Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«Казанский (Приволжский) Федеральный Университет»

Институт математики и механики им. Н.И. Лобачевского

Специальность: 050202.65 Информатика и английский язык

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«Мобильное приложение-головоломка Algorithmic Race для наглядности работы алгоритмов»

Работа завершена:

«__» _____ 20___ г. _____ (Цыганов. И.С)

(Студент 5 курса очного отделения

бюджетная форма обучения гр.05-009)

Работа допущена к защите:

д.т.н, профессор кафедры ТТПМИ

ИММ им. Н.И. Лобачевского

«___»_____ 20___ г. _____(Ахмадиев Ф.Г.)

Дата защиты:

« ____ » _____20___ г. Оценка _____

_____

Заведующий кафедрой

«____» _____ 20___ г. _____(Шакирова Л.Р.)

Казань 2015

Содержание

ВВЕДЕНИЕ

Мобильные устройства уже окончательно вошли в нашу жизнь. Наличие такого устройства делает нашу жизнь удобнее.

Они порой заменяют компьютер или ноутбук. Мобильные устройства оснащены разными операционными системами. Лидирующие позиции на сегодняшний день занимают Android, IOS. Androidc долей рынка в 81%, а iOS составляет 13%.

Нельзя недооценивать рынок мобильных устройств. Россия считается одной из быстроразвивающихся в этом плане стран. В 2014 году продажи смартфонов выросли на 61,3% до 5,58 миллиона устройств. Причина высокого темпа роста — увеличение ассортимента бюджетных смартфонов B-брендов. Выручка от продажи смартфонов в первом квартале 2014 года в сравнении с аналогичным периодом 2013 года выросла на 28% до 48,56 миллиарда рублей.[1]

Целью дипломной работы является разработка мобильного приложения для визуализации алгоритмов, а именно линейных, разветвляющихся и циклических.

Приложение разработано для операционной системы Androidи при небольших манипуляциях может быть перенесено на мобильную платформу IOS.

Решаемые задачи: изучение и улучшение знаний в разработке приложений для мобильных устройств, разработка приложения визуализации описания алгоритмов для школьников.

Так же в дополнение к приложению разработан сайт-справка для приложения. Для сайта использован язык разметки HTML, каскадные таблицы стилей CCSи фреймворк bootstrap.

Разработка мобильного приложения и его интерфейса разделена на несколько этапов:

1) Общая идея.

Были изучены средства разработки мобильных приложений для популярных мобильных операционных систем. Выбраны реализуемые алгоритмы и сформулирована задача программы.

2) Выбор средств разработки;

Выбор был сделан в пользу coronaSDK.

Плюсы CoronaSDK:

А)Симулятор позволяет видеть сделанные изменения моментально.

Б)Для работы достаточно скачать симулятор и использовать любой подходящий редактор кода поддерживающий язык программирования Lua редактор кода.

В)Качественная документация от разработчиков.

Минусы CoronaSDK:

А)Полная версия движка стоит от $79-$99/год.

В)Нет документации на русском языке.

3) Разработка алгоритма приложения.

На данном этапе был составлен алгоритм и написан код на языке программирования Lua.

4) Разработка графического интерфейса приложения.

При разработки графического интерфейса учитывались устоявшиеся принципы эргономичности.

5) Тестирование и отладка мобильного приложения.

Приложение протестировано на нескольких устройствах с различными расширениями дисплеев. Неполадок в работе приложения не обнаружено.

6) Создание веб-сайта.

Разработан сайт для презентации мобильного приложения.

Работа изложена на 80 страницах где 57 занимает основной текст и 23 приложение к диплому. Состоит из 6 глав. К диплому прилагается

приложение в виде компакт диска. На нем записаны исходники приложения и сайта.

1.Алгоритм и его исполнители

1.1Алгоритм и алгоритмизация

Понятие алгоритма одно из основных в информатике. Наука, изучающий алгоритмы связанна с математикой, информатикой и математической логикой называется теорией алгоритмов. Процессор компьютера способен выполнять конечное число простых операций. Поэтому более сложные операции программисты представляют в виде строгой последовательности простых операций. Для того, чтобы процессор мог выполнять сложные задачи, программисты весь процесс решения этой задачи расписывают в виде пошаговой инструкции в определенной последовательности.

Пошаговое описание процесса решения поставленной задачи называется алгоритмизацией, аконечное число правил, приводящих к решению задачи, записанных в строго определенной последовательности – алгоритмом.

Термин “алгоритм” впервые встречается в переводе на латинский язык трудов арабского математика IX века Аль-Хорезми “Арифметическое искусство индусов” (“Algoritmi de numero Indorum”).

Любой алгоритм должен удовлетворять следующим требованиям:

1. Возможность ввода первоначальных данных..
2. Наличие возможности вывода полученных данных.
3. Однозначность – выдача однозначных ответов.
4. Общность – алгоритм должен решать определенный класс задач.
5. Корректность – алгоритм должен вернуть правильное решение задачи.
6. Ограниченность – результат должен быть получен через определенное количество ограниченных шагов.

7. Эффективность – при решении задачи должно использоваться возможно малое количество ресурсов компьютера (процессорное время, объем оперативной памяти, и т.д.). [2]

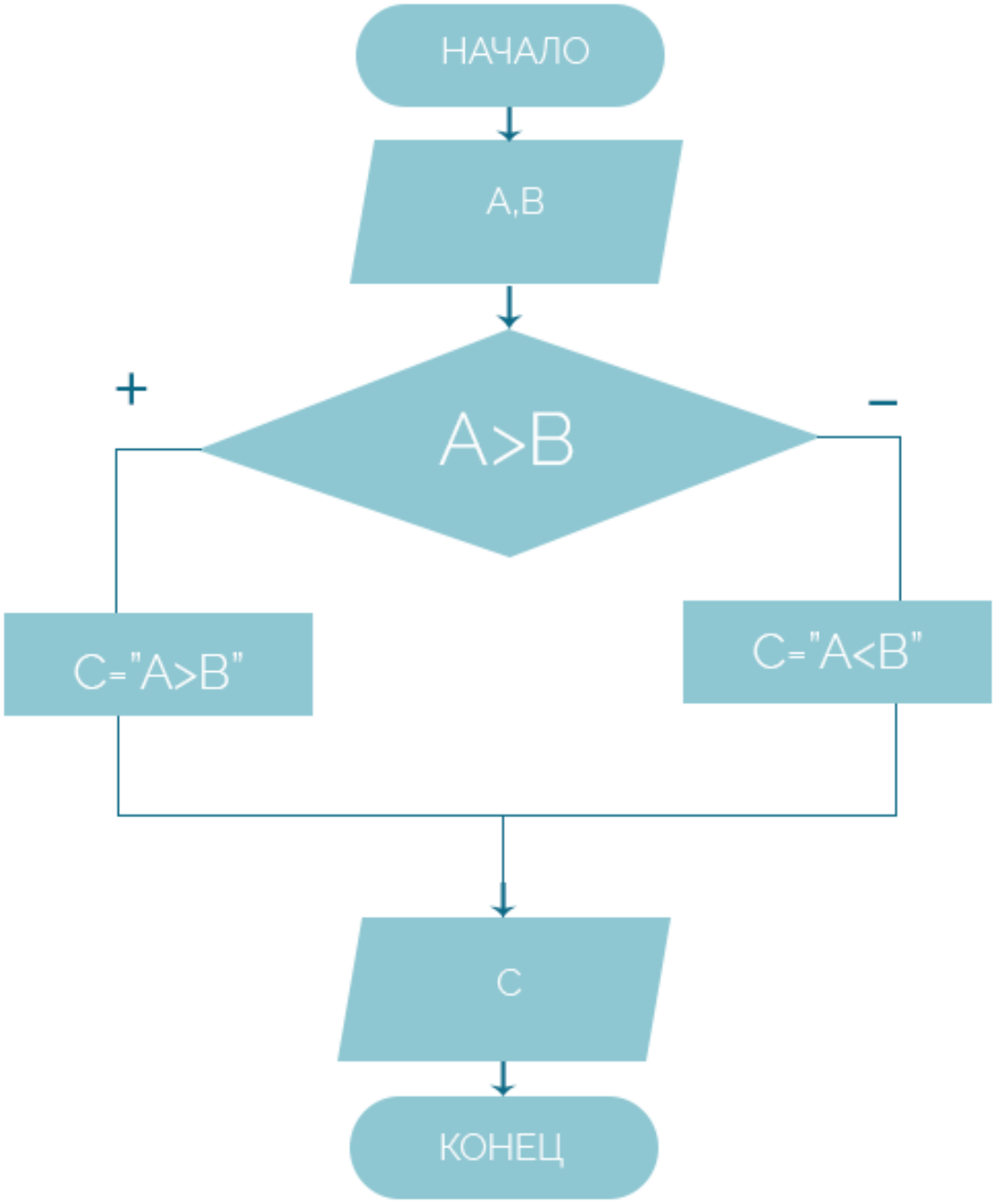
1.1 Классификация алгоритмов

Алгоритмы имеют 3 базовых структуры.

Линейная. Эта структура также называется простой. В этом алгоритме все операции выполняются последовательно одна за другой. Пример линейного алгоритма приведен ниже в виде блок-схемы:

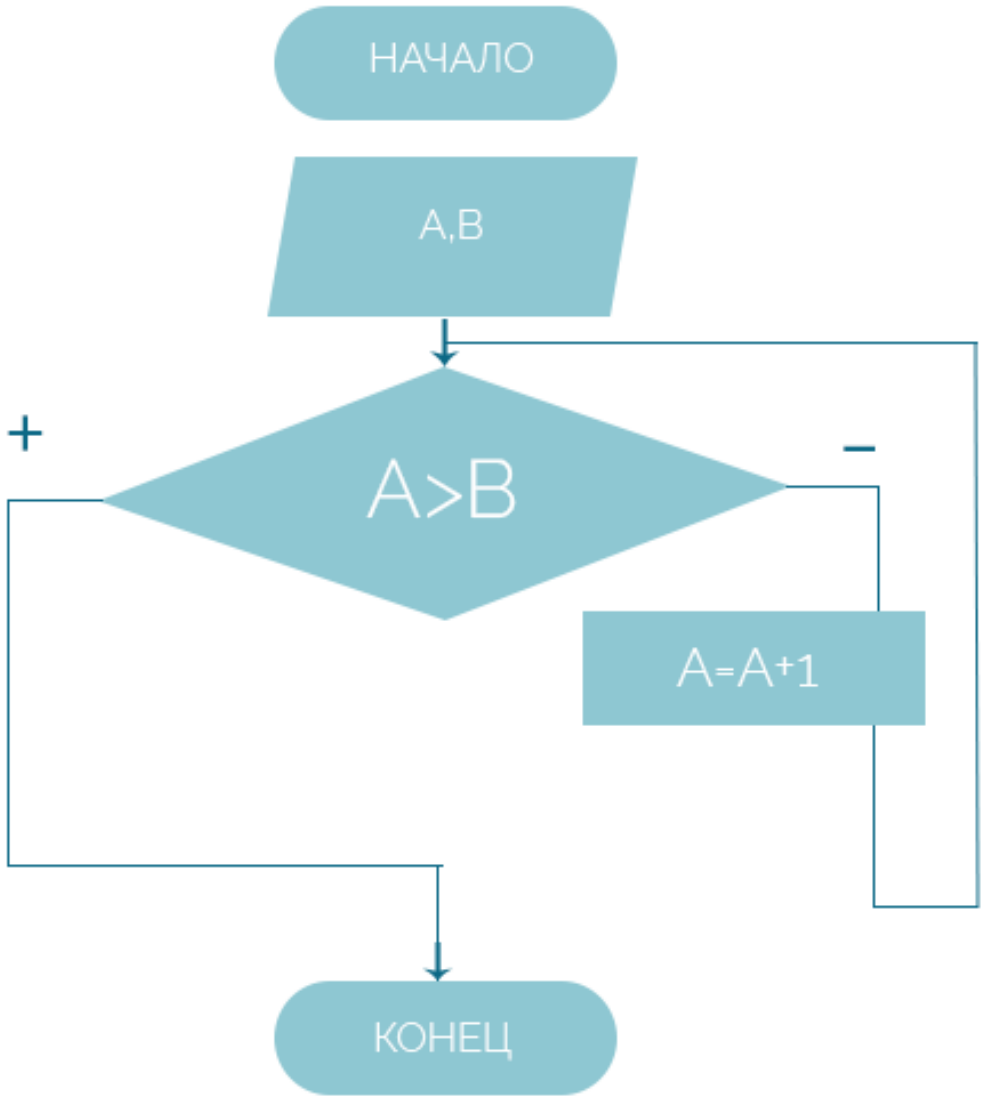
Разветвление. Данная структура обеспечивает выбор одной из двух альтернатив. Проверяется поставленное условие и если условие выполняется, то выбирается одно действие, если нет – другое. Это структура также

называется *if–then–else*(“если–то–иначе”). В итоге оба направления соединяются и независимо от выбора направления, программа продолжает выполнятся. Выполнение условия обозначается словами “да”, “true” или символом “+”, а невыполнение условия – словами “нет”, “false” или символом “–”. Пример разветвляющегося алгоритма приведен ниже в виде блок-схемы:



Цикл. Эта структура позволяет повторно выполнять группу операций несколько раз. Цикл – это последовательность многократно выполняемых операций. Как правило, цикл начинается с проверки условия. Если условие выполняется, операция выполняется и условие проверяется еще раз. Таким образом, цикл продолжается до тех пор, пока условеие не перестает выполняться. Как только условие перестает выполнятся, осуществляется выход из цикла. Данная структура называется “циклом с предусловием”.

“Цикл с постусловием”предусматривает сначало выполнение операции, только потом осуществляет проверку условия. Данная структура применяется в случае, если операцию необходимо осуществить один раз вне зависимости от выполнения условия. Пример циклического алгоритма приведен ниже в виде блок-схемы:



1.3Способы описания алгоритмов

Для строгого задания алгоритмов их обработки требуется иметь такую систему формальных обозначений и правил, чтобы смысл всякого используемого предписания трактовался точно и однозначно.

Языками описаний называются соответствующие системы правил.

К средствам описания алгоритмов относятся следующие основные способы их представления: словесный, графический, псевдокоды, программный. На практике используется также и другой способ описания: табличный (таблицы переключений (таблицы истинности); таблицы автоматов; циклограммы работы; таблицы решений).

Словесный способ описания алгоритмов

Словесный способ записи алгоритмов представляет собой последовательное описание основных этапов обработки данных и задается произвольно на естественном языке.

В качестве примера рассмотрим запись нахождения общего делителя двух натуральных чисел (m и n). Алгоритм может быть записан в следующем виде:

- если числа равны, то необходимо взять любое из них в качестве ответа, в противном случае – продолжить выполнение алгоритма;

- определить большее из чисел;

- заменить большее число разностью большего и меньшего чисел;

- повторить алгоритм сначала.

Способ основан на использовании общепринятых средств общения между людьми и с точки зрения написания трудностей не представляет. Такой способ записи удобно использовать на начальном этапе алгоритмизации задачи. К недостаткам словесного способа записи можно отнести следующее:

1) полное подробное описание алгоритма получается очень громоздким;

2) естественный язык допускает неоднозначность толкования отдельных инструкций;

3) при переходе к этапу программирования требуется дополнительная работа по формализации алгоритма, так как словесное описание может быть понятно человеку, но "непонятно" ПК. Поэтому словесный способ записи алгоритмов не имеет широкого распространения.

Графический способ описания алгоритмов

Блок-схемой называют графическое представление алгоритма, в котором он изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

выдержки

Структурное программирование опирается на основные принципы системного подхода:

а) программа должна составляться мелкими шагами;

б) размер шага определяется количеством решений, применяемых программистом на этом шаге;

в) сложная задача должна разбиваться на достаточно простые, легко воспринимаемые части (блоки), каждая из которых имеет только один вход и один выход;

г) логика алгоритма (программы) должна опираться на минимальное

число достаточно простых базовых управляющих структур.

Структурированная программа представляет собой последовательные или вложенные друг в друга блоки с одним входом и одним

выходом, причем размеры этих блоков могут доходить до уровня элементарных предложений языка программирования (операторов).[3]

2. Содержание учебного предмета.

2.1Структура содержания

Структура содержания общеобразовательного предмета информатики в седьмых, восьмых и девятых классах средней школы может быть определена следующими разделами:[4]

введение в информатику;

алгоритмы и начала программирования;

информационные и коммуникационные технологии.

Кратко остановимся на каждом из них.

2.2 Введение в информатику

Информация. Информационный объект. Информационный процесс. Субъективные характеристики информации, зависящие от личности получателя информации и обстоятельств получения информации: «важность», «своевременность», «достоверность», «актуальность» и т.п.

Представление информации. Формы представления информации. Язык как способ представления информации: естественные и формальные языки.

Кодирование информации. Исторические примеры кодирования. Универсальность дискретного (цифрового, в том числе двоичного) кодирования. Двоичный алфавит. Двоичный код. Разрядность двоичного кода. Связь разрядности двоичного кода и количества кодовых комбинаций.

Понятие о непозиционных и позиционных системах счисления. Знакомство с двоичной, восьмеричной и шестнадцатеричной системами счисления, запись в них целых десятичных чисел от 0 до 256. Перевод небольших целых чисел из двоичной системы счисления в десятичную. Двоичная арифметика.

Компьютерное представление текстовой информации. Кодовые таблицы. Американский стандартный код для обмена информацией, примеры

кодирования букв национальных алфавитов. Представление о стандарте Юникод.

Возможность дискретного представления аудио-визуальных данных (рисунки, картины, фотографии, устная речь, музыка, кинофильмы). Стандарты хранения аудио-визуальной информации.

Размер сообщения как мера количества содержащейся в нём информации. Достоинства и недостатки такого подхода. Другие подходы к измерению количества информации. Единицы измерения количества информации.

Основные виды информационных процессов: хранение, передача и обработка информации. Примеры информационных процессов в системах различной природы; их роль в современном мире.

Хранение информации. Носители информации (бумажные, магнитные, оптические, флэш-память). Качественные и количественные характеристики современных носителей информации: объем информации, хранящейся на носителе; скорости записи и чтения информации. Хранилища информации. Сетевое хранение информации.

Передача информации. Источник, информационный канал, приёмник информации. Скорость передачи информации. Пропускная способность канала. Передача информации в современных системах связи.

Обработка информации. Обработка, связанная с получением новой информации. Обработка, связанная с изменением формы, но не изменяющая содержание информации. Поиск информации.

Управление, управляющая и управляемая системы, прямая и обратная связь. Управление в живой природе, обществе и технике.

Модели и моделирование. Понятия натурной и информационной моделей объекта (предмета, процесса или явления). Модели в математике, физике, литературе, биологии и т.д. Использование моделей в практической деятельности. Виды информационных моделей (словесное описание, таблица, график, диаграмма, формула, чертёж, граф, дерево, список и др.) и

их назначение. Оценка адекватности модели моделируемому объекту и целям моделирования.

Графы, деревья, списки и их применение при моделировании природных и общественных процессов и явлений.

Компьютерное моделирование. Примеры использования компьютерных моделей при решении научно-технических задач. Представление о цикле компьютерного моделирования: построение математической модели, ее программная реализация, проведение компьютерного эксперимента, анализ его результатов, уточнение модели.

Логика высказываний (элементы алгебры логики). Логические значения, операции (логическое отрицание, логическое умножение, логическое сложение), выражения, таблицы истинности. [4]

2.3. Алгоритмы и начала программирования

Понятие исполнителя. Неформальные и формальные исполнители. Учебные исполнители (Робот, Чертёжник, Черепаха, Кузнечик, Водолей) как примеры формальных исполнителей. Их назначение, среда, режим работы, система команд.

Понятие алгоритма как формального описания последовательности действий исполнителя при заданных начальных данных. Свойства алгоритмов. Способы записи алгоритмов.

Алгоритмический язык – язык для записи алгоритмов. Программа – запись алгоритма на алгоритмическом языке. Непосредственное и программное управление исполнителем.

Линейные алгоритмы. Алгоритмические конструкции, связанные с проверкой условий: ветвление и повторение. Разработка алгоритмов: разбиение задачи на подзадачи, понятие вспомогательного алгоритма.

Понятие простой величины. Типы величин: целые, вещественные, символьные, строковые, логические. Переменные и константы. Знакомство с

табличными величинами. Алгоритм работы с величинами – план целенаправленных действий по проведению вычислений при заданных начальных данных с использованием промежуточных результатов.

Язык программирования. Основные правила одного из процедурных языков программирования (Паскаль, школьный алгоритмический язык и др.): правила представления данных; правила записи основных операторов (ввод, вывод, присваивание, ветвление, цикл) и вызова вспомогательных алгоритмов; правила записи программы.

Этапы решения задачи на компьютере: моделирование – разработка алгоритма – запись программы – компьютерный эксперимент. Решение задач по разработке и выполнению программ в выбранной среде программирования. [4]

2.4 Информационные и коммуникационные технологии

Компьютер как универсальное устройство обработки информации.

Основные компоненты персонального компьютера (процессор, оперативная и долговременная память, устройства ввода и вывода информации), их функции и основные характеристики (по состоянию на текущий период времени).

Программный принцип работы компьютера.

Состав программного обеспечения: системное программное обеспечение, прикладное программное обеспечение, системы программирования. Правовые нормы использования программного обеспечения.

Файл. Каталог. Файловая система.

Графический пользовательский интерфейс. Оперирование компьютерными информационными объектами в наглядно-графической форме: создание, именование, сохранение, удаление объектов, организация

их семейств. Стандартизация пользовательского интерфейса персонального компьютера.

Размер файла. Архивирование файлов.

Гигиенические, эргономические и технические условия безопасной эксплуатации компьютера.

Обработка текстов. Текстовые документы и их структурные единицы (раздел, абзац, строка, слово, символ). Технологии создания текстовых документов. Создание и редактирование текстовых документов на компьютере (вставка, удаление и замена символов, работа с фрагментами текстов, проверка правописания, расстановка переносов). Форматирование символов (шрифт, размер, начертание, цвет). Форматирование абзацев (выравнивание, отступ первой строки, междустрочный интервал). Стилевое форматирование. Включение в текстовый документ списков, таблиц, диаграмм, формул и графических объектов. Гипертекст. Создание ссылок: сноски, оглавления, предметные указатели. Инструменты распознавания текстов и компьютерного перевода. Коллективная работа над документом. Примечания. Запись и выделение изменений. Форматирование страниц документа. Ориентация, размеры страницы, величина полей. Нумерация страниц. Колонтитулы. Сохранение документа в различных текстовых форматах.

Графическая информация. Формирование изображения на экране монитора. Компьютерное представление цвета. Компьютерная графика (растровая, векторная). Интерфейс графических редакторов. Форматы графических файлов.

Мультимедиа. Понятие технологии мультимедиа и области её применения. Звук и видео как составляющие мультимедиа. Компьютерные презентации. Дизайн презентации и макеты слайдов. Звуковая и видео информация.

Электронные (динамические) таблицы. Использование формул. Относительные, абсолютные и смешанные ссылки. Выполнение расчётов.

Построение графиков и диаграмм. Понятие о сортировке (упорядочивании) данных.

Реляционные базы данных. Основные понятия, типы данных, системы управления базами данных и принципы работы с ними. Ввод и редактирование записей. Поиск, удаление и сортировка данных.

Коммуникационные технологии. Локальные и глобальные компьютерные сети. Интернет. Браузеры. Взаимодействие на основе компьютерных сетей: электронная почта, чат, форум, телеконференция, сайт. Информационные ресурсы компьютерных сетей: Всемирная паутина, файловые архивы, компьютерные энциклопедии и справочники. Поиск информации в файловой системе, базе данных, Интернете. Средства поиска информации: компьютерные каталоги, поисковые машины, запросы по одному и нескольким признакам.

Проблема достоверности полученной информация. Возможные неформальные подходы к оценке достоверности информации (оценка надежности источника, сравнение данных из разных источников и в разные моменты времени и т.п.). Формальные подходы к доказательству достоверности полученной информации, предоставляемые современными ИКТ: электронная подпись, центры сертификации, сертифицированные сайты и документы и др.

Основы социальной информатики. Роль информации и ИКТ в жизни человека и общества. Примеры применения ИКТ: связь, информационные услуги, научно-технические исследования, управление производством и проектирование промышленных изделий, анализ экспериментальных данных, образование (дистанционное обучение, образовательные источники). [4]

3.СРЕДСТВА РАЗРАБОТКИ ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

3.1Lua(языкпрограммирования)

Для разработки настольной версии приложения был использован язык Lua. Lua ([лу́а], порт. «луна») — скриптовый язык программирования, разработанный в подразделении Tecgraf (Computer Graphics Technology Group) Католического университета Рио-де-Жанейро (Бразилия). Интерпретатор языка является свободно распространяемым, с открытыми исходными текстами на языке Си.По возможностям, идеологии и реализации язык ближе всего к JavaScript, однако Lua отличается более мощными и гораздо более гибкими конструкциями. Хотя Lua не содержит понятия класса и объекта в явном виде, механизмы объектно-ориентированного программирования, включая множественное наследование, легко реализуются с использованием мета таблиц, которые также отвечают за перегрузку операций и т. п. Реализуемая модель объектно-ориентированного программирования — прототипная (как и в JavaScript).[5]

Язык широко используется для создания тиражируемого программного обеспечения (например, на нём написан графический интерфейс пакета Adobe Lightroom). Также получил известность как язык программирования уровней и расширений во многих играх (в том числе World of Warcraft) из-за удобства встраивания, скорости исполнения кода и лёгкости обучения.

3.2CoronaSDK

Corona SDK является комплектом разработки программного обеспечения (SDK), созданный Вальтером Лухом , основателем Corona Labs Inc. Корона SDK позволяет программистам создавать мобильные приложения для устройств на IOS и Android. На рисунках 1,2 представлен внешний вид окна .Corona SDK удобна тем, что в ней имеется встроенный эмулятор. Поэтому не ненужно скачивать дополнительный софт в виде AndroidSDK.[6]

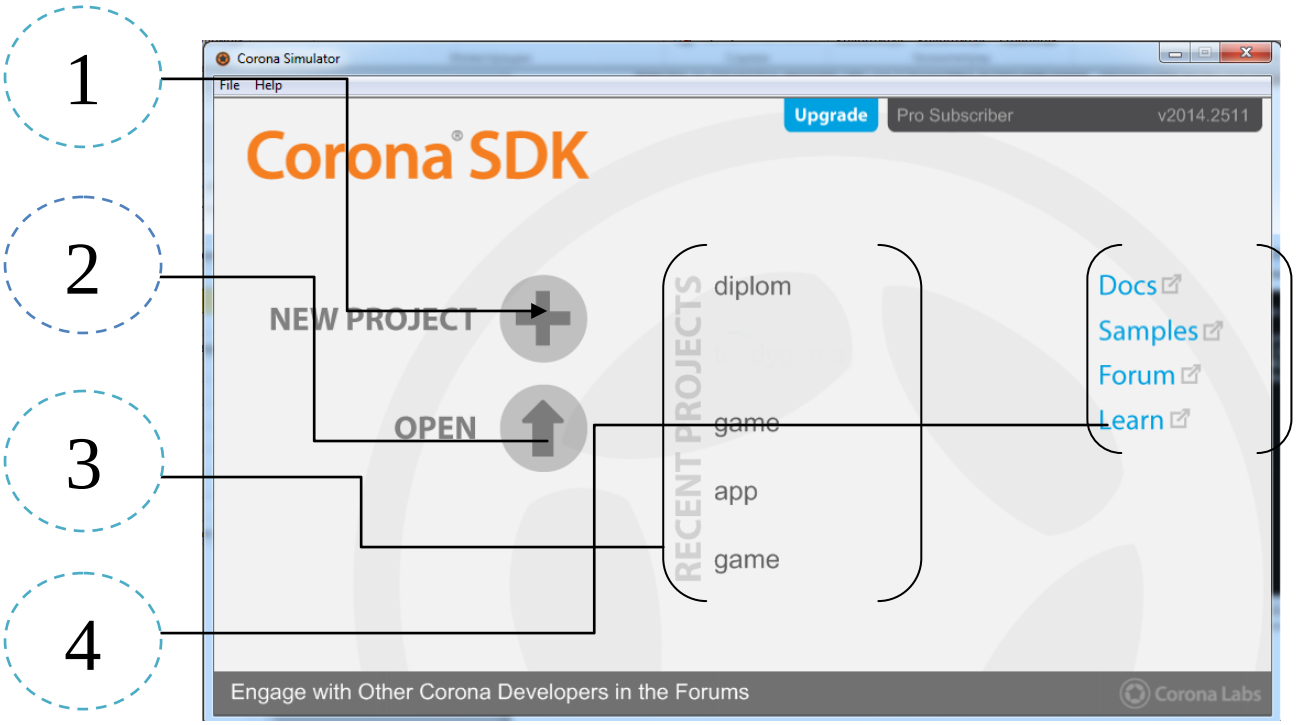


Рис. 1. Внешний вид Corona SDK

1)В окне при нажатии на «NEWPROJECT»будет создан новый проект.

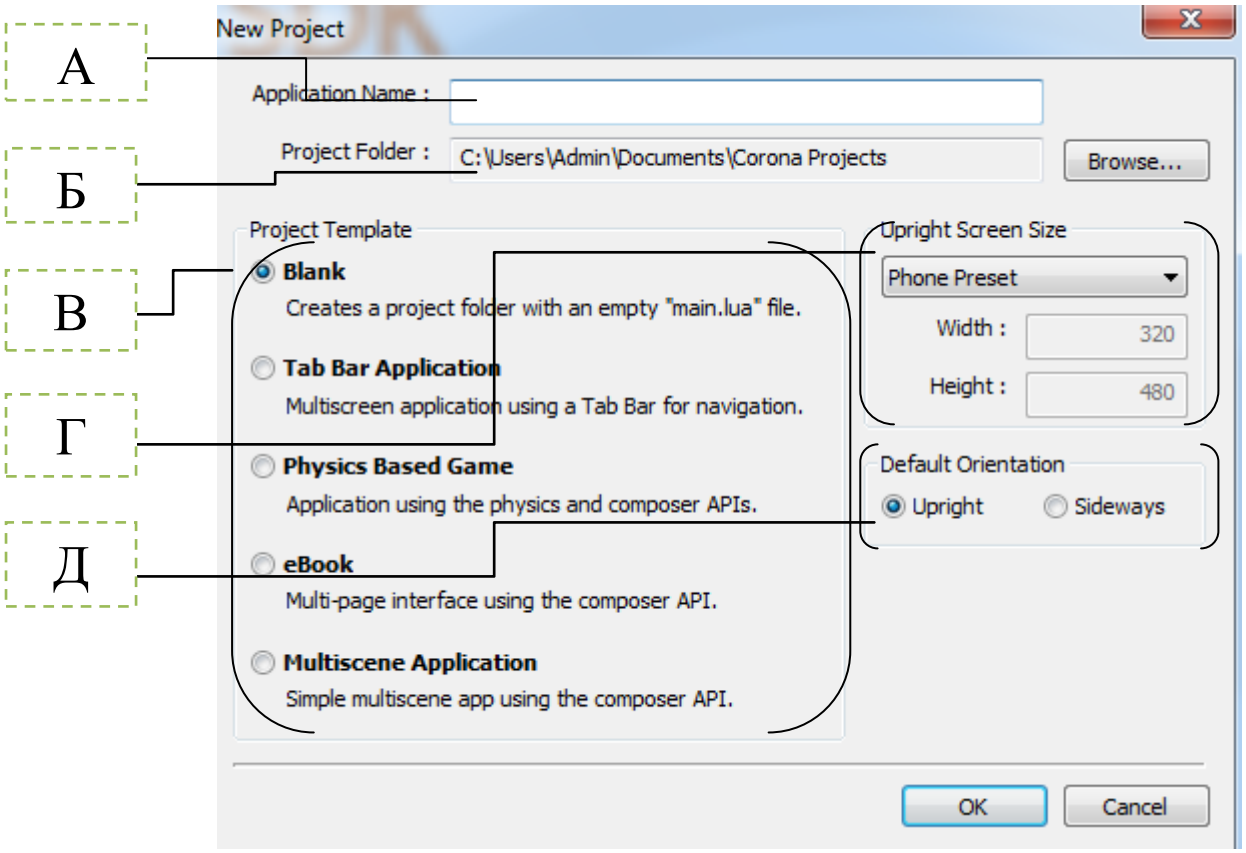


Рис. 2. окно создания проекта

А)Назначить имя для приложения.

Б)Сохранять проект в указанную папку или выбрать свою.

В)Выбрать пустой файл main.lua или шаблон с необходимыми подключенными функциями более подходящий для вашего проекта.

Г)Выбор разрешения дисплея из представленных или указать другое.

Д)Выбор горизонтальной или вертикальной ориентации дисплея.

2)В окне при нажатии на «OPEN» будет открыто приложение на который вы укажите путь.

3)Список последних открытых проектов.

4)Ссылки на официальный сайт с документацией, форумом и обучающими примерами на английском.

3.3Sublime text среда разработки

Sublime Text относится к тем текстовым редактором, которые могут почти все. Благодаря большому количеству плагинов Sublime Text можно настроить почти под любы нужды. Поэтому многие программисты используют его как среду разработки.

Sublime Text поддерживает большое количество языков программирования и имеет возможность подсветки синтаксиса для C, C++, C#, CSS, D, Dylan, Erlang, HTML, Groovy, Haskell, Java, JavaScript, LaTeX, Lisp, Lua, Markdown, MATLAB, OCaml, Perl, PHP, Python, R, Ruby, SQL, TCL и XML. В дополнение к тем языкам программирования, которые включены в первоначальный пакет, пользователи имеют возможность загружать дополнения для поддержки дополнительных языков. [7]

Существует множество сред разработки, но данная была выбрана из-за ее удобного интерфейса, подсвечивания кода и в ней просто комфортно работать. На рисунке 3представлен внешний вид среды разработки.

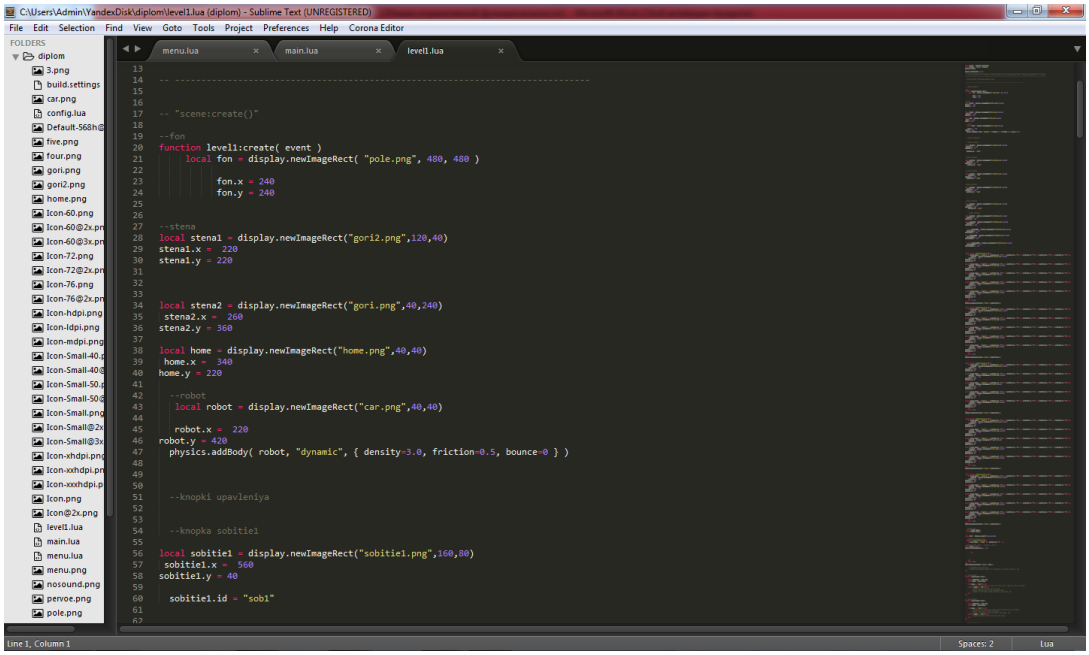


Рис. 3.

На рисунке выше представлен базовый вид среды разработки. Как видно из рисунка, среда разделена на 3 рабочих области, колонка слева содержит менеджер фалов проекта и предоставляет быстрый доступ и удобную навигацию по разрабатываемому приложению. Блок посередине содержит вкладки с открытыми на редактирование файлами. Блок справа быстрая навигация по редактируемому файлу.

Если в Sublime Text вы не нашли нужных функций вам может помочь Package Control. Package Control — это дополнение, упрощающее поиск и установку дополнений для Sublime Text. Кроме того, Package Control сам следит и устанавливает обновления для всех дополнений. Установка Package Control осуществляется в два этапа.

В Sublime Text имеется большое количество горячих клавишь, а если не найдется нужная ее можно настроить. Вкладка настроек изображена на рисунке 4.

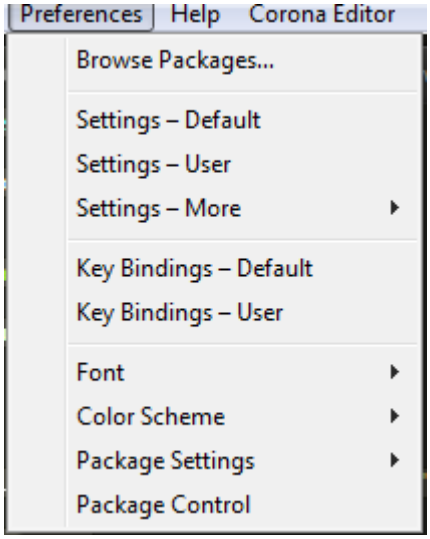


Рис. 4.

При нажатии сочетания клавиш ctr+p открывается мгновенный, живой поиск по открытым файлам и всему проекту. На рисунке 5 изображен пример поиска.

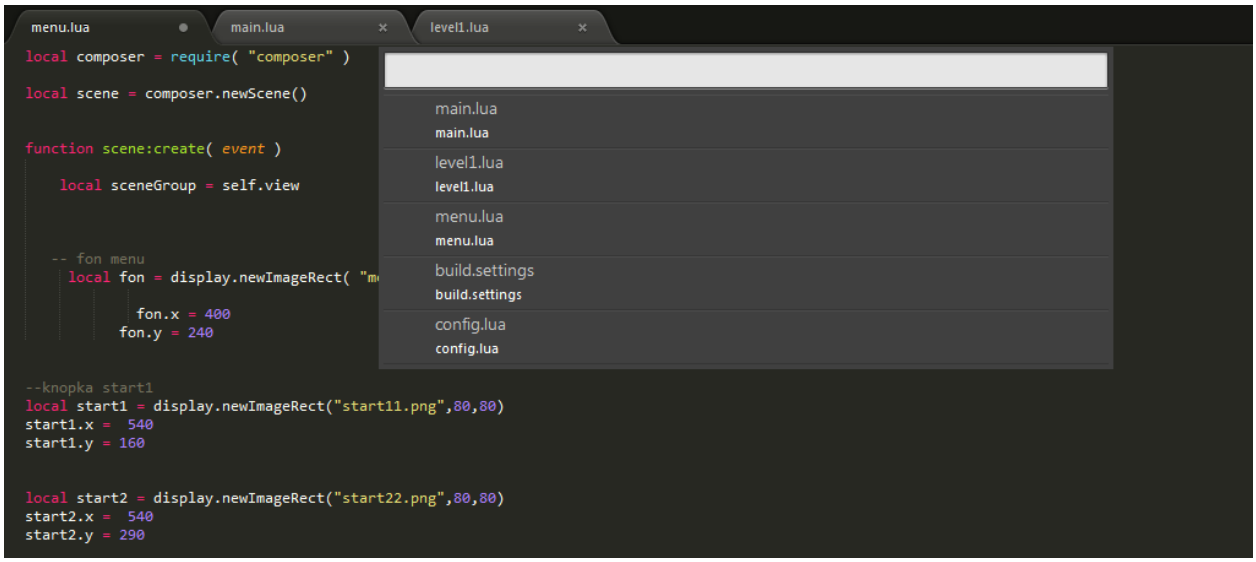


Рис. 5.

Перечисленные функции это основные функции с которыми приходилось мне работать, более подробно можно почитать на официальном сайте.

3.4Photoshop как средство разработки интерфейса

Для разработки интерфейса использовался графический редактор adobe photoshop 6 версии. Он отлично подошел для данной цели. Учитывая большую популярность данного редактора, в интернете можно найти много уроков и дополнений которыми приходилось пользовался для разработки графического интерфейса.

Adobe Photoshop — многофункциональный графический редактор, разработанный и распространяемый фирмой Adobe Systems. В основном работает с растровыми изображениями, однако имеет некоторые векторные инструменты. Продукт является лидером рынка в области коммерческих средств редактирования растровых изображений и наиболее известным продуктом фирмы Adobe. В настоящее время Photoshop доступен на платформах OS X, Windows, в мобильных системах iOS, Windows Phone и Android. Также существует версия PhotoshopExpress для Windows Phone 8.[8]

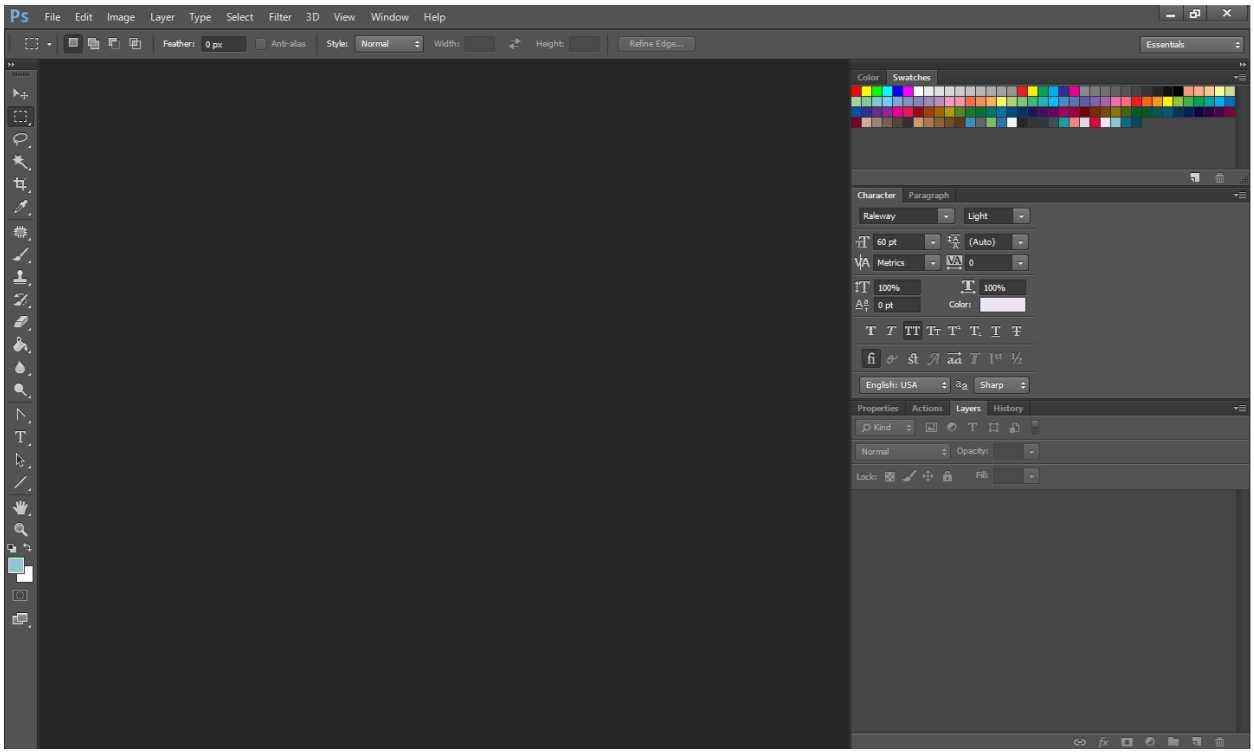


Рис. 6. Окно программы Adobe Photoshop

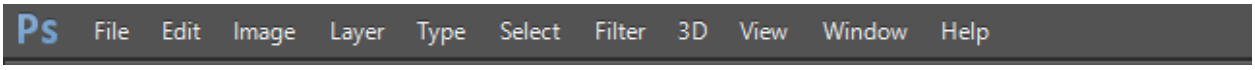


Рис. 7. Строка меню

Строка меню содержит в себе основные команды для настройки программы и редактирования изображения.

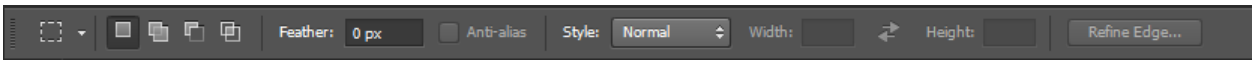


Рис. 8. Панель свойств

Панель свойств используется для настройки инструментов с которыми происходит работа в графическом редакторе.

Рис. 9. Панель инструментов

Панель инструментов имеет важную роль в программе Adobe Photoshop. Почти вся работа с графикой происходит с помощью этой панели. Панель инструментов включает в себя около 30 элементов.

Это все инструменты использовавшиеся для создания приложения. В следующих главах описано принцип работы приложения, а так же его интерфейс.

4.ОПИСАНИЕРАБОТЫ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

4.1 Работа со сценами

В приложении организована работа с восьмью сценами. Главной сценой является main.lua. В ней происходит начальная загрузка приложения. Робота со сценами осуществляется с помощью библиотеки Composer.

Composer является официальной библиотекой для создания сцен в Corona SDK. Пришедшей на смену библиотеки storyboard. Библиотека composer предоставляет разработчикам простой способ создания сцен и перехода между ними сценами.

Ниже представлен шаблон сцены.

```
1
2
3   local composer = require( "composer" )
4
5   local scene = composer.newScene()
6
```

Рис. 10.

На рисунке 10 первая строка подключает библиотеку composer к приложению, а во второй строке объявляется локальная переменная сцена являющаяся самой сценой.

```
 7
 8   function scene:create( event )
 9
10       local sceneGroup = self.view
11
12
13   end
14
```

Рис. 11.

На рисунке 11 изображена инициализация сцены. Сюда добавляются экранные объекты, обработчики действий.

```
15  -- "scene:show()"
16  function scene:show( event )
17
18      local sceneGroup = self.view
19      local phase = event.phase
20
21      if ( phase == "will" ) then
22          -- Called when the scene is still off screen (but is about to come on screen).
23      elseif ( phase == "did" ) then
24          -- Called when the scene is now on screen.
25          -- Insert code here to make the scene come alive.
26          -- Example: start timers, begin animation, play audio, etc.
27      end
28  end
29
```

Рис. 12.

На рисунке 12 изображена функция показа сцены. В ней может быть добавлен запуск таймера, начало анимации, проигрывание музыки.

if (phase == "will") then

Вызывается, когда сцена по-прежнему вне экрана (но собирается прийти на экран).

elseif (phase == "did") then

Вызывается, когда сцена на экране.

end

```
31  -- "scene:hide()"
32  function scene:hide( event )
33
34      local sceneGroup = self.view
35      local phase = event.phase
36
37      if ( phase == "will" ) then
38          -- Called when the scene is on screen (but is about to go off screen).
39          -- Insert code here to "pause" the scene.
40          -- Example: stop timers, stop animation, stop audio, etc.
41      elseif ( phase == "did" ) then
42          -- Called immediately after scene goes off screen.
43      end
44  end
45
```

Рис. 13.

На рисунке 13 изображена функция сворачивания сцены.

if (phase == "will") then

- Вызывается, когда сцена на экране (но собирается исчезнуть с экрана).

- Вставьте код здесь, чтобы "приостановить" сцену.

elseif (phase == "did") then

-- Вызывается сразу после сцены когда гаснет экран.

end

end

```
46
47  -- "scene:destroy()"
48  function scene:destroy( event )
49
50      local sceneGroup = self.view
51
52      -- Called prior to the removal of scene's view ("sceneGroup").
53      -- Insert code here to clean up the scene.
54      -- Example: remove display objects, save state, etc.
55  end
56
```

Рис. 14.

На рисунке 14 изображена функция для очистки сцены.

Обработчики сцен:

```
scene:addEventListener( "create", scene )
scene:addEventListener( "show", scene )
scene:addEventListener( "hide", scene )
scene:addEventListener( "destroy", scene )

return scene
```

4.2Удаление объектов для разгрузки приложения на примере menu.lua

Удаляет объект и освобождает его память, предполагая, что нет других ссылок на него. Это эквивалентно вызову group:remove() того же объекта отображения, но синтаксически проще. Объект: removeSelf () синтаксис также поддерживается в других случаях, таких как удаление физических соединений в физический движок.

Возьмем часть кода, где мы создаем и удаляем объекты.

```
--Создание фона
localfon = display.newImageRect( "menu.png", 800, 480 )

fon.x = 400
fon.y = 240


--Кнопка старт1
local start1 = display.newImageRect("start11.png",80,80)
start1.x =  540
start1.y = 160
```

```
--Кнопка старт2
local start2 = display.newImageRect("start22.png",80,80)
start2.x =  540
start2.y = 290

--Кнопка старт3
local start3 = display.newImageRect("start11.png",80,80)
start3.x=  540
start3.y = 420
```

Это функция обработки касания определенного объекта. При нажатии на кнопку первым делом удаляются объекты кнопки, фон. Далее удаляется сама сцена и мы переходим на сцену с тем уровнем за переход которого отвечает нужная кнопка.

```
function start1:touch( event )
ifevent.phase == "began" then
   start1:removeSelf()
   start2:removeSelf()
   start3:removeSelf()
fon:removeSelf()
composer.removeScene( "menu" )
local options = {
effect = "fade",
time = 800,
params = { level="level1", score=currentScore }
}
composer.gotoScene( "level1", options )

end
return true
```

```
end
start1:addEventListener( "touch", start1 )
```

4.3Описание работы события касания экрана

Когда палец пользователя касается экрана, событие генерируется и отправляется для отображения объектов в иерархии отображения. Только те объекты, которые пересекают место касания (расположение пальца на экране) будут принимать события.

```
localobject=display.newImage("ball.png")

functionobject:touch(event)

ifevent.phase=="began"then
print("Event dispatched; name: "..event.name)
end
returntrue
end
object:addEventListener("touch", object)
```

5. ВЗАИМОДЕЙСТВИЕ ПОЛЬЗОВАТЕЛЯ С ПРИЛОЖЕНИЕМ

5.1Экран загрузки и меню в приложении

Данный экран отображается при запуске приложения. Это первое действие которое выполняется в приложении.

Рис. 15.

После его загрузки мы переходим на экран меню (сцену menu.lua).

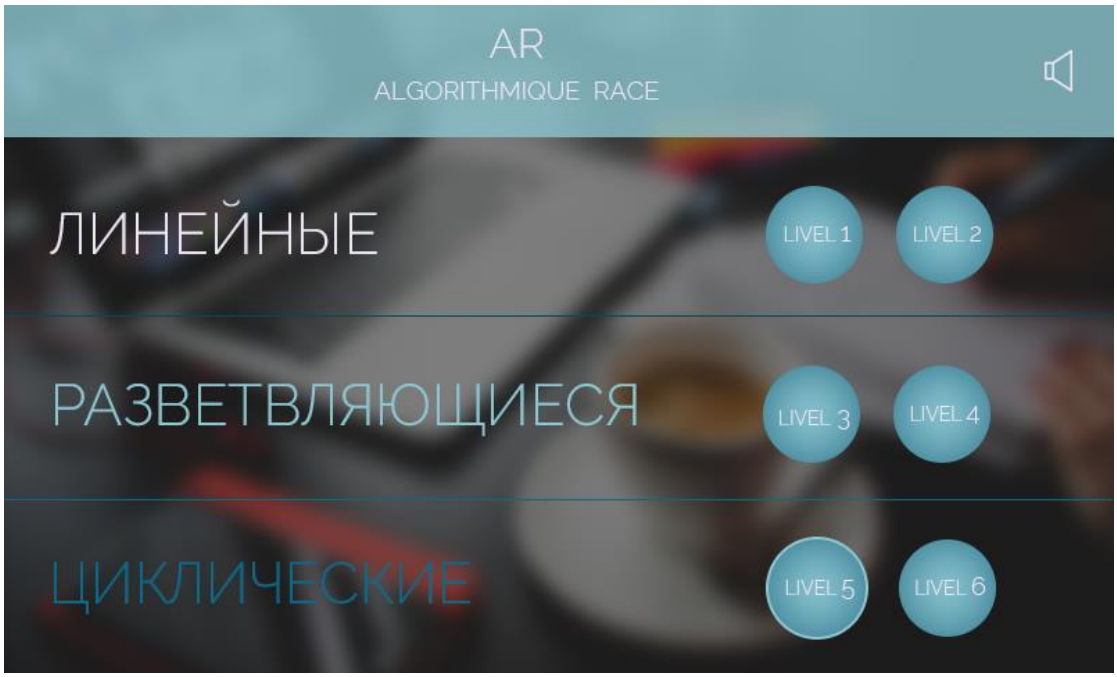


Рис. 16.

На экране меню мы видим 3 типа уровней. Линейный, разветвляющийся и циклический. В каждой из категорий по 2 уровня соответствующих типу уровней. На иконку сверху мы можем включить или выключить звук.

При переходе на какой либо уровень объекты находящиеся в меню будут удалены. Удаление объектов происходит для освобождения памяти на устройстве.

5.2Уровни в приложении

Для примера представлен первый уровень линейного алгоритма. Ниже представлен скриншот первого уровня.

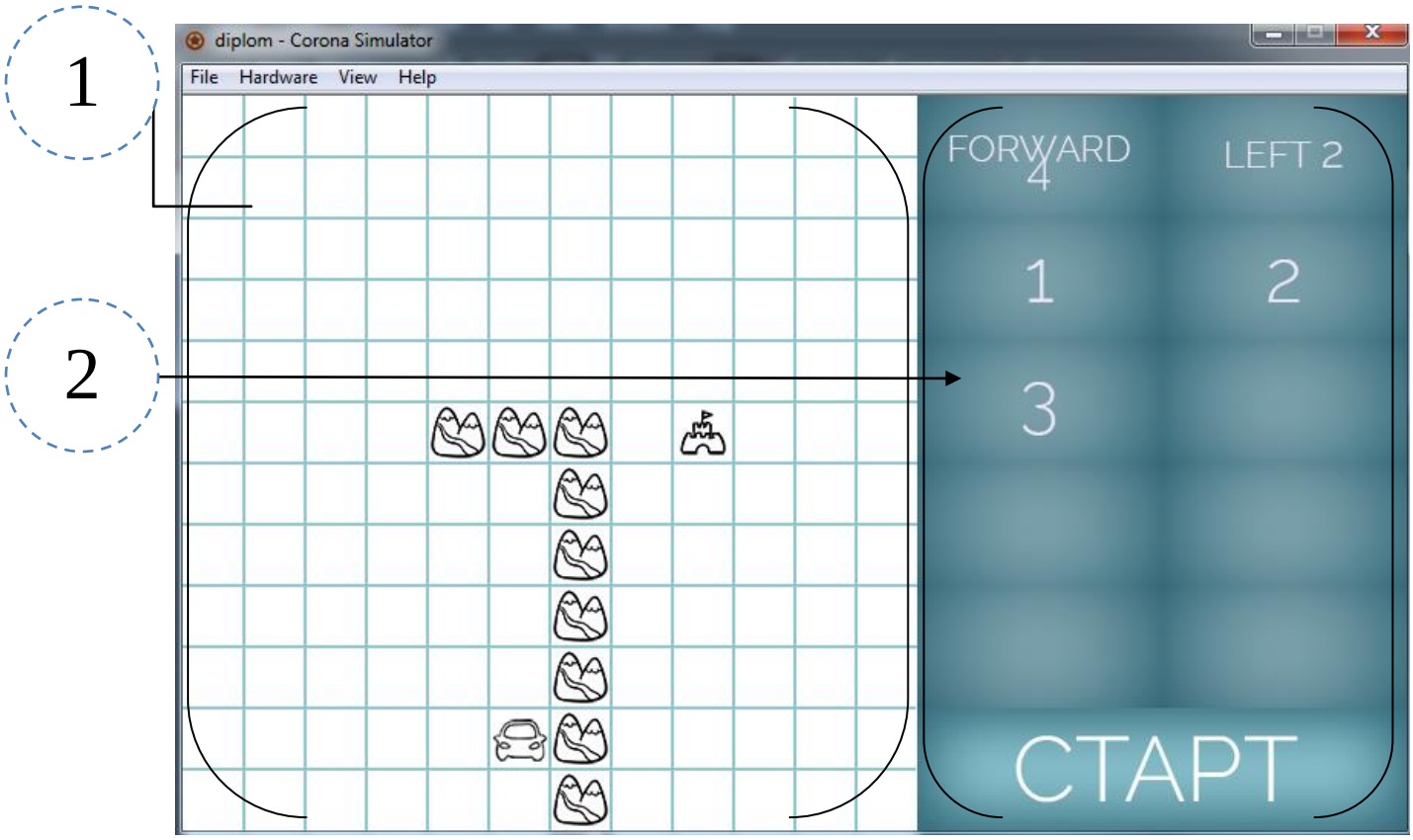


Рис. 17

1) . На рисунке 17 под номером 1 изображено поле, где автомобиль доезжает до замка в случае правильно указанного последствия алгоритма. Добравшись до замка мы автоматически переходим на сцену меню.
2) Справа изображено поле, где нужно производить действия. От них зависит доберется ли машина до конечной точки. В данном случае указанно пять действий которые должны произойти. Нажимая на какое либо действие ему присваивается последовательность. После указания последовательности нужно нажать на кнопку старт. Если

все окажется верно машина доберется до конечный точки и уровень будет считаться пройденным.

5.3Справка

Рис. 18.

На рисунке 18 изображена сцена справки. Эта сцена создана, как небольшое руководство к приложению.

6.СРЕДСТВА РАЗРАБОТКИ ДЛЯ САЙТА

6.1HTML

HTML – язык гипертекстовой разметки документов (HyperText Markup Language). С помощью HTML можно создавать веб-сайты, находящиеся в интернете. HTML – это не является языком программирования, он является языком разметки. С помощью HTML текстовый документ разбивают на блоки смысловой информации (заголовки, параграфы, таблицы, рисунки и т.п.).

Гипертекстовый документ – это документ, который содержит переходы (гиперссылки) на другие документы. При использовании гиперссылок происходит перемещение от одного документа к другому в Интернете. HTML-документ является гипертекстовым документом.[8]

Особенности HTML-документа:

1. HTML-документ содержит в себе текст, графику, видео.

2. HTML-документ – это один или несколько текстовых файлов, которые имеют расширение .htm или .html.

3. Создавать HTML-документ можно как с помощью специальных программ – редакторов HTML (например, Notepad++), так и с помощью любого текстового редактора, даже встроенного блокнота в Windows.

4. Для просмотра HTML-документов существуют специальные программы браузеры(chrome, opera, firefox). Они интерпретируют HTML-документы, т.е. переводят текст документа в веб-страницу, и отображают ее на экране пользователя. Существует очень много различных браузеров, но наиболее распространенными браузерами являются Chrome, Firefox и Opera.

5. Если при интерпретации HTML-документа используется устаревший браузер, то при отображение могут возникать ошибки в виде съехавших или не отображающихся элементов.

HTML-документ состоит из элементов HTML.

Элемент HTML – это в основном документ имеющий два тега (открывающий и закрывающий) и часть документа между ними. Кроме того, существуют элементы HTML, состоящие только из одного тега.

Тег – в переводе с английского – ярлык, этикетка. Тег определяет тип выводимого элемента HTML (например, заголовок, таблица, рисунок и т.п.). Сам тег не отображается браузером. Тег представляет собой последовательность элементов:[9]

- символ левой угловой скобки (<) – начало тега;
- необязательный символ слеша (/) – символ используется, чтобы обозначить закрывающий тег;
- имя тега;
- необязательные атрибуты в открывающем теге;
- символ правой угловой скобки (>)

Атрибуты – необязательный набор параметров, определяющих дополнительные свойства элемента HTML (например, цвет или размер). Атрибут

состоит:

- из имени атрибута;
- знака равенства (=);
- значения атрибута – строки символов, заключенной в кавычки

Пример элемента HTML:

<H1 ALIGN= "CENTER">Глава 1</H1>

В этом примере:

<H1 ALIGN= "CENTER"> – открывающий тег

</H1> – закрывающий тег

H1 – имя тега

ALIGN= "CENTER" – атрибут

ALIGN – имя атрибута

"CENTER" – значение атрибута

Содержание элемента HTML

Правила создания HTML-документов:

Теги и атрибуты можно записывать в любом регистре, т.е. </H1> и </h1> – это одно и то же.

Несколько пробелов подряд, символы табуляции и перевода строки при интерпретации браузером заменяются на один пробел. Это позволяет писать хорошо структурированные исходные тексты файлов HTML.

Рекомендуется давать имена файлам HTML строчными английскими буквами. Длина имени – до восьми символов. В принципе, можно не придерживаться данной рекомендации, но тогда пользователи, работающие в операционных системах, отличных от Windows, не смогут воспользоваться вашими HTML-документами.

Далее приведен код страницы сайта, а именно файл index с расширением html.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Algorithmic race</title>
    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
<link href="css/style.css" rel="stylesheet">

  </head>

  <body>

<div class="jumbotron">
<div class="nav">
     <div class="container">

       <ul class="pull-left">
```

```
            <li> <img src="img/logo.png" alt="..." "></li>

          </ul>
          <ul class="pull-right">
            <li><a href="#">Главная</a></li>
            <li><a href="#">Скачать</a></li>

          </ul>
        </div>
      </div>
    <div class="container">
     <h1>Algorithmic race</h1>
     <p>Мобильное приложение для Android.</p>

     <button type="button" class="btn btn-primary btn-lg" data-toggle="modal"
data-target="#exampleModal" data-whatever="@mdo">Скачать apk</button>

    </div>
    </div>

    <div class="container">
          <div class="row">
              <div class="col-xs-6 col-md-4">

                  <div class="thumbnail">
        <img src="img/1.png" alt="..." class="img-rounded">
         <div class="caption">
          <h3>ЛИНЕЙНЫЙ АЛГОРИТМ</h3>
          <p>В этом алгоритме все операции выполняются последовательно
одна за другой. </p>
```

```
  </div>
</div>
        </div>
            <div class="col-xs-6 col-md-4">
                <div class="thumbnail">
  <img src="img/2.png" alt="..." class="img-rounded">
  <div class="caption">
    <h3>РАЗВЕТВЛЯЮЩИЙСЯ АЛГОРИТМ</h3>
    <p>Данная структура обеспечивает выбор одной из двух
```
альтернатив. Проверяется поставленное условие и если условие выполняется, то выбирается одно действие, если нет – другое. </p>

```
  </div>
</div>
            </div>
            <div class="col-xs-6 col-md-4">
                <div class="thumbnail">
  <img src="img/3.png" alt="..." class="img-rounded">
  <div class="caption">
    <h3>ЦИКЛИЧЕСКИЙ АЛГОРИТМ</h3>
    <p>Эта структура позволяет повторно выполнять группу операций
```
несколько раз. Цикл – это последовательность многократно выполняемых операций. </p>

```
  </div>
</div>
            </div>
        </div>
    </div>
```

```
  <div class="make">
<div class="container">
        <div class="row">
            <div class="how">
                    <h2>Средства разработки</h2>
                    <p>Список используемых мною програм</p>
                <img src="img/how.png" >
                    </div>
              </div>
             </div>
            </div>

            <div class="container">
              <div class="row">
              <div class="afterhow">
              <div class="col-md-8 col-md-offset-2">
                    <div class="media">
          <div class="media-left media-middle">
           <a href="#">
             <img class="media-object" src="img/11.png" alt="...">
           </a>
          </div>
          <div class="media-body">
           <h4 class="media-heading">CoronaSDK</h4>
           <p>CoronaSDK позволяет создавать мобильные приложения для
устройств на IOS и Android.</p>
          </div>
        </div>
        <div class="media">
         <div class="media-left media-middle">
```

```
<a href="#">
<img class="media-object" src="img/22.png" alt="...">
</a>
</div>
<div class="media-body">
<h4 class="media-heading">Sublime Text</h4>
<p>Sublime Text относится к тем текстовым редактором, которые могут почти все. Благодаря большомуколичеству плагинов Sublime Text можно настроить почти под любы нужды. </p>
</div>
</div>
<div class="media">
<div class="media-left media-middle">
<a href="#">
<img class="media-object" src="img/33.png" alt="...">
</a>
</div>
<div class="media-body">
<h4 class="media-heading">Adobe Photoshop</h4>
<p>AdobePhotoshop — многофункциональный графический редактор, разработанный и распространяемый фирмой AdobeSystems.</p>
</div>
</div>
</div>
</div>
</div>
<div class="footer">
<div class="container">
<div class="row">
```

```
          <div class="col-xs-6 col-md-4">
          <div class="thumbnail">
            <img src="img/111.png" alt="...">
            <div class="caption">
              <h3>Цель задачи</h3>
              <p>Цель задачи довести машину до замка</p>
             </div>
            </div>
          </div>
                <div class="col-xs-6 col-md-4">
          <div class="thumbnail">
            <img src="img/222.png" alt="...">
            <div class="caption">
              <h3>Выбор действий</h3>
              <p>Выбрав правильные действия задача будет считаться
решенной</p>

            </div>
          </div>
        </div>
          <div class="col-xs-6 col-md-4">
            <div class="thumbnail">
              <img src="img/333.png" alt="...">
              <div class="caption">
                <h3>Конец</h3>
                <p>В случае правильного выбора машина доберется до замка и
будет осуществлен переход на следующую сцену</p>

            </div>
          </div>
```

```
</div>
        <div class="col-md-8 col-md-offset-2">
  </div>
  </div>

  </div>
</div>

        <div class="afterfooter">
  <div class="container">
    <p>Сайт к дипломной работе</p>
  </div>
  </div>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as
needed -->
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

6.2CSS

CSS - это каскадные таблицы стилей, определяющий представление данных в браузере.

Если формат HTML предоставляет информацию о составе документа, то таблицы стилей определяют как он должен выглядеть. Таким образом

каскадные таблицы стилей дают возможность хранить содержимое отдельно от его представления.

Стиль включает все типы элементов дизайна: шрифт, фон, текст, цвета ссылок, поля, рамки, тени, фильтры и расположение объектов на странице. CSS разрабатывались для обеспечения контроля над размещением текста и графики.

Каскадные таблицы стилей обеспечивают должный уровень единства оформления, организации и контроля во время разработки узла, который является недостижимым с помощью одного только HTML.

CSS предполагает 3 типа таблиц стилей - встроенные, внедренные (внутренние) и связанные (внешние).

Впервые идея форматирования HTML-документов с помощью CSS была рекомендована Консорциумом W3C в 1996 году. Эта рекомендация, которая была обновлена в 1998 году, используется веб-разработчиками и по сей день.[9]

Термин "каскадный" означает, что в одной странице HTML могут использоваться разные стили. Браузер, поддерживающий таблицы стилей, будет следовать их порядку (как по каскаду), интерпретируя информацию стилей.

Это означает, что вы можете использовать все три типа стилей, и браузер будет интерпретировать сначала связанные, затем внедренные и, наконец, встроенные стили. Даже если ко всему узлу будут применены образцы стилей, можно будет управлять отдельными аспектами страниц с помощью внедренных стилей, а отдельными областями внутри этих страниц - с помощью встроенных стилей.

Другой аспект каскадирования - наследование (inheritance). Наследование означает, что если не указано иное, то конкретный стиль будет унаследован другими элементами страницы HTML. Например, если вы примените

определенный цвет текста в теге <p>, то все теги внутри этого абзаца наследуют этот цвет, если не оговорено иное.

Далее приведена верстка сайта. Файл style с расширение css.

```
.nav img{
  margin-top:-4px;
}

.nav a {
  color:#f2f3f3;
  font-size: 11px;
  font-weight: bold;
  padding: 14px 10px;
  text-transform: uppercase;
}

.nav li {
  display: inline;
}

.jumbotron {
  background-image:url(../img/jumbotron.jpg) ;

  height: 600px;
  background-repeat: no-repeat;
  background-size: cover;
  text-align:center;
}

.jumbotron .container {
  position: relative;
```

```
}

.jumbotron h1{
margin-top:100px;
 color:#f2f3f3;
text-shadow:  1px 1px 1px #000;
padding-bottom:20px;:

}

.jumbotron p{
color:#f2f3f3;
text-shadow:  1px 1px 1px #000;
font-size:22px;
padding-bottom:20px;:
}

.jumbotron .btn {
 background-color:#16a1a4;
 border:none;

}

body {
  position: relative;

}

.make {
```

```
background-color:#8ec7d2;
      border-bottom: 1px solid #16a1a4;
      border-top: 1px solid #16a1a4;
      margin-bottom: 13px;

}

.thumbnail {
border:none;
}

.thumbnail h3{
color:#16a1a4;
padding-bottom:5px;
text-align:center;
}

.thumbnail p{
color:#0d6986;
text-align:center;

}

.how {

padding-top:40px;
padding-bottom:40px;
text-align:center;
}
.how h2{
```

```
color:#fff;
}

.how p{
color:#0d6986;
}

.how img{
padding-top:3px;
}

.media {
padding:20px;
}

.media p{
color:#777777;
padding-left:15px;
}

.media h4{
padding-top:15px;
padding-left:15px;
}

.footer{
  background:#16a1a4 ;
margin-top:50px;
  height: 400px;
  background-repeat: no-repeat;
```

```
  background-size: cover;

}

.afterfooter{
color:#fff ;
  background:#626262;
height: 50px;

}

.afterfooter p{

padding: 15px;
text-align:right;
}

.afterhow{
padding-top:50px;

}

.footer .thumbnail {
margin-top:20px;
  background:#16a1a4 ;
}

.footer .thumbnail h3{

color:#fff ;
```

```
}

.footer .thumbnail p{

color:#fff ;
}
```

6.3Bootstrap

TwitterBootstrap — свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML и CSS шаблоны оформления для веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейсов, включая JavaScript расширения.

Bootstrap использует самые современные наработки в области CSS и HTML, поэтому необходимо быть внимательным при поддержке старых браузеров.[10]

Основные преимущества TwitterBootstrap 3:

Экономия времени — Bootstrap позволяет сэкономить время и усилия, используя готовые решения, и сконцентрироваться на других разработках;

Высокая скорость — динамичные макеты Bootstrap адаптивны и с изменением разрешения экрана не делает изменений в разметке веб-страницы;

Гармоничный дизайн — все компоненты платформы Bootstrap используют единый стиль и шаблоны с помощью центральной библиотеки. ВДизайн и макеты веб-страниц согласуются друг с другом;

Простота в использовании — платформа проста в использовании, пользователь с базовыми знаниями HTML и CSS может начать разработку с TwitterBootstrap;

Совместимость с браузерами — TwitterBootstrap совместим с MozillaFirefox, GoogleChrome, Safari, InternetExplorer и Opera;

Открытое программное обеспечение — особенность TwitterBootstrap, которая предполагает удобство использования, посредством открытости исходных кодов и бесплатной загрузки.

Основные инструменты Bootstrap:

Сетки — заранее заданные размеры колонок, которые можно сразу же использовать, например ширина колонки 140px относится к классу .span2, который можно использовать в CSS описании документа.

Шаблоны — Фиксированный или резиновый шаблон документа.

Типографика — Описания шрифтов, определение некоторых классов для шрифтов, таких как код, цитаты и т. п.

Медиа — Представляет некоторое управление изображениями и Видео.

Таблицы — Средства оформления таблиц, вплоть до добавления функциональности сортировки.

Формы — Классы для оформления форм и некоторых событий происходящих с ними.

Навигация — Классы оформления для Табов, Вкладок, Страничности, Меню и Тулбара.

Алерты — Оформление диалоговых окон, Подсказок и Всплывающих окон.[13]

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломной работы был получен опыт в разработке мобильных приложений и сайтов. Мобильное приложение было сделано с помощью с помощью движка corona sdk, а сайт с помощью фреймворка bootstrap. Изучив несколько уроков по Adobe Photoshop были получены навыки работы в этом графическом редакторе. Получен опыт и понимания принципов стороннего взаимодействия систем в операционной системе Android. Работая над своим приложением я старался показать принципы работы алгоритмов путем игры. Данное приложение может быть использовано для школьников.

Задание на дипломную работу были выполнено в полном объёме.

Список используемой литературы

1) Сайт со статистикой рынка URL: http://volbusiness.ru

2)Макарова Н.В., Кочурова Е.Г., Николайчук Г.С., Нилова Ю.Н., Титова Ю.Ф. Информатика и ИКТ. Учебник. 8-9 класс / Под ред. проф. Н. В. Макаровой. — СПб.: Питер, 2010. - 416 с: ил

3) Информатика: Учебник для вузов. Стандарт третьего поколения. Макарова Н. В., Волков В. Б.: 2015 год, 576с.

4) Рабочая программа по информатике URL: http://nsportal.ru

5) Хабрахабр URL: http://habrahabr.ru

6)Сайт движка URL: http://coronalabs.com

7)Сайт среды разработки URL: http://www.sublimetext.com

8)Сайт со статистикой рынка URL: http://volbusiness.ru

7)Фремворк для сайта URL: http://getbootstrap.com

8) Свободная энциклопедия Википедия URL: https://ru.wikipedia.org

9) HTML, javascript, PHP и MySQL. Джентльменский набор Web-мастера. Прохоренок, БХВ-Петербург, Н. А. 900с.

10)Сайт с уроками по Adobe Photoshop URL: http://www.photoshop-master.ru

11) Форум о программировании для мобильных устройств URL: http://www.4pda.ru

12) Форум разработчиков под Андроид URL: http://www.gamedev.ru

ПРИЛОЖЕНИЕ ДИПЛОМУ

В данном приложении содержится базовый код разработки, именно 2 основные сцены приложения. На их базе созданы друге сцены.

1)menu.lua

```
 local composer = require( "composer" )

local scene = composer.newScene()



function scene:create( event )

    local sceneGroup = self.view



   -- fon menu
     local fon = display.newImageRect( "menu.png", 800, 480 )

            fon.x = 400
          fon.y = 240




--knopka help
local help = display.newImageRect("help.png",50,50)
help.x =  50
help.y = 45
```

```
--knopka sound
local sound = display.newImageRect("sound.png",30,30)
sound.x =  750
sound.y = 45

--knopka start1
local start1 = display.newImageRect("start11.png",80,80)
start1.x =  540
start1.y = 160


local start2 = display.newImageRect("start22.png",80,80)
start2.x =  540
start2.y = 290


local start3 = display.newImageRect("start11.png",80,80)
start3.x =  540
start3.y = 420


 function start1:touch( event )
    if event.phase == "began" then
        help:removeSelf()
        sound:removeSelf()
    start1:removeSelf()
    start2:removeSelf()
```

```
    start3:removeSelf()
    fon:removeSelf()
    composer.removeScene( "menu" )


composer.gotoScene( "level1" )

    end
    return true

end
start1:addEventListener( "touch", start1 )

 function help:touch( event )
              if event.phase == "began" then
    fon:removeSelf()
     help:removeSelf()
        sound:removeSelf()
    start1:removeSelf()
    start2:removeSelf()
    start3:removeSelf()
    composer.removeScene( "menu" )


composer.gotoScene( "help" )

                        end
                         return true
                         end
                help:addEventListener( "touch", help )
```

```
end

-- "scene:show()"
function scene:show( event )

    local sceneGroup = self.view
    local phase = event.phase

    if ( phase == "will" ) then
        -- Called when the scene is still off screen (but is about to come on screen).
    elseif ( phase == "did" ) then
        -- Called when the scene is now on screen.
        -- Insert code here to make the scene come alive.
        -- Example: start timers, begin animation, play audio, etc.
    end
end



-- "scene:hide()"
function scene:hide( event )

    local sceneGroup = self.view
    local phase = event.phase

    if ( phase == "will" ) then
        -- Called when the scene is on screen (but is about to go off screen).
        -- Insert code here to "pause" the scene.
        -- Example: stop timers, stop animation, stop audio, etc.
    elseif ( phase == "did" ) then
```

```
        -- Called immediately after scene goes off screen.
    end
end


-- "scene:destroy()"
function scene:destroy( event )

    local sceneGroup = self.view

    -- Called prior to the removal of scene's view ("sceneGroup").
    -- Insert code here to clean up the scene.
    -- Example: remove display objects, save state, etc.
end


-- ---------------------------------------------------------------------------

-- Listener setup
scene:addEventListener( "create", scene )
scene:addEventListener( "show", scene )
scene:addEventListener( "hide", scene )
scene:addEventListener( "destroy", scene )

-- ---------------------------------------------------------------------------

return scene
```

2)livel1.lua

```
local composer = require( "composer" )

local level1 = composer.newScene()
local physics = require "physics"
physics.start()

physics.setGravity( 0, 0 )
-- ---------------------------------------------------------------------------------------------
---------------
-- All code outside of the listener functions will only be executed ONCE unless
"composer.removeScene()" is called.
-- ---------------------------------------------------------------------------------------------
---------------

-- local forward references should go here

-- -------------------------------------------------------------------------------


-- "scene:create()"

--fon
function level1:create( event )
    local fon = display.newImageRect( "pole.png", 480, 480 )

            fon.x = 240
        fon.y = 240
```

```
--stena
local stena1 = display.newImageRect("gori2.png",120,40)
stena1.x =  220
stena1.y = 220



local stena2 = display.newImageRect("gori.png",40,240)
 stena2.x =  260
stena2.y = 360

local home = display.newImageRect("home.png",40,40)
 home.x =  340
home.y = 220
physics.addBody( home, "static", { density=2.0, friction=0.5, bounce=0 } )

          --robot
           local robot = display.newImageRect("car.png",40,40)

           robot.x =  220
robot.y = 420
  physics.addBody( robot, "dynamic", { density=5.0, friction=0.5, bounce=0 } )

           local point1 = display.newRect(220,260,1,1)

  physics.addBody( point1, "dynamic", { density=2.0, friction=0.5, bounce=0 } )

           local point2 = display.newRect(140,260,1,1)

  physics.addBody( point2, "dynamic", { density=2.0, friction=0.5, bounce=0 } )
```

```
        local point3 = display.newRect(140,180,1,1)

  physics.addBody( point3, "dynamic", { density=2.0, friction=0.5, bounce=0 } )

        local point4 = display.newRect(340,180,1,1)

  physics.addBody( point4, "dynamic", { density=2.0, friction=0.5, bounce=0 } )




        --knopki upavleniya


        --knopka sobitie1

local sobitie1 = display.newImageRect("sobitie1.png",160,80)
 sobitie1.x =  560
sobitie1.y = 40

        sobitie1.id = "sob1"



--knopka sobitie2

local sobitie2 = display.newImageRect("sobitie2.png",160,80)
        sobitie2.x =  720
```

```
sobitie2.y = 40
        sobitie2.id = "sob2"



--knopka sobitie3

local sobitie3 = display.newImageRect("sobitie3.png",160,80)
   sobitie3.x =  560
sobitie3.y = 120
  sobitie3.id = "sob3"



--knopka sobitie4

local sobitie4 = display.newImageRect("sobitie5.png",160,80)
  sobitie4.x =  720
sobitie4.y = 120
  sobitie4.id = "sob4"



--knopka sobitie5

local sobitie5 = display.newImageRect("sobitie5.png",160,80)
  sobitie5.x =  560
sobitie5.y = 200
```

```
    sobitie5.id = "sob5"


    --knopka sobitie0

local sobitie0 = display.newImageRect("sobitie0.png",160,80)
  sobitie0.x =  720
sobitie0.y = 200

local sobitie00 = display.newImageRect("sobitie0.png",160,80)
  sobitie00.x =  560
sobitie00.y = 280


   local sobitie000 = display.newImageRect("sobitie0.png",160,80)
  sobitie000.x =  720
sobitie000.y = 280

local sobitie0000 = display.newImageRect("sobitie0.png",160,80)
  sobitie0000.x =  560
sobitie0000.y = 360


   local sobitie00000 = display.newImageRect("sobitie0.png",160,80)
  sobitie00000.x =  720
sobitie00000.y = 360
```

```
local function onObjectTouch( event )
   if (event.phase == "began") and (sobitie1.id~="1") and (sobitie2.id~="1") and
(sobitie3.id~="1") and (sobitie4.id~="1") and (sobitie5.id~="1") then
       sobitie1 = display.newImageRect("pervoe.png",160,80)
    sobitie1.x =  560
sobitie1.y = 40
sobitie1.id="1"

elseif (event.phase == "began") and (sobitie1.id~="2") and (sobitie2.id~="2") and
(sobitie3.id~="2") and (sobitie4.id~="2") and (sobitie5.id~="2") then
    sobitie1 = display.newImageRect("vtoroe.png",160,80)
    sobitie1.x =  560
sobitie1.y = 40
sobitie1.id="2"

elseif (event.phase == "began") and (sobitie1.id~="3") and (sobitie2.id~="3") and
(sobitie3.id~="3") and (sobitie4.id~="3") and (sobitie5.id~="3") then
    sobitie1 = display.newImageRect("3.png",160,80)
    sobitie1.x =  560
sobitie1.y = 40
sobitie1.id="3"

elseif (event.phase == "began") and (sobitie1.id~="4") and (sobitie2.id~="4") and
(sobitie3.id~="4") and (sobitie4.id~="4") and (sobitie5.id~="4") then
     sobitie1 = display.newImageRect("four.png",160,80)
    sobitie1.x =  560
sobitie1.y = 40
sobitie1.id="4"
```

```
elseif (event.phase == "began") and (sobitie1.id~="5") and (sobitie2.id~="5") and
(sobitie3.id~="5") and (sobitie4.id~="5") and (sobitie5.id~="5") then
      sobitie1 = display.newImageRect("five.png",160,80)
     sobitie1.x =  560
sobitie1.y = 40
sobitie1.id="5"
    end
    return true
end
sobitie1:addEventListener( "touch", onObjectTouch )


local function onObjectTouch( event )
    if (event.phase == "began") and (sobitie1.id~="1") and (sobitie2.id~="1") and
(sobitie3.id~="1") and (sobitie4.id~="1") and (sobitie5.id~="1") then
        sobitie2 = display.newImageRect("pervoe.png",160,80)
     sobitie2.x =  720
sobitie2.y = 40
sobitie2.id="1"

elseif (event.phase == "began") and (sobitie1.id~="2") and (sobitie2.id~="2") and
(sobitie3.id~="2") and (sobitie4.id~="2") and (sobitie5.id~="2") then
     sobitie2 = display.newImageRect("vtoroe.png",160,80)
     sobitie2.x =  720
sobitie2.y = 40
sobitie2.id="2"

elseif (event.phase == "began") and (sobitie1.id~="3") and (sobitie2.id~="3") and
(sobitie3.id~="3") and (sobitie4.id~="3") and (sobitie5.id~="3") then
     sobitie2 = display.newImageRect("3.png",160,80)
```

```
    sobitie2.x =  720
sobitie2.y = 40
sobitie2.id="3"

elseif (event.phase == "began") and (sobitie1.id~="4") and (sobitie2.id~="4") and
(sobitie3.id~="4") and (sobitie4.id~="4") and (sobitie5.id~="4") then
     sobitie2 = display.newImageRect("four.png",160,80)
    sobitie2.x =  720
sobitie2.y = 40
sobitie2.id="4"

elseif (event.phase == "began") and (sobitie1.id~="5") and (sobitie2.id~="5") and
(sobitie3.id~="5") and (sobitie4.id~="5") and (sobitie5.id~="5") then
     sobitie2 = display.newImageRect("five.png",160,80)
    sobitie2.x =  720
sobitie2.y = 40
sobitie2.id="5"
   end
   return true
end
sobitie2:addEventListener( "touch", onObjectTouch )



local function onObjectTouch( event )
   if (event.phase == "began") and (sobitie1.id~="1") and (sobitie2.id~="1") and
(sobitie3.id~="1") and (sobitie4.id~="1") and (sobitie5.id~="1") then
      sobitie3 = display.newImageRect("pervoe.png",160,80)
    sobitie3.x =  560
sobitie3.y = 120
```

```
sobitie3.id="1"

elseif (event.phase == "began") and (sobitie1.id~="2") and (sobitie2.id~="2") and
(sobitie3.id~="2") and (sobitie4.id~="2") and (sobitie5.id~="2") then
    sobitie3 = display.newImageRect("vtoroe.png",160,80)
    sobitie3.x =  560
sobitie3.y = 120
sobitie3.id="2"

elseif (event.phase == "began") and (sobitie1.id~="3") and (sobitie2.id~="3") and
(sobitie3.id~="3") and (sobitie4.id~="3") and (sobitie5.id~="3") then
    sobitie3 = display.newImageRect("3.png",160,80)
    sobitie3.x =  560
sobitie3.y = 120
sobitie3.id="3"

elseif (event.phase == "began") and (sobitie1.id~="4") and (sobitie2.id~="4") and
(sobitie3.id~="4") and (sobitie4.id~="4") and (sobitie5.id~="4") then
     sobitie3 = display.newImageRect("four.png",160,80)
    sobitie3.x =  560
sobitie3.y = 120
sobitie3.id="4"

elseif (event.phase == "began") and (sobitie1.id~="5") and (sobitie2.id~="5") and
(sobitie3.id~="5") and (sobitie4.id~="5") and (sobitie5.id~="5") then
     sobitie3 = display.newImageRect("five.png",160,80)
    sobitie3.x =  560
sobitie3.y = 120
sobitie3.id="5"
   end
```

```
    return true
end
sobitie3:addEventListener( "touch", onObjectTouch )



local function onObjectTouch( event )
    if (event.phase == "began") and (sobitie1.id~="1") and (sobitie2.id~="1") and
(sobitie3.id~="1") and (sobitie4.id~="1") and (sobitie5.id~="1") then
        sobitie4 = display.newImageRect("pervoe.png",160,80)
     sobitie4.x =  720
sobitie4.y = 120
sobitie4.id="1"

elseif (event.phase == "began") and (sobitie1.id~="2") and (sobitie2.id~="2") and
(sobitie3.id~="2") and (sobitie4.id~="2") and (sobitie5.id~="2") then
     sobitie4 = display.newImageRect("vtoroe.png",160,80)
     sobitie4.x =  720
sobitie4.y = 120
sobitie4.id="2"

elseif (event.phase == "began") and (sobitie1.id~="3") and (sobitie2.id~="3") and
(sobitie3.id~="3") and (sobitie4.id~="3") and (sobitie5.id~="3") then
     sobitie4 = display.newImageRect("3.png",160,80)
     sobitie4.x =  720
sobitie4.y = 120
sobitie4.id="3"

elseif (event.phase == "began") and (sobitie1.id~="4") and (sobitie2.id~="4") and
(sobitie3.id~="4") and (sobitie4.id~="4") and (sobitie5.id~="4") then
```

```
      sobitie4 = display.newImageRect("four.png",160,80)
     sobitie4.x =  720
sobitie4.y = 120
sobitie4.id="4"

elseif (event.phase == "began") and (sobitie1.id~="5") and (sobitie2.id~="5") and
(sobitie3.id~="5") and (sobitie4.id~="5") and (sobitie5.id~="5") then
      sobitie4 = display.newImageRect("five.png",160,80)
     sobitie4.x =  720
sobitie4.y = 120
sobitie4.id="5"
    end
    return true
end
sobitie4:addEventListener( "touch", onObjectTouch )



local function onObjectTouch( event )
    if (event.phase == "began") and (sobitie1.id~="1") and (sobitie2.id~="1") and
(sobitie3.id~="1") and (sobitie4.id~="1") and (sobitie5.id~="1") then
        sobitie5 = display.newImageRect("pervoe.png",160,80)
     sobitie5.x =  560
sobitie5.y = 200
sobitie5.id="1"

elseif (event.phase == "began") and (sobitie1.id~="2") and (sobitie2.id~="2") and
(sobitie3.id~="2") and (sobitie4.id~="2") and (sobitie5.id~="2") then
     sobitie5 = display.newImageRect("vtoroe.png",160,80)
     sobitie5.x =  560
```

```
sobitie5.y = 200
sobitie5.id="2"

elseif (event.phase == "began") and (sobitie1.id~="3") and (sobitie2.id~="3") and
(sobitie3.id~="3") and (sobitie4.id~="3") and (sobitie5.id~="3") then
    sobitie5 = display.newImageRect("3.png",160,80)
    sobitie5.x =  560
sobitie5.y = 200
sobitie5.id="3"

elseif (event.phase == "began") and (sobitie1.id~="4") and (sobitie2.id~="4") and
(sobitie3.id~="4") and (sobitie4.id~="4") and (sobitie5.id~="4") then
     sobitie5 = display.newImageRect("four.png",160,80)
    sobitie5.x =  560
sobitie5.y = 200
sobitie5.id="4"

elseif (event.phase == "began") and (sobitie1.id~="5") and (sobitie2.id~="5") and
(sobitie3.id~="5") and (sobitie4.id~="5") and (sobitie5.id~="5") then
     sobitie5 = display.newImageRect("five.png",160,80)
    sobitie5.x =  560
sobitie5.y = 200
sobitie5.id="5"
   end
   return true
end
sobitie5:addEventListener( "touch", onObjectTouch )



        --knopka stat
```

```
  local xrobot=robot.x
  local yrobot=robot.y


local start = display.newImageRect("start.png",320,80)
start.x=640
start.y=440
  --obraborka najatiya knopok
  function start:touch( event )
    if event.phase == "began" and sobitie1.id=="1" and sobitie2.id=="2" and
sobitie3.id=="3" and sobitie4.id=="4" and sobitie5.id=="5"  then

transition.to( robot, {  y=robot.y - 160 } )


     local function onCollision( self, event )



      if ( event.phase == "began" ) then
      transition.to( robot, {  x=robot.x - 80 } )
   end
end

point1.collision = onCollision
point1:addEventListener( "collision", point1 )

          local function onCollision( self, event )


      if ( event.phase == "began" ) then
```

```
    transition.to( robot, {  y=robot.y - 80 } )
  end
end

point2.collision = onCollision
point2:addEventListener( "collision", point2 )

local function onCollision( self, event )



    if ( event.phase == "began" ) then
    transition.to( robot, {  x=robot.x + 200 } )
  end
end

point3.collision = onCollision
point3:addEventListener( "collision", point3 )

local function onCollision( self, event )



    if ( event.phase == "began" ) then
    transition.to( robot, {  y=robot.y + 40 } )
  end
end

point4.collision = onCollision
point4:addEventListener( "collision", point4 )

local function onCollision( self, event )
```

```
      if ( event.phase == "began" ) then
        robot:removeSelf()
     fon:removeSelf()
     sobitie0:removeSelf()
     sobitie1:removeSelf()
     sobitie2:removeSelf()
     sobitie3:removeSelf()
     sobitie4:removeSelf()
     sobitie5:removeSelf()
     sobitie00:removeSelf()
     sobitie000:removeSelf()
     sobitie0000:removeSelf()
     sobitie00000:removeSelf()
      stena1:removeSelf()
     stena2:removeSelf()
     home:removeSelf()
     start:removeSelf()
      point1:removeSelf()
     point2:removeSelf()
     point3:removeSelf()
     point4:removeSelf()
     physics.stop()
      composer.removeScene( "level1" )

composer.loadScene( "menu" )
   end
end
```

```
home.collision = onCollision
home:addEventListener( "collision", home )



    end
    return true
end
start:addEventListener( "touch", start )

    -- Initialize the scene here.
    -- Example: add display objects to "sceneGroup", add touch listeners, etc.
end


-- "scene:show()"
function level1:show( event )

    local sceneGroup = self.view
    local phase = event.phase

    if ( phase == "will" ) then
        -- Called when the scene is still off screen (but is about to come on screen).
    elseif ( phase == "did" ) then
        -- Called when the scene is now on screen.
        -- Insert code here to make the scene come alive.
        -- Example: start timers, begin animation, play audio, etc.
    end
end
```

```
-- "scene:hide()"
function level1:hide( event )

    local sceneGroup = self.view
    local phase = event.phase

    if ( phase == "will" ) then
        -- Called when the scene is on screen (but is about to go off screen).
        -- Insert code here to "pause" the scene.
        -- Example: stop timers, stop animation, stop audio, etc.
    elseif ( phase == "did" ) then
        -- Called immediately after scene goes off screen.
    end
end


-- "scene:destroy()"
function level1:destroy( event )

    local sceneGroup = self.view

    -- Called prior to the removal of scene's view ("sceneGroup").
    -- Insert code here to clean up the scene.
    -- Example: remove display objects, save state, etc.
end


-- ---------------------------------------------------------------------------
```

```
-- Listener setup
level1:addEventListener( "create", level1 )
level1:addEventListener( "show", level1 )
level1:addEventListener( "hide", level1 )
level1:addEventListener( "destroy", level1 )

-- ---------------------------------------------------------------------------

return level1
```