Федеральное государственное автономное образовательное учреждение

высшего образования

«Волгоградский государственный университет»

институт Математики и информационных технологий

кафедра Информационных систем и компьютерного моделирования

Допустить работу к защите

Зав. каф. ИСКМ

\_\_\_\_\_\_\_\_\_\_\_\_\_А.В. Хоперсков

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 201\_\_ г.

Минаев Дмитрий Владимирович

**Разработка игры для мобильной платформы Android**

Отчет о научно-исследовательской работе

Студент Минаев Д.В \_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись)

Направление подготовки Информатика и вычислительная техника

Группа ИВТ-161

Научный руководитель к.ф.-м.н., доц. каф. ИСКМ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Кузьмин Н.М. |  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_ (дата, подпись) |
| Нормоконтролер |  | Бутенко М. А. |  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_ |

(дата, подпись)

Волгоград, 2019

Содержание

[Введение 3](#_Toc11631019)

[1. Анализ необходимости разработки приложения 4](#_Toc11631020)

[2. Анализ требований 5](#_Toc11631021)

[2.1 Информационно-логическая модель 5](#_Toc11631022)

[2.2 Анализ функциональных требований 5](#_Toc11631024)

[2.3 Требования к интерфейсу приложения 5](#_Toc11631025)

[2.4 Выбор исходного кода и языка программирования 6](#_Toc11631026)

[3. Прототипирование 7](#_Toc11631027)

[4. Создание имиджа приложения 8](#_Toc11631028)

[4.1 Создание названия приложения 8](#_Toc11631029)

[4.2 Создание интерфейса приложения 8](#_Toc11631030)

[5. Разработка 10](#_Toc11631031)

[6. Тестирование 15](#_Toc11631032)

[7. Представление проекта 18](#_Toc11631033)

[Заключение 20](#_Toc11631034)

[Листинг 21](#_Toc11631035)

[Список использованной литературы 28](#_Toc11631036)

# Введение

Основной целью данной научной работы является создание Android-приложения для расчета подкачки давления в шинах автомобиля.

Для достижения этой цели следует выполнить следующие задачи:

1. Провести анализ необходимости разработки приложения

2) Проанализировать требования к приложению

a) Информационно-логической модели

б) Проанализировать функциональные требования

в) Выбрать технологию создания

3) Создать прототип приложения

4) Создать имидж приложения

а) Придумать название приложения

б) Спроектировать интерфейс

5) Создать приложение используя выбранный язык программирования

6) Протестировать приложение

7) Представить проект

Описание каждого пункта задач будет описано ниже.

# 1 Анализ необходимости разработки приложения

Необходимость создания приложения зависит от нескольких факторов, и прежде чем начать разрабатывать проект необходимо удостовериться в том, что данное приложение будет востребовано, а также провести анализ конкурентно образующих проектов на рынке приложений Play Market’a.

Проанализировав рынок приложений Play Market, можно прийти к выводу, что при вводе запроса «Подкачка шин» выдаются приложения только по назначению расчета шин относительно дисков (смотрите рисунок 1). Следовательно, данная «ниша» свободна для разработки, а, следовательно, необходимость данного приложения оправдана.

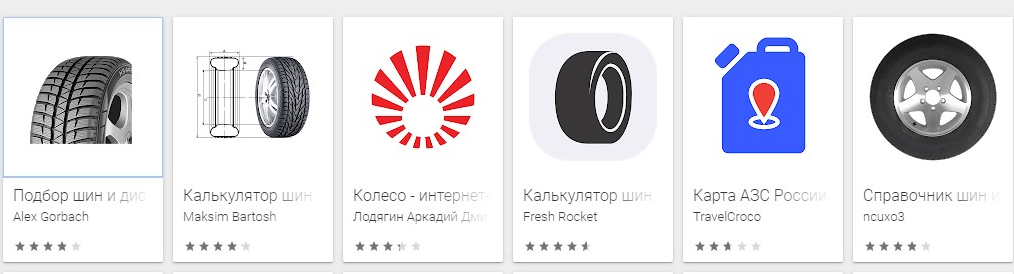


Рисунок 1 — Анализ возможных конкурентов приложения

# 2 Анализ требований

# 2.1 Информационно-логическая модель

Информационно-логическая модель отображает информационное описание работы приложения. Приложение на вход получает некоторые данные которые ввел пользователь, затем происходят вычисления. После получения результата, он выводится на экран. Информационно-логическая модель приложения представлена ниже на Рисунке 2.

# 

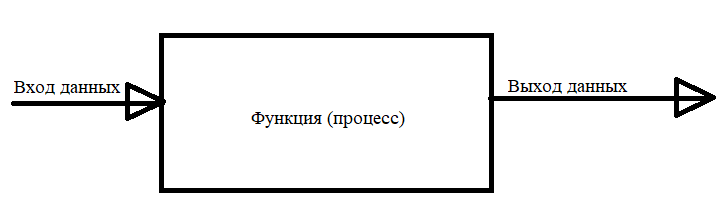


Рисунок 2 — Информационно-логическая модель приложения

# 2.2 Анализ функциональных требований

— Программа должна позволять вводить «Пользователю» (посредством ползунка на слайдере) нужные ему значения.

— Программа должна рассчитывать результат.

— Программа должна выводить полученный результат.

# 2.3 Требования к интерфейсу приложения

— Вверху на экране друг под другом должны быть расположены два слайдера, в которых «Пользователь» путем перемещения ползунка может выбрать ту или иную единицу размера шины.

— Ровно по середине экрана должны быть две кнопки «+» и «-», с левой и правой стороны соответственно, а между ними счетчик.

— Внизу экрана крупным шрифтом должен выводиться результат.

— В самом низу экрана должна быть размещена кнопка «рассчитать», но не занимать более 90% нижней части экрана.

— На заднем фоне должна присутствовать тематическая картинка на выбор разработчика.

# 2.4 Выбор текстового редактора и языка программирования

Так как данное приложение планировалась разрабатывать с помощью фреймворка Corona SDK, то на выбор предлагалось два языка программирования:

А) Язык программирования Lua.

Б) HTML+CSS+JavaScript

Фреймворк Corona SDK «из коробки» поставляется с собственным языком программирования Lua. Поэтому связка Corona SDK + Lua обладает следующими преимуществами:

— Кросс-платформенность. Написав один раз код, вы можете публиковать свой проект на самые разные устройства: IOS, Android, Mac, IPad, Windows и даже ТВ-приставок — Android TV, Apple TV.

* Язык программирования Lua. Да, это является преимуществом, так как Lua — это легкий, мощный, быстрый, а также легко изучаемый инструмент для разработки.
* Эмулятор. У Corona есть свой собственный эмулятор. Он имеет явное преимущество над эмулятором в Android Studio по скорости работы, даже на слабых устройствах.
* Тестирование в реальном времени. Вы можете писать код, сохранять и сразу же тестировать приложение, без необходимости его заново компилировать.
* Скорость и производительность. Corona, по заверению разработчиков, оптимизирована для ультра-быстрой графики OpenGL.
* Плагины. Сила плагинов в наше время играет достаточно важную роль. Плагины не обошли стороной и Corona. Плагины здесь имеются, как говорится, на ваш вкус и цвет.

Исходя из преимуществ связки Corona SDK и Lua, был выбран именно этот вариант разработки.

Данное приложение было разработано с использованием фреймворка Corona SDK (версии 2018.3326) и языком программирования Lua (версии 5.3.5).

В качестве текстового редактора был использован Atom (<https://atom.io/>). Данный редактор был выбран на мое усмотрение, так как давно им пользуюсь.

# 3 Создание прототипа

С соблюдением всех выше описанных технических указаний был создан прототип приложения (рисунок 3).



Рисунок 3 — прототип приложения

# 4 Создание имиджа приложения

# 4.1 Создание названия приложения

Создание название приложения производилось с использованием тематики приложения. Стоял выбор между «Wheel», «Wheels», «BarInWheels», «BarInWheel», «WheelsBar» и «WheelBar». После многочисленных раздумий, принято решение выбрать название «Wheel».

# 4.2 Создание интерфейса приложения

При создании приложения были использованы следующие компоненты:

1. Слайдер. Взят с сайта: https://itproger.com/course/corona-sdk/8 (смотрите рисунок 4).



Рисунок 4 — Слайдер

1. Кнопки «+» и «-». Картинки взяты с бесплатного сайта иконок: https://www.flaticon.com (смотрите рисунок 5).



Рисунок 5 — Кнопки «+» и «-»

1. Задний фон. Картинка взята с сайта: www.yandex.ru/images/cars/ (смотрите рисунок 6).



Рисунок 6 — Задний фон

1. Шрифты для текста взяты с сайта: https://fonts.google.com

# 5 Разработка

Разработка данного приложения начинается с главного файла в приложении это main.lua.

В качестве текстового редактора я буду использовать Atom (https://atom.io). Atom — это бесплатный текстовый редактор, который отличается поддержкой подсветки синтаксиса языка программирования Lua. Также, для более успешного и быстрого написания программы, в плагинах текстового редактора можно добавить себе поддержку сниппетов, которые по одной кнопке позволяют создавать примитивные конфигурации функций.

Сначала «нарисуем» прототип приложения. Для этого я использовал два вида фигур: круг и прямоугольник. Синтаксис создания данных фигур достаточно простой.

Для создания приложения мы используем библиотеку display, затем выбираем прямоугольник *newRoundedRect*. После этого вводим следующие параметры: группа, расположение по Х, расположение по Y, ширина прямоугольника, высота прямоугольника, закругленность краев (в процентах) соответственно. Все единицы измерения представлены в пикселях. Также использование *display.contentCenterX* и *display.contentCenterY* позволяет Corona SDK самой понимать где находится центр экрана, тем самым делая все фигуры адаптивными (смотрите листинг — A.1), тоже самое и с кругом *display.newCircle*: группа, расположение по Х, расположение по Y, радиус соответственно (смотрите листинг — A.2)

Листинг — А.1:

*display.newRoundedRect(zGroup,display.contentCenterX,*

*110, temp\_weight, 100, 15)*

Листинг — A.2:

*display.newCircle(zGroup, display.contentCenterX + 250, 160, 60):setFillColor(240/255)*

Также в конце каждого объекты мы можем указать его цвет использовав*: setFillColor(240/255, 0.8)*, где 240/255 — это цвет в палитре HSV, а *0.8* — это прозрачность в процентах (то есть 80%).

Далее мы будем использовать слайдер. Со слайдером все намного сложнее, так как наш скачанный слайдер не в .psd формате, и мы не можем вытащить оттуда каждую деталь. Благодаря встроенным функциям Corona SDK мы можем это сделать для этого достаточно импортировать библиотеку «widget» и полностью расписать встроенную конструкцию как указано в официальной документации http://docs.coronalabs.com/api/library/widget/newSlider.html (смотрите листинг — A.3).

Листинг — A.3:

*HeightSlider = widget.newSlider{*

*sheet = graphics.newImageSheet("corona\_slider\_image.png", optionsSlider),*

*leftFrame = 1,*

*middleFrame = 2,*

*rightFrame = 3,*

*fillFrame = 4,*

*handleFrame = 5,*

*frameWidth = 15,*

*frameHeight = 45,*

*handleWidth = 45,*

*handleHeight = 45,*

*top = 300,*

*left = 45,*

*width = 550,*

*height = 47,*

*orintation = "horizontal",*

*}*

После этого добавим переменную, которая будет меняться в зависимости от текущего положения ползунка. Для этого объявим глобальную переменную в коде приложения и локальную переменную в функции слайдера, которую будем присваивать глобальной после ее изменения (смотрите листинг — A.4).

Листинг — А.4:

*value = 100 \* (HeightValue - minHeightValue) / (maxHeightValue-minHeightValue),*

*listener = function(event)*

*HeightValue = math.round(minHeightValue + (maxHeightValue - minHeightValue) \**

*event.value / 100)*

*mHeight.text = HeightValue*

*end*

После этого еще добавим текст на слайдер путем использования библиотеки *display.newText*, в которую поместим нашу глобальную переменную.

Далее мы соединяем все вышесозданные объекты в один и получаем данный фрагмент (смотрите рисунок 7).

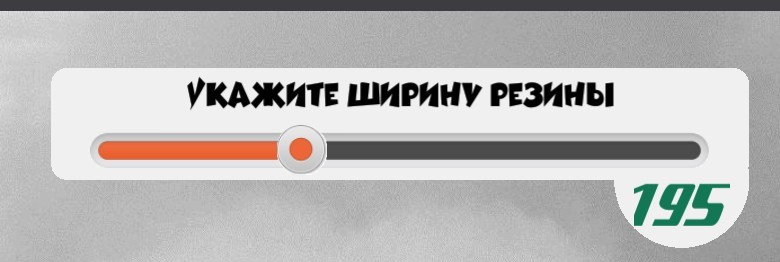


Рисунок 7 — Фрагмент готового приложения

По нашему прототипу таких слайдеров два. Поэтому мы дублируем код с изменением текста, координате положения по Y и переменных. Тем самым получая данный фрагмент готового приложения (смотрите рисунок 8).

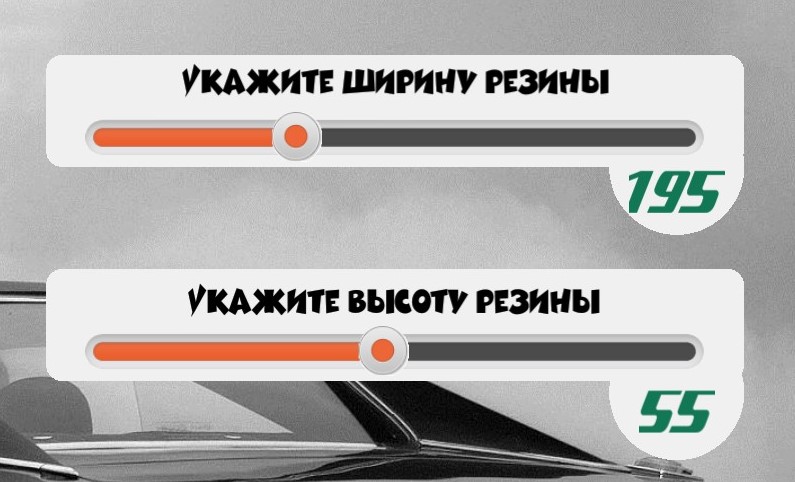


Рисунок 8 — Второй фрагмент готового приложения

Далее глядя на наш прототип нужно сделать две кнопки «+» и «-» и посреди них счетчик, который будет показывать диаметр диска в дюймах.

Для этого используем 4 прямоугольника: два для кнопок, один для счетчика и еще один для текста. Все прямоугольники серые и с коэффициентом прозрачности 0.8.

Для изменения размера диска я создал функцию добавления счетчика для каждой кнопки, причем в ней идет проверка минимального и максимального значения, чтобы не выходить за границы. Разницы между «+» и «-» нет, поэтому продемонстрирую только одну функцию (смотрите листинг — А.5).

Листинг — А.5:

*local size\_minus\_button = widget.newButton({*

*defaultFile = "image/size\_minus\_button.png",*

*overFile = "image/size\_minus\_button1.png",*

*onPress = function(event)*

*if (default\_size\_value > min\_size\_value) then*

*default\_size\_value = default\_size\_value - 1.0*

*size\_display.text = default\_size\_value*

*end*

*end*

*})*

Предпоследним шагом в создании приложения, является создание окна вывода. Делается это аналогично счетчика выше. Мы в текст передаем переменную результата.

К сожалению, я не смог найти способ, в котором результат меняется в реальном времени. Для этого я пошел другим путем. Я создал кнопку, которая при нажатии принимает в себя все введенные пользователем значения, считает их и передает на окно вывода (смотрите Листинг — А.6).

Листинг — А.6:

*local Button = widget.newButton {*

*parrent = Wheels,*

*shape = 'roundedRect',*

*radius = 10,*

*width =625, height = 80,*

*left = 10, top = 1050,*

*font = "16768",*

*label = "Рассчитать",*

*fontSize = 28,*

*labelColor = { default={ 1, 1, 1 }, over={ 0, 0, 0, 0.5 } },*

*onPress = function(event)*

*bum = math.round(size\_display.text \* (HeightValue + WidthValue)\* 0.75/ 11)/100*

*bar.text = bum*

*end*

*}:setFillColor(0.3)*

Создаем приложение. Указываем все необходимые данные, такие как: название приложения, иконка приложения, имя автора/название фирмы, выбор платформы Android/IOS. Спустя минут 15 получаем готовый APK-файл, который я буду тестировать в следующей главе.

# 6 Тестирование

Для тестирования я взял свой старый планшет Samsung GT-P5210 со следующими характеристиками:

* Операционная система Android
* Поддержка Android 4.2
* Процессор/чипсет Intel Atom Z2560 1600 МГц
* Количество ядер 2
* Оперативная память 1 Гб DDR3
* Встроенная память 16 Гб
* Поддержка карт памяти microSDXC, до 64 Гб

Сделаю маленькое замечание. На момент его покупки (2013 год) он был далеко не лучшим, что подтверждают тесты популярного бенчмарка Antutu (смотрите рисунок — 9).

По данным статистики Википедии (https://ru.wikipedia.org/wiki/История\_версий\_Android), первое место занимает Android 6.0 Marshmallow с результатом 16,9%, но по совокупности Android 7.0 и Android 7.1 вместе имеют уже 19,2%, таким образом считаем и восьмые версии. По итогу Android версии 8.0 и 8.1 имеют 28,3% на двоих. Из этого следует, что делать приложение для версии ниже Android 6.0 не имеет смысла, потому что, покупая новый телефон, вероятность нахождения на нем Android версии ниже чем 6.0 крайне мала.

Но вот так не задача, последнее обновление на планшете было в далеком 2013 году до версии 4.4 KitKat, что делает данный гаджет, мало приспособленный к тестам. Поэтому было принято решение перепрошить его на версию 6.0+.

Теперь посмотрим, что из этого получилось (смотрите рисунок 10) и начнем тестировать.



Рисунок 9 — Было

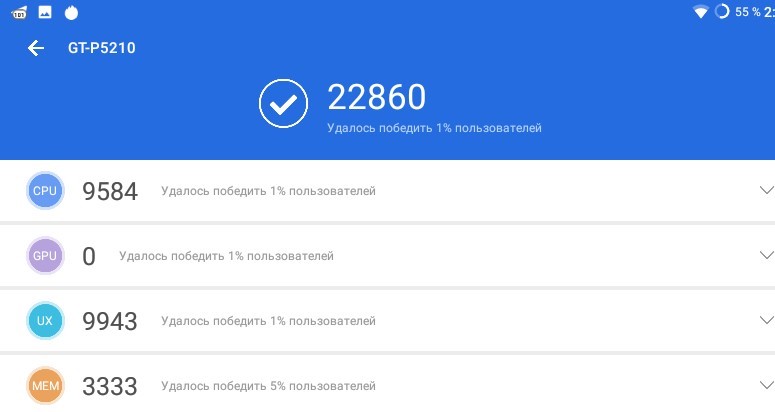


Рисунок 10 — Стало

Разница 3955 попугаев, что достаточно много. В прошивке нет Google services поэтому установить из Google Play не представляется возможным, но можно установить просто как приложение, перед этим разрешив установку из неизвестных источников.

После того как установили видим логотип и название фреймворка с помощью которого я создавал данное приложение (смотрите рисунок 11).



Рисунок 11 — Экран загрузки

Выбрав данные в своем приложении и нажав кнопку «Рассчитать», я получил результат. После этого взяв обыкновенный калькулятор, произвел расчёт вручную с использованием одних и тех же данных. Результаты полученные в ходе проведения тестирования были одинаковы. Из этого можно сделать вывод, что приложение работает так, как было задумано. Вылетов, зависаний и торможений приложения в ходу тестирования замечено не было.

# 7 Представление проекта

В этом разделе будут показаны результаты проделанной работы. Результаты в виде скриншотов, сделанные с различными входными данными (смотрите рисунок 12).

Рисунок 12 — Результаты работы

Почитав не малое количество статей на автомобильную тематику и посмотрев также не малое количество интервью с профессиональными спортсменами, я пришел к выводу, что одной точной универсальной формулы расчета давления в шинах нет. Спортсмены в такой дисциплине как «дрифт» используют повышенное давление в шинах, а вот в дисциплине «драг-рейсинг» наоборот — пониженное. Гонщики в кольцевых гонках используют пониженное давление сзади и повышенное спереди (это относится и к «Формула-3»). А обычным гражданским автомобилям при одинаковых колесах и шинах разные автоконцерны советуют разное значение давления в шинах. И это верное решение. Ведь давление в шинах зависит от огромного количества факторов начиная от погоды и заканчивая всевозможно неучтенных факторов. Однако, хочу заметить, что данное приложение создавалось для тех «Пользователей», у которых на трассе спустилось колесо, и они, достав запаску, введя данные в приложение могут спокойно накачать свои шины до значения, которое оно им выдало.

# Заключение

В данной работе был подробно рассмотрен метод моделирования и создания Android приложения на фреймфорке Corona SDK.

Описаны некоторые основные возможности Corona SDK, а также рассмотрены основные принципы его работы. А также было создано и протестировано Android-приложение.

В рамках выполнения научно-исследовательской работы изучено и продемонстрировано владение следующими компетенциями:

1. ОПК-1: способностью инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем.
2. ОПК-2: способностью осваивать методики использования программных средств для решения практических задач.
3. ОПК-4: способностью участвовать в настройке и наладке программно-аппаратных комплексов.
4. ОПК-5: способностью решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности.
5. ПК-1: способностью разрабатывать модели компонентов информационных систем, включая модели баз данных и модели интерфейсов "человек - электронно-вычислительная машина".
6. ПК-3: способностью обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности.

В дальнейшем планируется усовершенствование функциональности программы.

# Приложение

*local widget = require("widget")*

*minWidthValue = 125*

*maxWidthValue = 335*

*WidthValue = 195*

*-------------------------*

*minHeightValue = 30*

*maxHeightValue = 82*

*HeightValue = 55*

*-------------------------*

*min\_size\_value = 13*

*max\_size\_value = 21*

*default\_size\_value = 16*

*-------------------------*

*local main\_picture = display.newImage("main.png", 0, 550, 640, 1136)*

*display.setStatusBar(display.HiddenStatusBar)*

*display.setDefault("background", 60/255, 60/255, 65/255)*

*display.setDefault("fillColor", 0)*

*temp\_weight = display.contentWidth - 20*

*local zGroup = display.newGroup()*

*local Wheels = display.newGroup()*

*display.newRoundedRect(zGroup,display.contentCenterX,*

*110, temp\_weight, 100, 15)*

*:setFillColor(240/255)*

*display.newCircle(zGroup, display.contentCenterX + 250, 160, 60)*

*:setFillColor(240/255)*

*display.newText(zGroup, "Укажите ширину резины",*

*display.contentCenterX, 80, "16768", 28)*

*display.newRoundedRect(zGroup,display.contentCenterX,*

*300, temp\_weight, 100, 15)*

*:setFillColor(240/255)*

*display.newCircle(zGroup, display.contentCenterX + 250, 360, 60)*

*:setFillColor(240/255)*

*display.newText(zGroup, "Укажите высоту резины",*

*display.contentCenterX, 275, "16768", 28)*

*local mWidth = display.newText({*

*parrent = zGroup,*

*fontSize = 50,*

*text = WidthValue,*

*x = display.contentCenterX + 250,*

*y = 180 ,*

*font = "15348",*

*})*

*mWidth:setFillColor(17/255, 119/255, 85/255)*

*local mHeight = display.newText({*

*parrent = zGroup,*

*fontSize = 50,*

*text = HeightValue,*

*x = display.contentCenterX + 250,*

*y = 375 ,*

*font = "15348",*

*})*

*mHeight:setFillColor(17/255, 119/255, 85/255)*

*local optionsSlider = {*

*frames = {*

*{x = 0, y = 0, width = 15, height = 45},*

*{x = 16, y = 0, width = 130, height = 45},*

*{x = 332, y = 0, width = 15, height = 45},*

*{x = 153, y = 0, width = 15, height = 45},*

*{x = 353, y = 0, width = 47, height = 47},*

*},*

*SContentWidth = 400,*

*SContentHeight = 45,*

*}*

*WidthSlider = widget.newSlider{*

*sheet = graphics.newImageSheet("corona\_slider\_image.png", optionsSlider),*

*leftFrame = 1,*

*middleFrame = 2,*

*rightFrame = 3,*

*fillFrame = 4,*

*handleFrame = 5,*

*frameWidth = 15,*

*frameHeight = 45,*

*handleWidth = 45,*

*handleHeight = 45,*

*top = 110,*

*left = 45,*

*width = 550,*

*height = 47,*

*orintation = "horizontal",*

*value = 100 \* (WidthValue - minWidthValue) / (maxWidthValue-minWidthValue),*

*listener = function(event)*

*WidthValue = math.round(minWidthValue + (maxWidthValue - minWidthValue) \**

*event.value / 100)*

*mWidth.text = WidthValue*

*end*

*}*

*HeightSlider = widget.newSlider{*

*sheet = graphics.newImageSheet("corona\_slider\_image.png", optionsSlider),*

*leftFrame = 1,*

*middleFrame = 2,*

*rightFrame = 3,*

*fillFrame = 4,*

*handleFrame = 5,*

*frameWidth = 15,*

*frameHeight = 45,*

*handleWidth = 45,*

*handleHeight = 45,*

*top = 300,*

*left = 45,*

*width = 550,*

*height = 47,*

*orintation = "horizontal",*

*value = 100 \* (HeightValue - minHeightValue) / (maxHeightValue-minHeightValue),*

*listener = function(event)*

*HeightValue = math.round(minHeightValue + (maxHeightValue - minHeightValue) \**

*event.value / 100)*

*mHeight.text = HeightValue*

*end*

*}*

*--//////Size wheels//////*

*-------------------------------------------*

*display.newRoundedRect(Wheels, display.contentCenterX - 200,*

*600, 150, 150, 10):setFillColor( 0.3, 0.8 )*

*display.newRoundedRect(Wheels, display.contentCenterX + 200,*

*600, 150, 150, 10):setFillColor( 0.3, 0.8 )*

*display.newRoundedRect(Wheels, display.contentCenterX,*

*600, 150, 150, 10):setFillColor( 0.3, 0.8 )*

*-- size\_display = display.newText(Wheels, default\_size\_value, display.contentCenterX, 600, "16768", 100)*

*-- :setFillColor(0.8)*

*local size\_display = display.newText{*

*text = default\_size\_value,*

*x = display.contentCenterX,*

*y = 600,*

*font = "16768",*

*fontSize = 100*

*}*

*display.newText{*

*text = "Какой размер диска?",*

*x = display.contentCenterX,*

*y = 500,*

*font = "16768",*

*fontSize = 30*

*}:setFillColor(1)*

*local size\_minus\_button = widget.newButton({*

*defaultFile = "image/size\_minus\_button.png",*

*overFile = "image/size\_minus\_button1.png",*

*onPress = function(event)*

*if (default\_size\_value > min\_size\_value) then*

*default\_size\_value = default\_size\_value - 1.0*

*size\_display.text = default\_size\_value*

*end*

*end*

*})*

*size\_minus\_button.x = display.contentCenterX - 200*

*size\_minus\_button.y = 600*

*local size\_plus\_button = widget.newButton({*

*defaultFile = "image/size\_plus\_button.png",*

*overFile = "image/size\_plus\_button1.png",*

*onPress = function(event)*

*if (default\_size\_value < max\_size\_value) then*

*default\_size\_value = default\_size\_value + 1*

*size\_display.text = default\_size\_value*

*end*

*end*

*})*

*size\_plus\_button.y = 600*

*size\_plus\_button.x = display.contentCenterX + 200*

*---------------------------------------------*

*--/////////////Result//////////---------*

*display.newRoundedRect(Wheels, display.contentCenterX,*

*800, 250, 50, 5):setFillColor( 0.3, 0.8 )*

*display.newText(Wheels,"Результат", display.contentCenterX, 800, "16768", 30)*

*display.newText("Бар", display.contentCenterX + 100, 900, "16768",50):setFillColor(1)*

*display.newRoundedRect(Wheels, display.contentCenterX,*

*900, 350, 150, 10):setFillColor( 0.3, 0.8 )*

*----------------------------------------*

*-- display.newText( Wheels,mb ,display.contentCenterX - 50, 900, native.systemFont, 50):setFillColor(1)*

*bar\_default = 2.73*

*bar = display.newText( bar\_default, display.contentCenterX - 50, 900, "123", 50)*

*local Button = widget.newButton {*

*parrent = Wheels,*

*shape = 'roundedRect',*

*radius = 10,*

*width =625, height = 80,*

*left = 10, top = 1050,*

*font = "16768",*

*label = "Рассчитать",*

*fontSize = 28,*

*labelColor = { default={ 1, 1, 1 }, over={ 0, 0, 0, 0.5 } },*

*onPress = function(event)*

*bum = math.round(size\_display.text \* (HeightValue + WidthValue)\* 0.75/ 11)/100*

*bar.text = bum*

*end*

*}:setFillColor(0.3)*

# Список использованной литературы

1. Кнут, Д.Э. Искусство программирования (Том 1. Основные алгоритмы) 2000. - **947** c.

2. Крёнке, Д Теория и практика построения баз банных: 8-е избрание/ Д. Крёнке. - СПб, Питер, 2006, - 542 с.

3.  Кнут, Д.Э. Искусство программирования (Том 2. Получисленные алгоритмы) 2000. - **185** c.

4. Шандров, Б.В. Технические средства автоматизации: учебник для студентов высших учебных заведений/ Шандров Б.В. Москва; "Академия", 2007. - 310 с.

5. Волков, О.И., Девяткин, О.В. Экономика предприятия фирмы/ О.И. Волков, Москва, издательство инфра Ф 2008, 269c.

6. Агафонов, В.Н. Логическое программирование / В.Н. Агафонов. - М.: [не указано], **2017**. - **314** c.

7. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. - М.: Мир, **2016**. - 360 c.

8. https://docs.coronalabs.com/api/index.html Официальная документация Corona Sdk

9. https://developer.android.com/studio Официальная документация по архитектуре Android

10. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих 2018, 288 с

11. Фохт, Д. Проектирование и программная реализация систем на персональных ЭВМ/ Д. Фохт СПб.: БХВ - Петербург, 2005. - 96 с.

12. Четвериков, В.Н. Базы и банки банных / Четвериков, В.Н. СПб; БХВ-Петербург, 2005. - 1196 с.

13. https://itproger.com/course/corona-sdk Уроки по Corona SDK

14. https://github.com/coronalabs Официальный репозиторий Corona SDK

15. https://corona-renderer.com Официальный мануал по разработке приложений в Corona SDK.

16. Michelle M. Fernandez, Corona SDK Mobile Game Development, 2012, 408 с

17. Daniel Williams, Corona SDK Application Design, 2013, 98 с

18. https://habr.com/ru/post/344304/ Уроки по Corona SDK

19. Скотт, Т. Основы программирования. Курс программированного обучения / Т. Скотт. - М.: Советское радио, 2016. - 490 c

20. Голицына, О.Л. Основы алгоритмизации и программирования: Учебное пособие / О.Л. Голицына, И.И. Попов. - М.: Форум; Издание 2-е, 2015. - 432 c.

21. Майер, Рето Android 2. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер. - М.: "Издательство "Эксмо", 2011. - 672 c.

22. Майер, Рето Android 4. Программирование приложений для планшетных компьютеров и смартфонов / Рето Майер. - М.: Эксмо, 2013. - 816 c.

23. П. Дейтел, Х. Дейтел, А. Уолд. Android для разработчиков. 3-е издание (2016, PDF)

24. Брайан Харди, Билл Филлипс,Крис Стюарт, Кристин Марсикано. Программирование под Android. 2-е издание (2016, PDF)

25. Дж. Блэйк Мик, Зигард Медникс, Лайрд Дорнин, Масуми Накамура Программирование под Android СПб.: Питер, 2012