МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра компьютерных технологий

«Допущен к защите»
Заведующий кафедрой
Куралбаев З.К. профессор
_____ «_____»_____20__ г.
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: «Исследование разработки мобильных приложений»

Специальность «Вычислительная техника и программное обеспечение»

Выполнил Наурызбаев А.А. группа ВТу-11-1

Научный руководитель Турганбаев Е.С. доцент каф.КТ

Консультанты:
по экономической части:
Еркешева З.Д ст. преп.
_____ «21» мая 20 14 г.
(подпись)
по безопасности жизнедеятельности:
Бегимбетова А.С
_____ «28» мая 20 14 г.
(подпись)
по применению вычислительной техники:
_____
(Фамилия и инициалы, ученая степень, звание)
_____ «_____»_____20___г.
(подпись)

Нормоконтролер: Тусупов Д.М. ассистент
_____ «_____»_____20___г.
(подпись)

Рецензент: Бейбатшаев М.Ш
_____ «_____»_____20___г.
(подпись)

Алматы 2014 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра компьютерных технологий

ЗАДАНИЕ
на выполнение дипломного проекта

Студент Наурызбаев Алибек Арыстанович
Тема проекта «Исследование разработки мобильных приложений»
утверждена приказом ректора № 115 от «24» сентября 20___ г.
Срок сдачи законченной работы «___» _____ 20___ г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта
Разработать мобильное приложение с функцией GPS для получения гео-данных.

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:
1. Анализ мобильных ОС
2. Выбор языка программирования и рабочей среды
3. Создание схемы проекта
4. Реализация

Перечень графического материала (с точным указанием обязательных чертежей)

Рисунок архитектуры мобильных операционных систем
Рисунок интерфейса программного обеспечения
Рисунок интерфейса приложения
Рисунок интерфейса WINDOWS PHONE
Таблица команд
Таблица с данными раздела „Экономика"
Таблица с данными раздела „БЖД"

Рекомендуемая основная литература:

1 Библиотека профессионала Java 2 том 1. Кей С.Хорстманн. Гарри Корнелл.Москва 2003;

2 Библиотека профессионала Java 2 том 2. Кей С.Хорстманн. Гарри Корнелл.Москва 2005;

3 Применение Шаблонов Java. Библиотека профессионалов. Стивен Стетлинг Издательский дом Вильямс 2006;

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
Экономика	Ермекова З.Д.	Ермекова	Ермекова
БЖД	Бегимбетова А.С.		

Г Р А Ф И К

подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1	Постановка задачи	05.04.14	
2	Выбор среды программирования	10.04.14	
3	Сбор и изучение литературы	15.04.14	
4	Создание интерфейса	01.05.14	
5	Разработка главной формы	05.05.14	
6	Тестирование ПО	05.05.14	
7	Отладка ПП	15.05.14	
8	Подготовка справки "Экономика"	20.05.14	
9	Подготовка справки "Экология"	28.05.14	
10	Проверка и сдача проекта.	29.05.14.	

Дата выдачи задания «03» марта 2014 г.

Заведующий кафедрой _____ Курманбаев З.К.
(подпись) (Фамилия и инициалы)

Руководитель _____ Тургамбаев Е.С.
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ Жаурыкбаев А.А.
(подпись) (Фамилия и инициалы)

Аннотация

В данном дипломном проекте мы рассмотрим GPS программу, в системе android, для получения гео-данных местонахождения.

Кроме того, сделан анализ условий труда при эксплуатации программного продукта.

В экономической части составлен бизнес план проекта, подтвердивший его экономическую целесообразность.

Аңдатпа

Осы дипломдық жобада біз GPS бағдарлама android жүйесінде геолокациясы мекен жайы мәлімет аламыз.

Мұнан басқа бағдарламалық өнімді пайдалану кезіндегі еңбек жағдайларына сараптама жасалды.

Экономикалық бөлімде жобаның экономикалық тиімділігін дәлелдеп берген бизнес-жоспар жасалды.

Anotation

In this thesis project, we consider the GPS program in the system android, for geo-location data.

In addition, the analysis made by the working conditions in the operation of software.

In the economic part of the project, a business plan, which confirmed its economic feasibility.

Сод ержание

Введение

Актуальность темы исследования. Сейчас перед нами, как разработчиками приложений для мобильных платформ, открываются потрясающие возможности. Еще пару лет назад казалось невероятным, что обычный разработчик сможет продавать свои приложения миллионам пользователей по всему миру с помощью магазинов приложений (Marketplace), а пользователи будут эти приложения покупать. По разным оценкам объем мирового рынка мобильных приложений в 2011 году составил от 9 до 12 млрд долларов США, и в ближайшие пять лет этот показатель вырастет в 4 раза. Одной из причин является то, что конкуренция между основными производителями операционных систем для смартфонов, такими как Microsoft с платформой Windows Phone, Apple с ios и Google с Android, непрерывно усиливается. Это позволяет предположить, что в будущем данные платформы продолжат развиваться быстрыми темпами, а значит, потребность в разработчиках мобильных приложений станет только расти.

Microsoft уже в течение многих лет создает операционные системы для смартфонов. Однако Windows Phone - это полное переосмысление платформы. Ранее мобильные операционные системы от Microsoft назывались Windows Mobile (последняя версия 6.5.3), а до этого Pocket PC (2000 и 2002). Однако Microsoft столкнулась с тем, что старые подходы и принципы уже не удовлетворяют новым запросам пользователей. Поэтому Microsoft решила начать с нуля и создать операционную систему Windows Phone, которая не совместима с Windows Mobile ни с точки зрения пользователя, ни с точки зрения разработчика. Единственное общее у данных операционных систем то, что в их остове лежит ядро Windows Compact Edition, но ни пользователи Windows Phone, ни разработчики с Windows Compact Edition напрямую не взаимодействуют и взаимодействовать не могут. Приложения для Windows Mobile не работают на Windows Phone и наоборот. Windows Phone обладает новым пользовательским интерфейсом, построенным на принципах Metro-дизайна, что выгодно отличает данную платформу от других мобильных операционных систем. Операционная система Windows 8 также имеет пользовательский интерфейс, основанный на Metro-дизайне, как и последние версии консоли XЬox 360. Таким образом, Microsoft стремится унифицировать пользовательский интерфейс своих продуктов.

Раньше, когда система android не была совершенной, случались проблемы крэш системы, плохая оптимизация и так далее. Программировать было достаточно тяжело, но с выходом android V.2 можно сказать, что мир изменился и почти каждый выход новой ОС на андроиде доказывал, что Google стремится сделать лучше в целом структуру организации работы ОС.

Так же на данный момент мы видим таких монстров системы как Eclipse и android studio, которые набирают обороты и набирают больше сторонников системы android, в данном случае это програмисты всё же система является

бесплатной и за ней стоит Google. В то время как за ситемой VS стоит монстр Microsoft за которым огромные деньги. Но в принципе языки между собой очень схожи оба являются кроссплатформенными, но я со своей стороны решил отдать большее предпочтение Java чем c#, так как имеется сотовый аппарат на android для тестирования данных. Да и вообще на данный момент windowsphone последняя версия 8.1 против android 4.4 kitkat большее предпочтение на данный момент люди отдают андроидовидным телефонам нежели windowsphone, так же IOS является одним из конкурентов anroid, но по последним данным даже iphone уже не столь популярен как телефоны на android. Данная система щас везде в коммуникаторах, сотовых, планшетах, телевизорах и можно бесконечно перечислять данны виды оборудования, но факт на лицо и будущее по моему мнению стоит за Google.

Таким образом, учитывая вышеуказанные особенности android и дальнейшие перспективы развития программирования мобильных приложений, можно с уверенностью сказать, что разработка учебного курса найдет широкое применение в учебном процессе АУЭС. Также с течением времени спрос на программистов-разработчиков будет неуклонно расти. Этим и обусловлена актуальность данной работы.

Целью данной работы было изучение операционной системы android, программирование мобильного приложения и разработка методических указаний, которые будут внедрены в учебный процесс Алматинского Университета энергетики и связи.

1 Предметная область

1.1 Описание

Наш мир переживает революцию в мире компьютерных технологий каждый день выходит новая технология смартфонов, уплотнения пикселей экрана или же матрица для сотовых, телевизоров и т.д., угнаться за всем просто невозможно, но из всего что есть можно выделить одних из «акул» IT-бизнеса это 4 могущественные корпорации, такие как Nokia (SymbianOS), apple (IOS), Google (android), Microsoft (WindowsPhone 8).

Android - операционная система для смартфонов, планшетных компьютеров, электронных книг, цифровых проигрывателей, наручных часов,игровых приставок, нетбуков, смартбуков, очков Google и других устройств. Основана на ядре Linux и собственной реализации Java от Google. Изначально разрабатывалась компанией Android Inc., которую затем купила Google. Впоследствии Google инициировала создание альянса Open Handset Alliance (OHA), который сейчас занимается поддержкой и дальнейшим развитием платформы. Android позволяет создавать Java-приложения, управляющие устройством через разработанные Google библиотеки. Android Native Development Kit позволяет портировать (но не отлаживать) библиотеки и компоненты приложений, написанные на Си и других языках.

В 81,3 % смартфонов, проданных в третьем квартале 2013 года, была установлена операционная система Android.

На данном рисунке 1 мы видим обновление версий в разрезе 2008 по 2014 г.

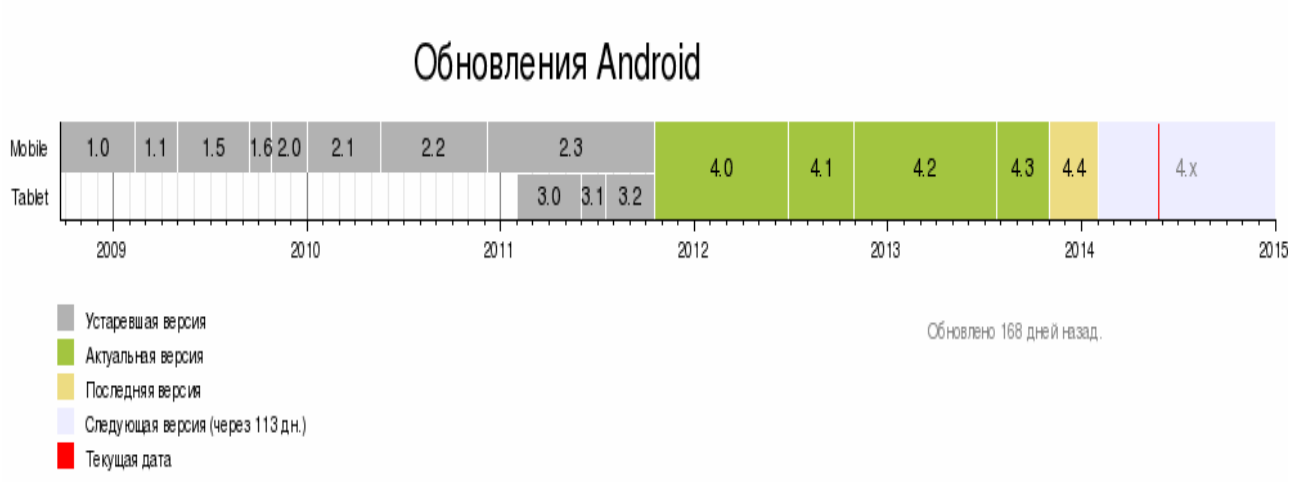

Рисунок 1 - Обновление версий android

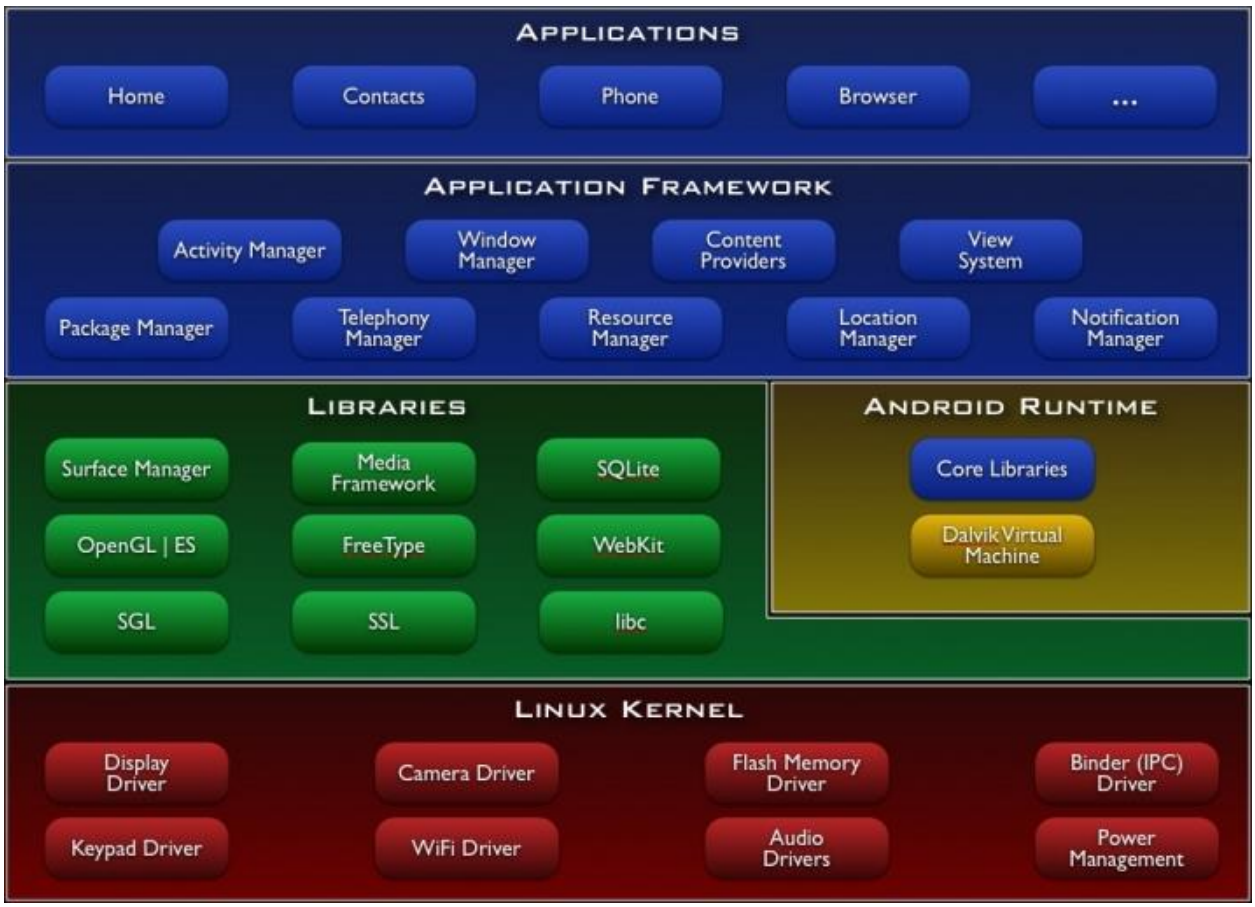

Рисунок 2 - Архитектура платформы android

Так же на рисунке 2 показана диаграмма взаимодействий в операционной системе android.

iOS (до 24 июня 2010 года - iPhone OS) - мобильная операционная система, разрабатываемая и выпускаемая американской компанией Apple. Была выпущена в 2007 году; первоначально - для iPhone и iPod touch, позже - для таких устройств, как iPad и Apple TV. В отличие от Windows Phone и Google Android, выпускается только для устройств, производимых фирмой Apple.

Пользовательский интерфейс iOS основан на концепции прямого манипулирования с использованием жестов мультитач. Элементы управления интерфейсом состоят из ползунков, переключателей и кнопок.

iOS разработана на основе OS X и использует тот же набор основных компонентов Darwin, совместимый со стандартом POSIX.

В iOS есть четыре слоя абстрагирования: слой Core OS, слой Core Services, слой Media Layer, и слой Cocoa Touch.

Для текущей версии операционной системы (iOS 7.1.1) выделяется 1,4-2 Гб флеш-памяти устройства для системного раздела и примерно 800 Мб свободного места (варьируется в зависимости от модели).

По состоянию на 19 мая 2013 года магазин приложений App Store содержит более 900 тыс. приложений для iOS, которые все вместе были загружены более 50 миллиардов раз.

Так же можно с уверенностью сказать, что iOS является самой идеальной системой в мире по причине того, что её энергопотребление просто потрясает. При батарейке 1570 мАч (iphone 5s) её хватает на 2 дня при нормальном использовании в отличии от таких флагманов как Galasys4 или HTC One при батарейке 2600 мАч хватает примерно на такое же время, а то и меньше. Так же потрясает оптимизация игр в системе iOSв то время как android надо ещё поработать и «закрыть лазейки» в своей системе. Но один из самых значительных как минусов, так и плюсов это закрытый исходный iOS, а также что почти каждый шаг является платным в iOS. В то время как система android имеет больше бесплатных приложений, а также имеет открытый исходный код. Вам не нравится как работает ваша система? И вы умеете программировать чудесно можно взять и написать widget ,переписать структуру ядра интерфейс здесь вы бог.

Symbian OS в то время, как Nokia вкладывала огромные деньги в создании новой операционной системы, что привело к низкой продаваемости телефонов nokia за счёт слабого функционала Symbian OS Belle одна из последних операционных систем созданных компанией Nokia стала провальной. В Symbian Os Belle имеется многозадачность, ограниченная системой, в то время как в смартфонах на базе android и Windows phone имеется более продуманная многозадачность, которая позволяет управлять ресурсами не тока системы, но и многих приложений.

Symbian OS является преемником операционной системы EPOC32, разработанной компанией Psion для своих карманных компьютеров. В 1998-2000 гг. значительная часть системы была переписана с целью оптимизации кода для работы на устройствах с ограниченными ресурсами. Разработчикам удалось добиться значительной экономии памяти, улучшения кэширования кода и, как следствие, ускорения работы программ, при пониженных требованиях к энергопотреблению. С точки зрения разработки, отличительной особенностью системы является полностью объектно-ориентированная архитектура (на уровне API). Начиная с версии системы 9.x появился серьёзный механизм защиты - разграничение API в соответствии с правами приложений (capabilities). Основной язык разработки приложений - C++, имеется поддержка Java. Также существуют библиотеки PIPS для портирования приложений с других ОС.

Windows Phone 8 - второе поколение операционной системы Windows Phone от Microsoft. Официальный запуск состоялся 29 октября 2012 года. Система также как и её предшественница, использует интерфейс Modern UI.

В Windows Phone 8 используется новая архитектура Windows NT, которая используется в настольных операционных системах Microsoft. Из-за смены ядра устройства под управлением Windows Phone 7.x, построенной на ядре Windows CE, не могут обновиться до Windows Phone 8. Новые приложения, созданные для Windows Phone 8, не могут запускаться на Windows Phone 7.x, тогда как приложения, написанные для Windows Phone 7.x, работают на Windows Phone 8, будучи автоматически перекомпилированными «в облаке».

Nokia подписала партнерское соглашение с Microsoft 11 февраля 2011, делающее Windows Phone основной операционной системой Nokia. Также телефоны на Windows Phone 8 выпускают Alcatel, HTC, Samsung и Huawei.

Windows Phone 8.1 - обновление мобильных устройств на базе ОС Windows Phone. Презентация нового обновления прошла 2 апреля 2014 года на ежегодном мероприятии компании Microsoft - Build 2014. 14 апреля 2014 года началась рассылка Developer Preview версии, которую можно скачать имея приложение "Preview for Developers".

Также в Windows Phone 8.1 был добавлен голосовой помощник "Cortana". Помощник базируется на конструкции Bing и находится на стадии бета тестирования. Чтобы воспользоваться бета версией Cortana, необходимо сменить язык и регион на США и, соответственно, Английский язык. Cortana имеет "NoteBook" - там вы разрешаете действия, которые она может воспроизводить на вашем телефоне.

1.2 Установка программного обеспечения

Для того чтобы начать программировать в любой среде программирования, необходимо установить нужное программное обеспечение. Начнем установку Java Eclipse. После загрузки системы необходимо скачать с официального сайта Eclipse for android. В разделе Java Platform, Standard Edition жмете JDK Download, ставите галку, что принимаете лицензионное соглашение и скачиваете файл соответственно Вашей операционной системе. С установкой проблем возникнуть не должно. После этого желательно перезагрузиться. Пристарте системы будет такой логотип как показано на рисунке 3.

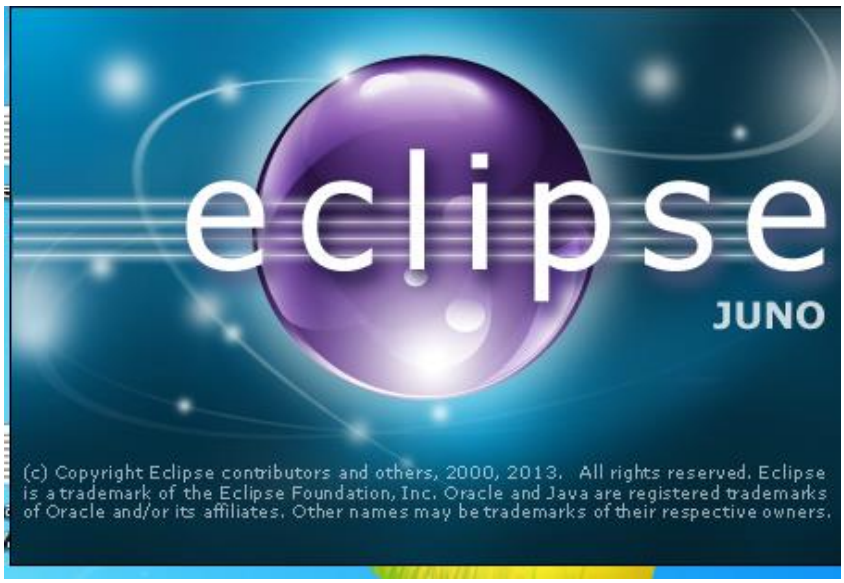

Рисунок 3 - Логотип при запуске приложения

Скачать готовую сборку среды разработки, включающую в себя Eclipse + Android SDK + ADT. Чтобы ее скачать проходите по ссылке

developer.android.com\sdk (Download for other platforms -> ADT Bundle) и выбирайте ссылку для вашей версии системы. Скачивайте, распаковывайте, запускайте ..\eclipse\eclipse.exe. Запускайте SDK Manager (меню Window> Android SDK Manager) и переходимк следующему пункту.

Android SDK включает в себя инструменты, необходимые для разработки Android-приложений.

Чтобы его скачать проходите developer.android.com\sdk (Download for other platforms -> SDK Tools Only) ивыбирайтессылкудлявашей версиисистемы. Рекомендуется скачивать EXE-шник, но я предлагаю скачать ZIP-версию и самим распаковать в какой-нибудь удобный для вас каталог.

Учтите, что это должен быть каталог "на века". И лучше его не перемещать никуда, иначе придется перенастраивать среду разработки. Предлагаю где-нибудь создать каталог Android. Крайне желательно, чтобы путь к нему был коротким. Идеально - <имя диска>:\android (у меня это будет f:\android). Для себя запомним этот каталог под псевдонимом <Android>**.** И в него распакуем наш архив SDK, получим <Android>\android-sdk-windows**.**

Либо вы можете скачать уже собранную среду разработки, выбрав на той же странице не SDK Tools Only, а ADT Bundle.

По умолчанию Eclipse не особо годится для разработки Android-приложений. ADT - плагин, который настраивает среду разработки для использования Android SDK и добавляет возможность удобной разработки.

Запускаем Eclipse (<Android>\eclipse\eclipse.exe). При первом запуске он попросит указать ему рабочий каталог, где он будет хранить информацию о проектах. Предлагаю опять же не ходить далеко и создать каталог <Android>\workspace , и указать этот каталог, как показано на рисунке 4.

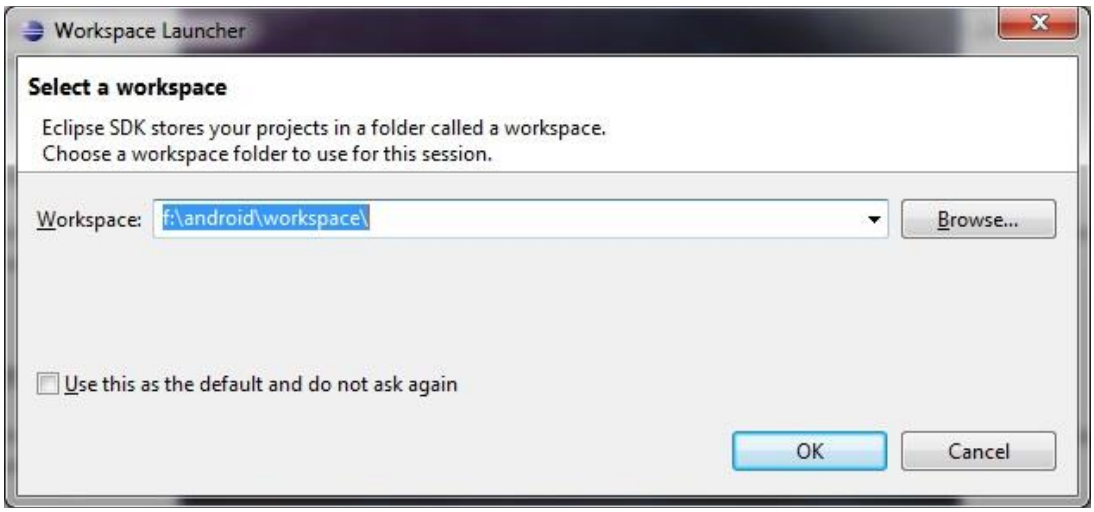

Рисунок 4 - Создание рабочего пространства

Далее требуется обновить версию Eclipse, а так же скачать дополнение для корректной работы android всё это находится на сайте разработчика. Заходим Help->Install new software, как показано на рисунке 5.

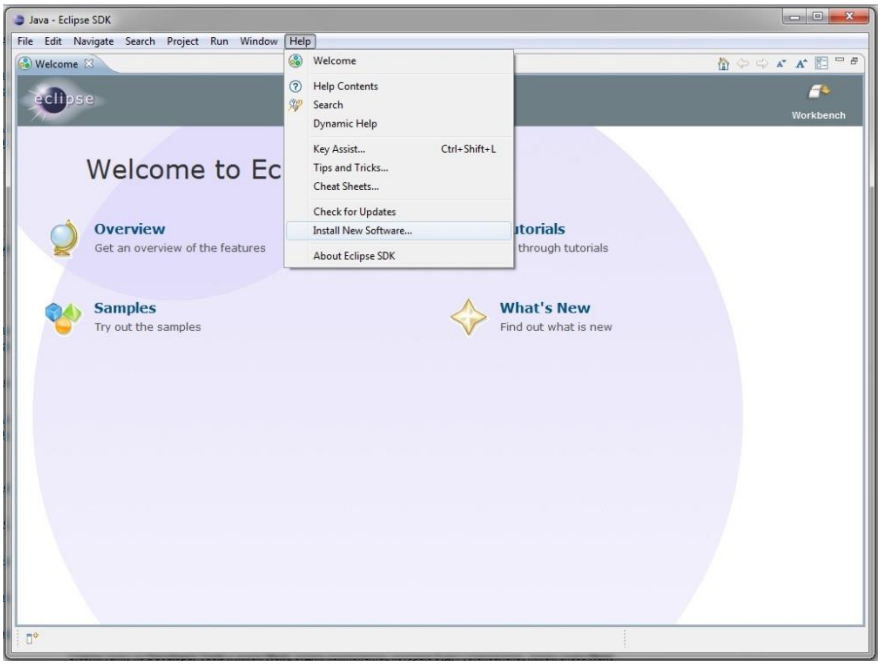

Рисунок 5 - Меню help

Жмем кнопку Add в правом верхнем углу как показано на рисунке 6.

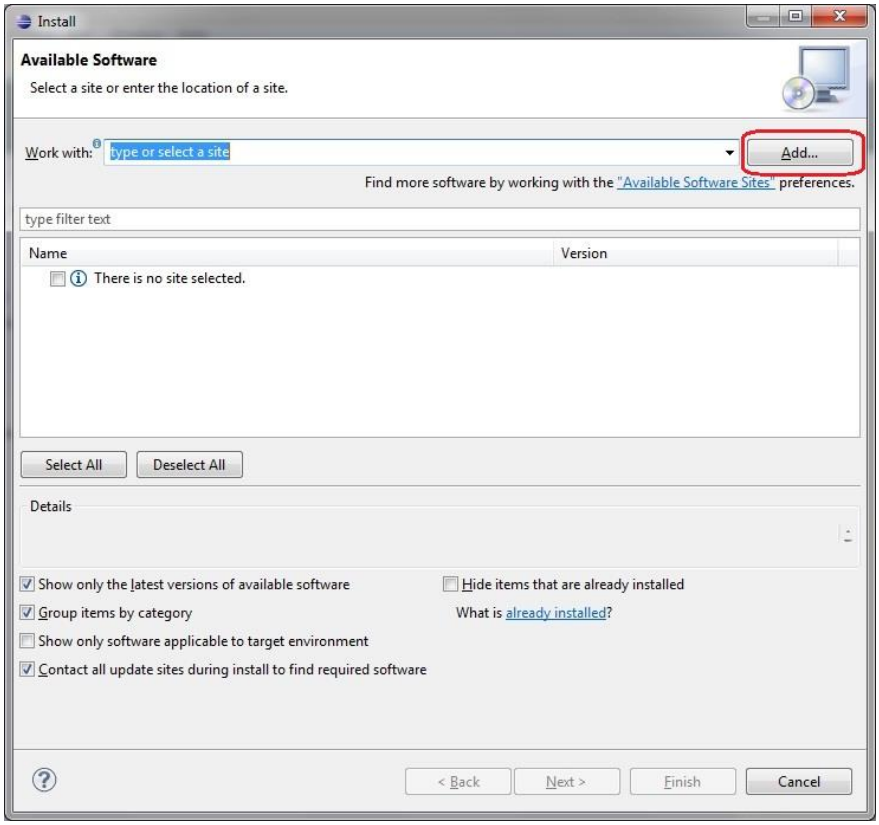

Рисунок 6 - Установка плагинов

Вводим "ADT Plugin" в поле Name URL адрес: https://dlssl.google.com/android/eclipse/ - в поле Location как показано на рисунке 7.

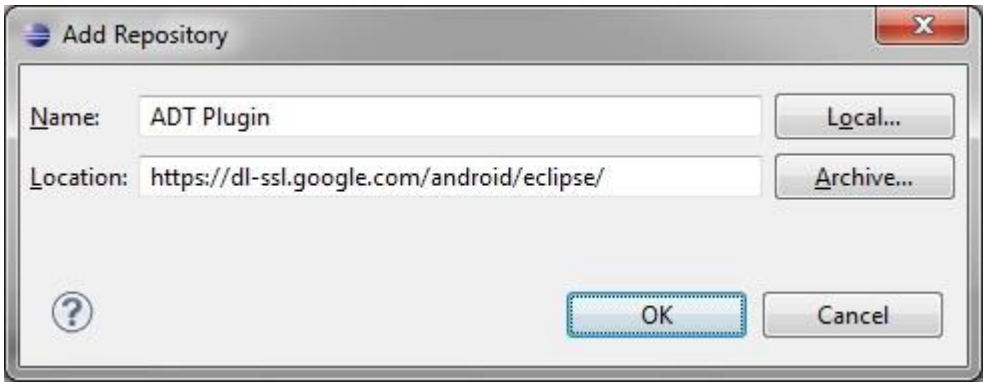

Рисунок 7 - Переход поссылке для скачивания дополнений

1. Элементы для ввода данных пользователем:
   - TextView- используется для ввода как короткого, так и многострочного текста;
   - CheckBox - представляет собой элемент управления «флажок». Может быть установлен или снят, опционально можно включить поддержку «неопределённого» состояния;
   - - RadioButton - позволяет пользователю выбрать только один вариант из нескольких. Сгруппированные элементы управления (один GroupName) позволяют выбрать только один вариант из группы;
   - -Slider - позволяет пользователю выбрать из нескольких последовательных вариантов. Позиция соотносится со значением свойства Value.
2. Элементы для указания выполнения действий:
   - Button - кнопка, при нажатии пользователем генерирует событие Click.
   - Элементы для вывода данных на страницу приложения:
   - Small text, Mediumtext,Large- позволяет отобразить простой текст, без возможности редактирования;
   - -Imageview- позволяет вывести на экран изображение;
   - ProgressBar - отображает текущий прогресс пользователю.
3. Элементы для разметки страницы:
   - ScrollViewer - добавляет возможность прокручивания дочерним элементам;
   - Gridlayout- является контейнером для других элементов управления. Позволяет располагать и упорядочивать дочерние элементы управления с привязкой к сетке;

- Canvas - определяет область, в рамках которой можно явно расположить дочерние элементы с помощью координат относительно области.

5. Специализированные элементы:
   - Map - отображает карту Google;
   - MediaController - позволяет воспроизводить звуки;
   - VideoView - позволяет воспроизводить видео;
   - WebView - отображает веб-страницу на языке HTML.

Ввод текста - важная часть взаимодействия пользователя с телефоном. Чтобы несколько упростить этот процесс, а также уменьшить количество ошибок ввода уже давно предлагается множество разнообразных технологий.

За то, какая именно клавиатура будет отображаться, отвечает контекст ввода (inputScope). Каждое текстовое поле (TextView). Такая настройка не обязательна, но пользователю будет приятнее работать с подходящей в каждом конкретном случае клавиатурой. Рассмотрим работу с текстовыми полями и контекстом ввода на практическом примере. Запустим Java for android и создадим новое приложение с помощью шаблона Приложение android application с названием inputtxt. После ввода имени проекта, нажимаем на кнопку ОК.

1.3 Концепции разработки приложений

Целевую платформу для приложения следует выбрать android 2.3.3(froyo) Будет создан новый проект, и откроется редактор пользовательского интерфейса главной страницы приложения. Пользовательский интерфейс приложений для android описывается декларативным образом на языке XML. Eclipse позволяет как редактировать XML-код напрямую, так и использовать графический Graphicallayout. В eclipse окно редактора пользовательского интерфейса приложений для android разделено вертикально на две части. Слева находится графический редактор, выполненный в виде телефона, а справа - редактор XML-кода. Eclipse предоставляет пустой шаблон страницы, хотя можно легко удалить почти любой элемент и сделать дизайн абсолютно не похожим на шаблон по умолчанию. Изучить структуру проекта прилоожения для android можно с помощью окна Обозреватель решений, который представлен на рисунке 8.

Для лучшего понимания следует описать все файлы. Main.xml - разметка главной страницы приложения на языке XML. Название страницы выбрано произвольно. Нет никаких требований к тому, как будет называться главная страница. Важно только указать в настройках приложения, какую страницу первой увидит пользователь. По умолчанию в качестве такой страницы указана именно эта страица. Main.xml - файл кода на языке java страницы Main.xml.

Если требуется добавить на какую-либо страницу код, то можно использовать для этого файл с расширением xml. В данном случае этим файлом является Main.xml. main.java-приложение кода, которая тесно работает вместе с

файлом main.xml. Данная разметка не имеет визуального представления и не является страницей приложения. Файл main.java является тем местом, где можно хранить данные и настройки для всего приложения. Кроме того, в этом файле удобно определять стили и подключать ресурсы, используемые на нескольких страницах приложения, хотя это и не обязательно. Main.java - файл кода на языке java для Main.java. Здесь можно обрабатывать события уровня приложения, такие как запуск, активация, деактивация и закрытие приложения.

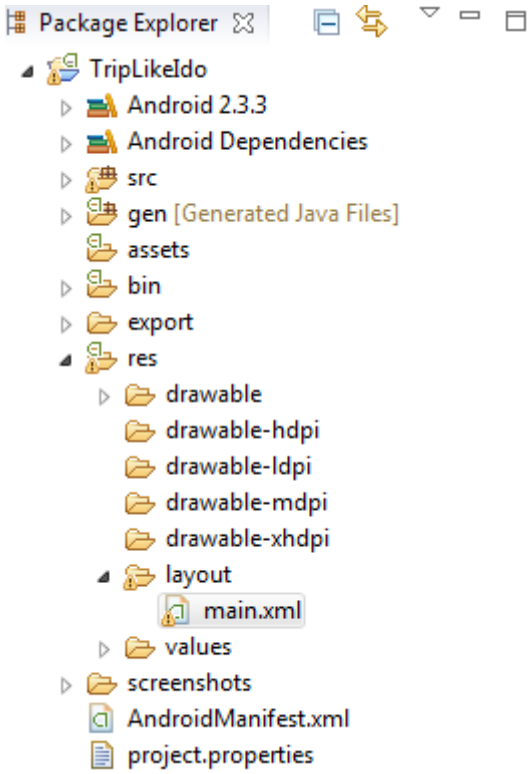

Рисунок 8 - Обозреватель решений проекта

Кроме того, в Main.java вы можете перехватывать необработанные исключения и отлавливать ошибки навигации между страницами. AndroidManifest.xml - файл метаданных, являющийся манифестом приложения. Он содержит множество настроек приложения: заголовок, имя первой страницы, пути к значкам, определение необходимых приложению системных возможностей и т. д. Во многих случаях данный файл редактируется не напрямую, а с помощью окна свойств приложения. AndroidManifcst.xml - простой манифест, необходимый для генерации xml-файла (пакета) приложения. Редактировать вручную данный файл не требуется. AssemblyInfo.es - еще один конфигурационный файл, в котором определяются некоторые метаданные главной сборки(Assembly) приложения.

После описания всех файлов проекта, вернемся к разработке нашего приложения. В данном примере мы разработаем самое очевидное приложение, которое можно создать для телефона, а именно приложение для звонков по заданному номеру. Приложение будут состоять из текстового поля для ввода

телефонного номера и кнопки Позвонить. Элементы управления будут находиться на странице MainPage.xml:

```
      <TextViewandroid:layout_width="wrap_content"android:id="@+id/
gpsstate"
      android:layout_height="wrap_content"android:typeface="sans"an
droid:textSize="25sp"android:lines="1"
      android:text="Detail">
      </TextView>
```

Добавим на страницу MainPage.xml соответствующую разметку и запустим приложение. Для того чтобы запустить приложение на отладку, нажмите клавишу <F5> или кнопку Start с зеленой стрелкой на панели инструментов либо выберите Debug - StartDebugging в меню. В выпадающем списке на панели инструментов вы можете указать, где будет запущено приложение: на эмуляторе или на реальном устройстве. В данном случае мы будем запускать приложение на реальном устройстве. Эмулятор работает независимо от Eclipse, поэтому его не требуется закрывать после каждого сеанса отладки. Вы можете просто остановить отладку в Eclipse, нажав комбинацию клавиш <Shift>+<F5> или кнопку Stop на панели инструментов либо выбрав в меню Debug-Stop Debugging, оставив при этом эмулятор работающим. В следующий раз приложение будет запущено уже в работающем эмуляторе, что займет немного меньше времени, чем "холодный" запуск. Кроме того, при закрытии и новом запуске эмулятора его состояние сбрасывается.

Внешний вид запущенного приложения очень похож на то, что мы видели в графическом редакторе в Eclipse. Большим отличием является наличие группы цифр, показываемых в правом верхнем углу приложения (красные и белые цифры). Это информация о производительности отрисовки приложения. По умолчанию в шаблон проекта добавлен код, включающий показ данной информации в режиме отладки.

После того, как мы запустили приложение, попробуем ввести телефонный номер. Появится стандартная QWERTY- клавиатура как на рисунке 8 для ввода текста, что нас не устраивает:

```
      Intent intent = new Intent(Intent.ACTION_DIAL,
Uri.parse("tel:1234567"));
      startActivity(intent);
```

Замена стандартного приложения для дозвона проходит в два этапа:

1) перехват Намерений, которые в настоящее время обслуживаются стандартным приложением;

2) осуществление исходящих звонков и при необходимости управление ими:

```
      <activity
      android:name=".MyDialerActivity"
```

```
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.CALL_BUTTON" />
<category android:name="android.intent.category.DEFAULT" />
</intent-filter>
<intent-filter>
<action android:name="android.intent.action.VIEW" />
<action android:name="android.intent.action.DIAL" />
<category android:name="android.intent.category.DEFAULT" />
<category android:name="android.intent.category.BROWSABLE" />
<data android:scheme="tel" />
</intent-filter>
</activity>
```

Стандартная клавиатура android как показано на рисунке 9.

Рисунок 9 - Клавиатура qwerty

Измените свойство inputscope элемента управления TextBox на TelephoneNumber:

```
    <TextBox x:Name="txtPhoneNumber"
InputScope="TelephoneNumber"></TextBox>
```

Запустите приложение. Теперь при попытке ввести телефонный номер появится подходящая для этих целей клавиатура, как показано на рисунке 10.

Рисунок 10 - Клавиатура для ввода телефонного номера

Для того чтобы позвонить по введенному номеру телефона, воспользуемся задачей запуска PhoneCallTask. Создайте обработчик события нажатия кнопки btnDial и подключите в файле манифесте read_phone_state, а также надо подключить простарнство имён telephonymanager:

```
Importandroid.telephonymanager;
```

В обработчике события нажатия кнопки позвоним по номеру телефона, введенному в текстовое поле txtPhoneNumber:

```
private void btnDial_Click(object sender, RoutedEventArgs e)
{
var phoneCallTask = new PhoneCallTask();
phoneCallTask.PhoneNumber = txtPhoneNumber.Text;
phoneCallTask.DisplayName = "Дед Мороз"; phoneCallTask.Show();
)
```

Так как был введен только номер абонента, в коде, осуществляющем звонок, мы указали, что имя абонента- Дед Мороз. Запустим приложение и попробуйте позвонить кому-либо. Рассмотрим другие контексты ввода, кроме Defaultи TelephoneNumber.

Text - данный контекст ввода задает раскладку клавиатуры, на которой присутствует кнопка в виде смайла. Нажатие кнопки переключает клавиатуру в режим ввода смайлов. Кроме того, при использовании данного контекста ввода предлагаются подсказки по вводу слов, а также производится автоматическое исправление ошибок.

URL - этот контекст ввода задает раскладку клавиатуры, на которой присутствует кнопка .com, предназначенная для быстрого ввода домена. Кроме того, если зажать данную кнопку, появятся другие возможные домены: org, .edu, .net. В локализованных версиях операционной системы поддерживаются кнопки для локальных доменов. Также в раскладке присутствует клавиша <Enter> для перехода по ссылке.

Number - задает раскладку клавиатуры для ввода чисел.

EmailNameOrAddress - раскладка клавиатуры, на которой кроме кнопки для ввода домена, как в режиме URL, присутствует кнопка @ для быстрого ввода соответствующего символа.

Каждый телефон под управлением Windows Phone имеет аппаратную кнопку «назад», которая позволяет перемещаться между страницами независимо от того, какому приложению принадлежит конкретная страница. В реальных приложениях таких страниц может быть множество. Это напоминает работу Web-браузера. С точки зрения пользователя различия между работой с Web-страницами и приложениями для android минимальны. Для разработчика

различия более существенны, в первую очередь потому, что все страницы конкретного приложения разделяют состояние этого приложения (статические объекты и т. д.), но базовые концепции аналогичны. Представьте, что ваше приложение для android является Web - сайтом, но вместо HTML - страниц и кода на JavaScript у вас есть XML - страницы и код на языке Java. Аналогом CSS в нашем примере будут стили, задаваемые также в XML - разметке. Важно учесть только, что XML - стили в отличие от CSS не являются каскадными. Кроме того, на телефоне доступны дополнительные возможности, такие как создание анимации перехода между страницами, не доступные для HTML - страниц, работающих в браузере.

Центральным элементом каждого приложения для android является фрейм, представляющий собой объект класса frame. В данный фрейм загружаются страницы приложения, унаследованные от класса frame. В приложении может существовать только один фрейм. Страниц же может быть произвольное количество. Программная навигация между страницами осуществляется с помощью класса NavigationService. Кроме явного вызова методов класса NavigationService, навигация может осуществляться с помощью гиперссылок на страницах приложения. Гиперссылки представляют собой объекты класса small text в object inspector есть функция hyperlink проставляем, что нужно и получаем ссылку. Рассмотрим создание страниц и навигацию между ними на практическом примере. Для этого создаем новое приложение с помощью шаблона androidApplication под названием NavigationApp.

Допустим, мы создаем приложение - каталог телефонов на платформе android, и нам требуется выводить информацию о фирмах-производителях, например о Nokia, HTC и Samsung. Создадим для каждого из производителей новую страницу. Для этого в окне Обозреватель решений щелкним правой кнопкой мыши на имени проекта и в контекстном меню выберите добавить - новый элемент. Откроется окно, предлагающее выбрать тип элемента, который мы хотим добавить в проект. Можно добавить страницы двух основных типов: Страница android в книжной или в альбомной ориентации. Данные типы страниц отличаются только ориентацией экрана по умолчанию, в первом случае она будет портретной (высота больше ширины), а во втором ландшафтной (ширина больше высоты). В процессе редактирования страницы всегда можно поменять ее ориентацию, кроме того, можно организовать работу страницы, как в портретном, так и в ландшафтном режиме, меняя ориентацию при повороте телефона. В нашем случае выберим страницу с книжной ориентацией и в поле Имя введем значение Lenovo.xml, как показано на рисунке 11.

В итоге создана новая страница. Поменяйте заголовок только что созданной страницы на lenovo, чтобы при работе приложения можно было понять, на какой странице мы находимся. Для этого в XML-коде страницы lenovo.xml измените значение свойства Text текстового блока с именем PageTitle.

```
<TextBlock x:Name=" PageTitle" Text="lenovo" Margin="9″
```

```
Style="{StaticResourcePhoneTextTitle1Style}"/>
```

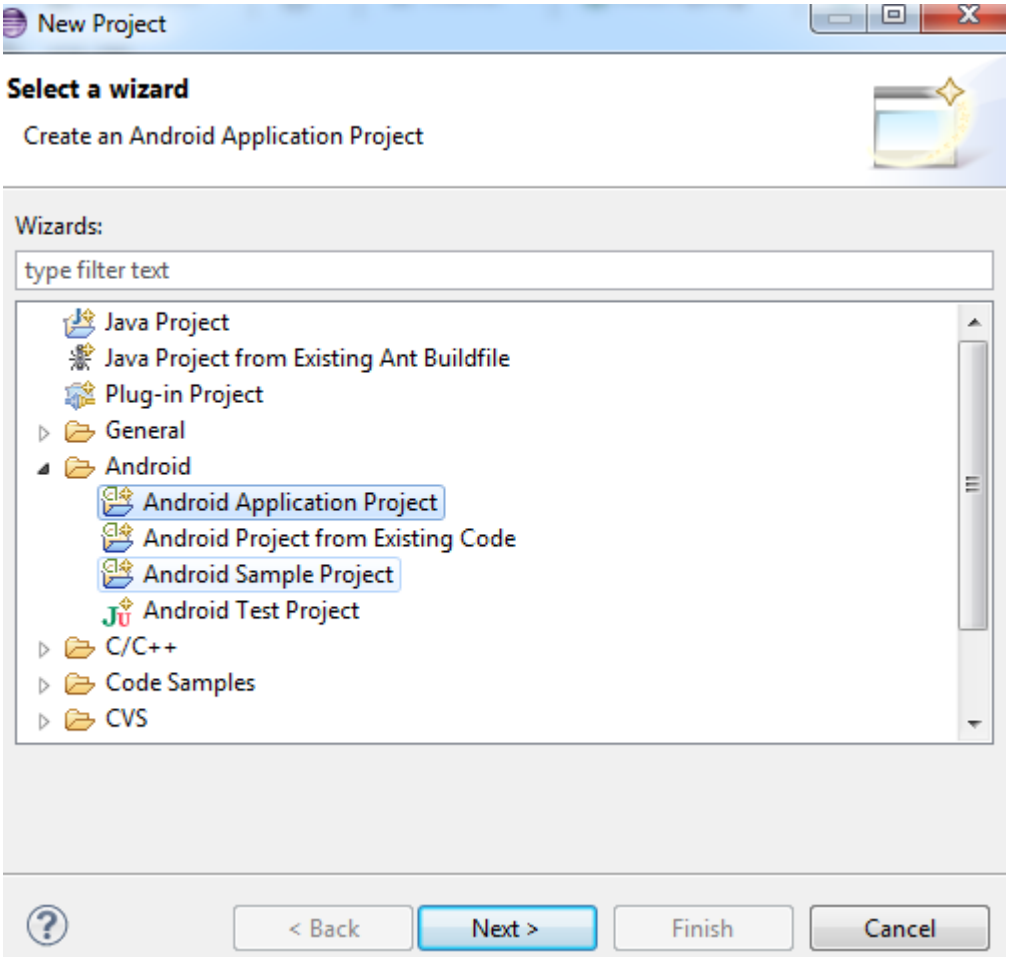

Рисунок 11 - Добавление нового элемента в проект

Создадим аналогичным образом еще две страницы с именами HTC.xml и Samsung.xml, а также изменим заголовки этих страниц на соответствующие названия компаний. Теперь поместим в главной странице MainPage.xml ссылки на каждую из страниц Nokia.xml, HTC.xml и Samsung.xml соответственно. Для этого в менеджер размещения Grid с именем ContentPanel добавьте несколько элементов управления HyperlinkButton, располагающихся один за другим по вертикали в менеджере размещения StackPanel:

```
        Text1.setText(
        Html.fromHtml(
        "<a href=\"http://www.google.com\">google</a> "));
        Text1.setMovementMethod(LinkMovementMethod.getInstance());
        Text2.setText(
        Html.fromHtml(
        "<a href=\"http://www.stackoverflow.com\">stackoverflow</a>
"));
        Text2.setMovementMethod(LinkMovementMethod.getInstance());
```

Свойство NavigateUri элемента управления HyperlinkButton содержит адрес страницы, на которую будет осуществлен переход при щелчке пользователем по гиперссылке. При запуске приложения убедимся в правильности работы и отображения контента, как показано на рисунке 12.

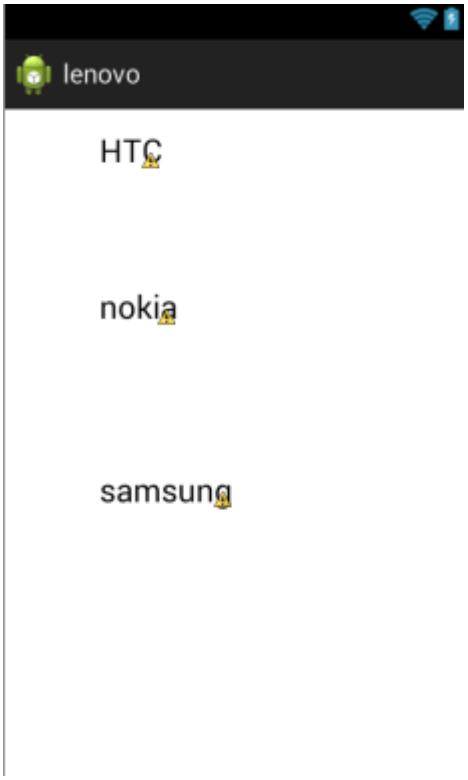

Рисунок 12 - Интерфейс приложения с элементом HyperlinkButton

Для возврата на главную страницу используйте аппаратную кнопку Назад. Если же нажать кнопку «Назад» на главной странице, то вы покинете приложение и вернетесь к тому интерфейсу телефона, который видели до запуска приложения.

Кроме использования элемента управления HyperlinkButtonмы можем перейти на другую страницу с помощью кода на языке C#. Добавим на страницу MainPage.xml в StackPanel после ссылок кнопку с текстом "Информация о компании Nokia":

```
    <Button x:Name="btnNokia" Content ="Информация компании Nokia"
Click="btnNokia_Click" />
```

Данная кнопка будет иметь имя btnNokia. В обработчике события нажатия кнопки добавим код перехода на страницу Nokia.xml:

```
    <Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
     android:layout_alignLeft="@+id/textView3"
     android:layout_below="@+id/textView2"
     android:layout_marginTop="37dp"
     android:text="test" />

     private void btnNokia_Click(object sender)
     {
     NavigationService.Navigate (new Uri ("/Nokia.xml",
UriKind.Relative));
     }
```

Приложение не всегда состоит из одного экрана. Например, мы создали очень полезную программу и пользователю хочется узнать, кто же её автор. Он нажимает на кнопку «О программе» и попадает на новый экран, где находится полезная информация о версии программы, авторе, веб-адресе сайта. Воспринимайте Activity как веб-страницу с ссылкой на другую страницу. Если вы посмотрите на код в файле HelloWorld.java из прошлых примперов, то увидите, что наш класс HelloWorld тоже относится к Activity или, если говорить точнее, наследуется от него.

1.4 Геолокация в android

Android устройства могут предоставить нам данные по нашему текущему местоположению. Это, конечно, очень удобно и вовсю используется для, например, пользования картой, получения актуальной для вашей местности информации (прогноз погоды), всевозможных чекинов и пр.

Реализация этого всего вполне проста. Мы вешаем слушателя на провайдера и получаем данные. На данный момент есть два провайдера: GPS и Network.

GPS - тут все понятно, это данные с GPS - спутников.

Network - это координаты, которые можно получить через сотовую связь или WiFi. Для этого провайдера нужен инет.

В onCreate определяем TextView - компоненты и получаем LocationManager, через который и будем работать.

В onResume вешаем слушателя с помощью метода requestLocationUpdates. На вход передаем:

- тип провайдера: GPS_PROVIDER или NETWORK_PROVIDER;
- минимальное время (в миллисекундах) между получением данных. Я укажу здесь 10 секунд, мне этого вполне хватит. Если хотите получать координаты без задержек - передавайте 0. Но учитывайте, что это только минимальное время. Реальное ожидание может быть дольше;

– минимальное расстояние (в метрах). Т.е. если ваше местоположение изменилось на указанное кол-во метров, то вам придут новые координаты;

– слушатель, объект locationListener, который рассмотрим ниже;

– Также здесь обновляем на экране инфу о включенности провайдеров;

– В onPause отключаем слушателя методом removeUpdates;

locationListener - слушатель, реализует интерфейс LocationListener с методами;

onLocationChanged - новые данные о местоположении, объект Location. Здесь мы вызываем свой метод showLocation, который на экране отобразит данные о местоположении;

onProviderDisabled - указанный провайдер был отключен юзером. В этом методе вызываем свой метод checkEnabled, который на экране обновит текущие статусы провайдеров;

onProviderEnabled - указанный провайдер был включен юзером. Тут также вызываем checkEnabled. Далее методом getLastKnownLocation (он может вернуть null) запрашиваем последнее доступное местоположение от включенного провайдера и отображаем его. Оно может быть вполне актуальным, если вы до этого использовали какое-либо приложение с определением местоположения;

onStatusChanged - изменился статус указанного провайдера. В поле status могут быть значения OUT_OF_SERVICE (данные будут недоступны долгое время),TEMPORARILY_UNAVAILABLE (данные временно недоступны), AVAILABLE (все ок, данные доступны). В этом методе мы просто выводим новый статус на экран.

Провайдеры включаются и отключаются в настройках системы. Тем самым, просто определяется доступен ли провайдер для получения от него координат. Чуть позже увидим, как можно отправить юзера в эти настройки. Программное включение/выключение провайдеров через стандартные методы недоступно.

Далее идут свои методы:

showLocation на вход берет Location, определяет его провайдера методом getProvider и отображает координаты в соответствующем текстовом поле.

formatLocation на вход берет Location, читает из него данные и форматирует из них строку. Какие данные он берет:

getLatitude - широта,

getLongitude - долгота,

getTime - время определения.

CheckEnabled определяет включены или выключены провайдеры методом isProviderEnabled и отображает эту инфу на экране.

Метод onClickLocationSettings срабатывает по нажатию кнопки Location settings и открывает настройки, чтобы пользователь мог включить или выключить провайдер. Для этого используется Intent с action = ACTION_LOCATION_SOURCE_SETTINGS.

Осталось в манифесте прописать разрешение на определение координат - ACCESS_FINE_LOCATION, которое позволит нам использовать и Network, и GPS. Также существует разрешение ACCESS_COARSE_LOCATION, но оно дает доступ только к Network-провайдеру.

С кодом все, давайте смотреть, что получилось. Все сохраняем и запускаем приложение.

У меня на планшете сейчас выключен GPS, выключен WiFi, вставлена симка и выключен мобильный интернет.

Запускаю приложение и вижу такую картину на рисунке 13:

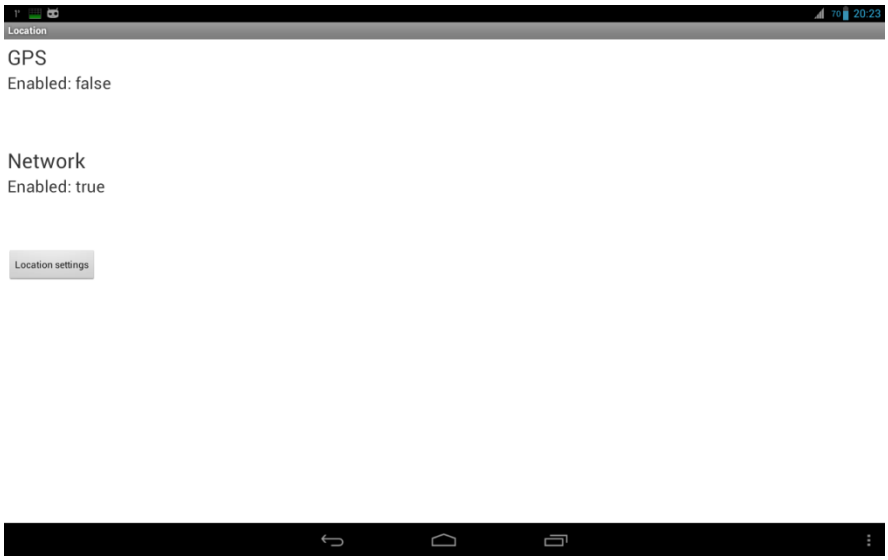

Рисунок 13 - Приложение показывает широту и долготу

1.5 Установка VS 2010 for windows phone

В отличии от java в данном случае мы видим установку «в 1 клик». То есть единственное, что надо сделать это скачать с официального сайта пакет для установки VS 2010 Express for Windows Phone, и он автоматически запускается, а в java приходилось немного попотеть над тем чтоб установить все плагины для работы в android системе. Так же в C# большая совместимость продуктов, то есть если вы пишите приложения на Windows 8, то его перенести на платформу windows phone 7, так же «в один клик».

Начальное окно при запуске vs 2010 рисунок 14.

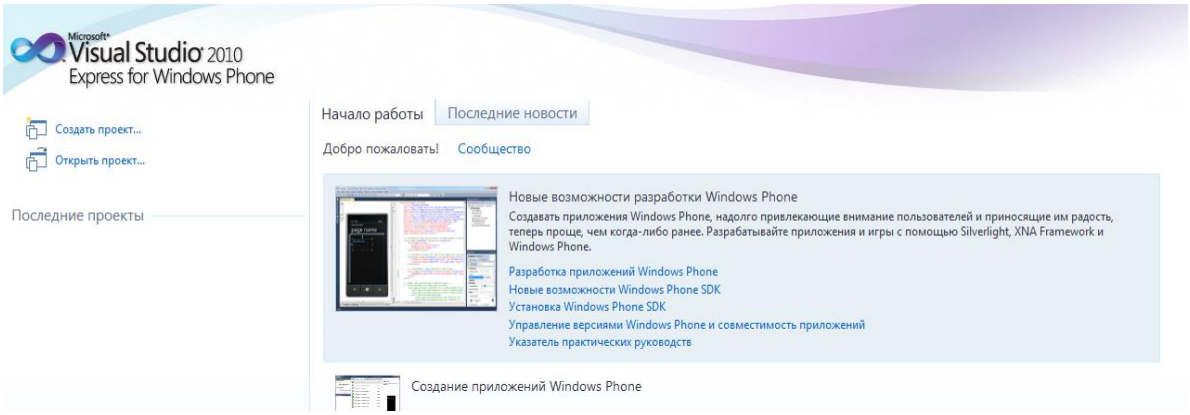

Рис 14 Начальная страница Windows phone.

1.6 Графический редактор

Так же можно сказать что в VS 2010 существуют очень хороший редактор WYSIWYG-Whatyouseeitswhatyouget очень интересное название но в данном случаем можно сказать что то что мы делаем отображается сразу же на графическом редакторе в принципе на данный момент существует AndroidStudio в нём реализована такая же функция но не столь хорошо как в VisualStudio, так же нужно отдать должное Microsoft которые постарались над управлением и интуитивным программированием в среде C# for windows phone то есть Intelsense (интеллектуальная система подсказок).

В графическом дизайнере в Visual Studio, а также при отладке приложений в верхней части экрана можно увидеть системную информацию телефона, такую как уровень сигнала, заряд батареи и т. д. Эта информация важна для пользователя. Область, где она отображается, называется *системным* трем (System Tray) рисунок 15 свидетельствует о месторасположении системного трея.

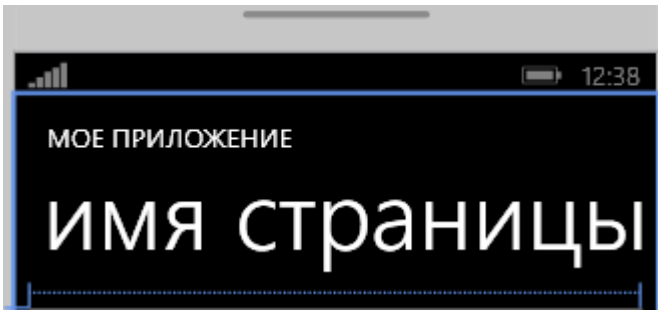

Рисунок 15 - Системный трей

Системный трей занимает 32 пиксела по высоте. Его можно скрыть. В этом случае приложению будет доступна на 32 пикселя большая область экрана. Но пользователю это вряд ли понравится. Часто при работе приложения информация в системном трее скрывается, и для того чтобы снова ее показать,

пользователю необходимо нажать на область системного трея. Однако даже если информация скрыта, системный трей все равно занимает пространство. По умолчанию системный трей показывается на всех страницах. За это отвечает свойство:

```
ShellSystemTray. IsVisible="True"
```

Чтобы скрыть системный трей, присвойте данному свойству значение False. Как уже было сказано ранее, скрывать системный трей - не лучшая идея. Если приложению необходимо дополнительное место на экране, системный трей можно сделать полупрозрачным. В этом случае он не будет забирать пространство экрана у приложения. Установим полу прозрачность системного трея в значение 0.5:

```
Shell: SystemTray.0pacity="0.5"
```

В этом случае над контентом приложения будет нависать полупрозрачный фон трея. Если установить полупрозрачность трея в значение 0, то фон показываться не будет. Кроме того, можно установить цвет фона и информации, отображаемой в системном трее. Для установки цвета фона воспользуйтесь свойством BackgroundColor, а для установки цвета содержимого - свойством Foregroundcolor. Зададим желтый и красный цвета соответственно:

```
shell:SystemTray.BackgroundColor="Yellow"
shell:SystemTray.ForegroundColor="Red"
```

Изменение цвета фона и содержимого трея полезно, когда создается приложение, не использующее системные цвета.

Системный трей предоставляет еще одну полезную возможность, а именно отображение индикатора процесса (Progressindicator). Элемент управления Progress indicator доступен и независимо от системного трея, однако в этом случае для него надо искать специальное место.

Индикатор процесса может работать в двух основных режимах - показ конкретного значения прогресса в виде полоски, а также в режиме информирования пользователя о том, что идет какой-либо процесс без конкретного значения прогресса. Во втором случае показываются четыре бегущие точки. Индикатор процесса позволяет задать текстовую надпись с дополнительной информацией для пользователя. Добавим в приложение индикатор процесса, значение которого не определено и будут показываться пять бегущих точек по экрану. Для этого в Mainpage.xaml после отображения системного трея зададим свойство Isindeterminate в значение True. Результат работы приложения представлен на рисунке 16.

```
shellSystemTray. IsVisible="True"
<shell:SystemTray.ProgressIndicator>
```

```
    <shell:ProgressIndicator IsVisible="True"
IsIndeterminate="True"/>
    </shell:SystemTray.ProgressIndicator>
```

Рисунок 16 - Индикатор процесса без конкретного значения

1.7 Варианты разметки для телефона

Для того чтобы удобно разместить элементы управления на странице приложения, нам требуются менеджеры размещения, основная задача которых- предложить нам разнообразные схемы разметки или компоновки других элементов управления.

Наиболее часто в приложениях для Windows Phone используются следующие менеджеры размещения:

- Canvas;
- StackPanel;
- Grid.

Рассмотрим, как каждый из менеджеров работает, поместив внутри них другие элементы управления. Для этого создадим новый проект, который будет называться ExploreLayoutControls.

Весь код, для простоты, будем вставлять на страницу MainPage.xaml внутрь элемента управления Grid с именем ContentPanel:

```
<!--ContentPanel- поместите здесь дополнительное содержимое>
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
</Grid>
```

Менеджер размещения Canvas предоставляет наиболее простой вариант разметки. Он применяется для абсолютного позиционирования элементов управления с использованием координат. Для позиционирования элементов управления на Canvas используются прикрепленные свойства (attached

properties). Такие свойства позволяют расширять свойства элементов управления. В нашем примере прикрепленное свойство Canvas.Top расширяет свойства элемента управления Button, позволяя задать для кнопки ее положение относительно верхнего края менеджера размещения Canvas. Разместим несколько кнопок (Button) на Canvas:

```
    <Canvas>
    <ButtonCanvas.Top="75"  Canvas.Left="75"  Content  ="Кнопка
T75.L75.Z-1" Canvas.ZIndex="-1" FontSize="18" Width="200"
    Height="75"   Background="Orange" />
    <Button  Canvas.Top="175"  Canvas.Left="5"  Content="Кнопка
T175.L45" FontSize="18" Width="200" Height="75" />
    <Button Canvas.Top=" 110" Canvas.Left="90" Content=" Кнопка
T95.L90" FontSize="18" Width="220" Height="75" />
    </Canvas>
```

При запуске приложения в эмуляторе отобразится страница, как показано на рисунке 17.

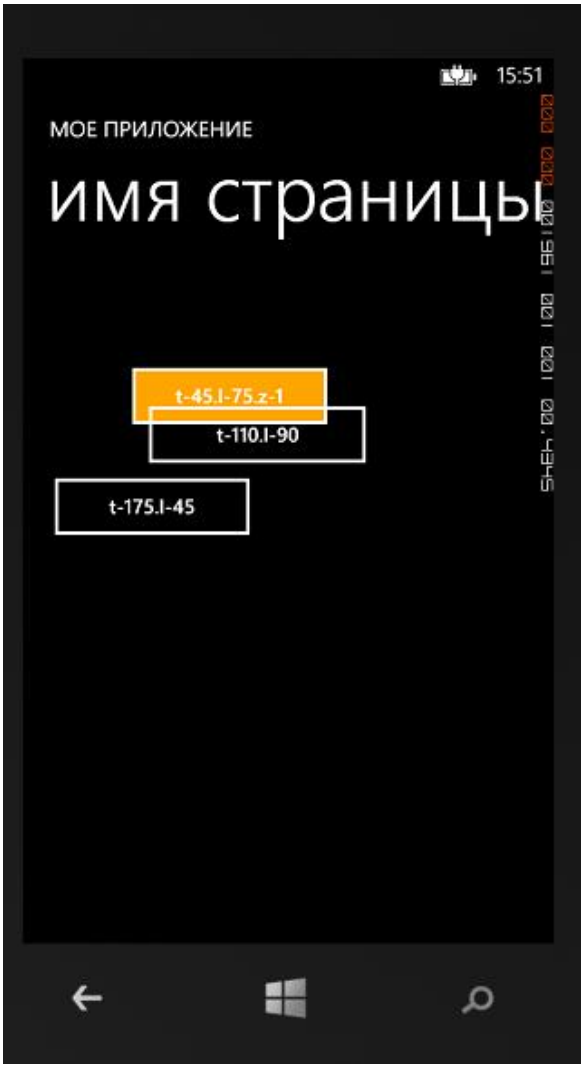

Рисунок 17 - Использование менеджера размещения Canvas

Как видно, элементы позиционированы по абсолютным координатам внутри Canvas. Обратите внимание, как свойство ZIndex, указанное для одной из кнопок, влияет на то, как один элемент перекрывает другой элемент.

Менеджер размещения Canvas удобно использовать, когда элементы управления внутри него не будут перемещаться, а окно приложения - менять ориентацию, или когда необходимо по тем или иным причинам позиционировать элементы интерфейса приложения в точности по заданным координатам. В противном случае использование Canvas может быть сложнее, чем использование таких менеджеров размещения, как Grid или StackPanel.

Менеджер размещения StackPanel предоставляет вариант разметки, который располагает помещенные в него элементы один за другим горизонтально или вертикально (по умолчанию - вертикально). Расположим несколько кнопок в StackPanel:

```
<StackPanel>
<ButtonMargin="0,0,0,10" Content="Кнопка 0.0.0.10"
FontSize="18" Width="200" Height="75" />
<Button Content="Кнопка 0.0.0.0" FontSize="18"
Width="200” Height="75" />
<Button Margin="0,50,0,0" Content=" Кнопка 0.50.0.0"
FontSize=”18" Width="200" Height="75" />
<Button Margin="-150,170,0,0" Content=" Кнопка -150.170.0.0"
FontSize="18" Width="200" Height="75" />
</StackPanel>
```

Свойство Margin позволяет выполнить относительное позиционирование элементов управления. Приложение, запущенное в эмуляторе, будет выглядеть так, как представлено на рисунке 18:

```
SupportedOrientations="Landscape" Orientation="Landscape"
...
<StackPanel Orientation="Horizontal">
<Button Margin="0,0,0,10" Content="Кнопка 0.0.0.10"
FontSize="18" Width="200" Height="75" />
<Button Content="Кнопка 0.0.0.0" FontSize="18"
Width="200” Height="75" />
<Button Margin="0,50,0,0" Content=" Кнопка 0.50.0.0"
FontSize=”18" Width="200" Height="75" />
<Button Margin="-150,170,0,0" Content=" Кнопка -150.170.0.0"
FontSize="18" Width="200" Height="75" />
</StackPanel>
```

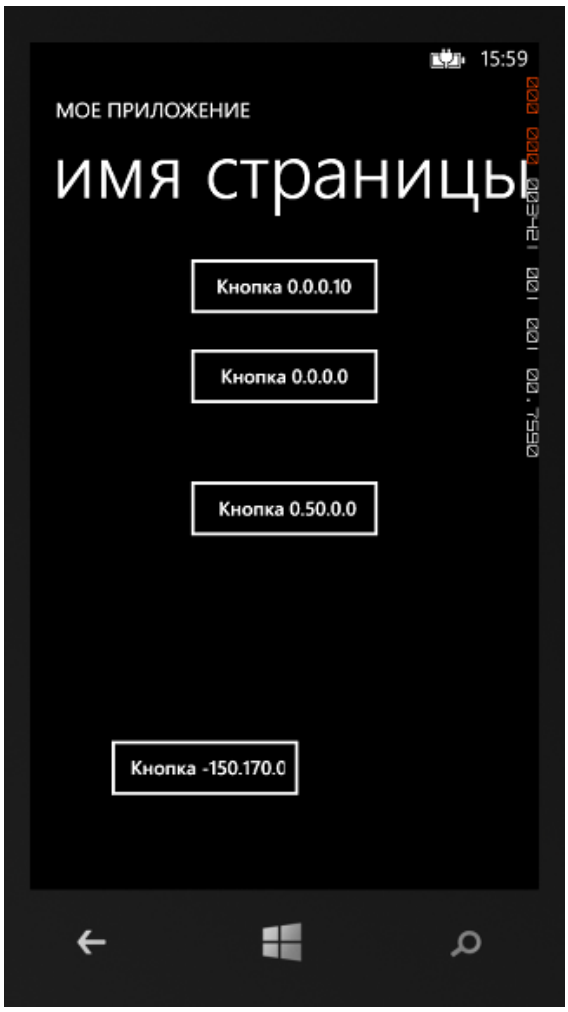

Рисунок 18 - Использование менеджера размещения StackPanel

Поменяем ориентацию StackPanel на горизонтальную. Добавим альбомную ориентацию в список поддерживаемых страницей MainPage.xml и изменим ориентацию экрана по умолчанию также на альбомную. Теперь приложение будет выглядеть, как показано на рисунке 19.

Это наглядный пример того, как можно, используя менеджер размещения StackPanel, создавать интерфейсы, которые будут работать в разных ориентациях экрана.

Менеджер размещения Grid позволяет позиционировать элементы внутри себя максимально гибко. Grid предоставляет возможность размещать элементы, используя строки и столбцы. При использовании менеджера размещения Grid разработчик определяет общую структуру сетки, а потом, используя присоединенные свойства, размещает элементы управления в ячейках сетки.

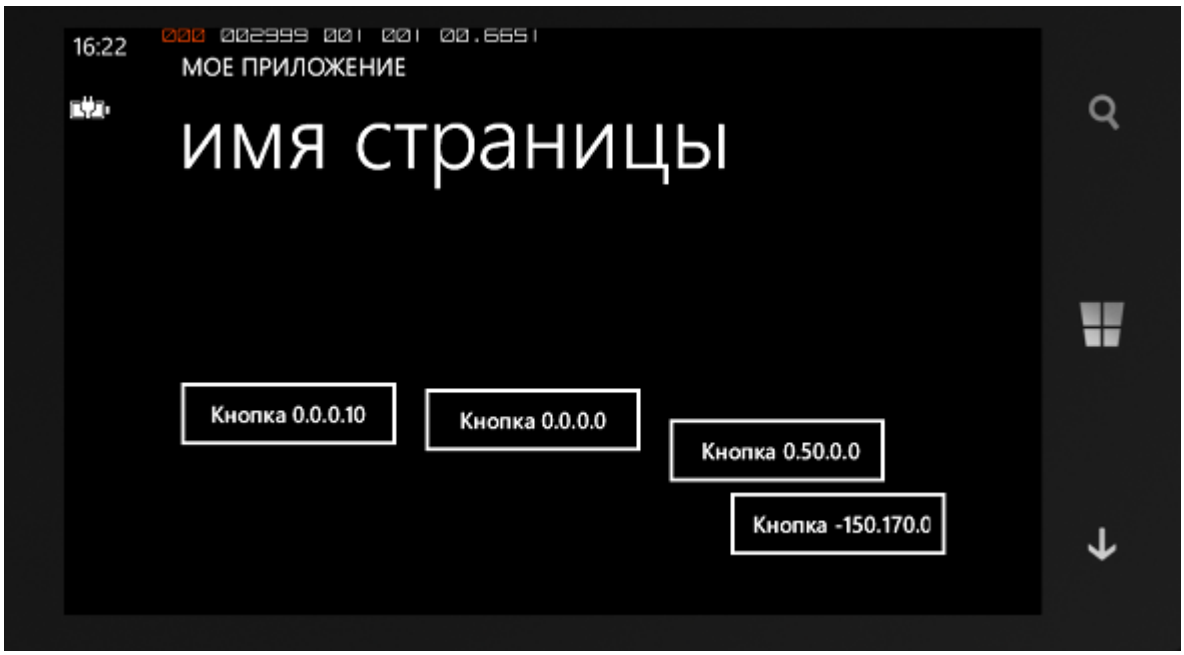

Рисунок 19 - Использование менеджера размещения StackPanel

```
<Grid ShowGridLines="True">
<Grid.RowDefinitions>
<RowDefinition Height="150"/>
<RowDefinition Height="*"/>
<RowDefinition Height="*"/>
<RowDefinition Height="100"/>
<RowDefinition Height="Auto"/>
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
<ColumnDefinition Width="150" />
<ColumnDefinition Width="*" />
<ColumnDefinition Width="150" />
</Grid.ColumnDefinitions>
<Button Grid.Column="0" Grid.Row="0" Content="Кнопка 0.0"
FontSize="18" Width="140" Height="75" />
<Button Grid.Column="2" Grid.Row="0" Margin="10"
Content=" Кнопка 2.0"
FontSize="18" Width="140" Height="75" />
<Button Grid.Column="1" Grid.Row="3" Margin="10"
Content=" Кнопка 1.2"
FontSize="18" Width="140" Height="75" />
</Grid>
```

Обратите внимание, что нумерация строк (Grid.Row) и столбцов (Grid.Column) начинается с 0. При определении строк и столбцов были использованы все три возможных способа указания размера:

- прямое указание размера в пикселах;
- автоматический размер, в зависимости от содержимого (Auto);
- пропорциональное разделение оставшегося пространства (*).

Таким образом, мы имеем в разметке две строки одинаковой высоты (1 и 2), которые пропорционально делят оставшееся от 0-й и 3-й строк место, а 4-я строка отсутствует, поскольку она не содержит ни одного элемента управления.

Приложение, запущенное в эмуляторе, будет выглядеть так, как представлено на рисунке 20.

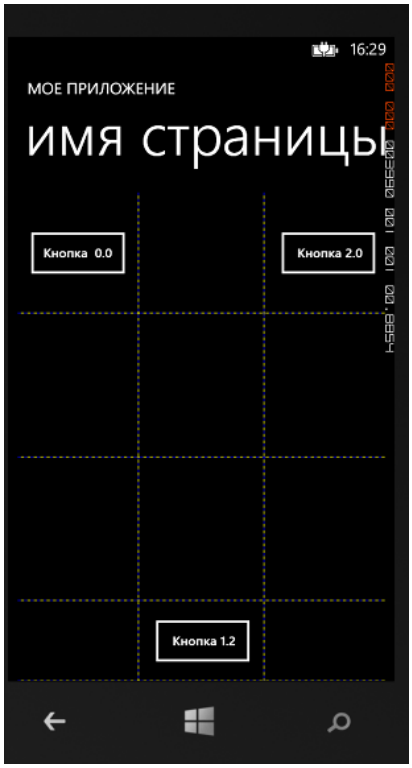

Рисунок 20 - Использование менеджера размещения Grid

Кроме стандартных для всех, основанных на XAML, технологий визуальной разработки. платформа WindowsPhone имеет доступные только для нее варианты разметки, представленные следующими элементами управления:

- Pivot;
- Panorama.

Данные элементы управления представляют собой часть Metro-дизайна, причем в поставке средств разработки присутствуют два специальных шаблона проектов: приложение WindowsPhonePivot и приложение WindowsPhonePanorama, которые позволяют сразу создать приложение на основе указанных элементов управления. Рассмотрим, как добавлять элементы управления Pivoti Panoramав уже существующие приложения. Для этого продолжим работу с ранее созданным приложением ExploreLayoutControls.

Pivot представляет собой элемент управления, предназначенный для создания своеобразных вкладок. Данный сценарий применяется в окне настроек телефона для переключения между настройками самого телефона и настройками приложений. В календаре телефона можно переключиться из режима *day* в режим *agend,* а также с помощью элемента управления pivot. Эти

режимы позволяют посмотреть на одни и те же данные как бы с различных сторон.

Рассмотрим работу с элементом управления Pivot. Двойным щелчком по файлу MainPage.xaml перейдем к его редактированию. В редакторе кода удалим все содержимое менеджера размещения Grid с именем LayoutRoot. Теперь надо добавить элемент управления Pivot на страницу MainPage.xaml внутрь менеджера размещения Grid с именем LayoutRoot. По умолчанию данный элемент управления не отображается на панели элементов в VisualStudio. Также в проект не подключена сборка, содержащая элемент управления pivot. Исправим это.

Чтобы добавить элемент Pivot на страницу приложения, сначала добавьте в проект (раздел ссылки в окне Обозреватель решений) ссылку на библиотеку Microsoft. Phone.Controls. Затем перейдите к корневому элементу страницы MainPage.xaml (phone: PhoneApplicationPage) и после строки:

```
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    в новой строке введите:
    xmlns:metrocontrols=
```

Система Intellisense добавит кавычки и отобразит список доступных пространств имен, как показано на рисунке 21.

Рисунок 21 - Список доступных пространств имен

Выберите пункт Microsoft. Phone.Controls (Microsoft. Phone.Cotrols). В результате строка кода юудет выглядеть как показано ниже:

```
    xmlns:controls=
    "clr-
namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Contro
ls"
```

Выполним сборку проекта (нажав комбинацию клавиш <Ctrl>+<Shift>+<B>). Это нужно для того, чтобы новое пространство имен гарантированно добавилось в систему IntelliSense. Теперь в наше приложение можно добавить элемент управления и разметки Pivot.

Двойным щелчком по файлу MainPage.xamlперейдем к его редактированию. В редакторе XAML-разметки перейдем к основному элементу Gridс именем LayoutRoot. Внутри данного элемента напишем:

```
<metrocontrols:
```

Система IntelliSenseподскажет нам какие элементы управления доступны. В этом пространстве имен доступны элементы управления Pivot, Panoramaи вспомогательные элементы управления Pivotltem и Panoramaltem.

Добавим в XAML-разметку элемент управления pivot и внутрь него 3 элемента управления Pivotltem:

```
<metrocontrols:Pivot Title="PIVOT ПРИЛОЖЕНИЕ>
<metrocontrols: Pivotltem Header="first">
</metrocontrols:PivotItem>
<metrocontrols: Pivotltem Header=" second ">
</metrocontrols:PivotItem>
<metrocontrols: Pivotltem Header=" third ">
</metrocontrols:PivotItem>
</metrocontrols:Pivot>
```

Запустим приложение и протестируем работу элемента управления Pivot. Работа элемента управления Pivotпохожа на классические страницы-закладки, однако это одна страница приложения. Результат работы приложения представлен на рисунке 22.

Рисунок 22 - Элемент управления Pivot

Элемент управления Panorama применяется очень часто, взять хотя бы хабы (центры) телефона, такие как хаб "люди" и хаб "игры". Данный элемент управления представляет собой прокручиваемую область с содержимым, которое является группой секций, выстроенных в ленту.

Несмотря на то, что элементы управления Panorama и Pivot выглядят как несколько страниц, они представляют собой одну страницу приложения.

Если вы выбираете между элементами управления Pivot и Panorama, прислушайтесь к нескольким советам:

- используйте элемент управления Panorama для вашего приложения, если оно представляется хабом (некой длинной лентой, которую пользователю будет интересно прокручивать);
- используйте элемент управления Pivot, когда вам надо показать одни и те же или связанные данные с различных сторон. Как бы расплывчато это не звучало.

Рассмотрим работу с элементом управления Panorama. В редакторе XAML-разметки страницы MainPage.xaml удалим все содержимое элемента управления Gridc именем LayoutRoot. Поскольку мы уже добавили необходимое пространство имен в XAML-файл, можно сразу добавить элемент управления Panorama и 3 элемента управления Panoramaltem внутрь пустого теперь элемента управления Gridc именем LayoutRoot:

```
<metrocontrols:Panorama Title=" Application panorama>
<metrocontrols:Panoramaltem Header="first">
</metrocontrols:PanoramaItem>
<metrocontrols:Panoramaltem Header ="second">
</metrocontrols:PanoramaItem>
<metrocontrols:Panoramaltem Header="third">
</metrocontrols:PanoramaItem>
</metrocontrols: Panorama>
```

О результате работы программы свидетельствует рисунок 23.

Рисунок 23- Элемент управления Panorama

1.7 Задачи запуска и выбора

Windows Phone предоставляет разработчикам способ взаимодействия со встроенными возможностями телефона, такими как осуществление звонка, отправка SMS, получение фотографий с камеры или запуск одного из стандартных приложений, например Internet Explorer, карт или поиска Bing. Это только некоторые из тех возможностей, которые нам доступны с помощью *задач запуска* (Launchers) или *задач выбора* (Choosers). Важно отметить, что с помощью обоих типов задач приложение не может произвести какое-либо потенциально опасное действие или получить данные без явного согласия на то пользователя, например, если приложение хочет осуществить звонок по заданному номеру телефона, то пользователю показывается стандартный системный диалог с кнопкам позвонить и отмена, где также будет отображаться номер, по которому осуществляется звонок. Аналогично и при выборе фотографий, пользователь должен явным образом указать в стандартном системном диалоге, какую фотографию он хочет передать приложению.

Рассмотрим задачи запуска и выбора более подробно. Задачи запуска можно описать как "запустил и забыл". С помощью данного типа задач мы не получаем обратно какие-либо данные. Например, запуская браузер InternetExplorer, мы передаем в параметре задачи запуска адрес страницы, которую хотим показать пользователю. Дальше пользователь работает в браузере. Задачи выбора имеют очень схожую с задачами запуска концепцию, но в отличие от последних они что-то возвращают приложению, будь то картинка с камеры или e-maiI-адрес из адресной книги.

При отладке приложений, в которых используются задачи запуска или выбора, надо учесть, что на эмуляторе многие задачи реально ничего не делают. Например, EmailComposeTask- отправка e-mail предполагает, что на телефоне настроена нужная учетная запись электронной почты. Поскольку на эмуляторе такую настройку произвести нельзя, реально отправить e-mailне получится. Похожая ситуация и со звонком. Реальной SIM-карты в эмуляторе нет. Но пользовательский интерфейс, эмулирующий звонок, показан будет. В целях тестирования данного интерфейса вполне достаточно. Эмулятор даже содержит записи в адресной книге и фотографии, которые можно использовать для тестирования. Тоже относится и к другим задачам запуска и выбора. Однако, несмотря на то что много функциональностей в эмуляторе работает, в целом возможности по тестированию многих задач запуска и выбора на эмуляторе ограничены.

Задачи запуска и выбора определены в пространстве имен Microsoft.Phone.Tasksбиблиотеки Microsoft.Phone.dll, которая подключается автоматически при создании проекта, поэтому для использования задач запуска и выбора требуется только подключить нужное пространство имен:

```
usingMicrosoft.Phone.Tasks;
```

Давайте рассмотрим, как осуществить звонок по определенному номеру телефона:

```
     var phoneCallTask = new PhoneCallTaskO;
phoneCallTask.PhoneNumber = "+77036935213";
     phoneCallTask.DisplayName = "Мария";
     phoneCallTask.Show();
```

ОтправитьSMS можно с помощью кода, приведенного ниже:

```
     var smsComposeTask = new SmsComposeTask();
     smsComposeTask.To = "+77036935213";
     smsComposeTask.Body = "СМС от Phone";
     smsComposeTask.Show();
```

А для отправки ссылки в социальную сеть воспользуемся ShareLinkTask:

```
     var shareLinkTask = new ShareLinkTask));
     shareLinkTask.Title = "Найдетсявсе";
      shareLinkTask.LinkUri = new Uri("http://yandex.kz/",
UriKind.Absolute);
     shareLinkTask.Message = "Портал Яндекс";
     shareLinkTask.Show();
```

Хочется отметить, что вызов метода show задач запуска и выбора необходимо обрамлять блоком try...catch, т. к. данный метод может генерировать исключения:

```
     try
     {
     shareLinkTask.Show() ;
     }
     catch (Exception ex)
     {
     IIобработка исключения
     }
```

В реальных приложениях корректная обработка исключений является очень важной задачей.

Для показа системного диалога настройки Wi-Fi воспользуемся задачей запуска ConnectionSettingsTask:

```
     var connectionSettingsTask = new ConnectionSettingsTask();
     connectionSettingsTask.ConnectionSettingsType =
ConnectionSettingsType.WiFi;
     connectionSettingsTask.Show();
```

Изменяя значение свойства ConnectionSettingsType, можно показать

различные диалоги, например настройки мобильного и 3G-соединения:

```
connectionSettingsTask.ConnectionSettingsType =
ConnectionSettingsType.Cellular;
```

Или настройки Bluetooth:

```
connectionSettingsTask.ConnectionSettingsType =
ConnectionSettingsType.Bluetooth;
```

Или настройкирежимаполета (Airplane Mode):

```
connectionSettingsTask.ConnectionSettingsType =
ConnectionSettingsType.AirplaneMode;
```

Важным отличием задач выбора отзадач запуска является то, что перед вызовом метода show для задач выбора необходимо подписаться нас обытие завершения выбора. В обработчике данного события можно получить данные, выбранные пользователем.

В таблице 1 представлены задачи запуска, которые доступны на телефоне.

Т а б л и ц а 1 - Задачи запуска

Имя задачи	Описание
PhoneCallTask	Запускает встроенный диалог операционной системы для осуществления звонка. Есть возможность задать номер и отображаемое имя абонента. Звонок не осуществляется, пока пользователь не нажмет кнопку Позвонить
SmsComposeTask	Запускает приложение для отправки сообщений. Можно определить получателя и тело сообщения, но пользователь должен сам нажать кнопку Отправить
EmailComposeTask	Позволяет отправить e-mail. На телефоне должен быть настроен аккаунт электронной почты
WebBrowserTask	Запускает браузер InternetExplorerпозволяет задать адрес страницы, открываемой в браузере
SearchTask	Запускает системное приложение поиска вBing. Позволяет задать поисковый запрос
BingMapsTask	Запускает системное картографическое приложение. Позволяет задать географическую точку, которая будет отображаться в центре экрана, а также уровень детализации и строку поиска, используемую для отметки объектов на карте. Если центральная точка не задается, карта отображается в соответствии с положением телефона
BingMapsDirectionsTask	Запускает системное картографическое приложение, автоматически прокладывающее маршрут между двумяточками. Если одна из точек маршрута не задана, то используется текущее местоположение телефона

Продолжение таблицы 1

Имя задачи	Описание
ShareLinkTask	Позволяет отправить ссылку в одну из социальных сетей, в которых зарегистрирова- -сльзс®г-=лг
ShareStatusTask	Позволяет отправить сообщение статуса в одну из социальных сетей, в которых зарегистрирован пользователь
MarketplaceHubTask	Запускает встроенное в Windows Phone приложениеMarketplace. Позволяет задать категорию отображаемого контента
ConnectionSettingsTask	Запускает системный диалог настройки подключения

Изучим теперь подробно задачи выбора. Рассмотрим, как получить фотографию с камеры телефона. Для этого используется задача выбора CameraCaptureTask:

```
var CameraCaptureTask = new CameraCaptureTask();
CameraCaptureTask.Completed += CameraCaptureTaskCompleted;
CameraCaptureTask.Show();
```

Мы подписались на обработчик события завершения задачи, в котором в случае успешного завершения можно получить фотографию:

```
void CameraCaptureTaskCompleted(object sender, PhotoResult e)
{
if (e.TaskResult == TaskResult.OK)
{
BitmapImage bmp = new Bitmaplmage();
bmp.SetSource(e.ChosenPhoto);
Image1.Source = bmp;
}
}
```

В данном примере imagel-это элемент управления Image на странице:

```
<image Name="image1" Stretch="Fill" />
```

Задачи выбора, доступные на телефоне, приведены в таблице 2.

Т а б л и ц а 2 - Задачи выбора

Имя задачи	Описание
CameraCaptureTask	Открывает системный диалог работы с камерой и возвращает приложению картинку, снятую пользователем
PhotoChooserTask	Открывает системный диалог выбора фотографии и возвращает фотографию, выбранную пользователем
PhoneNumberChooserTask	Позволяет пользователю выбрать телефонный номер из адресной книги и возвращает его приложению
Erna ilAddressChooserTask	Позволяет пользователю выбрать email-адрес из

	адресной книги и возвращает его приложению

Продолжение таблица 2

Имя задачи	Описание
SaveContactTask	Позволяет сохранить новый контакт в адресной книге. Данная задача не возвращает приложению собственно данные, но она позволяет узнать, был ли контакт сохранен пользователем или нет, возвращая, таким образом, статус операции, что позволяет отнести ее к задачам выбора
SaveEmailAddressTask	Позволяет сохранить email-адрес в адресной книге. Также как SaveContactTask, возвращает только статус операции
SavePhoneNumberTask	Позволяет сохранить телефонный номер в адресной книге. Также как SaveContactTask, возвращает только статус операции
SaveRingtoneTask	Позволяет сохранить MP3- или WMA-аудиофайл продолжительностью менее 40 с и объемом менее 1 Мбайт в системный список мелодий звонка. Эта задача выбора возвращает только статус операции
GamelnviteTask	Данная задача выбора может быть вызвана только из игр для Xbox LIVE, что ограничивает ее применение. Она позволяет пригласить пользователей в многопользовательскую игру

При вызове метода Show объекта CameraCaptureTask отобразится стандартный диалог работы с камерой, пользователь сделает фото, и в приложении произойдет событие, в обработчике которого, как уже говорилось ранее данное фото можно получить. При отладке приложения в эмуляторе вместо изображения с камеры вы увидите картинку, на которой черный прямоугольник перемещается по экрану. Даже если на вашем компьютере есть Web-камера, она, к сожалению, не будет использоваться эмулятором телефона. В принципе, перемещающегося прямоугольника достаточно в целях отладки, но рекомендуется протестировать приложение на реальном телефоне перед публикацией в Marketplace, чтобы понять, какую картинку дает реальная камера.

Рассмотрим задачу выбора адреса электронной почты:

```
     var emailAddressChooserTask = new EmailAddressChooserTask();
     emailAddressChooserTask. Completed +=
emailAddressChooserTaskCompleted;
     emailAddressChooserTask.Show();
     void EmailAddressChooserTaskCompleted (object sender,
EmailResult e)
      {
     if (e.TaskResult == TeskResult.OK)
     {
     varemail = e.Email;
     }
```

```
}
```

При запуске задачи эмулятор предложит несколько тестовых адресов в адресной книге на выбор. Другие задачи выбора работают похожим образом и также предоставляют тестовые данные. Рассмотрим задачу выбора фотографии из библиотеки телефона:

```
    var photoChooserTask = new PhotoChooserTask();
    photcChooserTask.Completed += PhotoChooserTaskCompleted;
photoChooserTask.Show();
```

1.8 Сенсоры

Современный смартфон или "умный" телефон давно уже больше, чем просто телефон. Это устройство взаимодействия пользователей с информационным пространством, с одной стороны, и с реальным миром, с другой стороны. Интеграция с информационным пространством, и передача данных реального мира позволяют создавать решения нового уровня, которые уже начинают казаться нам обычными. Например, пользователи информационного картографического сервиса в реальном времени строят карту загруженности дорог и отмечают проблемные точки, а потом все вместе пользуются этой информацией - это картографический сервис из информационного пространства и сервис местоположения на смартфоне. Кажущийся сейчас привычным и обычным, этот сервис еще совсем недавно казался фантастикой.

Платформа Windows Phone поддерживает сервис определения местоположения, акселерометр, гироскоп, компас и специальный Motion API, который аккумулирует данные со всех датчиков и, используя результаты исследований Microsoft Research, предоставляет разработчику данные о положении и движении устройства в пространстве.

Во всех устройствах, присутствующих на рынке, на момент написания книги был доступен сервис определения местоположения и акселерометр, на некоторых доступен гироскоп, компас наименее распространен.

Система предоставляет разработчику API, который позволяет узнать, поддерживает ли устройство, на котором исполняется программа, соответствующий сенсор.

Акселерометр предоставляет разработчику трехмерный вектор ускорения устройства в гравитационных единицах, ориентированный относительно устройства. На рисунке 24 показан акселерометр работы эмулятора.

Данные акселерометра включают в себя действие гравитации на устройство. Если устройство лежит на столе вверх экраном, Z-компонента вектора равна -1. Если устройство стоит на торце, экраном к разработчику, У-компонента вектора равна-1.

Акселерометр предоставляет структуру AccelerometerReading, содержащую вектор Acceleration, с ускорением устройства по осям. Работа с акселерометром осуществляется с помощью класса Accelerometer.

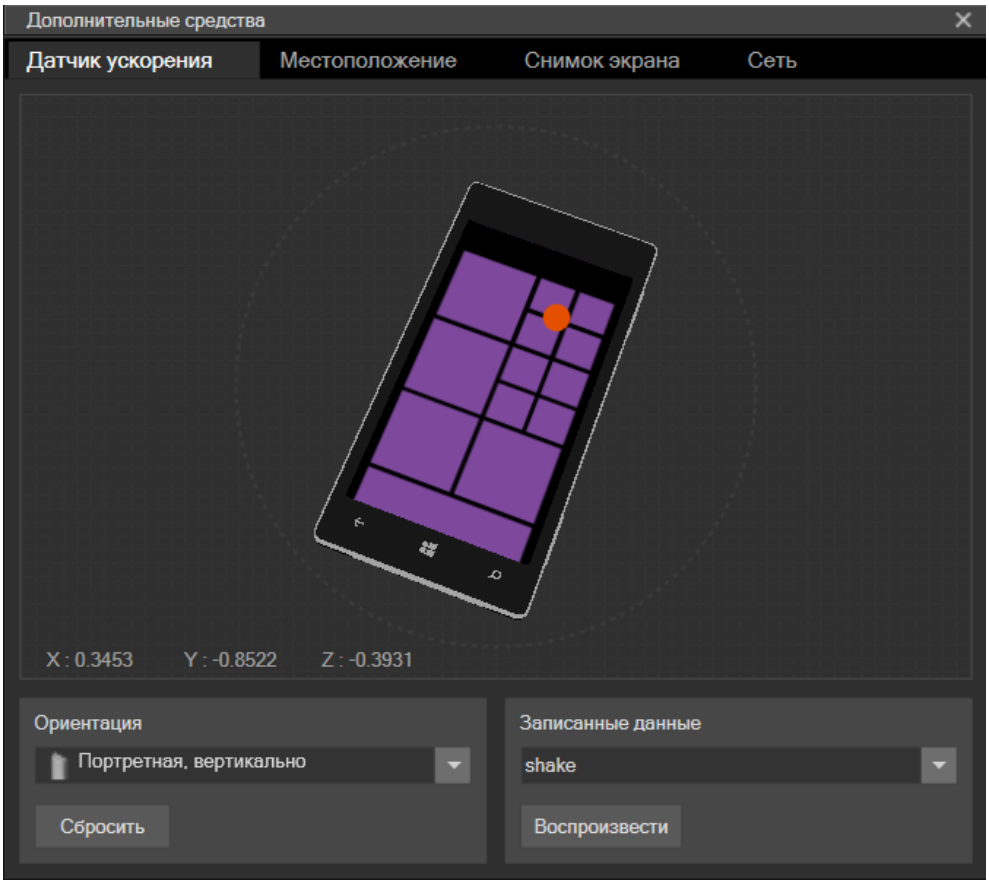

Рисунок 24 - Акселерометр окна Дополнительных средств эмулятора

Для работы с акселерометром к проекту надо подключить библиотеку Microsoft.Devices.Sensors. Необходимые классы определены в пространстве имен Microsoft.Devices.Sensors. Обработчик события изменения состояния акселерометра использует класс vector3 из библиотеки Microsoft.Xna.Framework для работы с вектором ускорения по осям. Чтобы работать с этой библиотекой, необходимо добавить в проект ссылку и подключите пространство имен Microsoft.Xna.Framework.

Гироскоп - это устройство для измерения ориентации в пространстве. Гироскоп предоставляет скорость вращения устройства вокруг каждой из осей в радианах в секунду. Как правило, вы не будете использовать гироскоп напрямую, а воспользуетесь Motion API, который объединяет данные акселерометра, гироскопа и компаса. Гироскоп есть не на всех аппаратах с Windows Phone, поэтому перед использованием гироскопа необходимо проверить его доступность:

```
if (Gyroscope.IsSupported)
{
```

```
// гироскоп доступен
}
```

Эмулятор не предоставляет способа работать с гироскопом, в отличие от работы с акселерометром. Поэтому значением свойства IsSupported класса Gyroscope будет false.

Класс Gyroscope, позволяющий работать с гироскопом, предоставляет структуру GyroscopeReading, содержащую вектор RotationRate, со скоростями вращения устройства. Код использования гироскопа похож на код использования акселерометра. Ниже демонстрируется получение данных от гироскопа:

```
if (Gyroscope.IsSupported)
{
var myGyroscope = new Gyroscope();
myGyroscope.CurrentValueChanged +=
MyGyroscopeCurrentValueCnanged;
myGyroscope.Start();
}
void MyGyroscopeCurrentValueChanged(object sender,
SensorReadingEventArgs<GyroscopeReading> e)
{
// код, обрабатывающий данные гироскопа
}
```

Нет необходимости рассказывать, что такое компас. Он знаком всем. Не забывайте только, что компас представляет собой магнитометр, это означает, что рядом с источниками магнитного поля его данные могут искажаться.

Отметим также, что эмулятор не поддерживает компас. Поэтому тестировать работу с компасом лучше на реальном устройстве. Компас предоставляет разработчикам следующие данные в виде структуры CompassReading:

- HeadingAccuracy - точность измерения в градусах;
- MagneticHeading- направление на магнитный полюс (в градусах, против часовой стрелки);
- TrueHeading- направление на географический полюс (в градусах, против часовой стрелки);
- MagnetoraeterReading - данные магнитометра в виде вектора в микротеслах.

Использование компаса аналогично использованию акселерометра и гироскопа. Единственное отличие состоит в том, что компасу может потребоваться калибровка. Событие калибровки (Calibrate) вызывается системой, если точность измерения (HeadingAccuracy) становится хуже 20°. Ниже показан код, демонстрирующий получение данных от компаса:

```
if (Compass.IsSupported)
{
```

```
     var myCompass = new Compass ();
     myCompass.Calibrate += MyCompassCalibrate;
     myCompass.CurrentValueChanged +=
MyCompassCurrentValueChanged;
     }

     void MyCompassCalibrate(object sender, CalibrationEventArgs
e)
     {
     // код, реализующий калибровку
     }

     void MyCompassCurrentValueChanged(object sender,
     SensorReadingEventArgs<CompassReading> e)
     {
     // код, обрабатывающий данные компаса
     }
```

Для калибровки компаса пользователю необходимо физически выполнять движение устройством в виде цифры 8 или знака бесконечности в любой из плоскостей. Например, выполнить движение перед собой или взять в руки устройство и пройти путь в виде цифры 8. Разработчик приложения должен сам отобразить интерфейс, который объяснит пользователю, как выполнить калибровку, а также завершить процесс калибровки, как только значение HeadingAccuracy станет приемлемым.

Motion API комбинирует в себе данные акселерометра, гироскопа и компаса и выдает обработанный результат в виде положения в пространстве и характеристик движения устройства. Motion API - это результат исследований Microsoft Research, которым теперь могут воспользоваться все разработчики приложений для Windows Phone.

Обратите внимание, чтобы на устройстве был доступен Motion API, в дополнение к акселерометру необходим, по крайней мере, компас. Эмулятор компас не поддерживает, поэтому и Motion API на нем не доступен.

Достаточно часто в качестве синонима Motion API можно встретить название Motion Sensor, поскольку Motion API доступен в пространстве имен Microsoft.Devices.Sensors и работа с ним аналогична работе со всеми остальными сенсорами на устройстве.

MotionAPI позволяет получить структуру MotionReading со следующими данными о положении устройства в пространстве (Attitude) в разнообразных представлениях (матрица (RotationMatrix), кватернион (Quaternion)):

- Pitch (наклон), Roll (вращение), Yaw (поворот);
- Quaternion;
- RotationMatrix.

Также структура содержит:

- DeviceAcceleration - линейное ускорение устройства;
- DeviceRotationRate - скорость вращения устройства;
- Gravity- гравитационный вектор.

Pitch, Roll и Yaw предоставляют данные о вращении устройства вокруг этих же осей. MotionAPI предоставляет всю необходимую информацию, чтобы создавать свои приложения дополненной реальности (augmentedreality) или активно использовать сенсоры устройства, как источники данных.

Кроме вышеназванных устройств, Windows Phone должно иметь ряд аппаратных возможностей - иногда называемых датчиками - и предоставлять некоторые программные сервисы, возможно, с аппаратной поддержкой. Рассмотрим те из них, которые наиболее интересны разработчикам:

– - Wi-Fi - для доступа к Интернету телефон снабжен Wi-Fi в дополнение к доступу к данным по технологиям 3G (3G data access), предоставляемому поставщиком мобильной связи. В программное обеспечение, установленное на телефоне, включена версия Internet Explorer;

– - Камера - телефон снабжен камерой с разрешением не менее 5 мегапикселей и вспышкой. Программы могут вызывать ПО камеры для осуществления ввода с нее или регистрироваться как приложение расширений для обработки фотографий. В этом случае они будут отображаться в меню для получения доступа к сфотографированным изображениям, например, для обработки этих изображений определенным образом;

– - Местоположение - по желанию пользователя телефон может применять множество стратегий определения своего географического местоположения. Телефон передает в аппаратное устройство GPS данные из Интернета или вышек сотовой связи. При его перемещении также могут предоставляться данные о направлении и скорости;

– - Вибрация - программное управление вибрацией телефона;

– - FM-радио - программный доступ к FM-радио.

Принудительные уведомления - для обновления данных, предоставляемых некоторыми Веб-сервисами, телефону пришлось бы регулярно опрашивать эти сервисы. Это могло бы приводить к разрядке батареи и сокращению времени автономной работы.

С согласия пользователя приложение для Windows Phone 8 может принимать географические координаты телефона по методу Assisted-GPS или A-GPS.

Наиболее точный метод определения местоположения - по сигналам спутников глобальной системы позиционирования (Global Positioning System, GPS). Но GPS может быть медленной. Эта система плохо работает в больших городах и в помещениях, ее применение энергоневыгодно, потому что приводит к большому расходу заряда батареи. Для энергосбережения и увеличения скорости, система A-GPS может определять местоположение по вышкам сотовой связи или сети. Эти методы намного более производительны и надежны, но менее точные.

Основным классом для определения местоположения является GeoCoordinateWatcher (Система отслеживания географических координат). Нам понадобится ссылка на сборку System.Device, и директива using для пространства имен System.Device.Location. В файле WMAppManifest.xml должен присутствовать такой тег:

```
<Capability Name="ID_CAP_LOCATION"/>
```

Конструктор GeoCoordinateWatcher принимает один из элементов перечисления GeoPositionAccuracy (Точность географических координат):

– - Default (По умолчанию);

– - High (Высокая);

После создания объекта GeoCoordinateWatcher необходимо задать обработчик события PositionChanged (Местоположение изменилось) и вызвать метод Start. Событие PositionChanged возвращает объект GeoCoordinate (Географические координаты), имеющий восемь свойств:

– Latitude (Широта), значение типа double в диапазоне от -90 до 90 градусов;

– Longitude (Долгота), значение типа double в диапазоне от -180 до 180 градусов;

– Altitude (Высота) типа double;

– HorizontalAccuracy (Точность в горизонтальном направлении) и VerticalAccuracy (Точность в вертикальном направлении) типа double;

– Course (Курс), значение типа double в диапазоне от 0 до 360 градусов;

– Speed (Скорость) типа double;

– IsUnknown (Неизвестно), булево значение, которое равно true, если Latitude или Longitude не является числом;

Если приложение не имеет разрешения на определение местоположения, свойства Latitude и Longitude будут иметь значение Double.NaN, и IsUnknown будет true. Кроме того, в GeoCoordinate есть метод GetDistanceTo (Определить расстояние до), вычисляющий расстояние между двумя объектами GeoCoordinate.

Мы рассмотрим первые два свойства. Их называют географическими координатами. Они определяют точку на поверхности Земли. Широта -это угловое расстояние от экватора. Как правило, широта обозначается как угол от 0 до 90 градусов с указанием С или Ю (сервер или юг). Например, широта Нью-Йорка примерно 40°С. В объекте GeoCoordinate широты к северу от экватора соответствуют положительным значениям, и широты к югу от экватора - отрицательным. Северный Полюс соответствует 90°, и Южный Полюс --90°.

Географические местоположения с одинаковой широтой образуют линию широты или параллель. Для заданной широты долгота - это угловое расстояние от нулевого меридиана, который проходит через Гринвичскую

астрономическую обсерваторию, Англия. Долготу определяют, как западную или восточную. Нью-Йорк располагается на 74° западной долготы, потому что он находится на запад от нулевого меридиана. В объекте GeoCoordinate положительные значения долготы соответствуют восточному полушарию, и отрицательные значения - западному. Значения 180 и -180 встречаются на линии перемены дат.

Пространство имен System.Device.Location включает классы, которые используют географические координаты для определения адресов (улиц и городов). Как и в случае с акселерометром, GeoCoordinateWatcher создается и инициализируется в перегруженном Initialize. Обработчик события вызывается в том же потоке.

1.9 Программирование игр на XNA

Давайте задумаемся: а чем же игра отличается от обычного приложения? На самом деле, разделение очень условно: например, стоит ли относить клиент сервиса foursquare, позволяющий нам "отмечаться" в различных местах и получать за это баллы, к играм? Или это бизнес-приложение с элементами игры?

Игры для Windows Phone обычно выделяют в отдельную категорию приложений по ряду причин. Во-первых, все игры в телефоне объединены в единый Xbox Hub, что выделяет их из остальных приложений. Многие игры используют сервисы Xbox Live, при этом игры и сам хаб интегрированы с достижениями Xbox, аватаром и т.д.

Во-вторых, для создания действительно динамичных графических приложений - особенно 3D-игр- технология Silverlight не очень подходит. Действительно, для написания игр на компьютере или Xbox часто используют трехмерную графику DirectX с возможностью программирования различных эффектов, отображаемых при помощи графического процессора. В то же время Silverlight предоставляет все необходимое для программирования 2D-примитивов, но в нем отсутствуют возможности по эффективному построению трехмерных моделей. Безусловно, 3D-приложения также можно программировать на Silverlight, но это будет неэффективно, поскольку не будут использоваться ресурсы графического процессора, да и на программиста ляжет значительная работа по переведению сцены из пространства в плоскость.

Таким образом, Silverlight действительно пригоден для программирования некоторых игр - особенно логических (в которых преобладает интерфейс) или двухмерных. Технологии, которые для этого необходимо знать, ничем не отличаются от технологии построения деловых приложений.

XNA - это набор библиотек, работающий поверх Microsoft .NET, причем как на телефоне, так и на других устройствах: полноценном персональном компьютере, Xbox 360 и Zune. Причем код игры для всех этих устройств может

отличаться весьма незначительно лишь, в первую очередь, в способе управления игрой и разном разрешении экранов. Остальные аспекты - вывод графики, звука, сохранение игры и т. д. - максимально унифицированы для всех устройств.

XNA появился за несколько лет до Windows Phone, как технология, которая позволила многим программистам-любителям создавать игры для Xbox 360 и для ПК. Ранее создание игр на Xbox было доступно только крупным студиям с большими бюджетами, способными купить соответствующие специализированные SDK и аппаратные средства разработки. Теперь же, благодаря платформе .NET, стало возможным писать игры сразу для нескольких устройств с минимальными изменениями кода.

Благодаря тому, что XNA работает поверх доступной платформы .NET (это полноценная .NET 4.0 на компьютере и .NET Compact Framework на других устройствах), вы можете использовать в играх все возможности платформы (например, средства сетевого взаимодействия). В частности, на телефоне вам доступны средства геолокации, отправки сообщений, акселерометр, доступ к адресной книге и все остальные описанные в этой книге средства.

Следует отметить, что XNA может использоваться не только для создания игр, но и для более широкого круга динамичных богатых графических приложений - например, для научной визуализации, или для создания инновационных 3D-интерфейсов. Однако так же, как Silverlight нечасто используют для написания игр, редко кому приходит в голову применять XNA для деловых интерфейсных приложений.

Тому есть веские основания - модель программирования XNA резко отличается от Silverlight. Используя Silverlight, мы привыкли, что интерфейс приложения задается отдельно от программного кода с помощью языка разметки, а затем поверх него на языке C# описывается поведение системы. Такая программная модель основана на реакции на события, которые генерируются в ответ на определенные действия пользователя (нажатие кнопки, переход в определенное поле и т. д.) и приводят к вызову различных фрагментов кода. Другая распространенная программная модель - это модель "запрос - ответ", она присуща Web-приложениям, которые отрисовывают страницу пользователю в зависимости от посланных данных формы.

Программная модель XNA-приложения весьма отличается от двух рассмотренных выше. Модель событий не может применяться в играх, поскольку она не очень подходит для решения задач реального времени - а ведь нам хочется, чтобы игра разворачивалась перед нашими глазами именно в реальном времени, со скоростью не менее чем 25 кадров в секунду! Поэтому игра имеет весьма простую программную модель, называемую *циклом игры*.

По сути дела, цикл игры - это почти бесконечный цикл, продолжающийся все время работы игры. В ходе цикла мы постоянно перерисовываем содержимое экрана, чтобы создать иллюзию непрерывности происходящего. Обычно в настольных играх цикл игры выполняется как можно чаще (многих

игроманов радует частота кадров больше 60!), в то время как на телефоне, с целью экономии заряда батареи, обычно используют частоту обновления 30 кадров в секунду.

Помимо отрисовки, в ходе цикла игры мы обрабатываем действия пользователя - в случае с телефоном это прикосновения к экрану и другие возможные взаимодействия. Как правило, выделяют некоторую модель игрового мира, которая обновляется на каждом цикле в зависимости от действий пользователя. Например, изменяются координаты движущихся объектов, подсчитывается новая скорость в зависимости от показаний акселерометра и т. д. Потом уже по этой модели происходит отрисовка экрана.

В XNA игровой цикл разбит на 3 основные функции, которые необходимо определить, чтобы описать игру:

1 В начале игры (или игрового уровня) вызывается специальная функция *LoadContent* для загрузки основных ресурсов (графических и звуковых элементов).

2 Затем в цикле попеременно вызываются:

- функция *Update*, служащая для обновления состояния игры в зависимости от действия пользователя с устройствами ввода;
- функция *Draw*, служащая для отрисовки состояния игры на экране.

Игровой цикл XNA схематически представлен на рисунке 25.

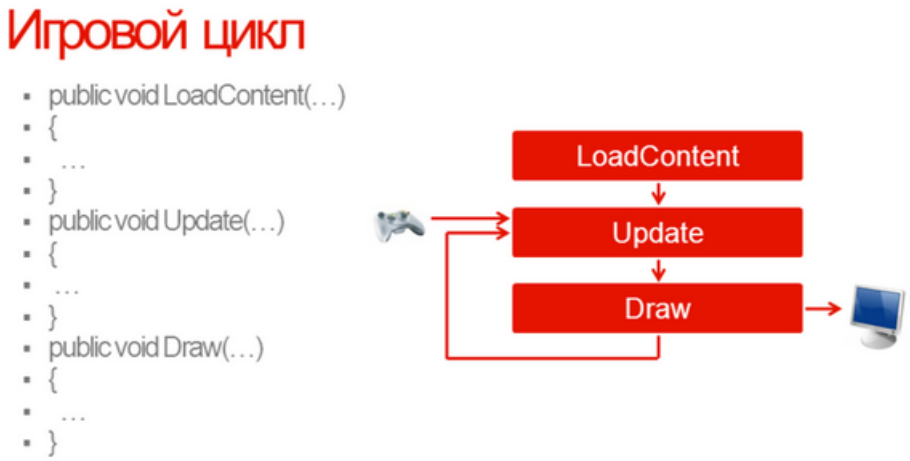

Рисунок 25 - Игровой цикл XNA

Прочитав о программной модели XNA, вы можете подумать, что для создания игры достаточно запрограммировать лишь один цикл игры. Однако это не совсем так. Даже для самой простой игры необходимо, помимо программирования, создать оригинальные графические элементы, музыкальное сопровождение и прочие артефакты - это могут быть представленные в каком-то внутреннем формате карты уровней игры или же видеофрагменты, вставляемые между уровнями.

Графические элементы могут быть как двумерными спрайтами, так и трехмерными моделями. Для двумерной игры графика обычно представляет собой набор картинок (спрайтов). которые затем выводятся на экран в соответствии с логикой игры. Такие модели удобно рисовать в инструментах семейства Microsoft Expression, в частности, с помощью Expression Design. Для рисования трехмерных объектов подойдет соответствующий редактор, например, Blender.

Звуковые компоненты обычно представляют собой либо короткие шумовые эффекты в формате WAV, либо музыку MP3. Для обработки звука удобно использовать звуковые редакторы, из свободно распространяемых можно порекомендовать Audacity.

Прежде чем программировать игру, надо определиться, что будет представлять собой состояние игры. В простейшем случае это может быть набор переменных, определяющих положение элементов на экране, их скорость, а также вспомогательную информацию: счет игрока и т. д. В более сложном случае состояние может содержать множество независимых взаимодействующих объектов или агентов, обладающих своим интеллектом и логикой.

Таким образом, для создания примитивной игры необходимо:

1 Создать графические и звуковые элементы оформления игры.

2 Описать переменные для хранения всех необходимых графических и музыкальных элементов и переопределить метод LoadContent для загрузки их в память.

3 Определить состояние игры. т. е. набор переменных и структур данных, которые будут описывать игру в каждый момент времени.

4 Обработать действия пользователя с устройствами ввода и запрограммировать логику изменения состояния игры в методе Update.

5 Запрограммировать отображение состояния на экран в методе Draw. Здесь опять же может использоваться 2D- или 30-графика, в зависимости от стиля игры.

6 Отладить игру сначала на эмуляторе, а потом и на реальном устройстве Windows Phone.

7 Играть, наслаждаться, делиться игрой с другими (публикация на Windows Phone Marketplace) и т. д.

Конечно, такие шаги уместны только для очень простой игры. Если игра более сложная, то разумно провести рефакторинг и создать свою объектную модель, чтобы было проще ориентироваться в различных аспектах игры и грамотно разделять исходный код:

1 Если игра содержит множество движущихся и ведущих себя по общим правилам объектов, то имеет смысл выделить логику загрузки, отрисовки и перемещения объектов в отдельный класс. Если объекты при этом отличаются по поведению, то можно выстраивать соответствующую объектную иерархию.

2 Если игра содержит несколько уровней, имеет смысл выделить каждый уровень в отдельный класс- тогда для каждого уровня придется частично

повторять описанные ранее действия. Помимо этого, придется создать класс, отвечающий за смену уровней и мультиплексирование методов LoadContent/update/Draw между уровнями.

Проиллюстрируем процесс создания игры на простом примере - морского боя. Сразу отмечу, что морской бой - это не та игра, в которую многие играли в школе на бумаге в клеточку. Мы будем реализовывать игру, которая была доступна лет 30 назад в игровых автоматах. В ней надо было попасть торпедой в плывущий вражеский корабль. В нашей игре также будет корабль, плавающий в верхней части экрана, и снаряды, которые можно будет выпускать из нижней части экрана путем прикосновения к нему. Основной экран игры изображен на рисунке 26.

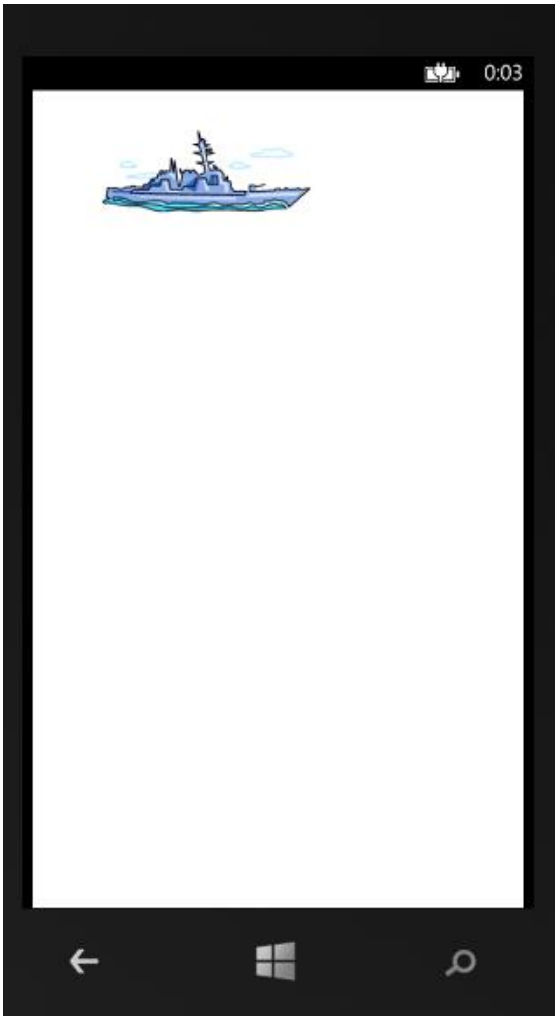

Рисунок 26 - Основной экран игры

Для игры нам потребуются три основных рисунка: корабль, ракета и изображение взрыва. Для данного примера все рисунки в формате PNG в виде графических файлов Explode.png, Ship.png, Rocket.png.

Запустим Visual Studio и выберем новый проект типа XNA Game Studio 4.0 - Игра для Windows Phone (4.0). Для примера, назовем проект WindowsPhoneShipBattle, как показано на рисунке 27.

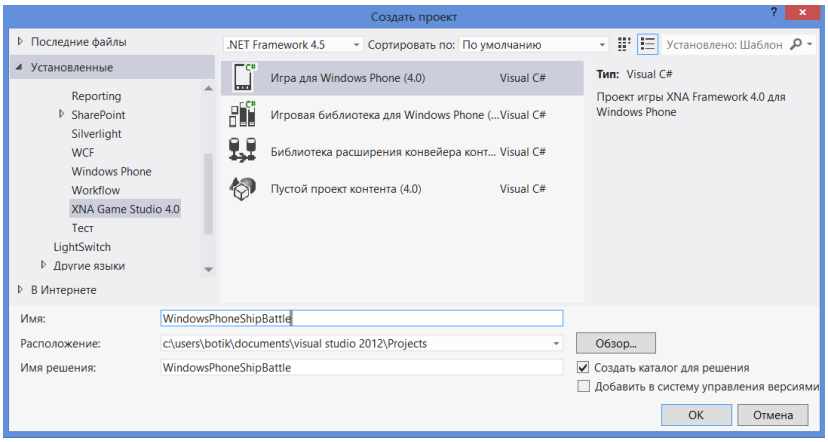

Рисунок 27 - Создание нового проекта игры

После этого выберем версию Windows Phone 7.1 - и Visual Studio создаст для нас каркас игры, который будет состоять из двух проектов:

1 WindowsPhoneShipBattie - это основной проект игры, в котором располагается вся логика, а также ресурсы, связанные с приложением Windows Phone (значки приложения, метаданные и т. д.). Опишем основные файлы этого проекта.

- Gamel.cs - это главный игровой файл, описывающий класс игры, в котором программируются упомянутые ранее функции игрового цикла LoadContent, Update и Draw;

- Program.cs - файл, запускающий игру на выполнение. Он не представляет собой особого интереса и почти никогда не требует модификации;

- Background.png и PhoneGameThumb.png- сгенерированные по умолчанию значки приложения.

2 WindowsPhoneShipBattleContent - это отдельный проект (так называемый Content Pipeline), в котором должны располагаться игровые артефакты: изображения, звуковые и видеофайлы и т. д. В процессе компиляции игры все артефакты в Content Pipeline преобразуются из исходных форматов во внутренний формат, используемый XNA, с помощью специальных процессоров. Для стандартных графических, музыкальных и 3D-форматов могут использоваться встроенные стандартные процессоры, однако возможно и создание своих процессоров для нестандартных ситуаций: например, преобразование формата хранения уровней и др.

Для того чтобы поместить в проект графические артефакты, достаточно просто перетащить их из Проводника Windows в проект WindowsPhoneShipBattle/, теперь обозреватель решений выглядит, как показано на рисунке 28.

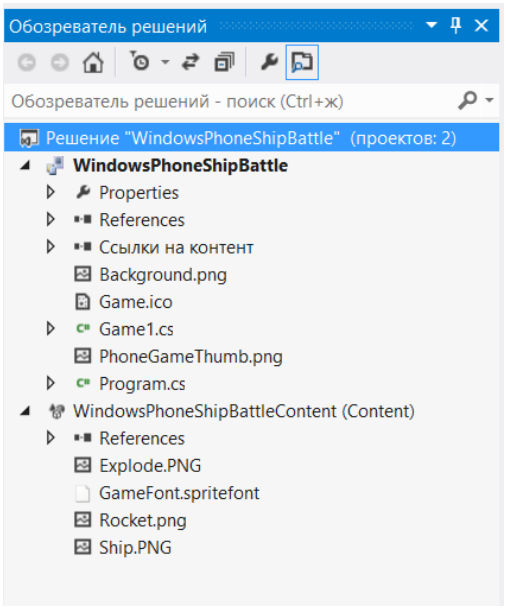

Рисунок 28 - Добавление графических файлов в проект

Если мы откомпилируем и запустим работающую игру, мы увидим закрашивание экрана в васильковый цвет. За это отвечает метод Draw в Gamel.cs (для лучшего понимания комментарии были удалены):

```
protected override void Draw(GameTime gameTime)
{
GraphicsDevice.Clear(Color.CornflowerBlue);
base. Draw (gameTiir.e);
}
```

Чтобы нарисовать на экране что-то другое, нужно явным образом изменить метод Draw. Для начала нарисуем корабль на экране. Для этого в классе игры создадим переменные, которые будут хранить Рисунок корабля (типа Texture2D), а также его положение и скорость (типа Vector2):

```
 Texture2D ship;
Vector2 ship_pos = new Vector2(0, 30);
Vector2 ship_dir = new Vector2(3, 0);
```

Метод LoadContent запрограммируем на загрузку изображения корабля (помимо этого, в нем останется еще одна строчка для создания объекта SpriteBatch, необходимого для рисования):

```
protected override void LoadContent()
{
spriteBatch = new SpriteBatch(GraphicsDevice);
 ship = Content.Load<Texture2D>("Ship”);
}
```

Наконец, метод Draw будет заниматься рисованием корабля:

```
protected override void Draw(GameTime gameTime)
{
GraphicsDevice.Clear(Color.White);
spriteBatch.Begin ();
spriteBatch.Draw(ship, ship_pos, Color.White);
spriteBatch.End();
 base.Draw(gameTime);
}
```

Здесь следует отметить одну тонкость - при рисовании спрайтов на экране мы рисуем картинки "порциями", называемыми SpriteBatch. Соответственно, мы открываем такую "рисовательную транзакцию" вызовом spriteBatch.Begin() и заканчиваем вызовом spriteBatch.End (), между которыми расположены вызовы spriteBatch.Draw() или DrawString(). В XNA принят подход к рисованию, называемый Immediate Mode Rendering, т. е. выполнение команд без промежуточной буферизации. При этом явно указываемая программистом транзакционность рисования позволяет избежать мерцания путем оптимизации процесса вывода на экран для последовательных кадров. Запустив игру на этом этапе, мы увидим, что XNA по умолчанию использует горизонтальную ориентацию для игры. Для изменения ориентации экрана на портретную добавим три строчки кода в конструктор игрового класса:

```
graphics.SupportedOrientations = DisplayOrientation.Portrait;
graphics.PreferredBackBufferHeight = 800;
graphics.PreferredBackBufferWidth = 480;
```

Чтобы заставить корабль двигаться, необходимо в методе Update изменять его координаты, которые в данном случае представляют собой состояние игры:

```
protected override void Update(GameTime gameTime)
{
width = GraphicsDevice.Viewport.Width;
 height = GraphicsDevice.Viewport.Height;
ship_pos += ship_dir;
if (ship_pos.X <= 0  ||  ship_pos.X > width - ship.Width)
{
ship_dir = -ship_dir;
}
base.Update(gameTime);
}
```

Здесь мы также определяем ширину и высоту экрана в переменных height и width (соответствующие переменные необходимо также описать в игровом

классе). Далее корректируем позицию корабля, а затем обрабатываем "отскакивание" корабля от края экрана.

Обратите внимание, что XNA поддерживает типы данных из линейной алгебры (в данном случае двумерные векторы, но также и трехмерные векторы, матрицы и т. д.) и операции с ними, что сильно упрощает многие операции, поскольку они выполняются естественным образом. Например, в данном случае мы корректируем положение корабля простым добавлением к вектору координат вектора скорости (неявно умноженного на время, предположительно равное 1).

В заключение, для более красивого рисования корабля, слегка модифицируем метод рисования. Теперь, если корабль плывет вправо, его картинка отражается вертикально вокруг свой оси. так что корабль все время плывет носом вперед, как это обычно происходит в реальной жизни:

```
spriteBatch.Draw(ship, ship_pos, null, Color.White,Of,
Vector2.Zero,1.Of,  ship_dir.X>0 ?
SpriteEffects.FlipHorizontally : SpriteEffects.None,Of);
```

Теперь добавим ракету и, собственно, возможность игры, т. е. взаимодействия программы с пользователем. Для начала в игровом классе нам потребуется определить переменные для графического изображения ракеты и взрыва. Дополнительно для отрисовки взрыва понадобится переменная explode она будет вести обратный отсчет числа кадров, во время которых вместо корабля отображается взрыв:

```
Texture2D rocket, explosion; // текстуры
Vector2 rocket_pos = new Vector2(0, 0); // положение ракеты
 int explode = 0; // признак того, что недавно был взрыв
```

Положение ракеты, равное нулевому вектору, соответствует ситуации, когда ракета не была выпущена.Отрисовка ракеты в методе Draw не представляет сложностей; кроме того, если недавно был взрыв и переменная explode не равна нулю, то вместо корабля нам надо отрисовывать соответствующий спрайт со взрывом:

```
protected override void Draw(GameTime gameTime)
{
GraphicsDevice.Clear(Color.White);
spriteBatch.Begin();
if (explode == 0)
{
spriteBatch.Draw(ship, ship_pos, null,
Color.White,Of,Vector2.Zero,1.Of, ship_dir.X>0 ?
SpriteEffects.FlipHorizontally :
SpriteEffects.None, Of);
}
else spriteBatch.Draw(explosion, ship_pos, Color.White);
```

```
if (rocket_pos != Vector2.Zero)
{
spriteBatch.Draw(rocket, rocket_pos, Color.Red);
}
spriteBatch.End();
base.Draw(gameTime);
}
```

Теперь обсудим по частям метод Update. Первая часть отвечает за отрисовку взрыва: когда переменная-флаг explode ненулевая, единственная задача нашей игры - отрисовать взрыв. Поэтому мы просто уменьшаем счетчик кадров, в течение которых демонстрируется взрыв, а когда он (счетчик) достигает нулевой отметки - возвращаем корабль в исходное положение, чтобы игра началась снова:

```
protected override void Update(GameTime gameTime)
{
if (explode > 0)
{
explode --;
 if (explode == 0)
{
ship_pos.X = 0;
ship_dir.X = 3;
}
base.Update(gameTime) ;
 return;
}
.......
base.Update(gameTime);
}
```

В конце метода Update вызывается метод Update базового класса. Следующий фрагмент кода отвечает за движение корабля влево-вправо:

```
ship_pos += ship_dir;
if (ship_pos.X + ship.Width >= width || ship_pos.X <= 0)
{
ship_dir = -ship_dir;
}
```

При обработке действий пользователя, следует решить, как будет происходить управление игрой - касанием экрана, жестами, поворотом или встряхиванием телефона (в этом случае надо задействовать акселерометр), нажатием на кнопки или как-то еще. В любом случае обработка действий пользователя занимает значимое место в функции Update.

В отличие от элементов управления Silverlight, управляемых событиями, в XNA обработка действий пользователя ведется по принципу опроса состояния в цикле игры. Например, нажатие на каком-то элементе управления в

Silverlight- приложении ведет к возникновению однократного события нажатия, в то время как XNA-приложение может опрашивать состояние сенсорной панели телефона и определять, касается ли его палец в данный момент или нет.

В нашей игре мы будем управлять игрой именно с помощью прикосновений к экрану. Для обработки прикосновений существует специальный объект TouchPanel, для которого метод TouchPanel.GetStateO возвращает текущее состояние панели телефона. Состояние в свою очередь может содержать информацию о нескольких одновременных касаниях (поддержка multi-touch). Свойство Count в этом случае содержит количество одновременных касаний или 0, если пользователь вообще не касается панели. Мы будем обрабатывать лишь одно (первое) касание, и в случае, если такое касание есть, будем запускать ракету из точки, X-координата которой совпадает с касанием. а координата по вертикали - фиксирована где-то внизу экрана:

```
var tc = TouchPanel.GetState();
 if (tc.Count>0)
{
rock_pos.X = tc[0].Position.X;
 rock_pos.Y = 750;
}
```

Ниже приведен код, отвечающий за движение ракеты и отработку столкновения ракеты с кораблем:

```
if (rock_pos != Vector2.Zero)
{
rock_pos += new Vector2(0, -7);
if (rock_pos.Y >= 0 && rock_pos.Y <= ship_pos.Y + ship.Height
&&
rock_pos.X >= ship_pos.X && rock_pos.X <= ship_pos.X +
ship.Width)
{
explode = 20;
ship_pos.X = rock_pos.X - explosion.Width / 2;
rock_pos = Vector2.Zero;
}
if (rock_pos.Y == 0) rock pos = Vector2.Zero;
}
```

Для движения ракеты мы просто прибавляем на каждом цикле игры значение скорости в виде двумерного вектора. Столкновение определяется "вручную" по координатам - в случае поражения устанавливается переменная explode, означающая, что следующие несколько кадров вместо корабля будут отображать взрыв. Если же ракета достигает верхней границы экрана - ее движение прекращается (устанавливаются нулевые координаты).

Чтобы игра выглядела профессионально, недостает немногого - звукового оформления и подсчета числа пораженных кораблей. Для подсчета успехов игрока введем переменную score, которую будем увеличивать на 1 при каждом поражении корабля.

Весь текст выводится на экран в растровой форме, и для его отображения необходимо использовать специальный ресурс - шрифт. Первое, что надо сделать - поместить в проект какой-нибудь шрифт. Для описания шрифта используется специальный файл с расширением spritefont, который задает, помимо собственно названия и размера используемого шрифта, еще и другие параметры отображения, вроде кернинга и дополнительного межсимвольного расстояния. Это XML-файл. поэтому вы можете легко править параметры прямо в редакторе Visual Studio. Соответственно, можно либо использовать и включить в проект готовый spritefont-файл, либо добавить новый файл шрифта и затем поменять параметры по необходимости.

Заметим, что по умолчанию в XNA есть проблема с отображением символов кириллицы. Чтобы ее преодолеть, следует в spritefont-файле прописать явным образом, что мы намерены использовать интервалы кириллических символов. Для этого необходимо найти секцию <CharacterRegions> и добавить туда кириллический диапазон:

```
<CharacterRegions>
<CharacterRegion>
<Start> </Start>
<End>~</End>
</CharacterRegion>
<CharacterRegion>
<Start>А</Start>
<End>я</End>
</CharacterRegion>
</CharacterRegions>
```

После этого нам необходимо описать переменную font типа SpriteFont и загрузить шрифт в методе LoadContent (предполагаем, что файл шрифта в контентном проекте называйся GameFont.spritefont):

```
font = Content.Load<SpriteFont>("GameFont");
```

Для вывода на экран строки текста используем специапьную функцию Drawstring в методе Draw:

```
spriteBatch.Drawstring(font, String.Format("Score = {0},
score), new Vector2(10, 10), Color.LightGray);
```

Помимо собственно строки и названия шрифтового ресурса, задается цвет и координаты надписи на экране.

Практически ни одна игра не обходится без различных звуковых эффектов. Поэтому нам надо добавить звуковые эффекты, например, если корабль при потоплении будет издавать шум взрыва, а в процессе плавания ему будет сопутствовать шум мотора. Чтобы добавить в проект звуковые эффекты, нам потребуются всего лишь два звуковых файла типа WAV, которые мы добавим в проект: Laser.wav и EngineLoop.wav. Звуки в XNA представляются переменными двух типов: SoundEffectи SoundEffectinstance. Класс SoundEffect используется для хранения основных данных звукового файла (waveform), которые стандартно загружаются в память в методе LoadContent и далее могут воспроизводиться простым вызовом метода P1ay():

```
laser = Content.Load<SoundEffect>("Laser");
```

Переменные типа SoundEffectinstance представляют собой конкретный экземпляр SoundEffect, для которого можно настраивать различные параметры, запускать и останавливать воспроизведение и т. д. При этом SoundEffectinstance не хранит в себе сами данные звукового файла, а лишь ссылается на соответствующий SoundEffect, используя данные оттуда и хранимые в переменной параметры для воспроизведения.

В нашем примере используем звук лазера и слегка модифицируем его (изменим высоту звучания), чтобы получить звук потопляемого корабля:

```
expl = laser.CreateInstance();
expl.Pitch = 0.5f;
```

Для SoundEffectlnstance доступны и другие параметры, модифицирующие звук: это в первую очередь громкость (.volume), панорамирование (.Pan) и установка 3D-координат звукового источника. Аналогичным образом создадим звук работающего двигателя. Мы загружаем звуковой эффект, сразу же создаем для него экземпляр SoundEffectlnstance и устанавливаем флаг зацикленности isLooped - такой эффект будет звучать до тех пор, пока его в явном виде не остановят вызовом stop:

```
engine = Content.Load<SoundEffect>
("EngineLoop").CreateInstance();
engine.IsLooped = true;
engine.Play () ;
```

Обратите внимание, что, запуская воспроизведение какого-то звукового эффекта, мы можем не сильно заботиться о том, воспроизводится ли какой-то другой эффект в данный момент или нет. XNA поддерживает автоматическое смешивание звуковых каналов, с учетом их индивидуальной громкости и панорамирования, поэтому в простых случаях вам не придется об этом заботиться.

Для воспроизведения музыки также есть специальное API, позволяющее манипулировать традиционными форматами звуковых файлов, такими как MP3. Для воспроизведения фоновой музыки в процессе игры создадим переменную music типа Song и в методе Loadcontent загрузим мелодию в память и запустим ее воспроизведение - для этого используется API MediaPlayer:

```
music = Content.Load<Song>("Music");
MediaPlayer.IsVisualizationEnabled = false;
MediaPlayer.IsRepeating = true;
MediaPlayer.Play(music);
```

2 Практическая часть

2.1 Разработка мобильного приложения

На сегодняшний день туризм является неотъемлимой частью человеческого мира и данное приложение может помочь как туристам, так и просто людям которым интересно узнать с какой скоростью они едут и где находятся так же данное приложение показывает пройденную дистанцию имеет очень простой интерфейс понятный самому неопытному пользователю.

С каждым днем популярность операционной системы android растет. Но в Play Маркет мало приложений Казахстанского содержания и тем более нет такой программы, которая бы показывала нахождение человека, пока лишь широту и долготу где находится тот или иной пользователь. Так же данное приложение будет иметь продолжение объёденённое вместе веб интерфейсом системы для этого нужно переопределить некоторые функции, а также написать Клиент-серверное приложение, где данное приложение будет являтся клиентской частью после чего данные координаты будут отправляться на сервер где будут обрабатываться и показывать ваше месторасположение на GoogleMaps или YandexКарты. Вам и вашим близким или ревнивым супругам или беспокойным родителям.

После того, как мы установили среду разработки, самое время создать мобильное приложение для android. На его примере, мы рассмотрим весь процесс создания, компиляции и отладки приложения на эмуляторе.

Запустим Eclipse Java, далее выберем Файл - Создать - Новый проект. В левой части открывшегося диалогового окна создать новый проект в разделе Шаблоны выберите сначала android project . В центральной части окна создать новый проект вам будет доступно множество шаблонов проектов, но мы выберем самый шаблон под названием Приложение android. Назовем приложение TripLikeIdo, а в поле Расположение каталог, в котором будет размещаться проект приложения. После этого кликнем на кнопку ОК. Затем откроется диалоговое окно, предлагающее выбрать минимальную версию платформы android, для которой предназначается создаваемое приложение. Мы выберем версию android 2.3.6(sandwich), так как хотим, чтобы пользователи и более ранних и новых версий операционной системы могли пользоваться приложением. Здесь есть небольшая особенность, о которой разрабочику мобильных приложений будет полезно знать. Новые приложения, созданные для android 3+, не могут запускаться на android 2.3.6, однако приложения, написанные для android 2.3.6, работают на android 3+, будучи автоматически перекомпилированными "в облаке". Таким образом, наше приложение будет многоплатформенное, т.е. его можно будет запускать и на Windows Phone 8.x и на ранних версиях операционной системы.

Будет создан новый проект, и откроется редактор пользовательского интерфейса главной страницы приложения. Как вы уже знаете,

пользовательский интерфейс приложений для android описывается декларативным образом на языке XML. Eclipse Juno позволяет как редактировать XML-код напрямую, так и использовать графический редактор. В Eclipse Juno окно редактора пользовательского интерфейса приложений для android разделено вертикально на две части. Слева находится графический редактор, выполненный в виде телефона, а справа - редактор XML-кода. Между ними располагается панель, с помощью которой вы можете поменять редакторы местами, изменить размер каждого из них, переключиться в режим, когда редакторы размещаются один под другим, а не слева направо, а также скрыть один из редакторов и работать, например, лишь с XML-кодом или только с графическим представлением.

Несмотря на то, что можно создать полноценный пользовательский интерфейс, просто перетаскивая элементы управления из панели элементов в графический редактор и изменяя их свойства в окне Свойства, большинство разработчиков приложений для android предпочитает редактировать XML-разметку напрямую. В большей части примеров данной книги мы также будем напрямую редактировать XML-разметку.

Рассмотрим XML-разметку графической страницы приложения более подробно:

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutandroid:id="@+id/LinearLayout01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical">
<LinearLayout
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/GpsLayout">
</LinearLayout>
<LinearLayout
android:id="@+id/LinearLayout02"
android:layout_width="wrap_content"
android:layout_height="wrap_content">
<ListView
android:id="@+id/traceView"
android:layout_width="wrap_content"
android:layout_height="match_parent"
android:choiceMode="multipleChoice">
</ListView>
</LinearLayout>
</LinearLayout>
```

2.2 Реализация

Данная программа реализована на языке Java для платформы android 2.3.3+.

Тестовое оборудование Lenovo VibeX.

В данной программе были использованы разрешения в манифесте для отправки gps, так же работы сети wi-fi. Помимо всего этого в графическом редакторе main.xml использовалось <LinearLayout>.

Расположение View-элементов на экране зависит от ViewGroup (Layout), в которой они находятся.

LinearLayout - отображает View-элементы в виде одной строки (если он Horizontal) или одного столбца (если он Vertical). Я использовал это в данном дипломном проекте.

TableLayout - отображает элементы в виде таблицы, по строкам и столбцам.

RelativeLayout - для каждого элемента настраивается его положение относительно других элементов.

AbsoluteLayout - для каждого элемента указывается явная позиция на экране в системе координат (x,y).

Так же для передачи и хранения данных была использована база данных sqlite встроенная в android ОС.

SQLite - компактная встраиваемая реляционная база данных. Исходный код библиотеки передан в общественное достояние. В2005 году проект получил награду Google-O'Reilly Open Source Awards.

Часть когда где объявляеться перменные LocationListener и Location Manager для получения данных со спутника:

```
public void onCreate(Bundle savedInstanceState) {

        LocationManager_GPS = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        LocationListener_GPS = new LocationListener_GPS();

        if (LocationListener_GPS != null) {

LocationManager_GPS.requestLocationUpdates(LocationManager.GPS_PRO
VIDER, GPS_Time, 10, LocationListener_GPS);
        }
```

В данной программе реализована простота и минимализм для того чтобы не «грузить» пользователей не нужной информацией как видно на рисунке 29.

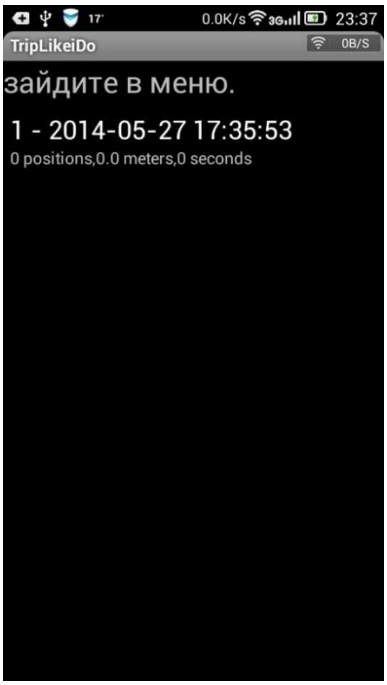

Рисунок 29 - Окно при открытии данного приложения

При нажатии кнопки «меню» увидим следующее меню как показано на рисунке 30.

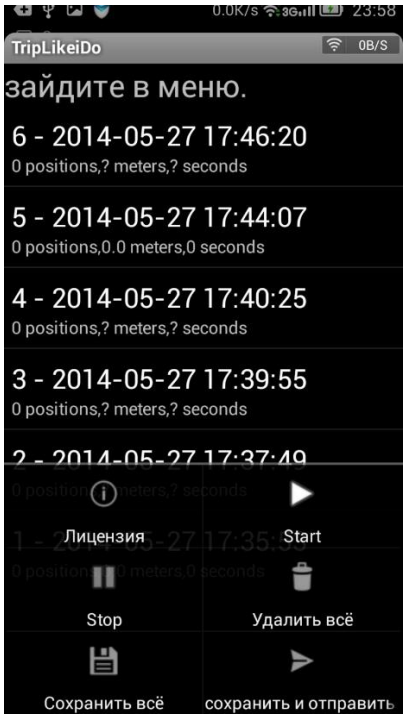

Рисунок 30 - Меню проекта

При нажатии на кнопку старт мы увидим либо данное сообщение как показано на рисунке 31 и рисунке 32.

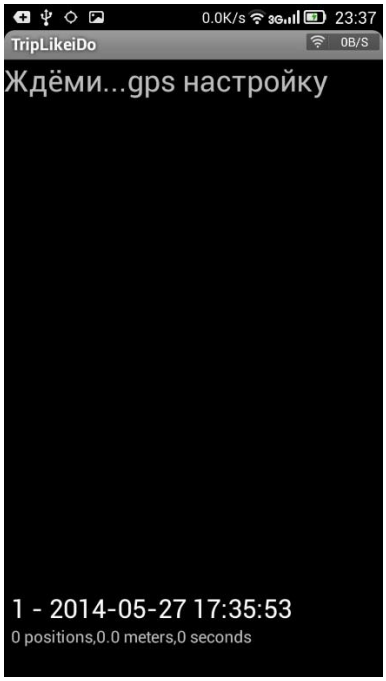

Рисунок 31 - Определение сигнала GPS

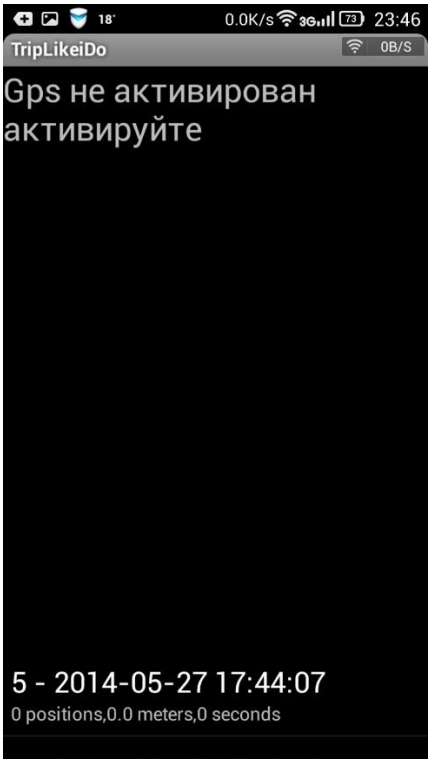

Рисунок 32 - При отсутствии сигнала GPS

Так же создал copyright как видно на рисунке 33.

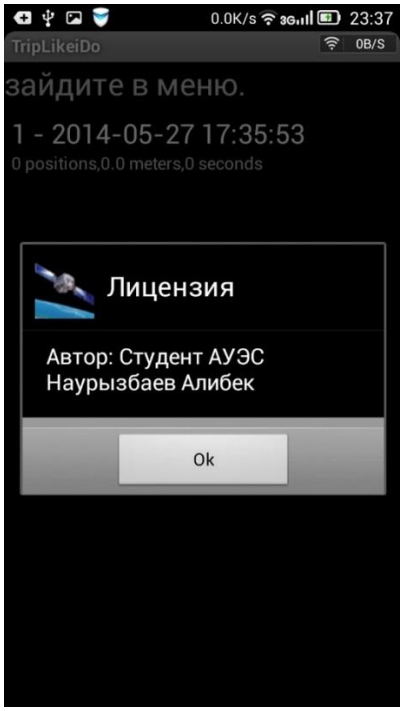

Рисунок 33 – Copyright

Как видно на данном рисунке 34 мы получаем уже данные с gps спутника.

Рисунок 34 - получение геолокационных данных

Так же полученные данные можно отправить по любой доступной программе которая существует в android как показано на рисунке 35.

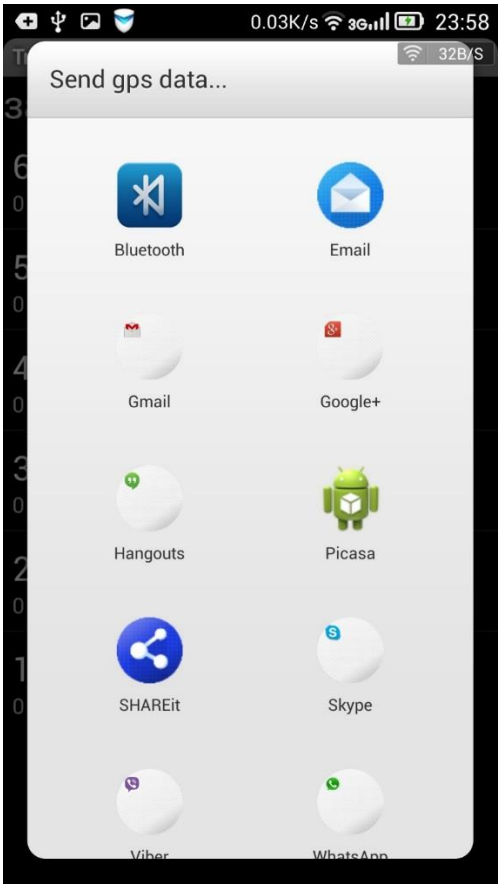

Рисунок 35 - Отправка данных

4 Технико-экономическое обоснование

4.1 Цель проекта

Целью данного дипломного проекта является разработка мобильных приложений для операционной системы Windows Phone и написание методичеких указаний для внедрения в учебный процесс АУЭС. Данные приложения реализованы на платформе .NET Framework в среде Visual Studio 2012 Ultimate на языке C#.

В данном разделе приводится рассмотрение экономической составляющей этого проекта, отражающей затраты на проект.

4.2 Трудовые ресурсы, используемые в работе

В работе задействованы:

– руководитель - постановка задачи, разработка схемы ПП;

– инженер-разработчик - разработка алгоритмов и программирование;

– консультанты по экономической части и по части БЖД.

Количество сотрудников, задействованных в разработке ПП представлено в таблице 3.

Таблица 3 - Данные о работниках, задействованных в проекте и их заработная плата

Наименование	Количество	Заработная плата в месяц
Разработчик	1	100 000
Руководитель проекта	1	90 000
Консультант по части «Экономика»	1	90 000
Консультант по части «БЖД»	1	90 000
Итого:	4	370 000

4.3 Оборудование, используемое в работе

Характеристики оборудования, используемого в работе, приведены в таблице 4.

Таблица 4 - Перечень оборудования при разработки ПП

Название оборудования	Характеристики	Ккол-во	Стоимость за единицу, с учетом НДС, тенге	Стоимость за единицу, без учета НДС, тенге
Ноутбук	ASUS K42J/ Intel Core i5 CPU M350 2,27 GHz/ 3072MB DDR3/250GB SATA/1024MB NVIDIA GeForce 310M/	11	156800	140000

Название оборудования	Характеристики	Ккол-во	Стоимость за единицу, с учетом НДС, тенге	Стоимость за единицу, без учета НДС, тенге
Итого:		1	156800	140 000

4.4 Программное обеспечение, используемое в работе

Программное обеспечение, используемое при разработке ПП, представлено в таблице 5.

Т а б л и ц а 5 - Перечень программного обеспечения, используемого при разработке ПП

Программное обеспечение	Стоимость, с учетом НДС, тенге	Стоимость, без учета НДС, тенге
MS Windows 8	44475	39 710
MS Visual Studio 2012 Ultimate	12 000	10 000
Итого:	56475	49 710

0

3.5 Сроки реализации проекта

Проектирование и разработка приложения состоит из пяти этапов и включает следующие виды работ:

1-й этап - Постановка задачи, сбор информации;
2-й этап - Разработка дизайна ПП;
3-й этап - Разработка главных форм ПП;
4-й этап - Проверка на работоспособность ПП;
5-й этап - Оформление отчетов.

График разработки и реализации проекта представлен в таблице 6.

Т а б л и ц а 6 - Этапы и сроки реализации проекта

Перечень работ		Неделя от начала работ								
		11	22	33	44	55	66	77	88	99
1 этап	Постановка задачи									
	Выбор среды программирования									
	Подбор и изучение литературы									
2 этап	Создание интерфейса									
3 этап	Разработка главных форм									
4 этап	Тестирование ПО									
	Отладка ПП									
5 этап	Подготовка раздела «Экономика»									
	Подготовка раздела «Безопасность жизнедеятельности»									
	Проверка и сдача отчёта									

3.6 Расчет стоимости произведенного проекта

Программирование мобильного приложения - сложный и трудоёмкий процесс, требующий наряду с интеллектуальными и техническими затратами, и финансовые затраты. Поэтому необходимо составить смету затрат на разработку данного программного продукта.

Смета затрат по разработке приложения состоит из расходов на материалы, затрат на оплату труда персонала, социальный налог, амортизационные отчисления на основные средства и прочие расходы.

Смета затрат определяется по формуле

$$C = ФОТ + C_н + A + Э + H \quad (1)$$

где ФОТ - фонд оплаты труда;
$С_н$- социальный налог;
А - амортизационные отчисления;
Э - затраты на электроэнергию;
Н - накладные расходы.

ФОТ состоит из основной и дополнительной заработной платы и рассчитывается по формуле

$$ФОТ = З_{ОСН} + З_{ДОП} \quad (2)$$

Для того, чтобы рассчитать затраты по основной зарплате, необходимы данные по среднедневному заработку и фактическому времени работы каждого участника, задействованного в проекте.

Заработная плата каждого работника за один рабочий день рассчитывается по формуле

$$D = \frac{З_П}{n} \quad (3)$$

где $З_П$ - заработная плата в месяц;
n - количество рабочих дней в месяце (21 день).

1) Руководитель

$$D = \frac{90000}{21} = 4286 \, тенге / день;$$

2) Консультант по части «Экономика»

$$D = \frac{90000}{21} = 4286\ тенге/день;$$

3) Консультант по части «Безопасность жизнедеятельности»

$$D = \frac{90000}{21} = 4286\ тенге/день;$$

4) Инженер-разработчик

$$D = \frac{100000}{21} = 4762\ тенге/день;$$

Заработная плата за один час рассчитывается по формуле

$$H = \frac{D}{Ч_P} \quad (4)$$

где D- заработная плата работника за один день;

$Ч_p$- количество часов рабочего дня (8-ми часовой рабочий день).

Заработная плата за один час:

1) руководитель

$$H = \frac{4286}{8} = 536\ тенге/час;$$

2) консультант по части «Экономика»

$$H = \frac{4286}{8} = 536\ тенге/час;$$

3) консультант по части «Безопасность жизнедеятельности»

$$H = \frac{4286}{8} = 536\ тенге/час;$$

4) инженер-разработчик

$$D = \frac{4762}{8} = 595\ тенге/час;$$

Длительность цикла в днях по каждому виду работ укрупнено рассчитывается по фомуле

$$t_n = \frac{T}{q_n \cdot z \cdot K} \quad (5)$$

где T - трудоемкость этапа, норма-час;
q_n - количество исполнителей по этапу;
z - продолжительность рабочего дня, $z = 8\,часов$;
K - коэффициент выполнения норм времени, $K = 1{,}1$.

Полученную величину t_n округляем в большую сторону до целых дней.

$t_1 = \frac{24}{1 \cdot 8 \cdot 1.1} \approx 3\,дн$; руководитель, постановка задачи;

$t_2 = \frac{16}{1 \cdot 8 \cdot 1.1} \approx 2\,дн$; руководитель, выбор среды программирования;

$t_3 = \frac{24}{1 \cdot 8 \cdot 1.1} \approx 3\,дн$; руководитель, подбор и изучение литературы;

$t_4 = \frac{32}{1 \cdot 8 \cdot 1.1} \approx 4\,дн$; Инженер-разработчик, создание интерфейса;

$t_5 = \frac{64}{1 \cdot 8 \cdot 1.1} \approx 8\,дн$; Инженер-разработчик, разработка форм приложения;

$t_6 = \frac{32}{1 \cdot 8 \cdot 1.1} \approx 4\,дн$; Инженер-разработчик, тестирование;

$t_7 = \frac{48}{1 \cdot 8 \cdot 1.1} \approx 6\,дн$; Инженер-разработчик, отладка;

$t_{8.1} = \frac{20}{1 \cdot 8 \cdot 1.1} \approx 3\,дн$; Консультант по экономической части, подготовка раздела «Экономика»;

$t_{8.2} = \frac{20}{1 \cdot 8 \cdot 1.1} \approx 3\,дн$; Инженер-разработчик, подготовка раздела «Экономика»;

$t_{9.1} = \frac{16}{1 \cdot 8 \cdot 1.1} \approx 2 дн;$ Консультант по части БЖД, подготовка раздела «БЖД»;

$t_{9.2} = \frac{16}{1 \cdot 8 \cdot 1.1} \approx 2 дн;$ Инженер-разработчик, подготовка раздела «БЖД»;

$t_{10.1} = \frac{16}{1 \cdot 8 \cdot 1.1} \approx 2 дн;$ Инженер-разработчик, проверка и сдача отчёта;

$t_{10.2} = \frac{32}{1 \cdot 8 \cdot 1.1} \approx 4 дн;$ руководитель проекта, проверка и сдача отчёта.

Сводные результаты расчета затрат на основную заработную плату работников, задействованных в разработке ПП представлены в таблице 7.

Т а б л и ц а 7 - Сводные результаты расчета затрат по основной заработной плате

Наименование этапов и содержание работ	Исполнитель	Трудоемкость		Длительность цикла, дни	Зар. плата за час работы, тенге	Сумма зар. платы, тенге
		Нормы-часы	% от общей трудоемкости			
1 Постановка задачи	Руководитель	24	6,7	3	536	12864
2 Выбор среды	Руководитель	16	4,4	2	536	8576
3 Подбор и изучение литературы	Руководитель	24	6,7	3	536	12864
4 Создание интерфейса	Инженер-разработчик	32	8,9	4	595	19040
5 Разработка форм	Инженер-разработчик	64	17,8	8	595	38080
6 Тестирование ПО	Инженер-разработчик	32	8,9	4	595	19040
7 Отладка ПП	Инженер-разработчик	48	13,3	6	595	28560
8 Подготовка раздела «Экономика»	Консультант по экономической части	20	5,6	3	536	10720
	Инженер-разработчик	20	5,6	3	595	11900
9 Подготовка раздела «Безопасность жизнедеятельности»	Консультант по экономической части	16	4,4	2	536	8576
	Инженер-разработчик	16	4,4	2	595	9520
10 Проверка и сдача отчёта	Инженер-разработчик	16	4,4	2	595	9520
	Руководитель	32	8,9	4	536	17152
ИТОГО:		360	100	46		206412

Дополнительная заработная плата составляет 10% от основной заработной платы и рассчитывается по формуле:

$$З_{ДОП} = З_{ОСН} \cdot 0{,}1 \quad (6)$$

$$З_{ДОП} = 206412 \cdot 0{,}1 = 20641{,}2 \; тенге$$

Таким образом, суммарный фонд оплаты труда согласно формуле 2 составит:

$$ФОТ = 206412 + 20641{,}2 = 227053{,}2 \; тенге$$

4.7 Расчет затрат по социальному налогу

Социальный налог составляет 11% от дохода работника, и рассчитывается по формуле:

$$С_{Н} = (ФОТ - ПО) \cdot 0{,}11 \quad (7)$$

где ПО - пенсионные отчисления, которые составляют 10% от ФОТ и рассчитывается по формуле:

$$ПО = ФОТ \cdot 0{,}1 \quad (8)$$

$$ПО = 227053{,}2 \cdot 0{,}1 = 22705{,}32 \; тенге$$

Таким образом, размер отчислений на социальные нужды составит:

$$С_{Н} = (227053{,}2 - 22705{,}32) \cdot 0{,}11 = 22478{,}27 \; тенге$$

4.8 Расчет амортизационных отчислений

Амортизационные отчисления на каждый актив, который задействован в проекте рассчитывается по формуле:

$$A = \frac{N_{АМ} \cdot С_{ПЕР} \cdot N}{100 \cdot 12 \cdot n} \quad (9)$$

где $N_{АМ}$ - норма амортизации;

$С_{ПЕР}$ - первоначальная стоимость оборудования;

N - количество дней на выполнение работы;

n - количество дней в рабочем месяце.

Таким образом, в соответствии с формулой 4.9 амортизационные отчисления по используемому оборудованию и программному обеспечению составят:

$$A_1 = \frac{40 \cdot 140000 \cdot 46}{100 \cdot 12 \cdot 21} = 10222 \, тенге$$

$$A_1 = \frac{40 \cdot 49710 \cdot 46}{100 \cdot 12 \cdot 21} = 3630 \, тенге$$

Суммарные затраты на амортизацию рассчитываются по формуле:

$$A = A_1 + A_2 \quad (10)$$

$$A = 10222 + 3630 = 13852 \, тенге$$

4.9 Расчет затрат на электроэнергию

Поскольку в процессе разработки программного продукта используется компьютер, то необходимо рассчитать затраты на электроэнергию.

Затраты на электроэнергию для производственных нужд включают в себя расходы электроэнергии на оборудование и дополнительные нужды.

$$Э = З_{ЭЛ.ЭН.ОБОР} + З_{ДОП.НУЖ} \quad (11)$$

где $З_{ЭЛ.ЭН.ОБОР}$ - затраты на электроэнергию оборудования;

$З_{ДОП.НУЖ}$ - затраты на электроэнергию на дополнительные нужды.

Расходы электроэнергии на оборудование рассчитывается по формуле:

$$З_{ЭЛ.ЭН.ОБОР} = W \cdot T \cdot S \cdot K_{ИСП} \quad (12)$$

где W - установленная мощность приборов, потребляющих электроэнергию, кВт;

S - стоимость киловатт-часа электроэнергии без НДС (11,84тнг/кВт*час);

$K_{ИМ}$ - коэффициент использования мощности (0,9-1);

T - время работы приборов, час.

$W = 0{,}09 \, Вт$ (мощность ноутбука).

$$T = 8 \cdot 45 = 360 \, ч$$

Сумма затрат на электроэнергию основного оборудования составляет:

$$З_{ЭЛ.ЭН.ОБОР} = 0{,}09 \cdot 360 \cdot 11{,}84 \cdot 0{,}9 = 345{,}25 \, тенге$$

Затраты на дополнительные нужды берутся по укрупненному показателю в размере 5% от затрат на оборудование и рассчитываются по формуле:

$$З_{ДОП.НУЖ} = 0{,}05 \cdot З_{ЭЛ.ЭН.ОБОР} \quad (13)$$

$$З_{ДОП.НУЖ} = 0{,}05 \cdot 345{,}25 = 17{,}26 \, тенге$$

Суммарные затраты на электроэнергию составляют:

$$Э = 345{,}25 + 17{,}26 = 362{,}51 \, тенге$$

4.10 Расчет накладных расходов

Накладные расходы составляют 25% от ФОТ и рассчитываются по формуле:

$$Н = ФОТ \cdot 0{,}25 \quad (14)$$

Накладные расходы согласно формуле 4.14 составляют:

$$Н = 227053{,}2 \cdot 0{,}25 = 56763{,}25 \, тенге$$

Таким образом, себестоимость разработки ПП согласно формуле 1 по всем перечисленным статьям составила:

$$С = 227053{,}2 + 22478{,}27 + 13852 + 362{,}51 + 56763{,}25 = 320509{,}23 \, тенге$$

Сводные результаты расчета стоимости разработки ПП и их структура представлены на рисунке 36 и в таблице 8.

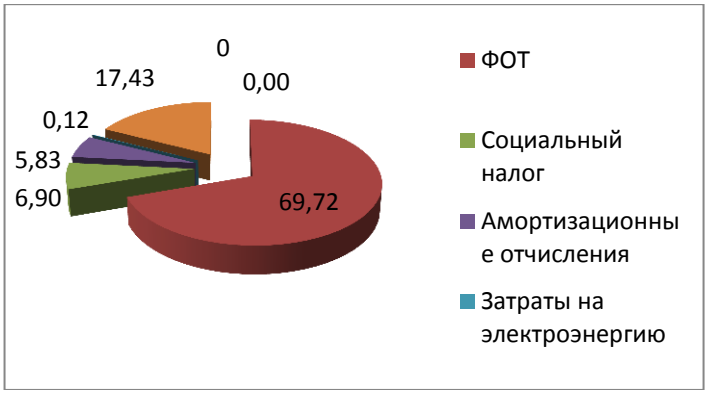

Рисунок 36 - Структура затрат по разработке мобильного приложения

Т а б л и ц а 8 - Стоимость разработки мобильного приложения

Наименование статей затрат	Сумма, тенге
ФОТ	227053,2
Социальный налог	22478,27
Амортизационные отчисления	13852
Затраты на электроэнергию	377,79
Накладные расходы	56763,25
Итого	**320524,51**

4.11 Цена реализации

Цена реализации программного продукта складывается из себестоимости и чистого дохода.

$$Ц_{П} = C + П \tag{15}$$

где C - себестоимость продукта;
$П$ - чистый доход.

При определении первоначальной цены задаемся желаемым уровнем рентабельности (*P=25%*) для отрасли информационных технологий.

$$Ц_{П} = C \cdot \left(1 + \frac{P}{100}\right) \tag{16}$$

$$Ц_{П} = 320509{,}23 \cdot \left(1 + \frac{25}{100}\right) = 400636{,}54 \ тенге$$

Цена реализации программного продукта рассчитывается по формуле:

$$Ц_{Р} = Ц_{П} + НДС \tag{17}$$

где НДС - налог на добавленную стоимость (12%)

Согласно Налоговому Кодексу Республики Казахстан НДС рассчитывается по формуле:

$$НДС = Ц_{П} \cdot 12\% \tag{18}$$

$$НДС = 400636{,}54 * 0{,}12 = 48076{,}38 \ тенге$$
$$Ц_{Р} = 400636{,}54 + 48076{,}38 = 448712{,}92 \ тенге$$

5 Безопасность жизнедеятельности

5.1 Анализ условий труда

Разработка мобильного приложения осуществляется с использованием компьютерной техники и электронного оборудования. В рассматриваемом помещении работают разработчик и администратор, которые имеют свое рабочее место.

При разработке программного продукта важную роль играет правильная организация условий труда в рабочем помещении. Так как данная работа связана с длительным нахождением за компьютером, необходимо учесть нормы естественного и искусственного освещения, кондиционирования для обеспечения благоприятных условий работы.

5.1.1 Рабочее помещение

План рабочего помещения представлен на рисунке 37.

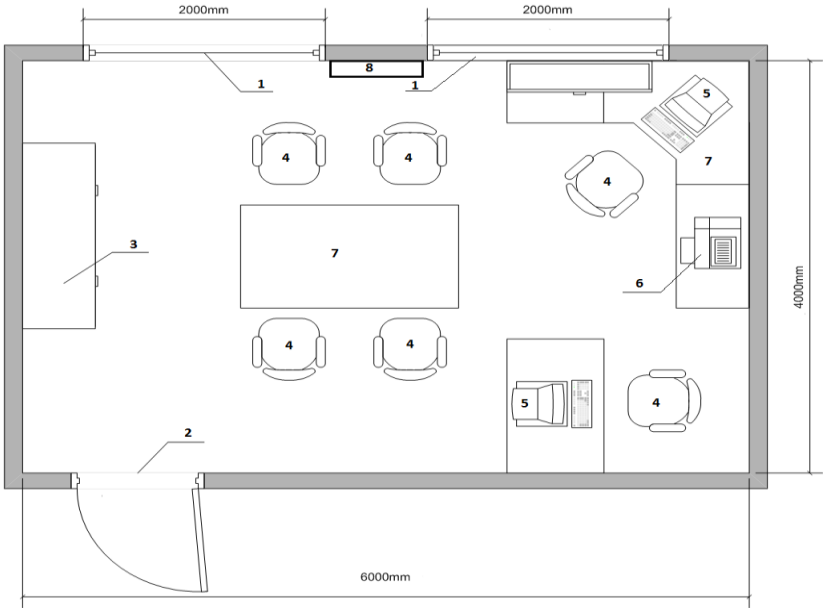

Рисунок - 38 рабочее помещение

где 1 - окна,
2 - дверь,
3 - шкаф,
4 - стулья,
5 - персональные
6 компьютеры,
7 - принтер,

8 - столы,

9 - кондиционер

Рисунок 39 - План помещения

Помещение имеет следующие параметры:

- размеры рабочей аудитории: высота - 3 м,
- ширина - 4 м,
- длина - 6 м.
- Общая площадь помещения составляет 24 м^2,
- площадь, занимаемая оборудованием и мебелью ≈ 15 м^2;
- помещение по зрительным условиям работы относится к категории легкихработ (легкая физическая, категория Iа, работа производится сидя и не требует физического напряжения) (ГОСТ 12.2.032-78);
- искусственное освещение - светильники: 2 светильника, в каждом по 2 люминесцентные лампы;
- остекление помещения - двойное (2 окна размером 2000х2000 мм) без стального переплетения.

При работе с ПЭВМ могут возникнуть потенциально опасные и вредные факторы, воздействие которых на организм человека может принести ему вред и привести к травматизму. Основные факторы с местом их возникновения и нормами изложены в ГОСТ 12.1.003-74/80 и сведены в таблице 8.

Т а б л и ц а 9 - Основные опасные и вредные факторы

N	Наименование фактора	Местопоявления	ПДУ, ПДК	Возможные последствия
1	Повышенное значение напряжения электрической цепи	Рабочее место разработчика	ГОСТ 12.1.038-82. В аварийном режиме Uпр≤36В переменного токапри длительном воздействии (более 1с).	Электротравма
2	Электрическая Дуга	Распределительный щит	ГОСТ 12.2.007.3-75 ГОСТ 12.2.007.4-75 ГОСТ 12.1.004-85	Ожоги, пожар
3	Повышенная напряженность электрического поля и электромагнитного излучения	Электроустановки: 220В., помещение	ГОСТ 12.1.002-84 ГОСТ 12.1.006-84 Время воздействия: 5 КВ/м-8 часов, 20..25 КВ/м-10 мин., ПДУ при частоте 60 КГц-3МГц, 50 В/м.	Профессиональные заболевания, электротравмы, пожары
4	Повышенная или пониженная температура воздуха, влажность, подвижность воздуха рабочей.	Рабочее место, помещение	ГОСТ 12.1.005-88, СН-4088-86, Т=20..24°С, влажность 40..60%, скорость воздуха менее 0.1 м/с СанПиН 2.2.4.548-96	Перегрев или переохлаждение организма

Продолжение таблицы 9

N	Наименование фактора	Местопоявления	ПДУ, ПДК	Возможные последствия
5	Недостаточная освещенность рабочей зоны	Помещение	СНиП 23-05-95, Е=300 Лк	Утомляемость, опасность травматизма, ухудшение зрения
6	Повышенный уровень шума	Рабочее место	ГОСТ 12.1.003-88 Уровень по полосам частот: менее 75 Дб.	Нервно-психическая перегрузка, заболевания органов слуха
7	Монотонность труда	Рабочее место	ГОСТ 12.1.003-80	Нервно-психическая перегрузка

Основное оборудование располагается в отделе разработки, который так же является центральным узлом управления всеми данными. Включая необходимые компьютеры, в этом помещении установлены источники бесперебойного питания (UPS) на случай отключения основного электропитания, климато-техническая установка для поддержания нужных температурных условий, специальный коммутационный шкаф и рабочие места для разработчика и администратора.

Для сохранения работоспособности и предупреждения развития заболеваний опорно-двигательного аппарата пользователей ПЭВМ необходимо организовать для них рабочие места, отвечающие требованиям ГОСТ.

Согласно ГОСТ 12.2.032-78 конструкция рабочего места и взаимное расположение всех его элементов должно соответствовать антропометрическим, физическим и психологическим требованиям. Большое значение имеет также характер работы. В частности, при организации рабочего места пользователя ПЭВМ должны быть соблюдены следующие основные условия:

- оптимальное размещение оборудования, входящего в состав рабочего места;
- достаточное рабочее пространство, позволяющее осуществлять все необходимые движения и перемещения;
- необходимо естественное и искусственное освещение для выполнения поставленных задач;
- уровень акустического шума не должен превышать допустимого значения:
- системы кондиционирования воздуха должны обеспечивать поддержание параметров микроклимата в соответствии с действующими нормами в течение всех сезонов года.
- применение специальных защитных экранных фильтров. Защитный фильтр должен быть плотно установлен на экране дисплея и надежно заземлен, ежедневно его следует очищать от пыли (так же как и экран дисплея).

- обязательный контроль уровней ЭМП не реже 1 раза в год, а также после проведения ремонтных работ, введения новых рабочих мест и др.

5.1.2 Характеристики используемого оборудования

При разработке и дальнейшей эксплуатации программного продукта используется следующее оборудование:

- Персональные компьютеры - 2 шт;
- Люминесцентные лампы T8\58W, 2 светильника по 2 лампы E=300лк;
- Источник бесперебойного питания: переменное напряжение 220-250 В, частотой 50 Гц. Мощность 400 Вт;
- Кондиционер SAMSUNG Monte AQ-09AWA.

5.1.3 Микроклиматические условия

В таблице 10 приведены оптимальные нормы параметров микроклимата с учетом периода года согласно ГОСТ 12.0.003-88. ССБТ для легкой физической работы. Оборудование, установленное в рабочем помещении, не является источником выделения тепла (очень незначительное выделение тепла аппаратурой никаким образом не оказывает влияние на микроклимат рабочего помещения).

Т а б л и ц а 10 - Оптимальные нормы микроклимата для помещений с ПК

Период года	Категория работ	Температура воздуха °С не более	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Холодный	Легкая - 1а	22-24	40-60	0,1
Теплый	Легкая - 1а	23-25	40-60	0,1

Здание относится к I степени огнестойкости (СНиП РК 2.02-05-2002). Рабочее помещение по вопросам пожарной безопасности относится к классу “Д”. В соответствии с типовыми правилами пожарной безопасности административные здания и отдельные помещения, и технологические установки обеспечиваются первичными средствами пожаротушения согласно нормативам.

5.2 Расчетная часть

Расчет искусственного освещения методом коэффициента использования отдела разработки.

Разряд зрительной работы - V. Нормируемая освёщенность по таблице 11 - 400 лк.

Т а б л и ц а 11 - Технические характеристики люминесцентных ламп Т8

Номинальная мощность, Вт	Номинальный световой поток ламп типа Т8	Размер ламп, мм	
58	4000	Диаметр	Длина по штырькам
		25,78	1500

В качестве светильника возьмем LZG 258E 3906a. Длина светильника 1570 мм, ширина 140 мм.

Расчёт искусственного освещения производим методом коэффициента использования.

Коэффициенты отражения от потолка стен и пола соответственно равны:

$\rho_{ПОТ} = 70\%;$

$\rho_{СТ} = 50\%;$

$\rho_{ПОЛ} = 30\%.$

Вычислим высоту подвеса светильника над рабочей поверхностью

$$H = h - h_P - h_c \quad (19)$$

где h_c- расстояние от светильника до перекрытия, h_c =0,11 м;

h_p- высота рабочей поверхности над полом, h_p=0,7 м.

h-высота помещения, h=3 м.

$$H = 3 - 0{,}7 - 0{,}11 = 2{,}19$$

Наивыгоднейшее расстояние от окна до светильника определяется как

$$L = \lambda \cdot H \quad (20)$$

где $\lambda = 1{,}2 \div 1{,}4$

$$L = 1{,}2 \cdot 2{,}19 = 2{,}628\ м$$

Расстояние от стены до ближайшего светильника, когда работа у стены не проводится, определяем по формуле

$$l_1 = (0{,}4 \div 0{,}5) \cdot L \quad (21)$$

$$l_1 = 0{,}4 \cdot 2{,}628 = 1{,}05$$

Определим индекс помещения

$$i = \frac{l \cdot s}{H(l+s)} \qquad (22)$$

$$i = \frac{6 \cdot 4}{2{,}19 \cdot (6+4)} = 1{,}1$$

Коэффициент использования в данном случае равен $\eta = 49$, коэффициент запаса равен $k_з = 1{,}2$.

Определим количество люминесцентных ламп по формуле

$$N = \frac{E_н \cdot S \cdot Z \cdot K_з}{n \cdot Ф_л \cdot \eta} \qquad (23)$$

где S- площадь помещения;
$k_з$- коэффициент запаса;
E-заданная минимальная освещенность, E=400 лк;
Z-коэффициент неравномерности освещения, Z=1,1;
n-количество ламп в светильнике;
$Ф_л$- световой поток выбранной лампы, $Ф_л$=4000 лм;
η - коэффициент использования, $\eta = 49$.

$$N = \frac{400 \cdot 24 \cdot 1{,}1 \cdot 1{,}2}{2 \cdot 4000 \cdot 0{,}49} \approx 4$$

Всего для создания нормируемой освещенности 400 лк необходимо 4 светильника по 2 люминесцентные лампы в каждом, мощность каждой лампы должна быть 58 Вт, значит, требуется увеличить количество имеющихся ламп до четырех, для соответствия санитарным нормам.

5.2.1 Расчет естественного освещения

Помещение имеет размеры: длина - a=6 м, ширина - b=4 м, высота -h=3 м. Высота рабочей поверхности над уровнем пола - 0,7 м, окно начинается с высоты 0,8 м, высота окна 2 м. Рабочее помещение находится в IV часовом поясе - город Алматы. Со всех сторон затеняющих зданий нет.

Нормированное значение коэффициента естественного освещения (КЕО) для работ средней точности V подразряда равна 1,2.

Общую требуемую площадь окон S_0, определяется по формуле:

$$S_0 = \frac{S_п \cdot e_п \cdot \eta_0 \cdot k_{зд} \cdot k_з}{100 \cdot \tau_0 \cdot r_1} \qquad (24)$$

Отсюда выразим нормированное значение КЕО - e_n:

$$e_n = \frac{S_0 \cdot 100 \cdot \tau \cdot r_1}{S_n \cdot \eta_0 \cdot k_{зд} \cdot k_з} \quad (25)$$

где S_n- площадь помещения, м^2;
S_0- общая площадь окон;
$k_з$ - коэффициент запаса;
$k_з = 1{,}2$ (учебные помещения, лаборатории, конструкторское бюро).
$S_n = a \cdot b = 6 \cdot 4 = 24\ м^2$
$S_0 = 2 \cdot (2 \cdot 2) = 8\ м^2$

Общий коэффициент светопропускания равен

$$\tau_0 = \tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4 \quad (26)$$

В качестве светопропускающего материала используем:

- стекло оконное листовое, двойное: $\tau_1 = 0{,}8$;
- вид переплёта - двойной раздельный: $\tau_2 = 0{,}6$;
- вид несущей конструкции - железобетонные фермы: $\tau_3 = 0{,}8$;

а. солнцезащитные устройства - убирающиеся регулируемые жалюзи $\tau_4 = 1$.

Общий коэффициент светопропускания:

$$\tau_0 = 0{,}8 \cdot 0{,}6 \cdot 0{,}8 \cdot 1 = 0{,}384$$

η_0 - световая характеристика окон, определяется из соотношения длины к ширине помещения и ширины к высоте от уровня рабочей поверхности до верха окна.

r_1 - коэффициент, учитывающий повышение КЕО при боковом освещении благодаря свету, отражённому от поверхностей помещения и подстилающего слоя, прилегающего к зданию.

Рассчитаем η_0:

Отношение длины помещения к его глубине

$$\frac{L}{B/2} = \frac{6}{2} = 3$$

$$h_1 = h_{ок} + h_{н.ок.} - h_{пов.} \quad (27)$$

где h_1 - высота от уровня условной рабочей поверхности до верха окна;
$h_{пов.}$ - уровень условной рабочей поверхности, $h_{пов.} = 0{,}7\ м$

$h_1 = 2 + 0{,}8 - 0{,}7 = 2{,}1\,м$

Отношение глубины помещения к его высоте от уровня условной рабочей поверхности до верха окна:

$$\frac{B}{h_1} = \frac{4}{2{,}1} = 1.9$$

Учитывая эти соотношения: $\eta_0 = 9{,}6$

Определим r_1:

Отношение длины помещения к его глубине:

$$\frac{L}{B} = \frac{6}{4} = 1{,}5$$

Отношение расстояния расчетной точки от наружной стены к глубине помещения:

$$\frac{H}{B} = \frac{3}{4} = 0{,}75.$$

Средний коэффициент отражения в помещении $\rho_{CP} = 0{,}3$, принимаем одностороннее боковое освещение, тогда по Таблице $r_1 = 1{,}05$.

$k_{зд}$- коэффициент, учитывающий затенение окон противостоящими зданиями. Поскольку затеняющих зданий поблизости нет, то $k_{зд}$=1.

Подставим все полученные данные в формулу для расчета нормированного значения КЕО:

$$e_n = \frac{8 \cdot 100 \cdot 0{,}384 \cdot 1{,}05}{24 \cdot 9{,}6 \cdot 1 \cdot 1{,}2} = 1{,}17$$

Так как, полученное значение КЕО $e_n = 1{,}17$, приблизительно равно нормированному значению ($e_n = 1{,}2$), значит, естественное освещение соответствует нормативам и дополнительных расчетов производить не нужно.

5.2.2 Расчет системы кондиционирования

В Таблице 12 приведены оптимальные нормы параметров микроклимата с учетом периода года согласно СНиП 2.04.05-91. для легкой физической работы. Оборудование, установленное в рабочем помещении, не является источником выделения тепла (очень незначительное выделение тепла аппаратурой никаким образом не оказывает влияние на микроклимат рабочего

помещения). Климатические условия эксплуатации оборудования полностью совпадают с климатическими условиями, нормируемыми для рабочего персонала.

Для вентиляции офисного помещения используются каналы естественной вентиляции, прокладываемые при строительстве здания и открытые окна летом. В теплый период года для поддержания оптимального микроклимата используется кондиционер. Нормальный микроклимат в офисе обеспечивает хорошее самочувствие сотрудников в любое время года, и соответственно продуктивность работы увеличивается. Таким образом, для поддержания условий микроклимата в помещении, целесообразно оборудовать его системой кондиционирования.

Ниже представлен расчет системы кондиционирования в рабочем помещении. Кондиционирование обеспечит соответствие климата в рабочем помещении нормативам.

Количество приточного воздуха $L_{ПР}, \frac{м^3}{ч}$ определяем по формуле

$$L_{пр} = \frac{Q_{изб}}{c \cdot p_{пр} \cdot (t_{выт} - t_{пр})} \qquad (28)$$

где $Q_{изб}$ - избыточное выделение явной теплоты, $\frac{кДж}{ч}$;

с- удельная теплоемкость воздуха при постоянном давлении, равная

$$c=1 \ \frac{кДж}{кг \cdot {}^0C};$$

$p_{пр}$ - плотность поступающего в помещение воздуха, равная $1{,}2 \frac{кг}{м^3}$;

$t_{выт}$ - температура удаляемого из помещения воздуха за пределы рабочей или обслуживаемой зоны, ${}^0C$;

$t_{пр}$ - температура приточного воздуха, ${}^0C$.

Температура удаляемого из помещения воздуха $t_{выт}, {}^0C$, определяется по формуле:

$$t_{выт} = t_{РЗ} + \Delta t \cdot (h_{вп} - z) \qquad (29)$$

где: $t_{РЗ}$ - температура в рабочей зоне, которая не должна превышать допустимую по нормам $(t_{РЗ} \le t_{ДОП})$, ${}^0C$;

$h_{вп}$ - расстояние от пола до центра вытяжных проемов (кондиционера),м;

Н - высота рабочей зоны, $м$.

Поскольку расчет производится для теплого периода года, то примем $t_{рз} = 22\ ^0C$

Внутренняя часть кондиционера расположена на высоте $h_{вп} = 2{,}5$м:

$$t_{выт} = 22 + 1{,}2 \cdot (2{,}5 - 3)\ ^0C$$

Температура приточного воздуха $t_{пр}$ при наличии избытка явной теплоты должна быть на $5 - 7\ ^0C$ ниже температуры воздуха в рабочей зоне:

$$t_{пр} = 22 - 7 = 15\ ^0C$$

Величину избыточного выделения явной теплоты $O_{ИЗБ}$ находят на основании баланса теплоты в помещении по формуле

$$O_{изб} = \sum Q - \sum Q_{ух} \qquad (30)$$

где: $\sum Q$ - суммарное количество поступающей в помещение явной теплоты;

$\sum Q_{ух}$ - суммарное количество уходящей из помещения теплоты (за счет теплопотерь ограждениями, нагрева поступающего в помещение воздуха и т. п.).

Основными источниками избыточного тепла являются светильники, люди и др. Кроме того, необходимо учитывать теплопоступления от солнечной радиации. В данном помещении тепловыделением электронного оборудования можно пренебречь. Поэтому учитываем тепловыделения от искусственного освещения, от людей, количество тепла, поступающего в помещение через окна от солнечной радиации.

Тепловыделения от искусственного освещения Q_2, рассчитывают, предполагая, что практически вся затрачиваемая энергия, в конечном счете, преобразуется в тепло, по формуле

$$Q_2 = 1000 \cdot N \qquad (31)$$

где N - расходуемая мощность светильников, кВт.

$Q_2 = 1000 \cdot 0{,}28 \cdot 4 = 1120$ кВт.

Тепловыделения от людей Q_3 определяют по формуле:

$$Q_3 = n \cdot q_ч \qquad (32)$$

где: n- число работающих;

$q_ч$ - количество тепла, выделяемое одним человеком, представлено в Таблице 5.3.

Таблица 12 - Количество тепла, выделяемое одним человеком в зависимости от категории работ и температуры окружающей среды

Категория работ	Тепло, Вт			
	Полное		Явное	
	при 100^0 С	При 350^0 С	при 100^0 С	При 350^0 С
Легкая	180^0 С	145^0 С	150^0 С	5^0 С

$$Q_з = 1*145 = 145 \text{ Вт}.$$

Количество тепла, поступающего в помещение от солнечной радиации $Q_{ост.рад}$, определяют по формуле

$$Q_{ост.рад} = F_{ост} \cdot q_{ост} \cdot A_{ост} \quad (33)$$

для покрытий:

$$Q_{п.рад} = F_п \cdot q_п \cdot k_п \quad (34)$$

где $F_{ост}$ и $F_п$ - площадь поверхности и покрытия, $м^2$;

$q_{ост}$ и $q_п$ - теплопоступления через $1 м^2$ поверхности остекления и поверхности покрытия, при коэффициенте теплопередачи, равном $1 \frac{Вт}{м^2 \cdot {}^0С}$;

$A_{ост}$ - коэффициент остекления;

$k_п$ - коэффициент теплопередачи покрытия, $1 \frac{Вт}{м^2 \cdot {}^0С}$.

Значение $q_{ост}$ в зависимости от географической ориентации поверхности и характеристики окон или фонарей принимается в пределах $70-210$, а коэффициента $A_{ост}$ в зависимости от вида остекления и его солнцезащитных свойств - в пределах $0,25-1,25$, средние значения теплопоступления от солнечной радиации через покрытие в зависимости от географической широты и вида покрытия принимают в пределах 6 - 24.

$$F_{ост} = 1,5 \cdot 1,2 \cdot 2 = 3,6 \text{ м}^2$$

Окно рабочего помещения направлено на север, поэтому примем значение $q_{ост}$ равным $140 \frac{Вт}{м^2 \cdot {}^0С}$. Примем $A_{ост} = 0,35$

$$Q_{ост.рад} = 3,6 \cdot 140 \cdot 0,35 = 176,4 \text{ Вт.}$$

Среднее значение теплопоступления для покрытия с учетом географической широты примем равным $Q_{п.рад} = 18 \text{ Вт.}$

Потери тепла из помещения $Q_{ух}$, кВт через стены двери, окна оценивают ориентировочно по формуле

$$Q_{ух} = \frac{\lambda \cdot S \cdot (t_{выт} - t_{пр})}{\delta} \quad (35)$$

где: λ - теплопроводность стен, $\frac{\text{Вт}}{\text{м} \cdot {}^0\text{C}}$;

S - площадь, м^2;

δ - толщина стен, м.

Стены рабочего помещения изготовлены из тяжелого бетона М600, теплопроводность которого равна $12 \frac{\text{Вт}}{\text{м} \cdot {}^0\text{C}}$. Толщина стен $\delta = 0,5 \text{ м}$.

$$Q_{ух} = \frac{1,2 \cdot 24 \cdot (21,4 - 15)}{0,5} = 368,64 \text{ Вт.}$$

Вычислим суммарное количество поступающей в помещение явной теплоты

$$\sum Q = Q_2 + Q_3 + Q_{ост.рад} + Q_{п.р.р} \quad (36)$$

$$\sum Q = 1120145 + 176,4 + 18 = 1120,3 \text{ кВт.}$$

Так как расчет производится для летнего периода величина избыточного выделения явной теплоты равна

$$O_{изб} = 1120,3 \text{ кВт.}$$

Вычислим количество приточного воздуха

$$L_{пр} = \frac{1120,3}{1 \cdot 1,2 \cdot (21,4 - 15)} = 145,9 \frac{\text{м}^3}{\text{ч}}$$

Чтобы обеспечивать расход воздуха L=145,9 м^3/ч, можно использовать 1 кондиционер фирмы SAMSUNGMonteAQ-09AWA с максимальным расходом воздуха 195 м^3/ч.

Краткие характеристики модели SAMSUNG Monte AQ-09AWA.

Тип: крышный моноблок промышленного кондиционера.

Номинальная холодопроизводительность Вт 23200 .

Номинальная потребляемая мощность Вт 9400.

Расход воздуха (min-max) м3/ч 100-195.

Что соответствует действительности и является достаточным для обеспечения комфортного микроклимата.

Заключение

Итак, наше первое Android-приложение готово. В процессе его создания мы узнали некоторые важные вещи об Android - и о разработке мобильных приложений в целом. мы установили Android SDK, загрузили Android версии 4.2 и создали AVD (или эмулятор) для имитации работы приложения на устройстве. Если мы используем Eclipse, то интерфейс IDE теперь настроен на создание и развертывание Android-приложений для выбранного AVD. Вся эта настройка требуется для того чтобы можно было работать с Android.

Вы также видели, как собирается простое Android-приложение: большую часть работы выполняют классы Activity, а AndroidManifest.xml определяет его структуру для базового устройства. И получили результат: приложение, отслеживающее местонахождение пользователя через GPS спутник, которое нам предстоит усовершенствовать.

В следующий раз мы продолжим заниматься Android. Пока же я буду эксперементировать с этим приложением. Добавлять какие-нибудь виджеты или изменения значения параметров в XML-файлах.

Список используемой литературы:

1 Библиотека профессионала Java 2 Том 1. Кей С.Хорстманн. Гарри Корнелл. Москва 2003;

2 Библиотека профессионала Java 2 Том 2. Кей С.Хорстманн. Гарри Корнелл. Москва 2003;

3 Применение шаблонов Java. Библиотека профессионалов. Стивен Стелтинг. Олав Маассен.Изд.дом Вильямс 2002;

4 Разработка web-служб средствами Java. Ильдар Хабибуллин. БХБ-Петербург 2003;

5 Работа в сети Интернет. Методическое пособие. Ютландова В.Ю.2000 г;

6 Изучение HTML 3.2 на примерах Jukka Korpela. Перевод "АКДИ Экономика и жизнь";

Заключение

Итак, наше первое Android-приложение готово. В процессе его создания мы узнали некоторые важные вещи об Android - и о разработке мобильных приложений в целом. мы установили Android SDK, загрузили Android версии 4.2 и создали AVD (или эмулятор) для имитации работы приложения на устройстве. Если мы используем Eclipse, то интерфейс IDE теперь настроен на создание и развертывание Android-приложений для выбранного AVD. Вся эта настройка требуется для того чтобы можно было работать с Android.

Вы также видели, как собирается простое Android-приложение: большую часть работы выполняют классы Activity, а AndroidManifest.xml определяет его структуру для базового устройства. И получили результат: приложение, отслеживающее местонахождение пользователя через GPS спутник, которое нам предстоит усовершенствовать.

В следующий раз мы продолжим заниматься Android. Пока же я буду эксперементировать с этим приложением. Добавлять какие-нибудь виджеты или изменения значения параметров в XML-файлах.

Список используемой литературы:

7 Библиотека профессионала Java 2 Том 1. Кей С.Хорстманн. Гарри Корнелл. Москва 2003.

8 Библиотека профессионала Java 2 Том 2. Кей С.Хорстманн. Гарри Корнелл. Москва 2003.

9 Применение шаблонов Java. Библиотека профессионалов. Стивен Стелтинг. Олав Маассен.Изд.дом Вильямс 2002.

10 Разработка web-служб средствами Java. Ильдар Хабибуллин. БХБ-Петербург 2003.

11 Работа в сети Интернет. Методическое пособие. Ютландова В.Ю.2000 г..

12 Изучение HTML 3.2 на примерах Jukka Korpela. Перевод "АКДИ Экономика и жизнь".