PictureCallback postview` - программе передаются полностью обработанные данные (если поддерживается аппаратно);

`Camera.PictureCallback jpg` - программе передается изображение в формате jpg. Здесь может быть организована запись изображения на карту памяти.

В нашем случае последним параметром является экземпляр класса `MyPictureCallback()`, который реализует *интерфейс Camera.PictureCallback*. В этом классе реализован единственный метод (см. [Листинг 7.1](#)):

```
public void onPictureTaken(byte[] paramArrayOfByte,  
    Camera paramCamera)...
```

В [листе 7.1](#) представлен код активности, реализующей работу с камерой и позволяющей выполнять снимки.

Чтобы разрешить приложению работать с камерой и сохранять фото-



Кафедра ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 422 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

графии на карте памяти, в *манифест* необходимо добавить следующие разрешения:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.
WRITE_EXTERNAL_STORAGE" />
```

24.4 Настройка интерфейса и реализация логики активности для воспроизведения аудио и видео

Настроим *интерфейс* активности для воспроизведения аудио и видео. Нам понадобится окно предварительного просмотра, добавим в окно активности элемент *SurfaceView*. Для работы приложения необходимо поле для задания расположения медиа контента для проигрывания, добавим элемент *EditText*, а также потребуются несколько кнопок, добавим в окно активности следующие кнопки:

- b_Start** - для запуска воспроизведения контента с начала, зададим свойству *On Click* этой кнопки значение *onClickStart*;
- b_Pause** - для приостановки воспроизведения, зададим свойству *On Click* этой кнопки значение *onClick*;
- b_Resume** для продолжения воспроизведения с места приостановки, зададим свойству *On Click* этой кнопки значение *onClick*;



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 423 из 469

Назад

На весь экран

Закрыть



Кафедра
ПМиИ

Начало

Содержание



Страница 424 из 469

Назад

На весь экран

Закрыть

b_Stop - для полной остановки воспроизведения, зададим свойству **On Click** этой кнопки значение **onClick**.

Для возможности выбора зацикленного воспроизведения медиа контента добавим в окно активности элемент **CheckBox**.

Далее предлагаем настроить *интерфейс* самостоятельно. В нашем случае *активность* выглядит так, как показано на рис 2.

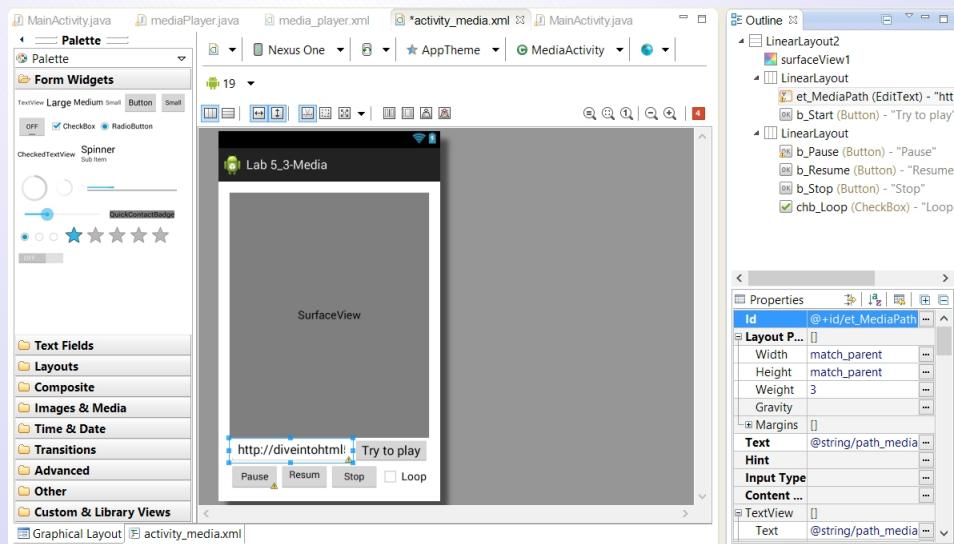


Рисунок 2.Интерфейс активности для воспроизведения аудио и видео

Реализуем логику активности, в данном случае необходимо выполнить следующие действия:

- при нажатии на кнопку **Try to play**, запускается воспроизведение с начала контента, расположенного по адресу, указанному в поле ввода, расположенному слева от кнопки;
- при нажатии на кнопку **Pause**, воспроизведение контента приостанавливается;
- при нажатии на кнопку **Resume**, воспроизведение контента продолжается с места приостановки;
- при нажатии на кнопку **Stop**, воспроизведение контента останавливается и может быть возобновлено только с начала, т. е. необходимо снова нажать **Try to play**.

Элемент **Loop** управляет возможностью повтора воспроизведения.

Работать будем с файлом **MediaActivity.java**, описывающим соответствующий класс активности.

Объявим следующие поля класса активности:

`MediaPlayer mediaPlayer; // управляет воспроизведением`

`CheckBox chbLoop; // управляет режимом повтора`

В методе `onCreate()` активности настроим чек-бокс так, чтобы он включал/выключал режим повтора для плеера.

```
chbLoop = (CheckBox) findViewById(R.id.chb_Loop);
chbLoop.setOnCheckedChangeListener(new
OnCheckedChangeListener() {
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 425 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 426 из 469

Назад

На весь экран

Закрыть

```
@Override
public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
    if (mediaPlayer != null)
        mediaPlayer.setLooping(isChecked);
}
```

Метод `onClickStart()` используется для обработки нажатий на кнопку **Try to play**. В этом методе сначала освобождаем ресурсы текущего проигрывателя, используя *вызов метода*:

```
releaseMP();
```

Этот метод рекомендуется вызывать по окончанию использования плеера, а также при `onPause/onStop`, если нет острой необходимости держать *объект*.

Реализация метода `releaseMP()`:

```
private void releaseMP()
if (mediaPlayer != null) {
    try {
        mediaPlayer.release();
        mediaPlayer = null;
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
}
```

Продолжим рассмотрение метода `onClickStart()`. В нем после освобождения ресурсов, занимаемся подготовкой плеера к воспроизведению.

```
MediaPlayer = new MediaPlayer();
MediaPlayer.setDataSource(DATA);
MediaPlayer.setDisplay(((SurfaceView)
findViewById(R.id.surfaceView1)).getHolder());
MediaPlayer.setOnPreparedListener(this);
MediaPlayer.prepareAsync();
```

Создаем новый *объект* класса `MediaPlayer`.

Используем метод `setDataSource()` для задания источника данных, в качестве параметра передаем строку из поля ввода, сохраненную в переменной `DATA`.

Используя метод `setDisplay()`, задаем экран для воспроизведения, в нашем случае элемент `SurfaceView`, добавленный в окно активности на этапе формирования интерфейса.

Метод `prepareAsync()` выполняет асинхронную подготовку плеера к воспроизведению и когда подготовка будет завершена сообщает об этом слушателю, указанному в методе `setOnPreparedListener()`. В нашем случае слушателем является сам *класс* активности, для этого он объявлен, как *класс* реализующий *интерфейс* `OnPreparedListener`. В случае готовности плеера вызывается метод `onPrepared(MediaPlayer mp)`, объявленный в



Кафедра ПМиМ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 427 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



указанном интерфейсе и реализованный в классе активности, в этом методе выполняется запуск воспроизведения:

```
mp.start();
```

Существует еще метод `prepare()`, он также выполняет подготовку плеера, но в синхронном режиме. Для прослушивания файлов из интернета, необходимо использовать асинхронный режим, что мы и сделали.

Последние две строки метода `onClickStart()`:

```
mediaPlayer.setLooping(chbLoop.isChecked());  
mediaPlayer.setOnCompletionListener(this);
```

В первой из них определяется возможность циклического воспроизведения в зависимости от значения чек-бокса.

Во второй - задается слушатель для получения сообщения о достижении конца воспроизводимого контента. В нашем случае этим слушателем будет сам класс активности для этого он объявлен, как класс реализующий интерфейс `OnCompletionListener`. Полную версию метода `onClickStart()` можно найти в [листе 7.2](#).

Метод `onClick()` вызывается при нажатии на любую кнопку: **Pause**, **Resume**, **Stop**. В этом методе выполняется проверка какая кнопка была нажата и после этого выполняются соответствующие действия.

Если нажата кнопка **Pause**, выполняется следующая конструкция:

```
if (mediaPlayer.isPlaying()) mediaPlayer.pause();
```

Если нажата кнопка **Release**, выполняется следующая конструкция:

```
if (!mediaPlayer.isPlaying()) mediaPlayer.start();
```

Кафедра ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 428 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 429 из 469

Назад

На весь экран

Закрыть

Если нажата кнопка **Stop**, выполняется следующая конструкция:

```
mediaPlayer.stop();
```

Имеет смысл обратить внимание на метод активности:

```
protected void onDestroy() {  
    super.onDestroy();  
    releaseMP();  
}
```

В этом методе обязательно необходимо освободить ресурсы, что и выполняется вызовом метода `releaseMP()`.

Полный код класса `MediaActivity` представлен в [листе 7.2](#).

24.5 Настройка интерфейса и реализация логики активности для просмотра изображений

Настроим *интерфейс* активности для просмотра изображений. Нам потребуется элемент для просмотра изображений, добавим в окно активности элемент `ImageView`. Добавим информационное *поле*, т. е. элемент `TextView`, для отображения информации об общем количестве изображений в папке и номере просматриваемого изображения. Добавим две кнопки, для перемещения от одного изображения к другому вперед и назад.

Предлагаем настроить *интерфейс* самостоятельно. В нашем случае *активность* выглядит так, как показано на [рис 3](#) Но это, разумеется,

не единственно возможный вариант.

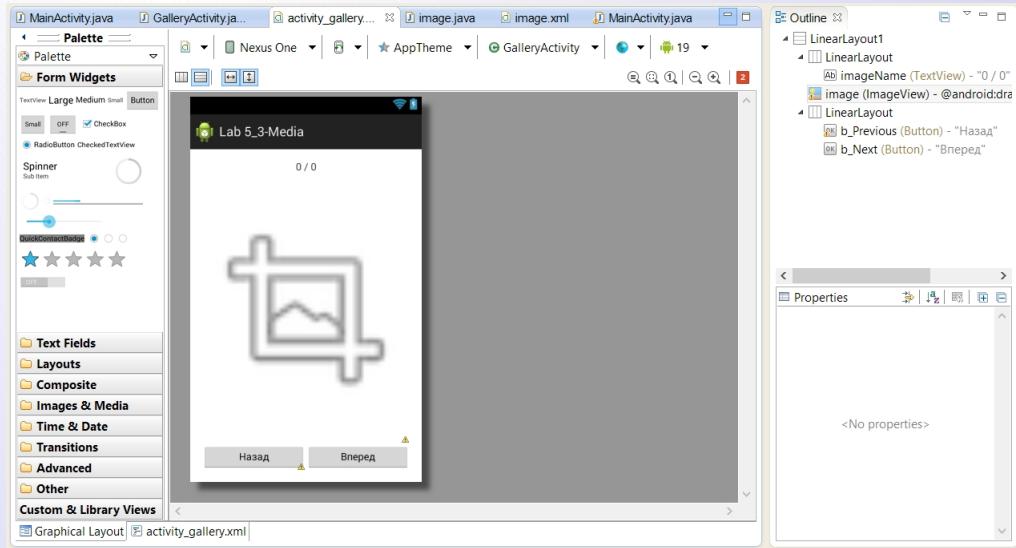


Рисунок 3.Интерфейс активности для просмотра изображений

Реализуем логику активности, данная *активность* ориентирована на просмотр снимков, сделанных в этом же приложении и сохраненных в папке: **/sdcard/TrainingMedia/**. Кнопка **Назад** выводит в окно просмотра предыдущий снимок, по отношению к уже отображеному, зададим свойству **On Click** этой кнопки значение **onPrevious**. Кнопка **Вперед** выводит в окно просмотра следующий снимок, по отношению к



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀ ▶

Страница 430 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 431 из 469

Назад

На весь экран

Закрыть

уже отображенному, зададим свойству **On Click** этой кнопки значение **onNext**.

Работать будем с файлом **GalleryActivity.java**, описывающим класс соответствующей активности.

Объявим поля класса активности:

```
int currentImage=0;  
ArrayList<String> images;  
ImageView imageView;  
TextView nameView;
```

Настройку основных элементов для вывода изображений на экран выполним в методе **onResume()** активности, этот метод вызывается каждый раз, перед выводом активности на передний план (см. жизненный цикл активности в ["Виды приложений и их структура"](#)).

```
images=new ArrayList<String>();  
imageView=((ImageView)findViewById(R.id.image));  
try{  
    File imagesDirectory=new File("/sdcard/TrainingMedia/");  
    images=searchImage(imagesDirectory);  
    updatePhoto(Uri.parse(images.get(currentImage)));  
}catch(Exception e){  
    nameView.setText("Ошибка: Папка '/sdcard/TrainingMedia/' не найдена" );  
}
```



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 432 из 469

Назад

На весь экран

Закрыть

Для получения списка изображений в переменную `images` используется метод `searchImage()`, этот метод, используя переданный в него в качестве параметра *адрес* директории с изображениями, находит *файл* с расширением `.jpg`, `.png` или `.jpeg` и добавляет его к списку изображений. Метод возвращает *список* доступных файлов с изображениями. Код метода представлен в [листе 7.3](#).

Метод `updatePhoto()` выполняет обновление счетчика фотографий в информационном *поле* и выводит на экран изображение, соответствующее переданному в метод *URI*. Код метода представлен в [листе 7.3](#).

В методе `onPause()` активности выполняется очистка списка изображений и *освобождение памяти*. Этот метод вызывается каждый раз, как *активность* теряет фокус ввода. Код метода представлен в [листе 7.3](#).

Осталось рассмотреть еще два метода: `onPrevious()` и `onNext()`.

Первый метод вызывается, когда нажата кнопка **Назад**. В нем уменьшается номер текущего изображения и вызывается метод `updatePhoto()`.

Второй метод вызывается, когда нажата кнопка **Вперед**. В нем увеличивается номер, текущего изображения и вызывается метод `updatePhoto()`.

Полный код класса `GalleryActivity` представлен в [листе 7.3](#).

24.6 Настройка интерфейса и реализация логики главной активности приложения

Настроим *интерфейс* главной активности приложения. Эта *активность* содержит три кнопки (**Image Button**):

- bCamera** - для вызова активности, предоставляющей возможности работы с камерой;
- bGallery** - для вызова активности, предоставляющей возможности просмотра изображений;
- bMusic** - для вызова активности, предоставляющей возможности воспроизведения аудио и видео.

Предлагаем настроить *интерфейс* самостоятельно. В нашем случае *активность* выглядит так, как показано на рис 4. Но это, разумеется, не единственный возможный вариант.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 433 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание



Страница 434 из 469

Назад

На весь экран

Закрыть

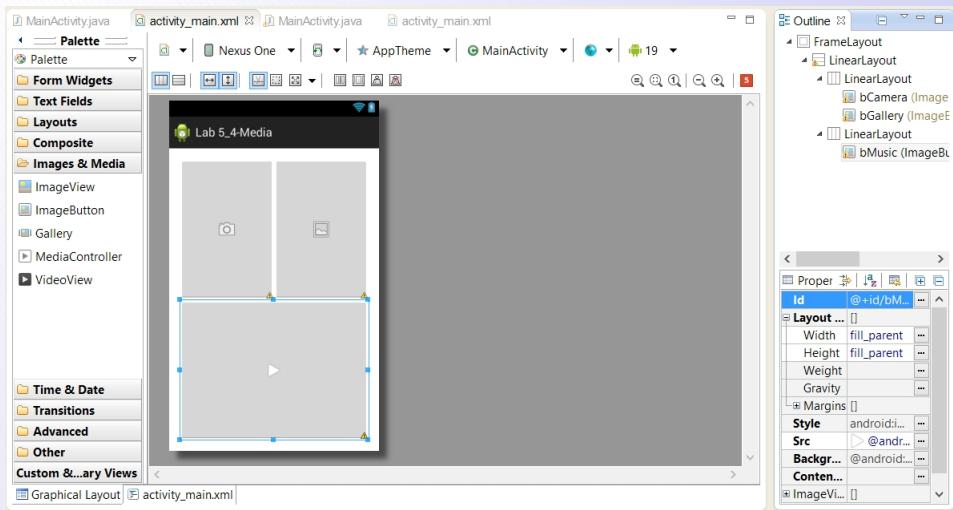


Рисунок 4. Интерфейс главной активности приложения

Реализуем логику главной активности. В данном случае достаточно настроить обработку событий нажатия на кнопки, таким образом, чтобы при нажатии на кнопку запускалась соответствующая активность. Работаем с java файлом, описывающим класс активности.

- Создадим переменную `btnClick`, как реализацию интерфейса-слушателя `OnClickListener`:
 - `OnClickListener btnClick=new OnClickListener() {
 @Override`



```
public void onClick(View v) {  
    Click(v.getId());  
}  
};
```

- Добавим слушателя события нажатия к каждой кнопке активности:

```
- ((ImageButton)findViewById(R.id.bMusic)).  
setOnTouchListener(btnClick);  
((ImageButton)findViewById(R.id.bCamera)).  
setOnTouchListener(btnClick);  
((ImageButton)findViewById(R.id.bGallery)).  
setOnTouchListener(btnClick);
```

- В методе `onClick()` слушателя вызываем метод `Click()`, в котором выполняется запуск соответствующей активности:

```
- protected void Click(int view){ Intent intent=null;  
switch (view){  
case R.id.bMusic: intent=new Intent(this,mediaPlayer.class);  
break;  
case R.id.bGallery: intent=new Intent(this,GalleryActivity.class);  
break;
```

*Кафедра
ПМиИ*

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 435 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 436 из 469

Назад

На весь экран

Закрыть

```
case R.id.bCamera: intent=new Intent(this,CameraActivity.class);
break;
default: break;
}
if(intent!=null){
startActivity(intent);
}
}
```

В [листе 7.4](#) представлен код главной активности.

```
package com.example.lab5_4_media;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import android.hardware.Camera;
import android.hardware.Camera.Size;
import android.os.Bundle;
import android.app.Activity;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.util.Log;
import android.view.SurfaceHolder;
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 437 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

```
import android.view.SurfaceView;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.ViewGroup.LayoutParams;
public class CameraActivity extends Activity
private Camera camera;
private SurfaceHolder surfaceHolder;
private SurfaceView preview;
private View shotBtn;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
// если хотим, чтобы приложение постоянно имело портретную ори-
ентацию
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_
LANDSCAPE);
// если хотим, чтобы приложение было полноэкранным
getWindow().addFlags(WindowManager.LayoutParams.FLAG_
FULLSCREEN);
// и без заголовка
requestWindowFeature(Window.FEATURE_NO_TITLE);
setContentView(R.layout.activity_camera);
```

```
preview = (SurfaceView) findViewById(R.id.surfaceCamera);
surfaceHolder = preview.getHolder();
surfaceHolder.addCallback(new MyCallback(this));
surfaceHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_
BUFFERS);
shotBtn = findViewById(R.id.bCameraShot);
shotBtn.setOnClickListener(new MyViewListener());
}
@Override
protected void onResume() {
super.onResume();
camera = Camera.open();
}
@Override
protected void onPause(){
super.onPause();
if (camera != null){
camera.setPreviewCallback(null);
camera.stopPreview();
camera.release();
camera = null;
}
}
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 438 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

```
class MyCallback implements SurfaceHolder.Callback{  
    Activity host;  
    MyCallback(Activity act){  
        host=act;  
    }  
    @Override  
    public void surfaceChanged(SurfaceHolder holder, int format, int width,  
        int height){}  
    @Override  
    public void surfaceCreated(SurfaceHolder holder){  
        try {  
            camera.setPreviewDisplay(holder);  
            camera.setPreviewCallback(new MyPreviewCallback());  
        }  
        catch (IOException e){  
            Log.d("myLogs " "Ошибка камеры";  
            e.printStackTrace();  
        }  
        Size previewSize = camera.getParameters().getPreviewSize();  
        float aspect = (float) previewSize.width / previewSize.height;  
        int previewSurfaceWidth = preview.getWidth();  
        int previewSurfaceHeight = preview.getHeight();  
        LayoutParams lp = preview.getLayoutParams();
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 439 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

// здесь корректируем размер отображаемого preview, чтобы не было искажений

```
if (host.getResources().getConfiguration().orientation != Configuration.ORIENTATION_LANDSCAPE){  
    // портретный вид  
    camera.setDisplayOrientation(90);  
    lp.height = previewSurfaceHeight;  
    lp.width = (int) (previewSurfaceHeight / aspect);  
}  
else {  
    // ландшафтный  
    camera.setDisplayOrientation(0);  
    lp.width = previewSurfaceWidth;  
    lp.height = (int) (previewSurfaceWidth / aspect);  
}  
preview.setLayoutParams(lp);  
camera.startPreview();  
}  
@Override  
public void surfaceDestroyed(SurfaceHolder holder){}  
}  
class MyViewListener implements View.OnClickListener{  
    @Override
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 440 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 441 из 469

Назад

На весь экран

Закрыть

```
public void onClick(View v) {  
    if (v == shotBtn) {  
        // либо делаем снимок непосредственно здесь // либо включаем об-  
        //работчик автофокуса  
        //camera.takePicture(null, null, null, this);  
        camera.autoFocus(new MyAutoFocusCallback());  
    }  
}  
}  
}  
  
class MyAutoFocusCallback implements Camera.AutoFocusCallback{  
    @Override  
    public void onAutoFocus(boolean paramBoolean, Camera paramCamera){  
        if (paramBoolean){  
            // если удалось сфокусироваться, делаем снимок  
            paramCamera.takePicture(null, null, null, new MyPictureCallback());  
        }  
    }  
}  
}  
  
class MyPictureCallback implements Camera.PictureCallback{  
    @Override  
    public void onPictureTaken(byte[] paramArrayOfByte,  
        Camera paramCamera){  
        // сохраняем полученные jpg в папке /sdcard/CameraExample/
```



Кафедра ПМиИ

```
// имя файла - System.currentTimeMillis()
try {
File saveDir = new File("/sdcard/CameraExample/";
if (!saveDir.exists()) {
saveDir.mkdirs();
}
FileOutputStream os = new FileOutputStream(String.format
("/sdcard/CameraExample/System.currentTimeMillis()));
os.write(paramArrayOfByte);
os.close();
}
catch (Exception e) {}
// после того, как снимок сделан, показ превью отключается.
// Необходимо включить его paramCamera.startPreview();
}

}

class MyPreviewCallback implements Camera.PreviewCallback{
@Override
public void onPreviewFrame(byte[] paramArrayOfByte, Camera
paramCamera) {
// здесь можно обрабатывать изображение, показываемое в preview
}
}
```

Начало

Содержание

◀ ▶

◀▶

Страница 442 из 469

Назад

На весь экран

Закрыть

}

Листинг 17.1. Класс CameraActivity для работы с камерой

```
package com.example.lab5_4_media;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.media.MediaPlayer.OnPreparedListener;
import android.os.Bundle;
import android.view.Gravity;
import android.view.SurfaceView;
import android.view.View;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.Toast;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.app.Activity;
public class MediaActivity extends Activity implements OnPreparedListener,
OnCompletionListener{
    MediaPlayer mediaPlayer;
    CheckBox chbLoop;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 443 из 469

Назад

На весь экран

Закрыть

```
setContentView(R.layout.activity_media);
chbLoop = (CheckBox) findViewById(R.id.chb_Loop);
chbLoop.setOnCheckedChangeListener(new OnCheckedChangeListener()
{
    @Override
    public void onCheckedChanged(CompoundButton buttonView,
        boolean isChecked) {
        if (mediaPlayer != null)
            mediaPlayer.setLooping(isChecked);
    }
});
}

public void onClickStart(View view) {
    releaseMP();
    String DATA=((EditText)findViewById(R.id.et_MediaPath)).getText().
    toString();
    try {
        mediaPlayer = new MediaPlayer();
        mediaPlayer.setDataSource(DATA);
        mediaPlayer.setDisplay(((SurfaceView)
        findViewById(R.id.surfaceView1)).getHolder());
        //mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
        mediaPlayer.setOnPreparedListener(this);
    }
}
```



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 444 из 469

Назад

На весь экран

Закрыть

```
mediaPlayer.prepareAsync();
} catch (Exception e) {
showMessage("Ошибка воспроизведения");
}
if (mediaPlayer == null)
return;
mediaPlayer.setLooping(chbLoop.isChecked());
mediaPlayer.setOnCompletionListener(this);
}
private void showMessage(String text){
Toast toast = Toast.makeText(getApplicationContext(), text,
Toast.LENGTH_SHORT);
toast.setGravity(Gravity.CENTER, 0, 0);
toast.show();
}
private void releaseMP() {
if (mediaPlayer != null) {
try {
mediaPlayer.release();
mediaPlayer = null;
} catch (Exception e) {
e.printStackTrace();
}
}
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 445 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

```
}

}

public void onClick(View view) {
    if (mediaPlayer == null)
        return;
    switch (view.getId()) {
        case R.id.b_Pause:
            if (mediaPlayer.isPlaying())
                mediaPlayer.pause();
            break;
        case R.id.b_Resume:
            if (!mediaPlayer.isPlaying())
                mediaPlayer.start();
            break;
        case R.id.b_Stop:
            mediaPlayer.stop();
            break;
    }
}

@Override
public void onPrepared(MediaPlayer mp) {
    mp.start();
}
```



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 446 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

```
@Override
public void onCompletion(MediaPlayer mp) {}

@Override
protected void onDestroy() {
    super.onDestroy();
    releaseMP();
}
}
```

Листинг 7.2. Класс MediaActivity

```
package com.example.lab5_4_media;
import java.io.File;
import java.util.ArrayList;
import android.net.Uri;
import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
public class GalleryActivity extends Activity
{
    int currentImage=0;
    ArrayList<String> images;
```

[Начало](#)

[Содержание](#)

[◀](#)

[▶](#)

[◀◀](#)

[▶▶](#)

Страница 447 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

```
ImageView imageView;
TextView nameView;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_gallery);
}
@Override
public void onResume(){
super.onResume();
currentImage=0;
Log.d("myLogs "onResume cI="+currentImage);
nameView=((TextView)findViewById(R.id.imageName));
images=new ArrayList<String>();
imageView=((ImageView)findViewById(R.id.image));
try{
File imagesDirectory=new File("/sdcard/TrainingMedia/");
images=searchImage(imagesDirectory);
updatePhoto(Uri.parse(images.get(currentImage)));
}catch(Exception e){
nameView.setText("Ошибка: Папка '/sdcard/TrainingMedia/' не найдена");
Log.d("myLogs" "Ошибка";

```



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 448 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

```
}

}

@Override
protected void onPause()
{ super.onPause() ; images.clear();
Log.d("myLogs "onPause cI="currentImage);
}
private ArrayList<String> searchImage(File dir){
ArrayList<String> imagesFinded=new ArrayList<String>();
for(File f:dir.listFiles()){
if(!f.isDirectory()){
String fileExt=getFileExt(f.getAbsolutePath());
if(fileExt.equals("png" || fileExt.equals("jpg" || fileExt.equals("jpeg")){
Log.d("myLogs "Файл найден " +f.getAbsolutePath());
imagesFinded.add(f.getAbsolutePath());
}
}
}
return imagesFinded;
}
public static String getFileExt(String filename){
return filename.substring(filename.lastIndexOf("." + 1);
}
```

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 449 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

```
public void updatePhoto(Uri uri){  
try{  
nameView.setText((currentImage+1)+"/" +images.size());  
imageView.setImageURI(uri);  
}catch(Exception e){ nameView.setText("Ошибка загрузки файла";  
}  
}  
}  
  
public void onNext(View v){  
if(currentImage+1<images.size() images.size()>0){  
currentImage++;  
updatePhoto(Uri.parse(images.get(currentImage)));  
}  
}  
  
public void onPrevious(View v){  
if(currentImage>0 images.size()>0){  
currentImage--;  
updatePhoto(Uri.parse(images.get(currentImage)));  
}  
}  
}
```

Листинг 7.3. Класс GalleryActivity

```
package com.example.lab5_4_media;  
import android.os.Bundle;
```



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 450 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 451 из 469

Назад

На весь экран

Закрыть

```
import android.app.Activity;
import android.content.Intent;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;
public class MainActivity extends Activity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
OnClickListener btnClick=new OnClickListener() {
@Override
public void onClick(View v) {
Log.d("myLogs" + v.getId() + ;
Click(v.getId());
}
};
((ImageButton)findViewById(R.id.bMusic)).setOnClickListener
(btnClick);
((ImageButton)findViewById(R.id.bCamera)).setOnClickListener(
btnClick);
```



Кафедра
ПМиИ

```
((ImageButton)findViewById(R.id.bGallery)).setOnClickListener  
(btnClick);  
}  
protected void Click(int view){  
Intent intent=null;  
Log.d("myLogs" view+;  
switch (view){  
case R.id.bMusic: intent=new Intent(this,MediaActivity.class);  
break;  
case R.id.bGallery: intent=new Intent(this,GalleryActivity.class);  
break;  
case R.id.bCamera: intent=new Intent(this,CameraActivity.class);  
break;  
default: break;  
}  
if(intent!=null){  
Log.d("myLogs" "Интент = "intent.toString());  
startActivity(intent);  
}  
}  
}  
}
```

Листинг 7.4. Класс MainActivity

Начало

Содержание



Страница 452 из 469

Назад

На весь экран

Закрыть

§25. Лабораторная работа №8

Геолокационные возможности

Аннотация: Разработка приложения, демонстрирующего геолокационные возможности.

Цель лабораторной работы:

Разработка приложения, демонстрирующего геолокационные возможности.

Задачи лабораторной работы:

- Разработать приложение, получающее координаты устройства и отслеживающее их изменения.

25.1 Введение

В данной работе рассмотрим процесс создания приложения.

25.2 Разработка приложения, получающего координаты устройства и отслеживающего их изменение

Создадим *приложение*.

Далее будем править *java файл*, определяющий *класс* активности приложения. Внесем в этот *класс* следующие дополнения:

- Нам потребуется экземпляр класса **LocationManager**, поэтому в ме-



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 453 из 469

Назад

На весь экран

Закрыть



Кафедра ПМиИ

[Начало](#)

[Содержание](#)



Страница 454 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

тоде `onCreate()` запишем следующую конструкцию:

- `LocationManager mlocManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);`

экземпляр класса `LocationManager`, как и большинство системных сервисов, создается с помощью вызова метода `getSystemService()`.

- Укажем, что в приложении необходимо получать обновления координат, сделаем это с помощью вызова метода `requestLocationUpdates()` класса `LocationManager`:
 - `mlocManager.requestLocationUpdates
(LocationManager.GPS_PROVIDER, 0, 0, mlocListener);`

первый параметр метода указывает способ получения координат, для этого используются константы класса `LocationManager`:

`GPS_PROVIDER`

- сообщает, что координаты определяются с помощью GPS;

`NETWORK_PROVIDER`

- сообщает, что координаты определяются с использованием сигналов сотовых вышек и WiFi.

В случае, когда необходимо получать координаты и с GPS, и от сотовых



вых вышек необходимо вызвать метод `requestLocationUpdates()` дважды: один раз первый параметр должен быть равен `GPS_PROVIDER`, второй раз - `NETWORK_PROVIDER`.

Второй и третий параметры метода управляют частотой обновлений. Второй определяет минимальное время между извещениями об обновлениях, третий - минимальное изменение расстояния между извещениями об обновлениях. Если оба эти параметра имеют нулевое значение, то извещения об обновлениях появляются настолько часто, насколько это возможно.

Последний параметр указывает на слушателя, который получает вызовы при обновлениях координат. В нашем случае слушателем является переменная `mlocListener` - реализация интерфейса `LocationListener`.

- Теперь необходимо объявить переменную `mlocListener`:

```
- LocationListener mlocListener = new LocationListener(){  
    public void onLocationChanged(Location location) {  
        tvLat.append(location.getLatitude());  
        tvLon.append(location.getLongitude());  
    }  
    public void onStatusChanged(String provider, int status,  
        Bundle extras) {}  
    public void onProviderEnabled(String provider) {}
```

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

[Страница 455 из 469](#)

[Назад](#)

[На весь экран](#)

[Закрыть](#)

```
public void onProviderDisabled(String provider) {}  
};
```

Переменная `mlocListener` инициализируется реализацией интерфейса `LocationListener`. Этот интерфейс содержит 4 метода, каждый из которых должен быть определен в реализации интерфейса. Нас интересует метод `onLocationChanged()`, который вызывается каждый раз при изменении показаний GPS датчика, в этом методе всего лишь выполняется вывод полученных координат в информационные поля.

- Чтобы разрешить получать обновления координат с помощью сигналов от сотовых вышек (`NETWORK_PROVIDER`) или с GPS датчика (`GPS_PROVIDER`), необходимо в файле манифеста добавить строку:
 - `<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />`
(для работы с сигналами сотовых вышек) или
 - `<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />`
(для работы GPS датчиком).

Без этих полномочий приложение "сломается" во время выполнения, когда попробует запросить обновление координат. Если в приложении в



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 456 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 457 из 469

Назад

На весь экран

Закрыть

качестве источника координат используются и **NETWORK_PROVIDER**, и **GPS_PROVIDER**, в манифесте достаточно запросить только полномочия на **ACCESS_FINE_LOCATION**. Тогда как **ACCESS_COARSE_LOCATION** позволяет работать только с **NETWORK_PROVIDER**.

В [листе 8.1](#) представлен код приложения, позволяющего получать *координаты* устройства и отслеживать их изменения, используя *GPS* приемник.

После создания приложения, разумеется, необходимо протестировать его работу. Проще всего это сделать, используя реальное устройство под управлением *Android*, но если устройства под рукой нет, можно использовать *виртуальное* устройство. Однако при этом необходимо решить вопрос, каким образом имитировать передачу данных о местоположении на эмулятор. Проще всего воспользоваться для этого *DDMS*.

В Eclipse откройте компоновку (*Perspective*) *DDMS*, в этой компоновке выберите вкладку **Emulator Control**. С помощью инструмента *DDMS*, можно имитировать обновление данных о местоположении несколькими способами:

- вручную передать значения широты и долготы на виртуальное устройство;
- использовать GPX файл, описывающий маршрут для считывания эмулятором;

- использовать KML файл, описывающий отдельные метки местности для последовательной передачи на виртуальное устройство.

25.3 Заключение

В этой части работы рассмотрели вопросы разработки приложений, способных получать *координаты* устройства и отслеживать их изменения. Разработанное *приложение* реагирует на изменения координат устройства и выдает новые *координаты* в соответствующие информационные поля.

```
package com.example.lab5_4_geolocation;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.widget.TextView;
public class MainActivity extends Activity {
    TextView tvOut;
    TextView tvLon;
    TextView tvLat;
    @Override
```



Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 458 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМиИ

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    tvOut = (TextView)findViewById(R.id.textView1);  
    tvLon = (TextView)findViewById(R.id.longitude);  
    tvLat = (TextView)findViewById(R.id.latitude);  
    //Получаем сервис  
    LocationManager mlocManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);  
    LocationListener mlocListener = new LocationListener(){  
        public void onLocationChanged(Location location) {  
            //Called when a new location is found by the network location provider.  
            tvLat.append(location.getLatitude());  
            tvLon.append(location.getLongitude());  
        }  
        public void onStatusChanged(String provider, int status, Bundle extras)  
    }  
    public void onProviderEnabled(String provider) {}  
    public void onProviderDisabled(String provider) {}  
};  
//Подписываемся на изменения в показаниях датчика  
mlocManager.requestLocationUpdates  
(LocationManager.GPS_PROVIDER, 0, 0, mlocListener);
```

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 459 из 469

Назад

На весь экран

Закрыть

```
//Если gps включен, то ... , иначе вывести "GPS is not turned on..."  
if (mlocManager.isProviderEnabled(  
LocationManager.GPS_PROVIDER))  
{  
    tvOut.setText("GPS is turned on...");  
    else {  
        tvOut.setText("GPS is not turned on...");  
    }  
}  
}  
}
```

Листинг 8.1. Получение геолокационных данных



Кафедра
ПМиИ

Начало

Содержание

◀

▶

◀◀

▶▶

Страница 460 из 469

Назад

На весь экран

Закрыть



Вопросы и задания для самоконтроля

Выполните пожалуйста этот тест

*Кафедра
ПМиИ*

Начало

Содержание

◀

▶

◀◀

▶▶

Страница 461 из 469

Назад

На весь экран

Закрыть



Литература

1. Мобильная платформа Android: пять лет истории [Электронный ресурс]/Мобильная платформа Android. - Режим доступа: <http://itc.ua/articles/mobilnaya-platforma-android-pyat-let-istorii/>. - Дата доступа: 9.08.2015.
2. Что это за Android? [Электронный ресурс]/ Что это за Android? - Режим доступа: http://android.com.ua/android_os.html. - Дата доступа: 9.08.2015.
3. Версии Android [Электронный ресурс]/ Версии Android. - Режим доступа: <http://android-phones.ru/android/>. - Дата доступа: 8.09.2015.
4. Подробный обзор Android 4.3 [Электронный ресурс]/ Подробный обзор Android 4.3. - Режим доступа: <http://habrahabr.ru/post/188344/>. - Дата доступа: 9.08.2015.
5. Архитектура операционной системы Android [Электронный ресурс] / Архитектура операционной системы Android. - Режим доступа: <http://android-shark.ru/arkhitektura-operatsionnoy-sistemyi-android/>. - Дата доступа: 9.08.2015.
6. Майер, Р. Android 2 : программирование приложений для планшетных компьютеров и смартфонов/ Р.Майер. - Москва: Эксмо, 2011. - 672 с.

Кафедра
ПМиИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 462 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)



Кафедра ПМИИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 463 из 469

Назад

На весь экран

Закрыть

7. Android для новичков: что такое виджет? [Электронный ресурс] / Android для новичков: что такое виджет? - Режим доступа: <http://www.androidpit.ru/chto-takoe-vidzhet>. - Дата доступа: 9.08.2015.
8. Application Fundamentals [Электронный ресурс] / Application Fundamentals. - Режим доступа: <http://developer.android.com/guide/components/fundamentals.html>. - Дата доступа: 9.08.2015.
9. Введение в разработку для платформы Android [Электронный ресурс] / Введение в разработку для платформы Android. - Режим доступа: <http://www.ibm.com/developerworks/ru/library/os-android-devel/>. - Дата доступа: 9.08.2015.
10. Тидвелл, Дж. Разработка пользовательских интерфейсов / Дж. Тидвелл. - Москва: СПб, Питер, 2008. - 416 с.
11. Кронин, Д. Аллан Купер об интерфейсе. Основы проектирования взаимодействия / Д. Кронин, А. Купер, Р. Рейман. - СПб.: Символ'Плюс, 2009. – 688 с.
12. Универсальное разрешение Android: идеально на всех экранах [Электронный ресурс] / Универсальное разрешение Android: идеально на всех экранах. - Режим доступа: <http://habrahabr.ru/post/177093/>. - Дата доступа: 9.08.2015.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀▶

Страница 464 из 469

Назад

На весь экран

Закрыть

13. Android M developer Preview [Electronic resource] / Android M developer Preview. - Mode of access: <http://developer.android.com/index.html>. - Date of access: 9.08.2015.
14. Dialogs [Electronic resource] / Developer Android. - Mode of access: <http://developer.android.com/guide/topics/ui/dialogs.html>. - Date of access: 9.08.2015.
15. Программирование под Android / З. Медникс [и др.]. - СПб.: Питер, 2012. - 496 с.
16. [Азбука] Жизнь на кончиках пальцев. Тактильное взаимодействие с сенсорным экраном смартфона [Электронный ресурс] / CMS magazine. - 2013. - Режим доступа: <http://www.cmsmagazine.ru/library/items/mobile/tactile-interaction/>. - Дата доступа: 9.08.2015.
17. Особенности проектирования тачевых интерфейсов [Электронный ресурс] / Особенности проектирования тачевых интерфейсов. - Режим доступа: <http://habrahabr.ru/post/150905/>. - Дата доступа: 9.08.2015.
18. Реализация сенсорного интерфейса в новых и существующих играх [Электронный ресурс] / Реализация сенсорного интерфейса в новых и существующих играх. - Режим доступа: <http://software.intel.com/ru-ru/node/394259>. - Дата доступа: 9.08.2015.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)

[◀](#)

[▶](#)

[◀◀](#)

[▶▶](#)

Страница 465 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

19. Датчики и сенсоры современных мобильных устройств [Электронный ресурс] / INFOCITY. - 2013. - Режим доступа: <http://www.infocity.az/?p=8233>. - Дата доступа: 9.08.2015.
20. Введение в Gestures [Электронный ресурс] / Введение в Gestures. - Режим доступа: <http://habrahabr.ru/post/120016/>. - Дата доступа: 9.08.2015.
21. Android Research Blog [Electronic resource] / Android Research Blog. - Mode of access: <http://androidresearch.wordpress.com/tag/gesture-builder/>. - Date of access: 9.08.2015.
22. android.gesture [Electronic resource] / android.gesture. - Mode of access: <http://developer.android.com/reference/android/gesture/package-summary.html>. - Date of access: 9.08.2015.
23. Location and Sensors APIs [Electronic resource] / Location and Sensors APIs. - Mode of access: <http://developer.android.com/guide/topics/sensors/index.html>. - Date of access: 9.08.2015.
24. Location Strategies [Electronic resource] / Location Strategies. - Mode of access: <http://developer.android.com/guide/topics/location/strategies.html>. - Date of access: 9.08.2015.
25. Библиотеки [Электронный ресурс] / Сайт Александра Климова. - Режим доступа: <http://developer.alexanderklimov.ru/android/library/>.



Кафедра
ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 466 из 469

Назад

На весь экран

Закрыть

- Дата доступа: 9.08.2015.
- 26. Android Support Library [Electronic resource] / Support Library. - Mode of access: <http://developer.android.com/tools/support-library/index.html#overview>. - Дата доступа: 9.08.2015.
- 27. Библиотека NineOldAndroids [Электронный ресурс] / NineOldAndroids. - Режим доступа: <http://nineoldandroids.com/>. - Дата доступа: 9.08.2015.
- 28. Библиотека ActionBarSherlock [Электронный ресурс] / Библиотека ActionBarSherlock. - Режим доступа: <http://actionbarsherlock.com/>. - Дата доступа: 9.08.2015.
- 29. Библиотека Яндекс-метрика [Электронный ресурс] / yandex.ru. - Режим доступа: <http://api.yandex.ru/metrica-mobile-sdk/>. - Дата доступа: 9.08.2015.
- 30. Facebook SDK for Android [Electronic resource] / Facebook SDK for Android. - Mode of access: <https://developers.facebook.com/docs/android>. - Date of access: 9.08.2015.
- 31. Universal Image Loader for Android [Electronic resource] / Universal Image Loader for Android. - Mode of access: <https://github.com/nostra13/Android-Universal-Image-Loader>. - Date of access: 9.08.2015.



Кафедра ПМиИ

Начало

Содержание

◀ ▶

◀◀ ▶▶

Страница 467 из 469

Назад

На весь экран

Закрыть

32. jsoup: Java HTML Parser [Electronic resource] / jsoup: Java HTML Parser. - Mode of access: <http://jsoup.org/>. - Date of access: 9.08.2015.
33. Android Holo ColorPicker [Electronic resource] / Android Holo ColorPicker. - Mode of access: <https://github.com/LarsWerkman/HoloColorPicker>. - Date of access: 9.08.2015.
34. MapNavigator [Electronic resource] / MapNavigator. - Mode of access: <https://github.com/tyczj/MapNavigator>. - Date of access: 9.08.2015.
35. AChartEngine [Electronic resource] / AChartEngine. - Mode of access: <http://code.google.com/p/achartengine/>. - Date of access: 9.08.2015.
36. Обзор Android-библиотек [Электронный ресурс] / Обзор Android-библиотек. - Режим доступа: <http://www.androidviews.net/>. - Дата доступа: 9.08.2015.
37. Ad Vulna: A Vulnaggressive (Vulnerable & Aggressive) Adware Threatening Millions [Electronic resource] / freeeye.com. - Mode of access: <http://www.fireeye.com/blog/technical/2013/10/ad-vulna-a-vulnaggressive-vulnerable-aggressive-adware-threatening-millions.html>. - Date of access: 9.08.2015.
38. Бурнет, Э. Привет, Android! Разработка мобильных приложений / Э. Бурнет, - СПб: Питер, 2012. - 256 с.



Кафедра ПМИ

[Начало](#)

[Содержание](#)

[◀](#) [▶](#)

[◀◀](#) [▶▶](#)

Страница 468 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

39. Понимание SQL (Understanding SQL). [Электронный ресурс] / Понимание SQL. - Режим доступа: http://www.sql.ru/docs/sql/u_sql/. - Дата доступа: 9.08.2015.
40. Animation and Graphics Overview [Electronic resource] / Animation and Graphics Overview. - Mode of access: <http://developer.android.com/guide/topics/graphics/overview.html>. - Date of access: 9.08.2015.
41. Основные принципы разработки игр [Электронный ресурс] / Основные принципы разработки игр. - Режим доступа: <http://habrahabr.ru/post/188372/>. - Дата доступа: 9.08.2015.
42. Место сценариста в команде разработки игр [Электронный ресурс] / Место сценариста в команде разработки игр. - Режим доступа: <http://habrahabr.ru/company/mailru/blog/197152/>. - Дата доступа: 9.08.2015.
43. Создание Игр Для Начинающих [Электронный ресурс] / 3dg.me. - Режим доступа: <http://3dg.me/ru/gamedev/basics/sozdanie-igr-dlya-nachinayushchih>. - Дата доступа: 9.08.2015.
44. Суровый геймдев на примере трех игр для Android [Электронный ресурс] / Суровый геймдев на примере трех игр для Android. - Режим доступа: <http://habrahabr.ru/post/195828/>. - Дата доступа: 9.08.2015.



Кафедра ПМиИ

[Начало](#)

[Содержание](#)



Страница 469 из 469

[Назад](#)

[На весь экран](#)

[Закрыть](#)

45. 4pda.ru [Электронный ресурс] / 4pda.ru. - Режим доступа: <http://4pda.ru/forum/lofiversion/index.php?t315915-0.html>. - Дата доступа: 9.08.2015.
46. Блог Андрея Монахова [Электронный ресурс] / Блог Андрея Монахова. - Режим доступа: <http://andmonahov.blogspot.se/2012/03/glsurfaceview.html>. - Дата доступа: 9.08.2015.
47. Peter Wayner [Electronic resource] / InfoWorld. - Mode of access: <http://www.itworld.com/software/383227/10-reasons-browser-becoming-universal-os>. - Date of access: 9.08.2015.
48. HTML [Electronic resource] / ru.wikipedia. - Mode of access: <http://ru.wikipedia.org/wiki/HTML>. - Date of access: 9.08.2015.
49. Основы интернет-технологий: учебное пособие / А.А. Липницкий [и др.]. - Сев. (Арктич.) федер. ун-т. Архангельск: ИПЦ САФУ, 2013. – 365 с.
50. Node.js [Electronic resource] / Node.js. - Mode of access: <http://ru.wikipedia.org/wiki/Nodejs>. - Date of access: 9.08.2015.
51. Видеоролики по работе с Intel XDK [Электронный ресурс] / Видеоролики по работе с Intel XDK.- Режим доступа: <http://software.intel.com/ru-ru/html5/tools#tab-panel-begin>. - Дата доступа: 9.08.2015.