Санкт – Петербургский Государственный Университет

Физический факультет

Кафедра вычислительной физики



**Кузьмин Николай Витальевич**

**Кросс-компиляция компонентов моделей виртуальных лабораторий программного комплекса BARSIC в модели Android**

**Cross-compilation of components of BARSIC virtual laboratories models to Android models**

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Научный руководитель

к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_\_ Монахов В. В.

Рецензент

к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_\_ Смирнов А. В.

**Санкт-Петербург**

**2017**

Оглавление

[Введение 3](#_Toc483482463)

[1. Система интернет-олимпиад по физике и модели виртуальных лабораторий BARSIC 7](#_Toc483482464)

[1.1 Программный комплекс distolymp 7](#_Toc483482465)

[1.2 Модели виртуальных лабораторий BARSIC и их взаимодействие с сервером 8](#_Toc483482466)

[1.3 Пути переноса моделей виртуальных лабораторий BARSIC на платформу Android 9](#_Toc483482467)

[2. Написание Android-моделей виртуальных лабораторий на языке Java 10](#_Toc483482468)

[2.1 Структура проекта 10](#_Toc483482469)

[2.2 Библиотека ABarsicLib 11](#_Toc483482470)

[2.3 Парсер выражений 14](#_Toc483482471)

[2.4 Модель виртуальной лаборатории «Тележка на наклонном рельсе» 15](#_Toc483482472)

[2.5 Проигрыватель моделей виртуальных лабораторий ABarsicPlayer 17](#_Toc483482473)

[3. Система кросс-компиляции моделей виртуальных лабораторий BARSIC на платформу Android 26](#_Toc483482474)

[3.1 Структура кросс-компиляции 26](#_Toc483482475)

[3.2 Библиотека ABarsicLib в системе кросс-компиляции 27](#_Toc483482476)

[3.3 Схема запуска кросс-компилированных моделей 28](#_Toc483482477)

[3.4 Оболочечные компоненты 29](#_Toc483482478)

[4. Объединение проектов для платформы Android 33](#_Toc483482479)

[4.1 Объединение проектов 33](#_Toc483482480)

[4.2 Тестирование объединенного проекта 34](#_Toc483482481)

[Выводы 35](#_Toc483482482)

[Литература 36](#_Toc483482483)

[Приложения 39](#_Toc483482484)

[Листинг 3. Код измененных методов класса Button 39](#_Toc483482485)

[Листинг 4. Код измененных методов класса TextEdit 40](#_Toc483482486)

[Листинг 5. Код измененных методов класса Subwindow 43](#_Toc483482487)

# Введение

Начиная с 2005 года, СПбГУ проводит интернет-олимпиаду школьников по физике [1] среди 7-11 классов. Интернет-олимпиада школьников по физике с каждым днём становятся всё более востребованной по следующим причинам.

В рамках любой олимпиады школьникам следует предложить решить задачи и выполнить лабораторные работы. Задачи служат для проверки теоретических знаний. Экспериментальные умения и навыки проверяются в лабораторных работах. Умения и навыки по проведению физических экспериментов очень важны для будущего физика-исследователя.

Однако на практике, в абсолютном большинстве олимпиад по физике требуется решить только задачи, а экспериментальная часть в лучшем случае может присутствовать лишь на заключительном этапе. Такое положение дел связано с большой сложностью обеспечения участников экспериментальными установками, ведь для проведения массовых олимпиад практически нереально обеспечить для тысяч человек однотипное оборудование, необходимые площади, персонал, обслуживающий это оборудование (сборку, настройку, наблюдение за правильностью использования).

Вопрос о проведении экспериментальной части олимпиады полностью снимается, если использовать вместо настоящих установок их компьютерные модели. Виртуальные физические установки, в отличие от настоящих, не требуют значительных материальных затрат и не могут выйти из строя. Программное обеспечение виртуальных лабораторий разрабатывается так, чтобы поведение виртуальных установок не отличалось от поведения их реальных аналогов.

Дальнейшее развитие интернет-олимпиады школьников по физике зависит от возможности работы программного обеспечения олимпиады на различных программных платформах. На данный момент прохождение заданий олимпиады возможно только на настольных компьютерах под управлением операционной системы семейства Windows, либо под управлением Linux с помощью дополнительного ПО Wine и установки браузерного ядра Gecko. Задействование мобильных устройств поможет значительно расширить аудиторию олимпиады. Согласно статистике, представленной на сайте Рейтинг@Mail.ru [2, 3], по состоянию на 1 мая 2017 года более 53% пользователей выходят в интернет с мобильных устройств. Более 76% мобильных устройств работают под управлением операционной системы Android.

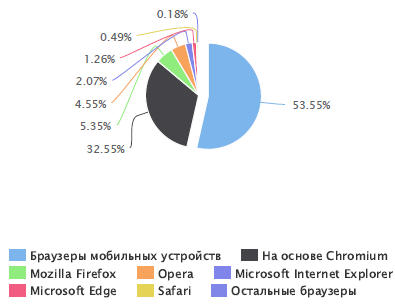


Рис. 1. Статистика выхода в интернет из различных браузеров.  
Рейтинг@mail.ru.

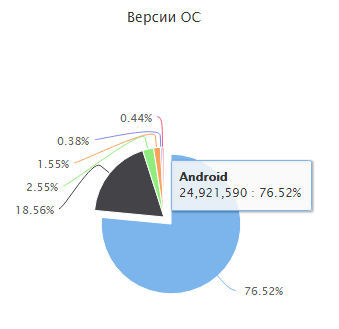


Рис. 2. Статистика операционных систем мобильных устройств.  
Рейтинг@mail.ru.

Встал вопрос о переносе интернет-олимпиады по физике на платформу Android. Ранее Монаховым В.В. и др. [4] рассматривалась возможность создания моделей виртуальных лабораторий на языке Java без использования среды BARSIC. Программы для операционной системы Android пишут на языке Java, однако нет совместимости двоичного кода виртуальной Java-машины и виртуальной машины Android. Оказалось, что размеры файлов моделей получались около 5 Мб и большая часть кода дублировалась. Другой подход состоял в вынесении всей повторяющейся части кода в приложение, однократно загружаемое с сервера и устанавливаемое на клиентское устройство. Модели виртуальных лабораторий содержали только специфическую для данного приложения часть кода. Для разработки кодов моделей использовалось два способа. Первый состоял в написании моделей непосредственно на Java. Второй, при помощи системы кросс-компиляции (разрабатываемой Максимовым М.А. [5]) из исходных кодов, написанных на языке BARSIC, в коды Java.

В связи с этим целью данной работы являлась разработка в программном комплексе Distolymp системы, позволяющей работать на устройствах Android с кросс-компилированными моделями виртуальных лабораторий.

Для достижения поставленной цели решались следующие задачи:

* Разработка структуры проекта, позволяющего работать как с кросс-компилированными, так и разработанными непосредственно на Java моделями;
* Доработка «проигрывателя» моделей виртуальных лабораторий ABarsicPlayer;
* Доработка модели виртуальной лаборатории «Тележка на наклонном рельсе»;
* Усовершенствование библиотеки ABarsicLib;
* Объединение проектов проигрывателя моделей виртуальных лабораторий ABarsicPlayer, библиотеки ABarsicLib, моделей «Тележка на наклонном рельсе» и «Электрическая лаборатория» и парсера выражений (калькулятор);
* Тестирование объединенного проекта.

# 1. Система интернет-олимпиад по физике и модели виртуальных лабораторий BARSIC

## **1.1 Программный комплекс distolymp**

В настоящее время создание компьютерных моделей, имитирующих реальный эксперимент, реализовано с помощью программного обеспечения BARSIC [6].

BARSIC (Business And Research Scientific Interactive Calculator) - это язык программирования интегрированной среды, также называющейся BARSIC. В основном программный комплекс применяется в области физики и создания программ для численных расчетов.

Программный комплекс Distolymp разрабатывался для проведения интернет-олимпиады школьников по физике. Основу олимпиады составляют модели виртуальных лабораторий, написанные на языке BARSIC. Интернет-олимпиада является мощным средством профессиональной ориентации – за последние годы около половины поступивших на физический факультет СПбГУ абитуриентов являлись её участниками [8].

Программный комплекс Distolymp основан на клиент-серверной модели взаимодействия. Сервер передает клиенту модели виртуальных лабораторий вместе с параметрами. Клиент взаимодействует с полученными моделями в специальном «проигрывателе» BarsicPlayer.

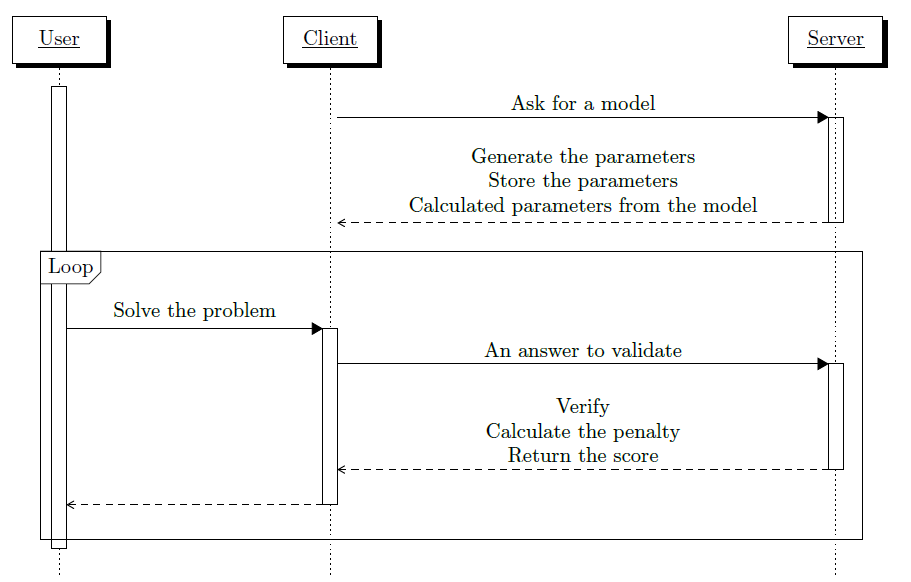


Рис. 3. Диаграмма клиент-серверного взаимодействия [5].

## **1.2 Модели виртуальных лабораторий BARSIC и их взаимодействие с сервером**

Исходные коды моделей виртуальных лабораторий — это проекты BARSIC. Проект BARSIC состоит из bpr-, brm- и bfm-файлов. bpr-файл – это файл заголовка проекта. В нем содержатся ссылки на модули – brm-файлы. К brm-файлу могут прилагаться bfm-файлы, описывающие форму. Разделение на brm- и bfm-файлы условное. Отличие brm- и bfm-файлов в том, что исходные коды bfm-файлов генерируются автоматически, а код brm-файлов разрабатывается пользователем.

Файл заголовка проекта BARSIC (bpr-файл) состоит из ссылок на модули из файлов с расширением brm, к каждому из которых может прилагаться несколько bfm-файлов c описанием формы. brm- и bfm-файлы имеют одинаковый синтаксис, при этом разделение является лишь логическим. Данный шаг был сделан с целью отделения автоматически сгенерированного кода форм от исходных кодов модулей, разрабатываемых пользователем [5].

Параметры модели изначально задаются в самой модели. При загрузке модели с сервера они меняются в соответствии с программой-сценарием («скриптом»), написанным для этой модели со стороны сервера.

## **1.3 Пути переноса моделей виртуальных лабораторий BARSIC на платформу Android**

Ранее Максимовым М.А. была предложена схема кросс-компиляции в клиент-серверную систему виртуальных лабораторий и реализован прототип кроссплатформенного транслятора языка BARSIC в Java и Dalvik/ART [5]. Мартынюком С.А. были разработаны компоненты библиотеки ABarsicLib, среди которых: базовый класс модели, средства рисования графическими примитивами и построения диаграмм, графическое подокно, кнопка, служебная система управления выполнением обработчиков событий [9].

На момент начала настоящей работы требовалось дальнейшее развитие библиотеки ABarsicLib, совершенствование ее существующих компонентов, доработка моделей виртуальных лабораторий и согласование с «проигрывателем» ABarsicPlayer.

# 2. Написание Android-моделей виртуальных лабораторий на языке Java

## **2.1 Структура проекта**

Работа выполнялась при помощи интегрированной среды разработки Android Studio (v. 2.3.2). В нее включены эмулятор устройств (Android Emulator, Intel x86 Emulator Accelerator (HAXM)) и все необходимые библиотеки. Приложение рассчитано на использование на устройствах с версией ОС Android 5.0 (API 21) и выше.

Проект состоит из нескольких основных частей:

* ABarsicPlayer – аналог проигрывателя BARSIC на платформе Windows;
* Библиотека ABarsicLib, содержащая в себе базовый класс модели виртуальной лаборатории, средства рисования графическими примитивами, инструменты для построения диаграмм и компоненты пользовательского интерфейса, необходимые для взаимодействия с моделями;
* Модели виртуальных лабораторий – «Тележка на наклонном рельсе» и «Электрическая лаборатория»;
* Парсер выражений – калькулятор.

Одной из задач данной работы было объединение составляющих в единый проект. Взаимодействие частей программного комплекса происходит следующим образом. Пользователь устанавливает на своем Android-устройстве приложение-клиент, которое включает в себя проигрыватель виртуальных лабораторий ABarsicPlayer. Проигрыватель ABarsicPlayer, в свою очередь, включает в себя библиотеку ABarsicLib. Пользователь проходит регистрацию, аутентификацию на сайте интернет-олимпиады и приступает к выполнению заданий. Задания олимпиады состоят из текста задания, модели виртуальной лаборатории и отчета. Модели виртуальных лабораторий используют необходимые им классы библиотеки ABarsicLib. Далее более подробно рассмотрим каждую из частей проекта.

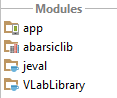


Рис. 4. Структура проекта.

## **2.2 Библиотека ABarsicLib**

Библиотека ABarsicLib создавалась Сергеем Мартынюком [9] с целью объединения минимального набора компонент, необходимого для разработки моделей виртуальных лабораторий и создания базового класса модели.

Сергеем Мартынюком [9] было предложено строить иерархию классов таким образом, чтобы классы моделей виртуальных лабораторий наследовались от базового класса Model, содержащегося в библиотеке. Однако предложенная им реализация данного класса как наследника класса Fragment создала проблемы при реализации моделей виртуальных лабораторий со структурой, отличной от модели ModelCar. В рамках данной работы автором было предложено разрабатывать классы моделей виртуальных лабораторий «Тележка на наклонном рельсе» (ModelCar) и «Электрическая лаборатория» (ModelElectric), наследуя их от абстрактного класса Model. Благодаря такой схеме наследования становится возможным загрузка данных моделей с сервера с помощью проигрывателя ABarsicPlayer. Модели виртуальных лабораторий могут подключаться к проигрывателю как независимые модули.

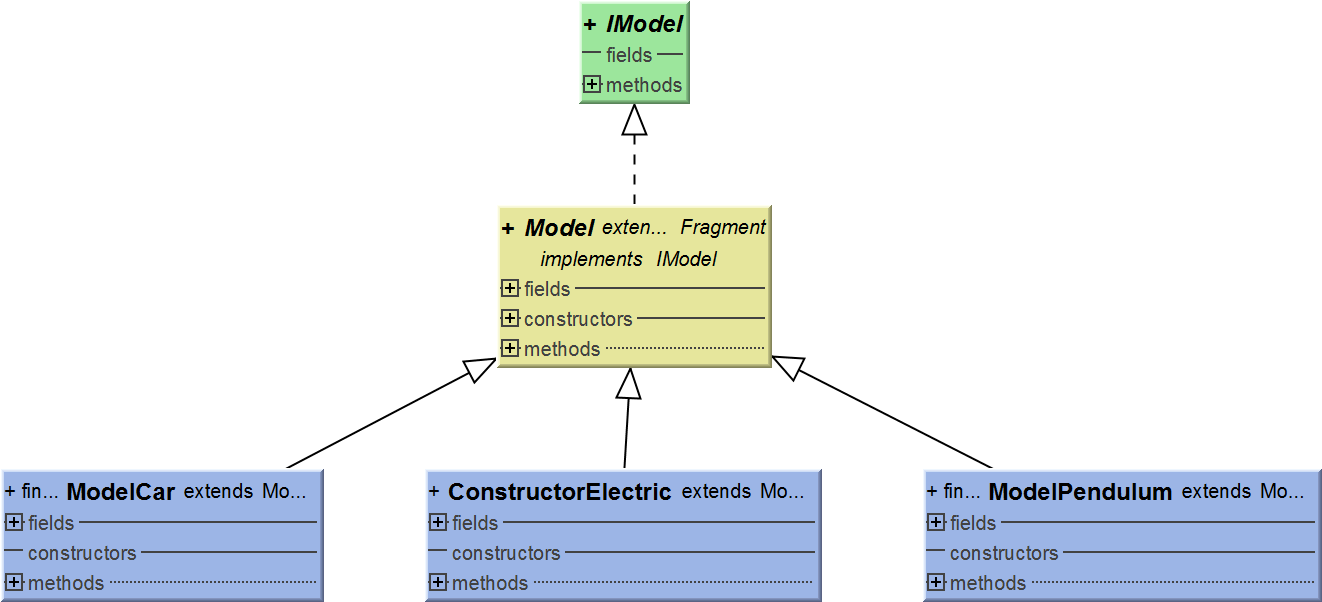


Рис. 5. Структура наследования моделей от базового класса Model.

В проекте доступно два класса MainActivity. Первый из них, находится в пакете ru.spbu.barsicmodel. Он служит для создания моделей виртуальных лабораторий [Листинг 1]. С его помощью возможно создать независимое Android приложение, которое впоследствии может быть загружено с сервера Distolymp и подключено проигрывателем ABarsicPlayer.

Листинг 1. Код класса ru.spbu.barsicmodel.MainActivity

**package** ru.spbu.barsicmodel;  
  
**import** android.app.Activity;  
**import** android.content.pm.ActivityInfo;  
**import** android.os.Bundle;  
**import** android.util.Log;  
  
**import** ru.spbu.abarsic.Model;  
  
**public final class** MainActivity **extends** Activity {  
  
 @Override  
 **protected void** onCreate(Bundle state) {  
 **super**.onCreate(state);  
 Log.*d*(**"state = "**, **""**+state);  
 **if** (state == **null**) {  
 setRequestedOrientation(ActivityInfo.***SCREEN\_ORIENTATION\_LANDSCAPE***);  
 Model model = **new** ModelCar();  
 *//Model model = new ConstructorElectric();  
 //Model model = new ModelPendulum();* getFragmentManager().beginTransaction().add(android.R.id.***content***, model).commit();  
 }  
 }  
}

Второй класс MainActivity находится в пакете ru.spbu.abarsic.activities и отвечает за создание интерфейса приложения проигрывателя ABarsicPlayer и будет описан ниже. Переключение между этими классами происходит в файле AndroidManidest.xml в блоке кода <**activity ...** </**activity**>, позволяя работать в одном проекте над моделями виртуальных лабораторий, усовершенствованием библиотеки ABarsicLib и проигрывателя ABarsicPlayer. Такое объединение частей проекта позволит в дальнейшем избежать конфликтов их совместимости между собой.

Работа с графикой в библиотеке ABarsicLib, написанной Сергеем Мартынюком [9], организована с помощью объектов graph, plot, subwindow, находящихся в классе виртуальной лаборатории. Объекты graph, plot, subwindow отвечают за рисование графическими примитивами, построение диаграмм и управление параметрами вывода графики виртуальных лабораторий соответственно.

В данной работе автором усовершенствованы компоненты пользовательского интерфейса, представленные кнопкой (класс Button), полем редактирования текста (класс TextEdit) и текстовой панелью (класс TextArea), а также графическое подокно (класс Subwindow). В этих классах автором написана полноценная реализация для методов-заглушек, у которых в работе С. Мартынюка была сделана пустая реализация, и усовершенствована реализация методов, у которых была реализация, но она не обеспечивала необходимую функциональность моделей, что позволило корректно отображать разрабатываемые в данный момент модели «Тележка на наклонном рельсе» и «Электрическая лаборатория» в соответствии с моделями, на языке BARSIC.

Основное внимание обращалось на разработку общей структуры проекта, дающей возможность работы с различными виртуальными лабораториями. При этом были реализованы только те Android-компоненты, являющиеся оболочками над компонентами BARSIC, которые были необходимы для кросс-компиляции простейшей модели ModelCar. Для поддержки на платформе Android кросс-компилированного кода в дальнейшем требуется обеспечить реализацию таких Android-компонентов, являющихся оболочками над компонентами BARSIC, как: метка, флажок, переключатель, группа кнопок, текстовая таблица, прокручиваемый список, панель. Также требуется усовершенствование метода onTouchEvent в классе Subwindow, позволяющего увеличивать показываемое в графическом подокне изображение (аналог лупы в приложениях BARISC на Windows). Важным аспектом последующей работы является понижение доступного уровня API для работы приложения на устройствах с ОС Android версии младше 5.0.

## **2.3 Парсер выражений**

Для поддержки вычислений в калькуляторе моделей виртуальных лабораторий Еленой Монаховой была разработана библиотека парсера выражений (JEvalParser) [10]. В рамках данной работы она была включена автором в библиотеку ABarsicLib. Автором был добавлен калькулятор в модель виртуальной лаборатории «Тележка на наклонном рельсе». Парсер позволяет вычислять значения функции, находящейся в строке ввода.

Парсер выражений может быть применен в других моделях виртуальных лабораторий.

Для того, чтобы начать использование парсера, необходимо создать экземпляр класса Parser.

## **2.4 Модель виртуальной лаборатории «Тележка на наклонном рельсе»**

Разрабатываемый автором класс модели виртуальной лаборатории «Тележка на наклонном рельсе» ModelCar был унаследован от абстрактного класса Model для того, чтобы он мог быть загружен и подключен проигрывателем ABarsicPlayer.

В данный класс автором добавлен парсер выражений, описанный выше. С его помощью был реализован калькулятор, аналогичный калькулятору в модели на языке BARSIC. Он позволяет получить такую же точность вычислений выражений с плавающей точкой (15 знаков после запятой). Корректно считываются вводимые числа как в формате с точкой, так и с запятой между целой и дробной частью, математические функции и некоторые константы (число Пи).

Также автором были добавлены все недостающие компоненты, присущие оригинальному приложению, такие как функциональные кнопки, линейка со шкалой, блокнот.

Была реализована загрузка параметров с сервера (класс paramVariableSet). Она устроена таким образом, что даже при перехвате параметров участником олимпиады, он не сможет извлечь из них полезной информации. Псевдослучайный набор параметров для модели генерируется сервером.

Параметры, загруженные с сервера, устанавливают значения переменных, использующихся в модели. Шкалы линейки и графика, начальная координата и скорость автоматически рассчитываются исходя из загруженных параметров.

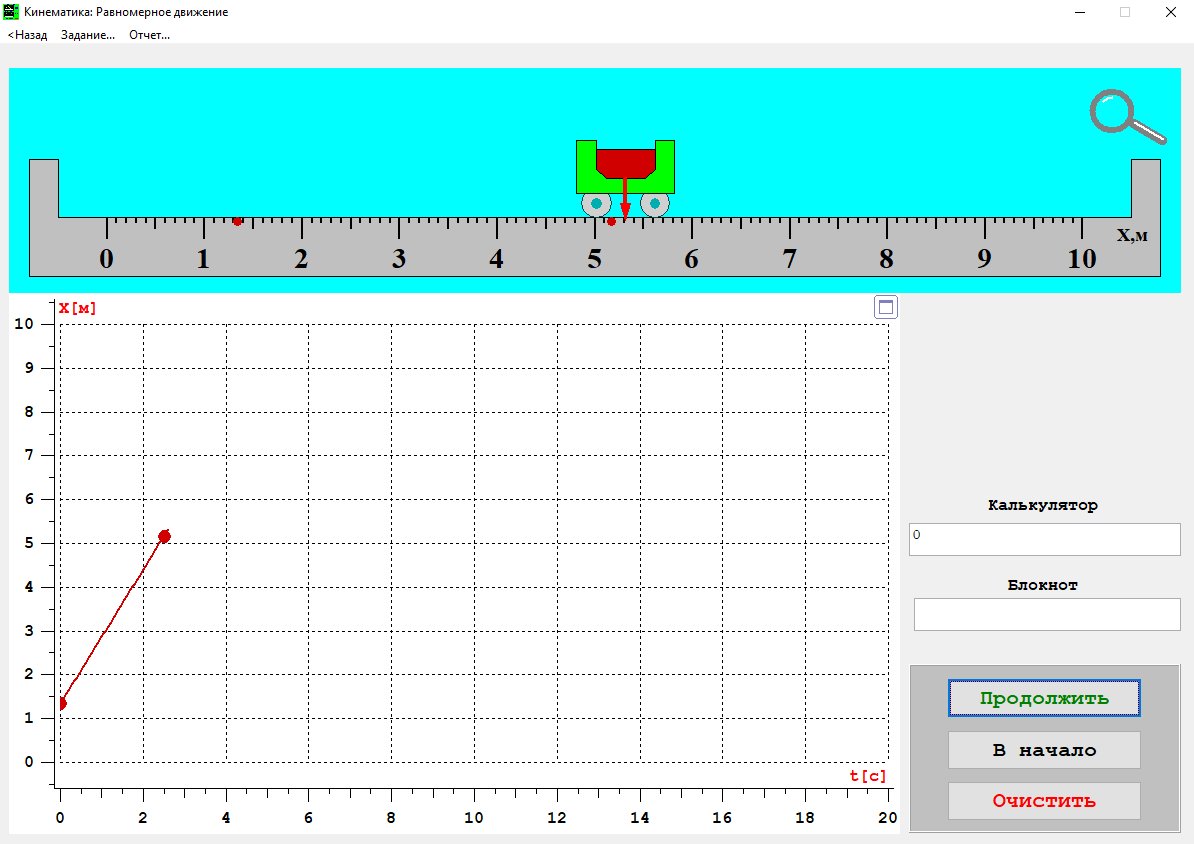


Рис. 6. Модель виртуальной лаборатории "Тележка на наклонном рельсе" на языке BARSIC.

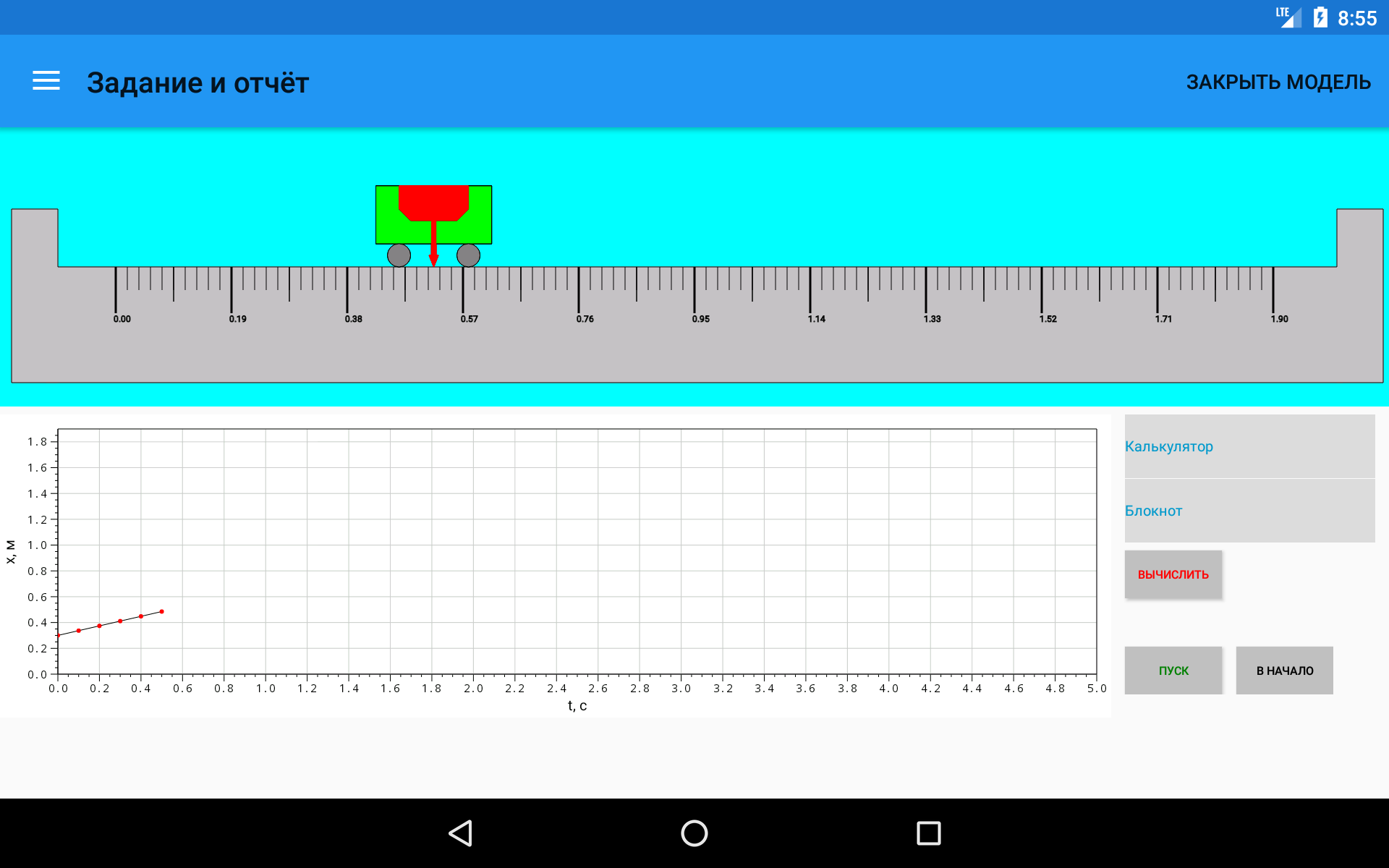


Рис. 7. Модель виртуальной лаборатории "Тележка на наклонном рельсе" на платформе Android (на эмуляторе Nexus 7).

## **2.5 Проигрыватель моделей виртуальных лабораторий ABarsicPlayer**

Проигрыватель ABarsicPlayer [11] является аналогом проигрывателя BARSIC на платформе Windows [12].

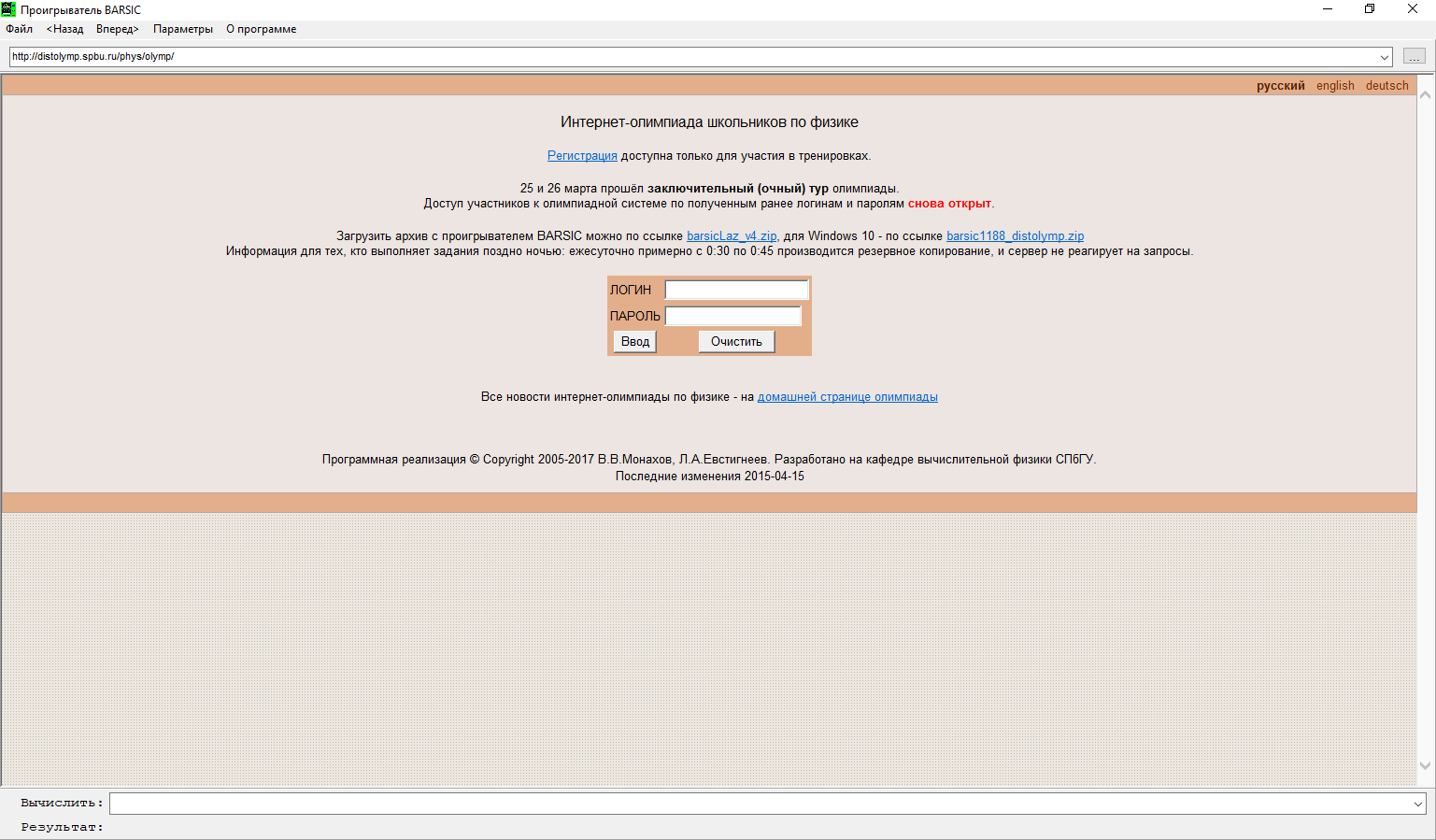


Рис. 8. Проигрыватель BARSIC на Windows.

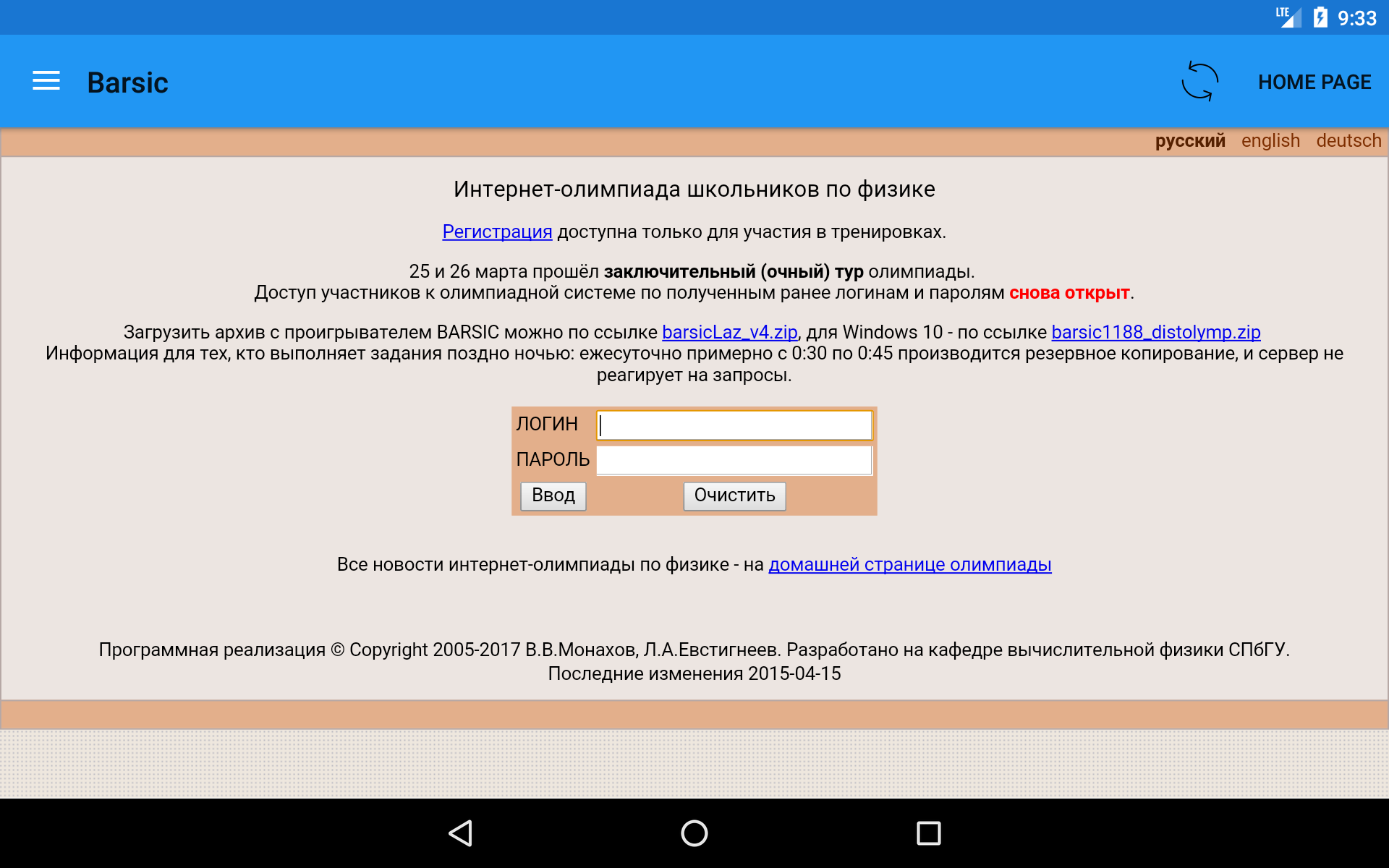


Рис. 9. Проигрыватель ABarsicPlayer на Android (на эмуляторе Nexus 7).

Функции проигрывателя:

* Регистрация и аутентификация пользователей
* Навигация по сайту интернет-олимпиады школьников по физике
* Воспроизведение заданий олимпиады
* Отсылка отчета на сервер

Для регистрации, аутентификации пользователей и навигации по сайту интернет-олимпиады проигрыватель содержит браузер, основанный на компоненте WebView. Также проигрыватель содержит фрагмент с разделом help, отображающий страницу ответов на вопросы по интернет-олимпиаде школьников по физике [13]. В проигрыватель добавлен фрагмент настроек Settings. В данном разделе можно устанавливать значение адреса домашней страницы. Изменение адреса домашней страницы необходимо при работе с локальным сервером, установленном на компьютере для отладки приложения. Адрес локального сервера в браузере ОС Windows <http://localhost/dev/>, а в браузере эмулятора устройства на ОС Android <http://10.0.2.2/dev/>.

Для установки локального сервера на компьютер необходимо сперва установить сервер Apache 2.2.9, php5 и MySQL-5.1.40. Затем возможно установка сервера distolymp и его настройка согласно инструкции [14, 15].

Для того чтобы добавить новую модель на локальный сервер необходимо пройти аутентификацию как root пользователь, перейти во вкладку модели, заполнить необходимые поля, в соответствии с инструкцией [14], загрузить файл BARSIC с расширением .brc по кнопке «Выберите файл». Затем нажать кнопку «Создать». После этого открыть папку C:\Apache2\htdocs\dev\data\models. В ней будет содержаться файл, такой же как файл, который был загружен на сервер, только в конец его имени будет добавлен идентификатор задания, типа .[\*\*\*\*]. В папке должны содержаться 3 файла: файл модели BARSIC с расширением .brc, файл модели Android с расширением .apk и текстовый файл с расширением .abrc, содержащий полное имя класса модели. Имена файлов должны быть одинаковыми и соответствовать имени модели BARSIC (файл с расширением .brc).

После успешной аутентификации пользователя на сайте интернет-олимпиады и перехода по ссылке на начало выполнения задания в проигрывателе ABarsicPlayer загружаются файл с именем модели с расширением .abrc и файл самой модели с расширением .apk с соответствующими идентификаторами задания, описанными выше. После получения имени класса модели, содержащегося в файле с расширением .abrc становится возможно подключение модели виртуальной лаборатории, находящейся в файле с расширением .apk.

Далее рассмотрим более подробно структуру проекта проигрывателя ABarsicPlayer.

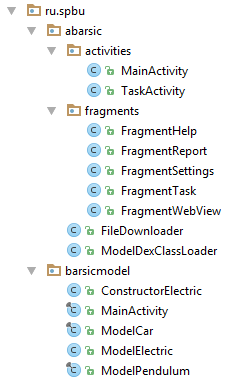


Рис. 10. Структура проигрывателя ABarsicPlayer.

Как видно из рисунка 10, проигрыватель ABarsicPlayer разделяется на два пакета. Пакет ru.spbu.abarsic содержит в себе активности и фрагменты проигрывателя, а пакет ru.spbu.barsicmodel – модели виртуальных лабораторий и класс MainActivity, описанный выше. Напомним, что класс MainActivity [Листинг 1] служит для создания самостоятельного приложения с моделью виртуальной лабораторий, которое может быть в последующем загружено проигрывателем ABarsicPlayer.

В проигрывателе ABarsicPlayer возможна работа как с кросс-компилированными, так и с разработанными непосредственно на java моделями. В первом случае, речь идет о подключении к проигрывателю уже скомпилированных dex-class файлов, содержащихся в загружаемом с сервера Android приложении (с расширением .apk). Во втором случае, говорится о работе с классами моделей виртуальных лабораторий, находящихся в пакете ru.spbu.barsicmodel. Такая структура проекта позволяет организовать удобную отладку моделей виртуальных лабораторий, без загрузки изменений на сервер. Когда разработка и отладка модели внутри проигрывателя ABarsicPlayer полностью завершена, ее файл-приложение однократно загружается на сервер. Для того, чтобы после загрузки модели на сервер, проигрыватель использовал именно загружаемую модель, а не модель из пакета ru.spbu.barsicmodel необходимо не включать соответствующий класс модели в приложение. В финальной версии проигрывателя ABarsicPlayer, которая будет устанавливаться на клиентские устройства, планируется полностью исключить пакет ru.spbu.barsicmodel для уменьшения размеров приложения проигрывателя.

В пакете ru.spbu.abarsic.activities представлено два класса: MainActivity и TaskActivity. В классе MainActivity задается интерфейс приложения проигрывателя, создаются экземпляры классов фрагментов из пакета ru.spbu.abarsic.fragments: FragmentWebView, FragmentSettings и FragmentHelp. Фрагмент fragmentWebView отвечает за навигацию по сайту интернет-олимпиады и отслеживание ссылок, содержащих строку «.brc» [Листинг 2].

Листинг 2. Код класса ru.spbu.abarsic.fragments.onPageStarted

@Override  
**public void** onPageStarted(WebView view, String url, Bitmap favicon) {  
 **super**.onPageStarted(view, url, favicon);  
 **if** (url.contains(**".brc"**)) {  
 **onModelUrlFoundListener**.onModelUrlFound(url);  
 }  
}

Фрагмент fragmentSettings отвечает за установку адреса домашней страницы, имеет две предустановленные кнопки, устанавливающие адреса сервера интернет-олимпиады и локального сервера. Фрагмент fragmentHelp показывает страницу помощи <http://distolymp2.spbu.ru/olymp/index_faq.html>.

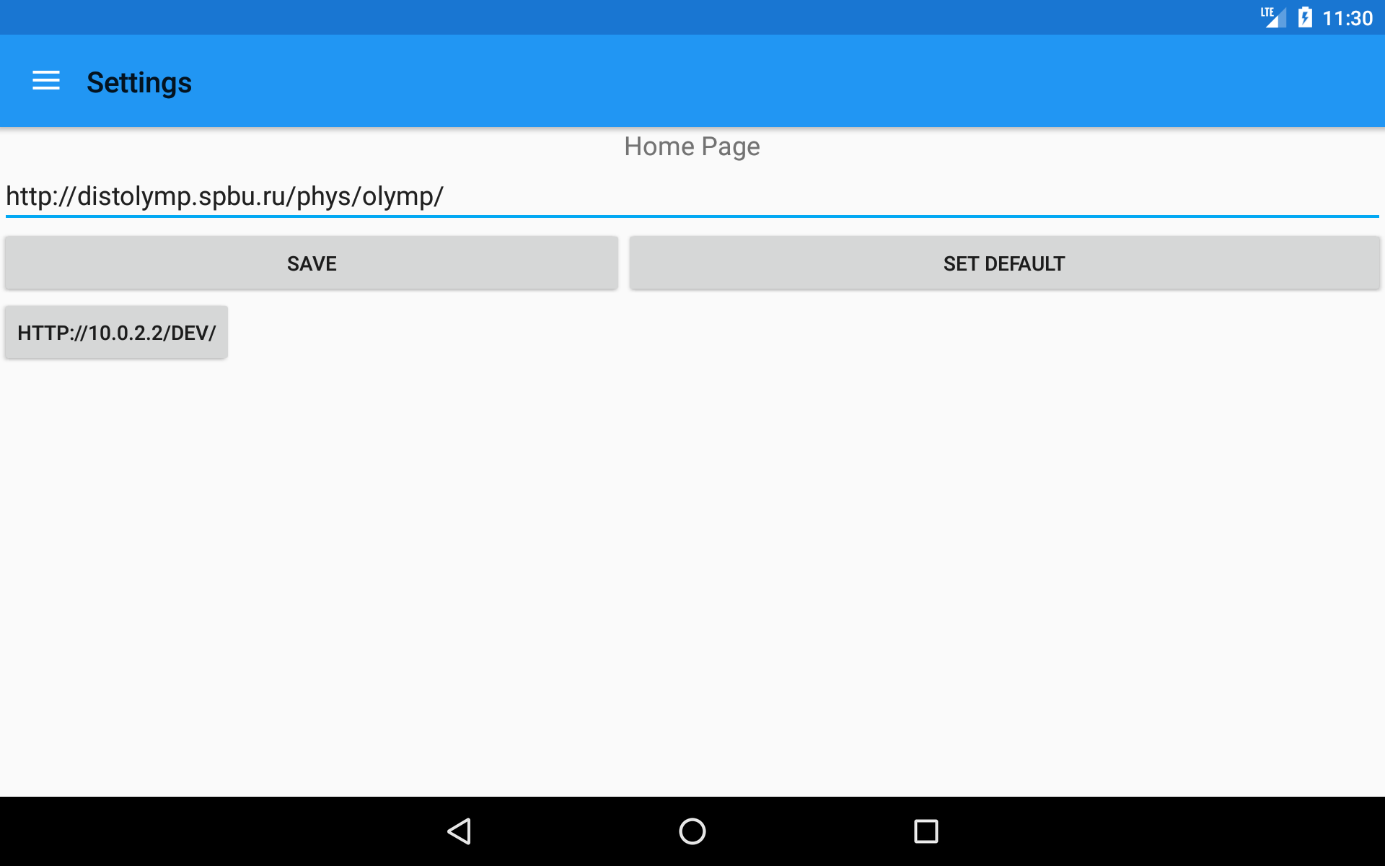


Рис. 11. Фрагмент настройки адреса домашней страницы (на эмуляторе Nexus 7).

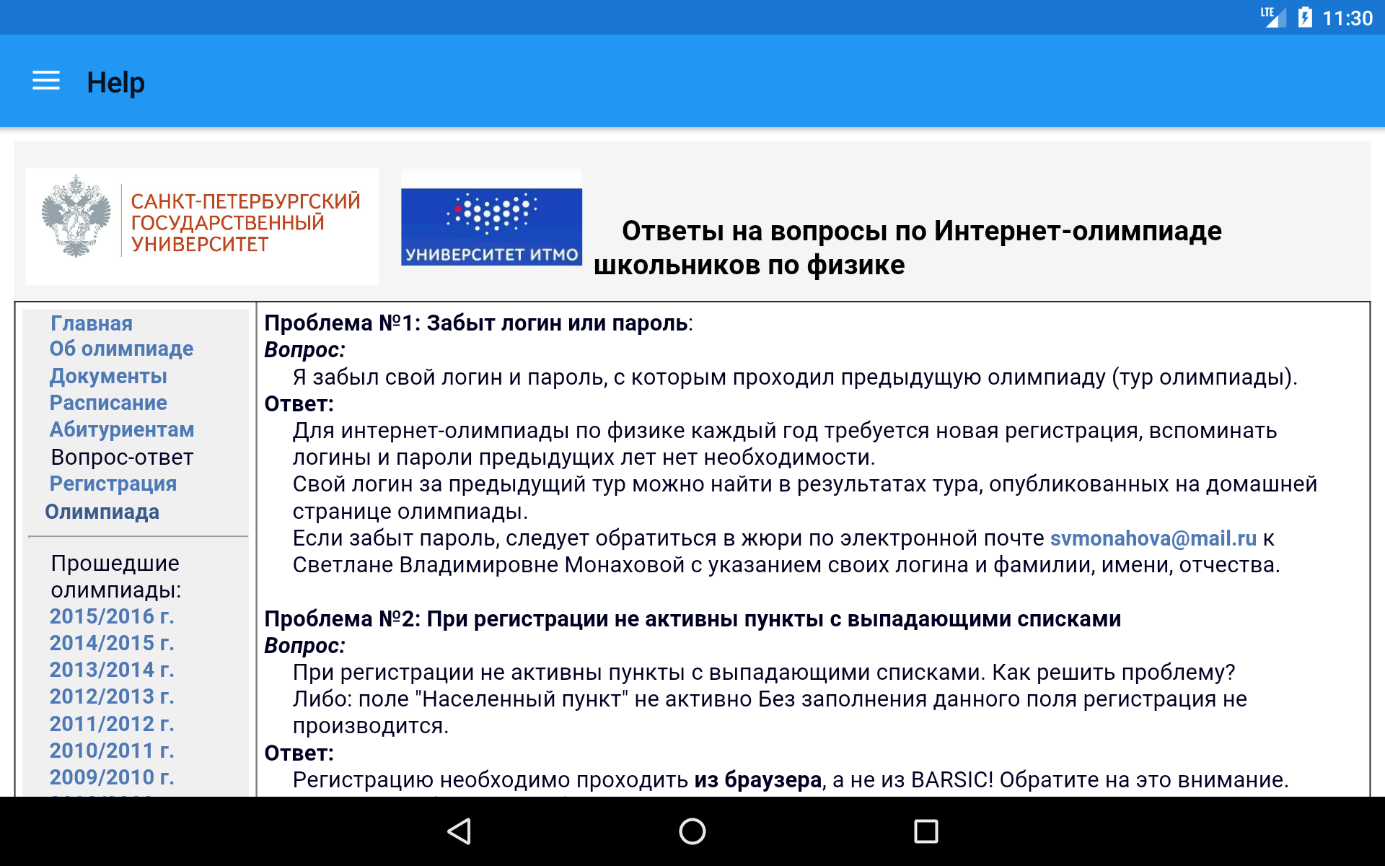


Рис. 12. Фрагмент «Help» (на эмуляторе Nexus 7).

В классе MainActivity задается fragmentManager, который управляет добавлением, показом и скрытием фрагментов. Изначально показывается fragmentWebView, а fragmentSettings и fragmentHelp – скрыты.

В отдельном потоке происходит загрузка файла с расширением .abrc, обработка имени этого файла и определение имени класса модели согласно содержанию файла. Метод onModelUrlFound отвечает за загрузку файла модели с расширением .apk с сервера. После загрузки файлов происходит переход к TaskActivity.

В классе TaskActivity задается интерфейс приложения в части выполнения задания, создаются экземпляры классов фрагментов из пакета ru.spbu.abarsic.fragments: FragmentTask и FragmentReport. Фрагмент FragmentTask отвечает за показ текста задания. Фрагмент FragmentReport отвечает за отправку отчета с результатами на сервер и получение результатов. Форма отчета генерируется на сервере. Если задание выполнено правильно, в окне появляется сообщение об этом. В противном случае сообщается о неправильном выполнении задания и предлагается повторить его выполнение.

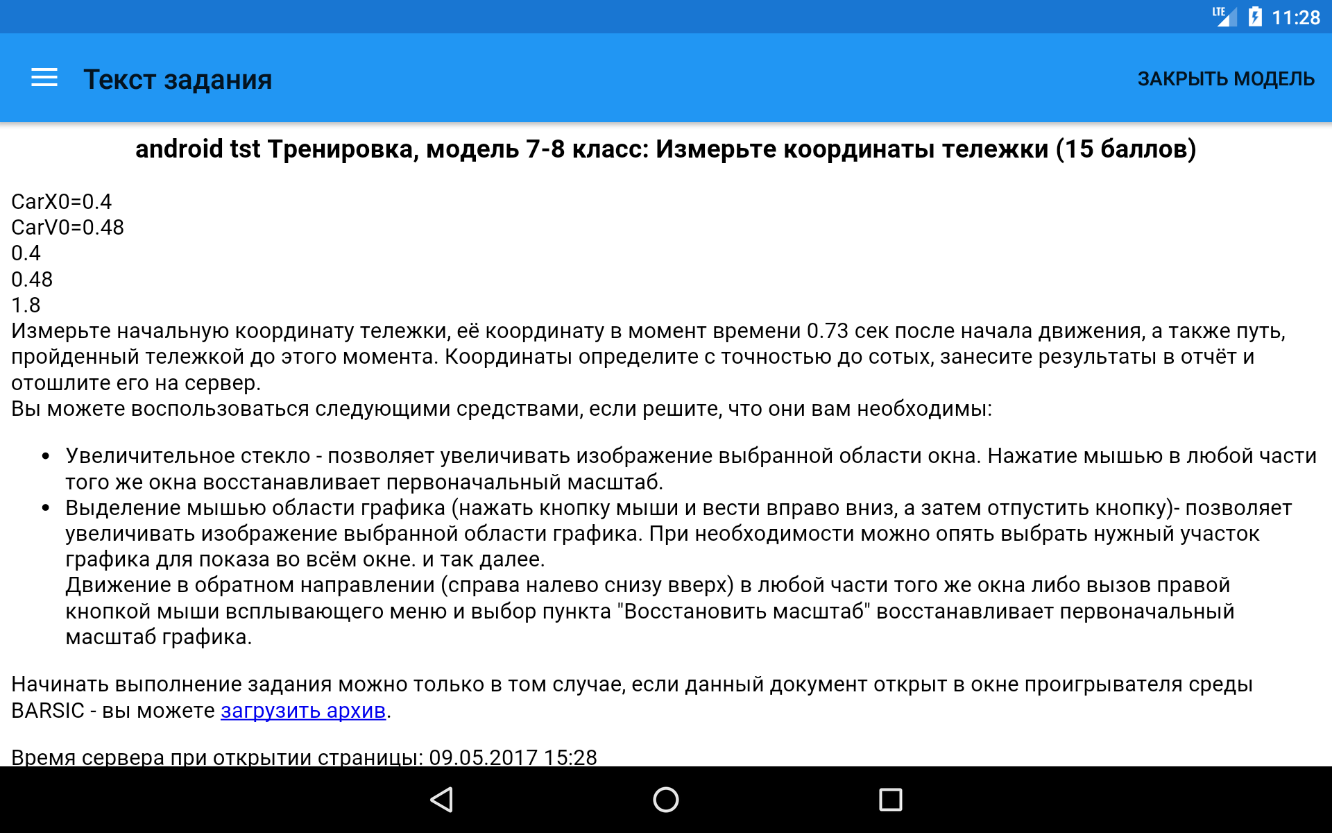


Рис. 13. Фрагмент «Текст задания» (на эмуляторе Nexus 7).

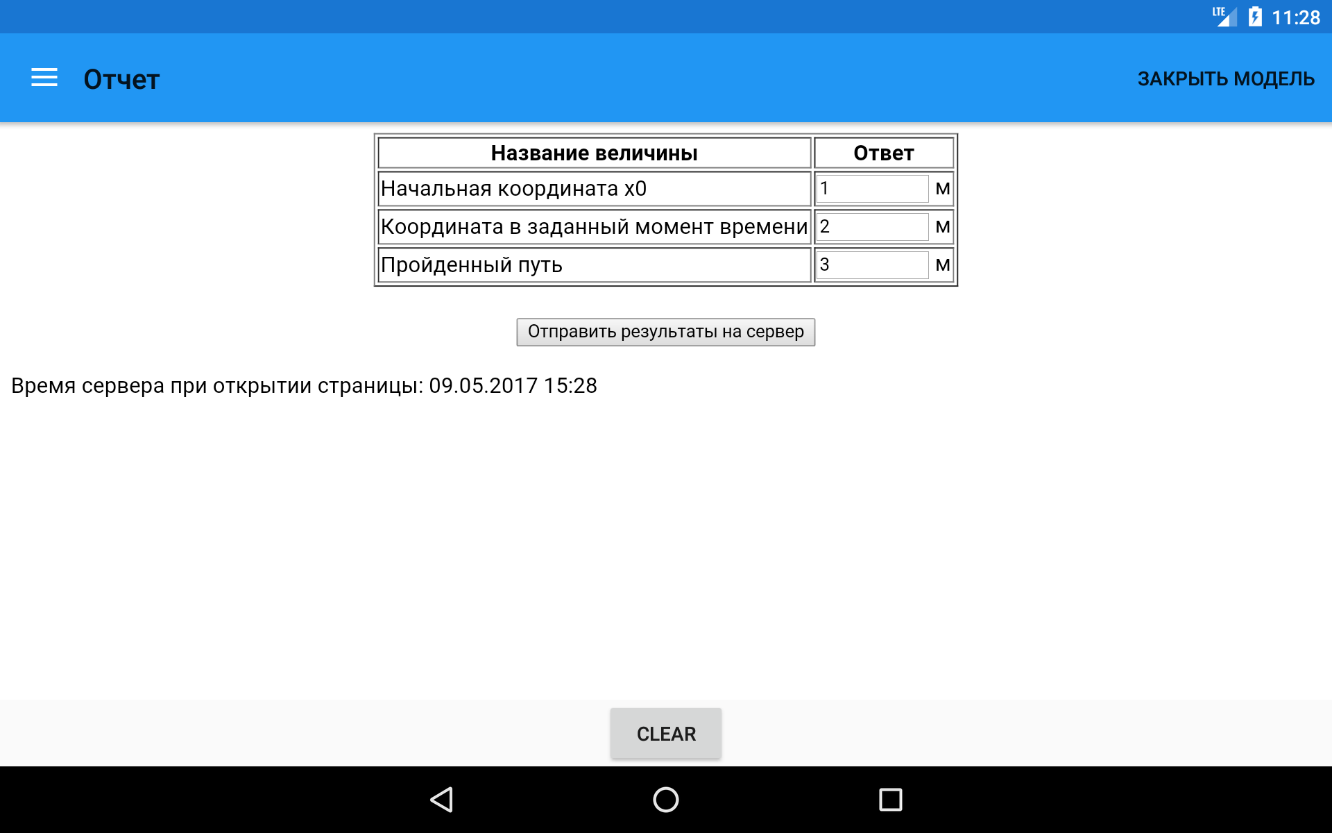


Рис. 14. Фрагмент «Отчет» (на эмуляторе Nexus 7).

Также, как и в классе MainActivity, в классе TaskActivity задается fragmentManager, который управляет добавлением, показом и скрытием фрагментов.

В методе loadModel происходит подгрузка фрагмента модели fragmentModel и его добавление с помощью менеджера фрагментов.

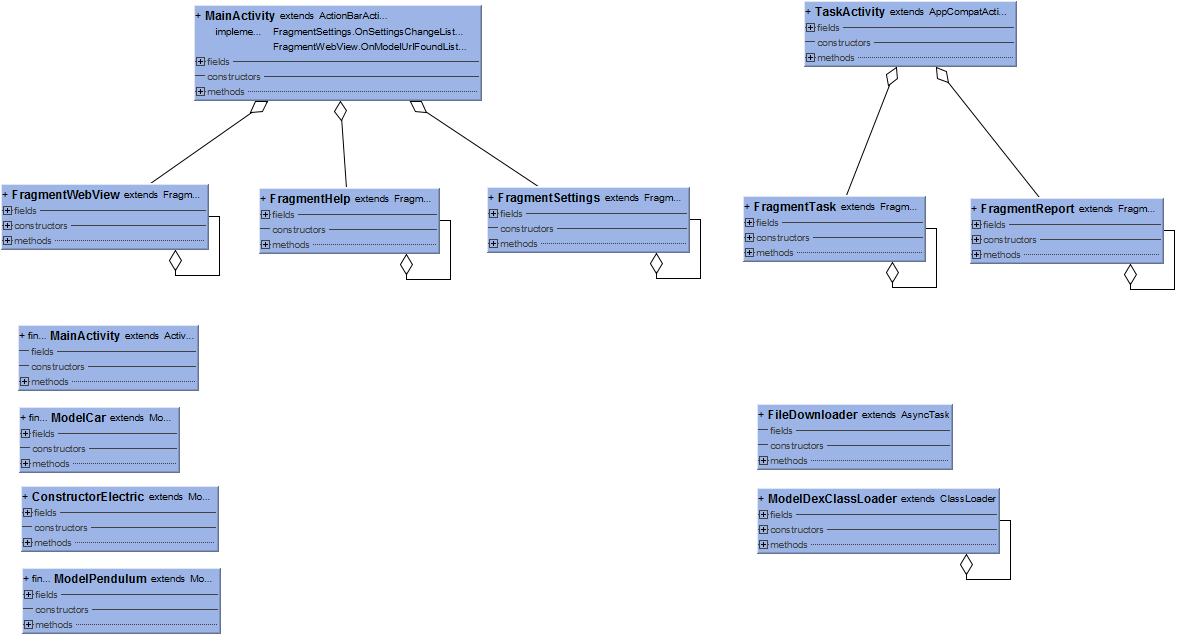


Рис. 15. Диаграмма классов проигрывателя ABarsicPlayer.

Класс FileDownloader осуществляет загрузку apk файла модели с сервера.

Класс ModelDexClassLoader используется для загрузки классов моделей. Класс модели выгружаются из apk-файла. Данный класс позволяет запускать код, не являющийся частью приложения.

В отдельном потоке загружаются параметры модели. Для этого код загрузки страницы параметров вынесен в тело метода doInBackground. После этого выполняется код, находящийся в теле метода onPostExecute. Здесь с помощью метода parsParam на странице параметров находятся специально помеченные имена и значения параметров и добавляются в объект LinkedHashMap. После чего выполняется метод paramVariableSet. Метод paramVariableSet в классе Model является абстрактным, его необходимо реализовать в классе модели виртуальной лаборатории в соответствии с параметрами, заданными на сервере.

# 3. Система кросс-компиляции моделей виртуальных лабораторий BARSIC на платформу Android

## **3.1 Структура кросс-компиляции**

Данная работа посвящена созданию структуры проекта, обеспечивающего поддержку работы на платформе Android кросс-компилированных моделей виртуальных лабораторий, и разработку или модификацию соответствующих классов Java. Преимуществом подхода кросс-компиляции по сравнению с написанием моделей с нуля является низкая степень дублирования исходного кода. Проект состоит из транслятора, который переводит код BARSIC в код платформы Android, и клиентского приложения, которое включает в себя стандартную библиотеку.

Рассмотрим процесс сборки Android-приложения.

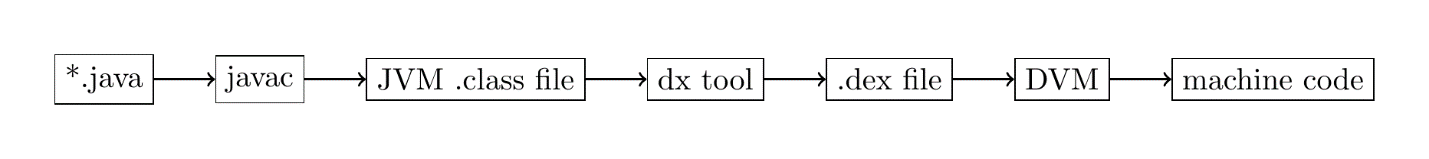


Рис. 16. Сборка и исполнение Android-проекта [4].

Он принципиально отличается от сборки Java-приложения. Байт-код JVM переводится транслятором dx в байт-код DVM (Dalvik VM). Затем транслятор запаковывает dex-файл с конкретной реализацией модели в apk.

Для платформы Android структура кросс-компиляции моделей BARSIC выглядит следующим образом:

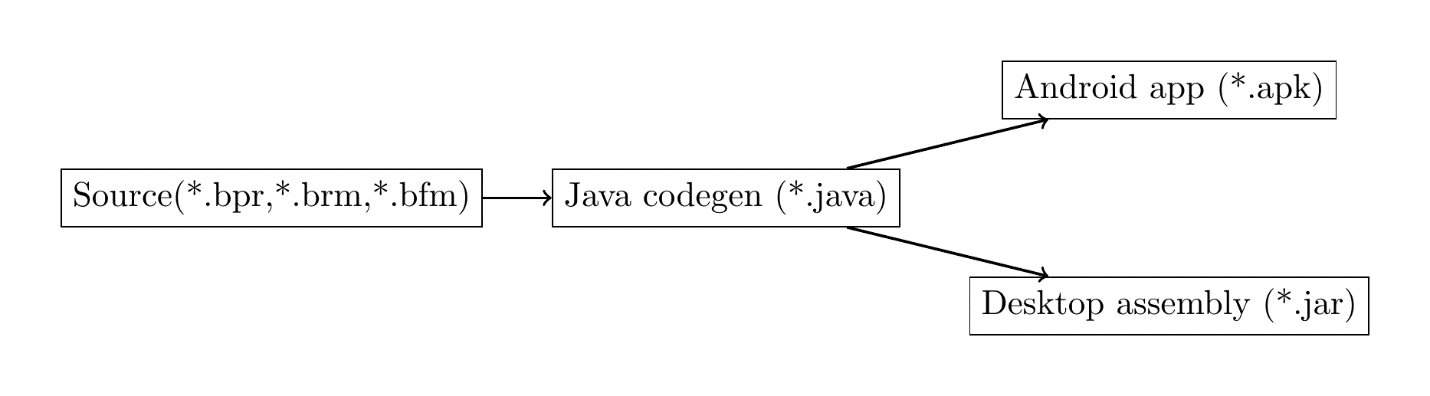


Рис. 17. Кросс-компиляция моделей BARSIC [5].

Происходит кросс-компиляция исходного кода языка BARSIC в байт-код Java и используемой в Android виртуальной машины Dalvik/ART.

## **3.2 Библиотека ABarsicLib в системе кросс-компиляции**

Библиотека ABarsicLib, разработанная Сергеем Мартынюком [9], включает компоненты пользовательского интерфейса и графики, реализующие интерфейсы системы кросс-компиляции проектов моделей на языке BARSIC. Данные компоненты призваны скрыть специфику поведения ОС Android и показать особенности аналогичных компонент на платформе BARSIC. Классы библиотеки были разработаны Сергеем Мартынюком таким образом, чтобы набор полей и сигнатура методов соответствовала классам на языке BARSIC.

Как отмечалось в главе 2.2, в данной работе были усовершенствованы некоторые компоненты библиотеки ABarsicLib, такие как кнопка, поле редактирования текста, текстовая панель, графическое подокно, однако основной частью работы в направлении построения системы кросс-компиляции стала разработка схемы запуска в проигрывателе ABarsicPlayer кросс-компилированных моделей.

## **3.3 Схема запуска кросс-компилированных моделей**

Автором предложена структура проекта виртуальных лабораторий, обеспечивающая схему запуска кросс-компилированных моделей под управлением ABarsicPlayer, аналогичную схеме запуска моделей, написанных непосредственно на Java. Как описывалось в главе 2.5 на сервере содержатся файлы модели с расширениями brc, abrc и apk. Сперва проигрыватель ABarsicPlayer при попытке перейти по гиперссылке на brc-файл модели BARSIC заменяет расширение brc на расширения abrc и apk и загружает файлы с расширениями abrc и apk. Затем получает имя класса модели, содержащееся в файле с расширением abrc, выгружает класс из полученного ранее файла с расширением apk согласно этому имени и исполняет его.

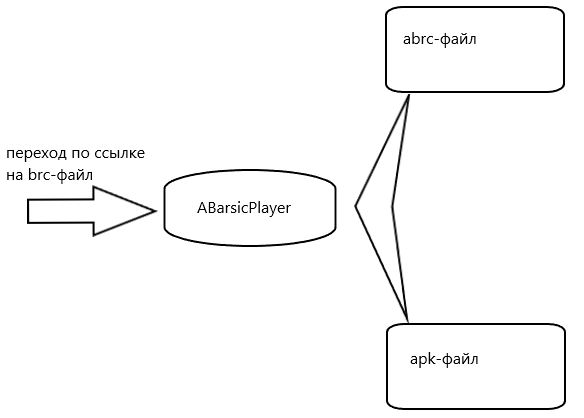


Рис. 18. Загрузка файлов.

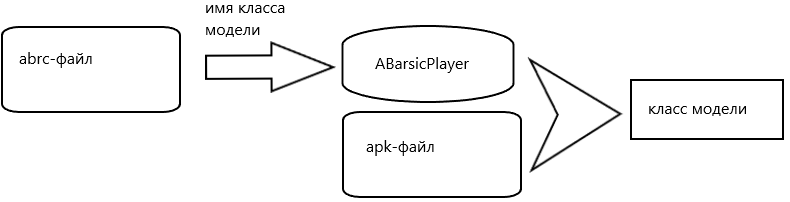


Рис. 19. Подгрузка класса.

Отличие проигрывателя моделей виртуальных лабораторий на Windows от проигрывателя на платформе Android заключается в том, что на Windows модели интерпретировались исполняющей средой BARSIC, а на платформе Android проигрыватель ABarsicPlayer лишь исполняет подгружаемый кросс-компилированный код модели. При этом кросс-компилятор создает код, обеспечивающий визуализацию с помощью компонентов-оболочек, находящихся в ABarsicPlayer. Поэтому усовершенствование этих компонентов-оболочек позволит усовершенствовать работу модели без изменения кросс-компилированного кода. Например, когда в ABarsicPlayer в компоненте-оболочке Subwindow будет реализован выбор пользователем масштаба графиков, установка на клиентское устройство нового варианта ABarsicPlayer позволит после загрузки с сервера той же модели виртуальной лаборатории просматривать в ней графики в разных масштабах.

## **3.4 Оболочечные компоненты**

В данной работе автором были усовершенствованы оболочечные компоненты: кнопка, поле редактирования текста, текстовая панель, графическое подокно [Листинги 3, 4, 5]. Во всех данных компонентах было реализовано изменение размеров и позиционирование на экране в зависимости от его размеров.

На экранной форме BARSIC нет проблемы с позиционированием компонент. В BARSIC размер формы и расположенных на ней компонентов задаётся исходя из размера экрана 1024х768 пикселей. Если разрешение экрана другое, форма и компоненты автоматически масштабируются так, чтобы форма занимала в процентном отношении такую же долю экрана, как при размере экрана 1024х768 пикселей. Экраны смартфонов и планшетов малы по сравнению с экранами компьютеров. В связи с этим требуется использовать всю доступную площадь экрана. К тому же размеры экранов Android устройств отличаются друг от друга. Методы display.getHeight() и display.getWidth() позволяют определить размеры экрана. Автором было предложено вычислять размеры компонентов и их координаты на экране из пропорции: x1/x0=d1/d0, где

x1 – величина размера либо координаты компонента для конкретного экрана, которую требуется найти;

x0 – универсальная величина размера либо координаты компонента, задаваемая разработчиком модели;

d1 – размер экрана конкретного устройства;

d0 – величина принятая за эталонный размер экрана.

Под размером экрана понимается ширина либо высота, в зависимости от высчитываемого свойства компонента.

В компонентах кнопки и поля редактирования текста автором были реализованы методы изменения цвета фона, цвета текста, изменения размера текста в соответствии с размером экрана. В BARSIC используется кодировка цветов – RGB. Такая же кодировка цветов реализована автором в данном приложении.

В компоненте поля редактирования текста добавлен метод setEnterKeyCount, который позволяет добавлять дополнительную кнопку на экранную клавиатуру и назначать необходимое событие при ее нажатии. Этот метод может быть полезен в поле редактирования текста – «Калькулятор» на тех экранах смартфонов или планшетов, где экранная клавиатура закрывает большую часть экрана (Рис. 22). В компоненте графического подокна был учтен размер верхнего меню toolbar – теперь при размещении компонента на форме он не может быть скрыт данным элементом меню, а корректно размещается под ним.

Размеры текста на некоторых компонентах малы для полноценной работы с моделями виртуальных лабораторий. Данную проблему еще предстоит решить при помощи увеличения экранных форм – аналога лупы в моделях BARSIC.

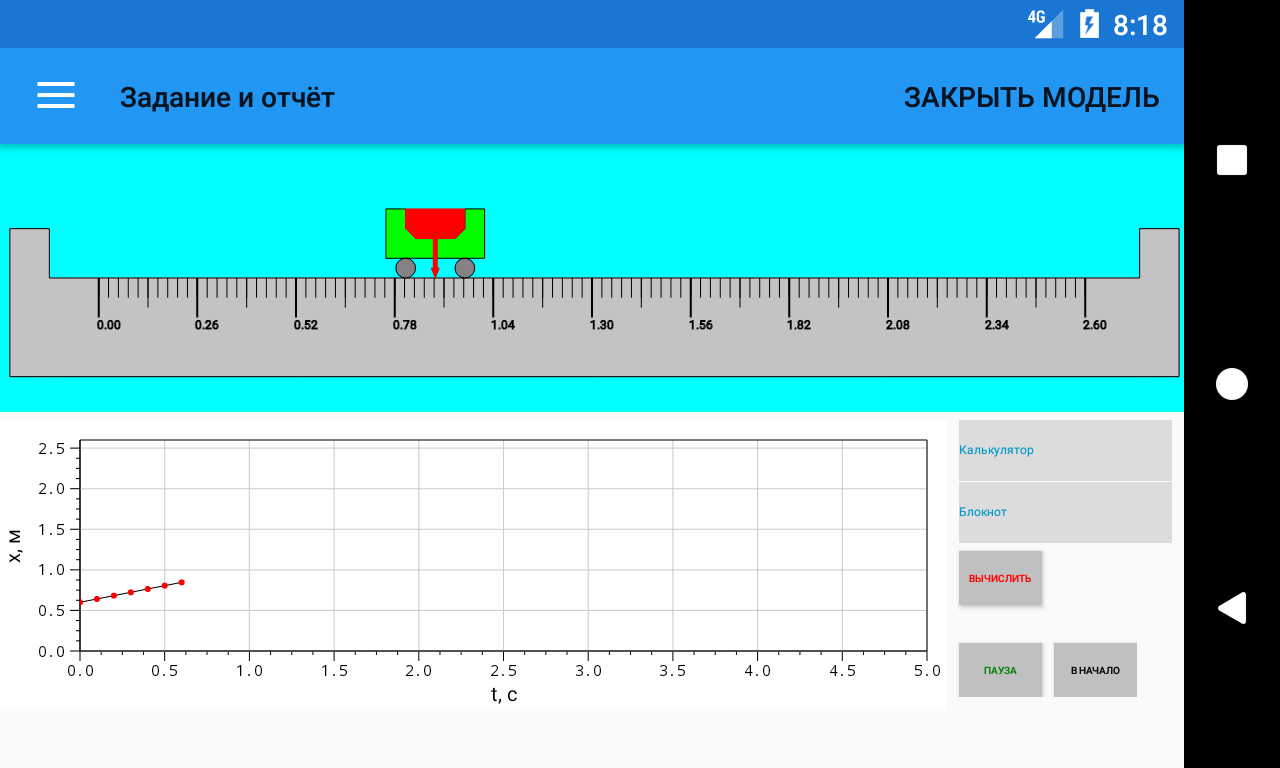


Рис. 20. Корректное отображение размеров и положения компонентов (на эмуляторе Nexus 4).

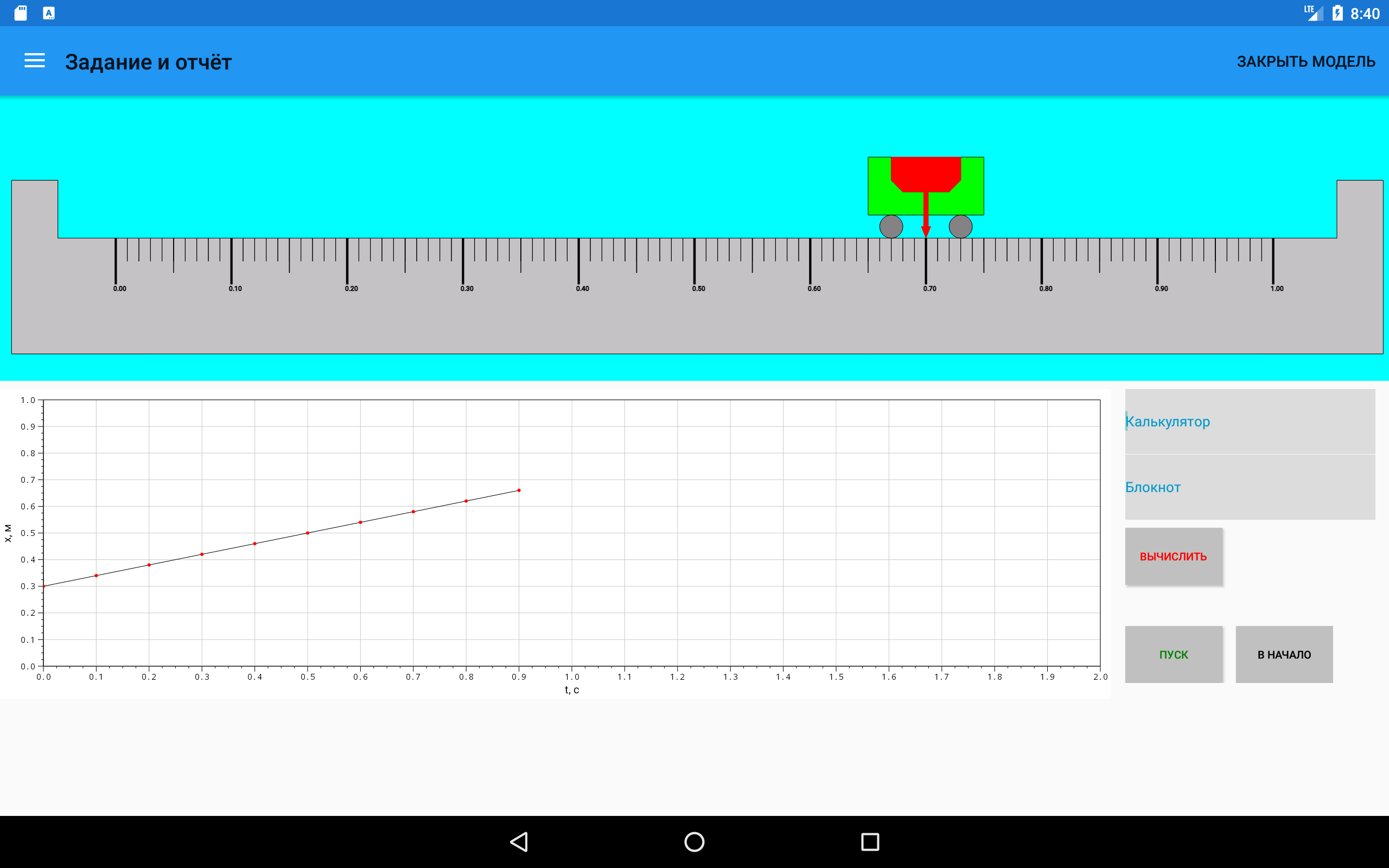


Рис. 21. Корректное отображение размеров и положения компонентов (на эмуляторе Nexus 10).

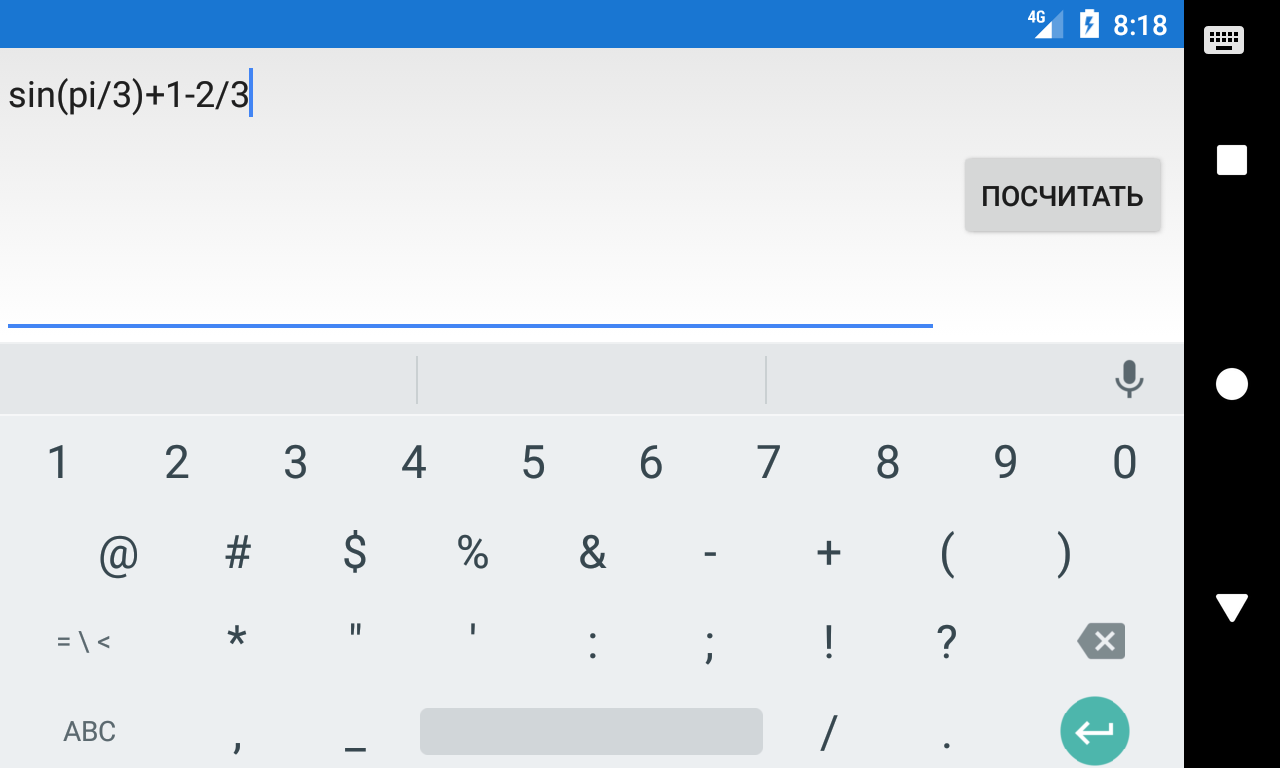


Рис. 22. Работа калькулятора (на эмуляторе Nexus 4).

# 4. Объединение проектов для платформы Android

## **4.1 Объединение проектов**

Для того, чтобы загружаемые модели были работоспособными, в проект ABarsicPlayer была включена библиотека ABarsicLib. В проект была добавлена модель «Электрическая лаборатория», разрабатываемая Евгением Файзулиным. В библиотеку ABarsicLib были добавлены все классы, необходимые для работы модели «Электрическая лаборатория».

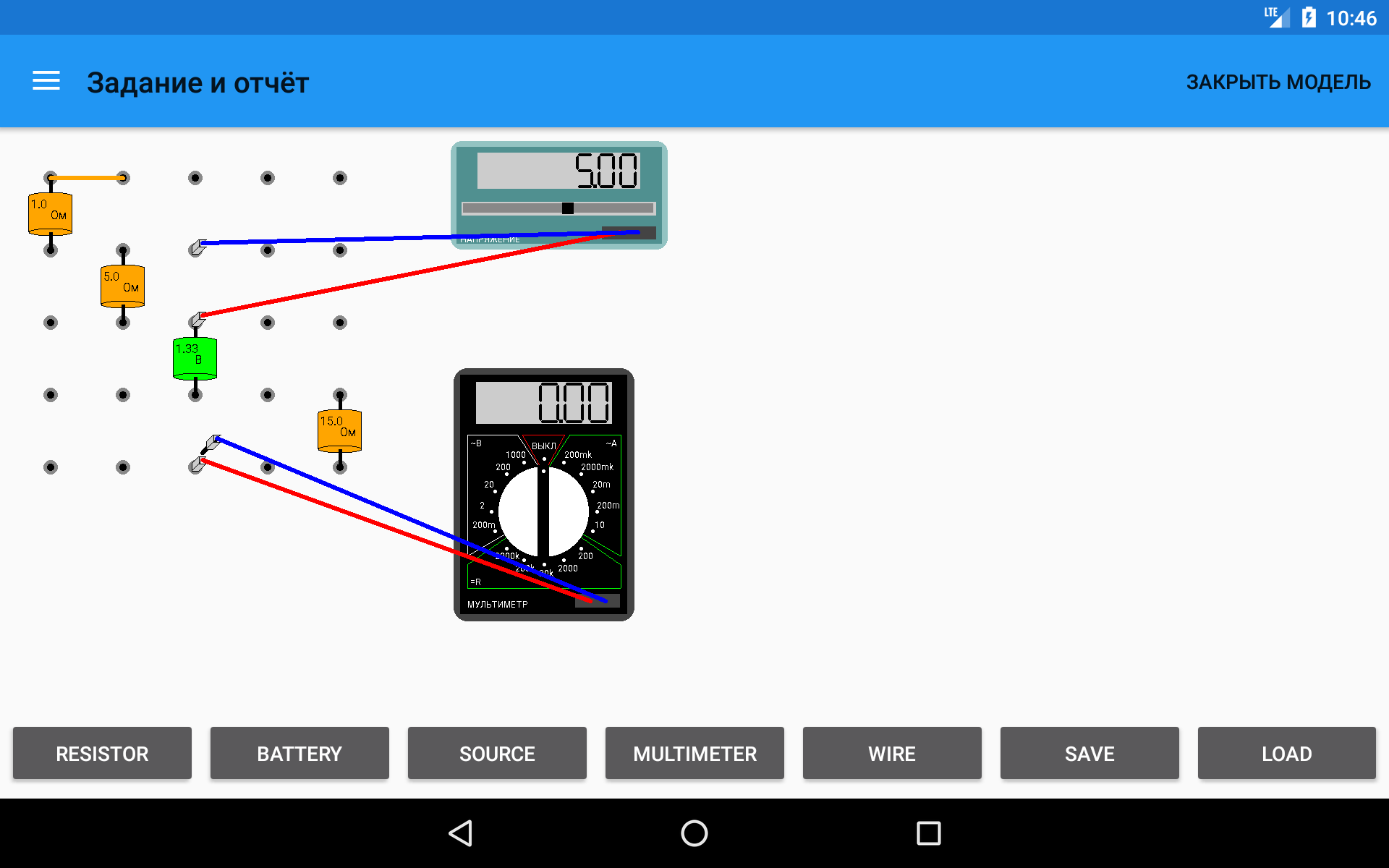


Рис. 23. Модель "Электрическая лаборатория" (на эмуляторе Nexus 7).

По состоянию на данный момент не все модели виртуальных лабораторий были перенесены на платформу Android. Также не все методы классов библиотеки ABarsicLib были реализованы для полноценной кросс-компиляции всех приложений BARSIC. В связи с этим, автором было принято решение вести разработку проекта, объединяющего библиотеку, проигрыватель и модели виртуальных лабораторий. Данное решение позволило тестировать модели виртуальных лабораторий без их загрузки на сервер олимпиады, что существенно ускорило отладку. Как отмечалось выше, в финальной версии проигрывателя, устанавливаемой на клиентские устройства, должны быть убраны все модели виртуальных лабораторий.

## **4.2 Тестирование объединенного проекта**

Приложение тестировалось на различных эмуляторах (Nexus 4, Nexus 7, Nexus 10, Pixel) и реальных Android-устройствах (HTC desire, Xiaomi Redmi Note 3 и др.) с различными параметрами экрана и характеристиками производительности.

Тестирование проводилось как на локально установленном сервере distolymp, так и на реально используемом при проведении олимпиад сервере.



Рис. 24. Тестирование проекта на локальном сервере.

# **Выводы**

1. Разработана структура проекта, позволяющего работать как с кросс-компилированными, так и разработанными непосредственно на Java моделями;
2. Доработан «проигрыватель» моделей виртуальных лабораторий ABarsicPlayer, а именно подгрузка параметров;
3. Доработана модель виртуальной лаборатории «Тележка на наклонном рельсе». Добавлен рельс, линейка со шкалой, функциональные кнопки, калькулятор, блокнот. Реализована установка загружаемых параметров;
4. Доработана библиотека ABarsicLib. Предложено разрабатывать классы моделей виртуальных лабораторий, наследуя их от абстрактного базового класса Model, содержащегося в библиотеке. Усовершенствованы разработанные С. Мартынюком классы Button, TextEdit, Subwindow. Добавлена библиотека JEvalParser;
5. Объединены проекты проигрывателя моделей виртуальных лабораторий ABarsicPlayer, библиотеки ABarsicLib, моделей «Тележка на наклонном рельсе» и «Электрическая лаборатория» и парсера выражений (калькулятор);
6. Проведено тестирование объединенного проекта.

Планируется дальнейшее усовершенствование библиотеки ABarsicLib, что позволит кросс-компилировать все модели виртуальных лабораторий и проводить олимпиаду на платформе Android.

# Литература

1. http://distolymp2.spbu.ru/olymp/ Домашняя страница интернет олимпиады школьников по физике [Электронный ресурс] // Интернет олимпиада школьников по физике / СПбГУ, университет ИТМО.
2. <https://top.mail.ru/browsers?id=250&period=0&date=&aggregation=sum&sids=mob,chrome,firefox,opera,msie,msedge,safari,sum,others&ytype=value&gtype=line> Глобальная статистика [Рейтинг@mail.ru](mailto:Рейтинг@mail.ru) [Электронный ресурс]
3. <https://top.mail.ru/oses?id=250&period=0&date=&aggregation=sum&sids=win,mob,mac,lin,tv&ytype=value&gtype=line> Глобальная статистика [Рейтинг@mail.ru](mailto:Рейтинг@mail.ru) [Электронный ресурс]
4. Монахов В.В., Воропаев Р.А., Бушманова Е.А. и др. Особенности разработки компьютерных тестов, задач и виртуальных лабораторий // Компьютерные инструменты в образовании. – 2013. – № 2. – С. 28–39.
5. Выпускная квалификационная работа аспиранта Максимова М.А. «Разработка кросс-компилятора из языка программирования BARSIC в код ART/Dalvik», 2016
6. В. В. Монахов и др. BARSIC: программный комплекс, ориентированный на физика-исследователя. Программирование, 2005, N3, с.68-80.
7. Бакалаврская работа Кузьмина Н.В. «Разработка средств программного комплекса aBarsic для отображения моделей виртуальных лабораторий по физике», 2015
8. Монахов В.В., Басов Л.В., Воропаев Р.А., Пивоваров А.М., Зуган М.С. distolymp – программный комплекс для проведения интернет-олимпиад и дистанционного обучения // В материалах XII международной конференции “Физика в системе современного образования”. – Петрозаводск, 2013, т.2. – С. 221-223.
9. Магистерская диссертация студента Мартынюка С.А. «Разработка компонента графики для виртуальных Java лабораторий по физике на платформе Android», 2016
10. Магистерская диссертация студентки Монаховой Е.В. «Разработка системы HTML5-визуализации виртуальных лабораторий по физике на платформе Android», 2016
11. В.В. Монахов, М.А. Максимов, В.А. Саликов, К.Р. Зубов, А.А. Салатич DISTOLYMP - программный комплекс для мультиплатформенной проверки знаний и практических умений. Современные информационные технологии. Теория и практика Материалы II Всероссийской научно-практической конференции в рамках ИТ-форума «ICITY 2015: Информатизация промышленного города» (г. Череповец, 19 ноября 2015 г.). c.124-129
12. <http://barsic.spbu.ru/www/license.html> Проигрыватель BARSIC (BARSIC player) [Электронный ресурс]
13. <http://distolymp2.spbu.ru/olymp/index_faq.html> Ответы на вопросы по Интернет-олимпиаде школьников по физике [Электронный ресурс]
14. <https://subversion.assembla.com/svn/barsic/trunk/source/Barsic/barsic/source/serverInstall/> Репозиторий [Электронный ресурс]
15. <https://subversion.assembla.com/svn/barsic/trunk/source/Barsic/barsic/source/serverModel/> Репозиторий [Электронный ресурс]
16. Монахов В.В., Кожедуб А.В., Уткин А.Б. «Особенности заданий интернет-олимпиады школьников по физике» Компьютерные инструменты в школе. №6, 2011
17. Монахов В.В., Стафеев С.К., Парфенов В.Г., Кожедуб А.В., Евстигнеев Л.А, Кавтрев А.Ф., Пономарев А.И. «Проведение дистанционных туров олимпиад по физике с использованием программного комплекса BARSIC» // Компьютерные инструменты в образовании. - СПб.: Изд-во ЦПО "Информатизация образования", 2005, N11, С. 05-15.
18. M.A. Maksimov, V.V. Monakhov, A.V. Kozhedub. Virtual Laboratories in Physics with Autogenerated Parameters. Journal of Physics: Conference Series, Vol. 633, Number 1, 2015, pp. 12009-12011(3).
19. <http://barsic.spbu.ru/www/barsic/bprog/bprog.htm> Цвета в RGB кодировке [Электронный ресурс]

# Приложения

## **Листинг 3. Код измененных методов класса Button**

**public class** Button **extends** android.widget.Button **implements** IButton {  
  
 **…**  
  
 Display **display** = ((WindowManager) getContext().getSystemService(***WINDOW\_SERVICE***)).getDefaultDisplay();  
 **public final double heightPixels1** = **display**.getHeight();  
 **public final double widthPixels1** = **display**.getWidth();  
 **public final double diagonal** = Math.*round*(Math.*sqrt*(Math.*pow*(**heightPixels1**,2)+Math.*pow*(**widthPixels1**,2)));  
 **public final double standartDiagonal**=800; *//640x480* **public final double ratio** = **diagonal**/**standartDiagonal**;  
 **private** ReentrantLock **mMainLock**;  
  
  @Override  
 **public void** setX(**int** value) {  
 **x1**=value;  
 **x1** = (**int**) Math.*round*(**x1**\***widthPixels1**/100);  
 AbsoluteLayout.LayoutParams params =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(params);  
 }  
  
  
 @Override  
 **public void** setY(**int** value) {  
 **y1**=value;  
 **y1**= (**int**) Math.*round*(**y1**\***heightPixels1**/100);  
 AbsoluteLayout.LayoutParams params =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(params);  
 }  
  
 @Override  
 **public void** width(**int** value) {  
 **width1**=value;  
 **width1** = (**int**) Math.*round*(**width1**\***widthPixels1**/100);  
 AbsoluteLayout.LayoutParams params =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(params);  
 }  
  
  
 **public void** height(**int** value){  
 **height1**=value;  
 **height1**= (**int**) Math.*round*(**height1**\***heightPixels1**/100);  
 AbsoluteLayout.LayoutParams parampampam =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(parampampam);  
 }  
  
 @Override  
 **public void** setHint(String value){  
 **int** hintColor = android.R.color.***holo\_blue\_dark***;  
 **super**.setHintTextColor(getResources().getColor(hintColor));  
 **super**.setHint(value);  
 }  
  
 **public void** setIsEnabled(**boolean** value){  
 setEnabled(value);  
 }  
  
 **public void** setIsVisible(**boolean** value){  
 **if** (value==**true**) {  
 **this**.setVisibility(**this**.***VISIBLE***);  
 }  
 **else  
 this**.setVisibility(**this**.***INVISIBLE***);  
 }  
  
 **public int textSize2**=0;  
 @Override  
 **public void** setTextSize(**int** value){  
 **textSize2**= (**int**) Math.*round*(value\***ratio**/10);  
 **super**.setTextSize(**textSize2**);}  
  
 @Override  
 **public void** setBackColor(String value) {  
 **int** color = *formatColorAsInt*(value);setBackgroundColor(color);  
 }  
  
 **public void** setTextColor(String value) {  
 **int** color = *formatColorAsInt*(value);  
 setTextColor(color);  
 }

…

}

## **Листинг 4. Код измененных методов класса TextEdit**

**package** ru.spbu.abarsic;  
  
**import** ru.spbu.barsic.ITextField;  
**import static** android.content.Context.***WINDOW\_SERVICE***;  
**import static** ru.spbu.abarsic.graphics.ColorUtils.*formatColorAsInt*;

**import**  
…

**public class** TextEdit **extends** EditText **implements** ITextField {  
  
 **int width1**=0,**height1**=0,**x1**=0,**y1**=0;  
  
 Display **display** = ((WindowManager) getContext().getSystemService(***WINDOW\_SERVICE***)).getDefaultDisplay();  
 **public final double heightPixels1** = **display**.getHeight();  
 **public final double widthPixels1** = **display**.getWidth();  
 **public final double diagonal** = Math.*round*(Math.*sqrt*(Math.*pow*(**heightPixels1**,2)+Math.*pow*(**widthPixels1**,2)));  
 **double standartDiagonal**=800;  
 **public final double ratio** = **diagonal**/**standartDiagonal**;**public** TextEdit(Context context, AttributeSet attrs) {  
 **super**(context, attrs);  
 }  
  
 **public int width**=100, **height**=100, **x**=300, **y**=500;  
  
 **public void** setX(**int** value){  
 **x1**=value;  
 **x1** = (**int**) Math.*round*(**x1**\***widthPixels1**/100);  
 AbsoluteLayout.LayoutParams params =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(params);  
  
 }  
  
 **public void** setY(**int** value){  
 **y1**=value;  
 **y1**=(**int**) Math.*round*(**y1**\***heightPixels1**/100);  
 AbsoluteLayout.LayoutParams params =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(params);  
 }  
  
 @Override  
 **public int** width() {  
 **return** getWidth();  
 }  
  
 @Override  
 **public void** width(**int** value){  
 **width1**=value;  
 **width1** = (**int**) Math.*round*(**width1**\***widthPixels1**/100);  
 AbsoluteLayout.LayoutParams parampampam =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(parampampam);  
 }  
  
 @Override  
 **public int** height() {  
 **return** getHeight();  
 }  
  
 **public void** height(**int** value){  
 **height1**=value;  
 **height1**= (**int**) Math.*round*(**height1**\***heightPixels1**/100);  
 AbsoluteLayout.LayoutParams params =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(params);  
 }  
  
 @Override  
 **public void** setHint(String value){  
 **int** hintColor = android.R.color.***holo\_blue\_dark***;  
 **int** normalTextColor = android.R.color.***holo\_blue\_dark***;  
 **super**.setHintTextColor(getResources().getColor(hintColor));  
 **super**.setTextColor(getResources().getColor(normalTextColor));  
 **super**.setHint(value);}  
  
 **public void** setIsEnabled(**boolean** value){  
 setEnabled(value);  
 }  
  
 **public void** setIsVisible(**boolean** value){  
 **if** (value==**true**) {  
 **this**.setVisibility(**this**.***VISIBLE***);  
 }  
 **else  
 this**.setVisibility(**this**.***INVISIBLE***);  
 }  
  
 **public int textSize1**=0;  
 @Override  
 **public void** setTextSize(**int** value){  
 **textSize1**= (**int**) Math.*round*(value\***ratio**/10);  
 **super**.setTextSize(**textSize1**);  
 }  
  
 **public void** setTextColor(String value) {  
 **int** color = *formatColorAsInt*(value);  
 setTextColor(color);  
 }  
  
 **public void** setBackColor(String value) {  
 **int** color = *formatColorAsInt*(value);  
 setBackgroundColor(color);  
 }  
  
 @Override  
 **public void** setText(**final** String value) {  
 **final** TextEdit textEdit = **this**;  
 **if** (Thread.*currentThread*() != Looper.*getMainLooper*().getThread()) {  
 post(**new** Runnable() {  
 @Override  
 **public void** run() {textEdit.setText(value);  
 }  
 });  
 } **else** {  
 **if** (**textSize1**!=0) {}  
 **super**.setText(value);  
 }  
 }  
  
   
 **public void** setEnterKeyCount(**final** Button button) {  
 setImeOptions(EditorInfo.***IME\_ACTION\_DONE***);  
 setImeActionLabel(**"Посчитать"**, EditorInfo.***IME\_ACTION\_DONE***);  
 setOnEditorActionListener(**new** EditText.OnEditorActionListener() {  
 @Override  
 **public boolean** onEditorAction(TextView v, **int** actionId, KeyEvent event) {  
 **if** ((event != **null** && (event.getKeyCode() == KeyEvent.***KEYCODE\_ENTER***)) || (actionId == EditorInfo.***IME\_ACTION\_DONE***)) {  
 button.callOnClick();  
 InputMethodManager imm = (InputMethodManager) getContext().getSystemService(Activity.***INPUT\_METHOD\_SERVICE***);  
 imm.toggleSoftInput(InputMethodManager.***HIDE\_IMPLICIT\_ONLY***, 0);  
 }  
 **return false**;  
 }  
 });  
 }

…

}

## **Листинг 5. Код измененных методов класса Subwindow**

**public final class** Subwindow **extends** SurfaceView **implements** ISubwindow {  
  
 …  
  
 *//zoom* **private** ScaleGestureDetector **mScaleDetector**;  
 **private float mScaleFactor** = 1.f;  
  
  
   
  
 Display **display** = ((WindowManager) getContext().getSystemService(***WINDOW\_SERVICE***)).getDefaultDisplay();  
 **public final int heightPixels1** = **display**.getHeight();  
 **public final int widthPixels1** = **display**.getWidth();  
  
 **public int toolbarSize**;  
  
   
  
 *//zoom* **mScaleDetector** = **new** ScaleGestureDetector(getContext(), **new** Subwindow.ScaleListener());  
  
 TypedValue tv = **new** TypedValue();  
 getContext().getTheme().resolveAttribute(android.R.attr.***actionBarSize***, tv, **true**);  
 **toolbarSize** = getResources().getDimensionPixelSize(tv.**resourceId**);  
 }  
 @Override  
 **public void** setX(**int** value) {  
 **x1**=value;  
 **x1** = Math.*round*(**x1**\***widthPixels1**/100);  
 AbsoluteLayout.LayoutParams params =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(params);  
 }  
  
 @Override  
 **public void** setY(**int** value) {  
 **y1**=value;  
 **if** (**y1**==0) {  
 **y1**=Math.*round*(**toolbarSize**/**heightPixels1**);  
 }  
 **else** {  
 **y1**=Math.*round*(**y1**\***heightPixels1**/100);  
 }  
 AbsoluteLayout.LayoutParams params =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(params);  
 }  
  
 @Override  
 **public void** setHeight(**int** value) {  
**height1**=value;  
 **height1**=Math.*round*(**height1**\***heightPixels1**/100);  
 AbsoluteLayout.LayoutParams parampampam =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(parampampam);  
 }  
  
  
 @Override  
 **public void** setWidth(**int** value) {  
 **width1**=value;  
 **width1** = Math.*round*(**width1**\***widthPixels1**/100);  
 AbsoluteLayout.LayoutParams params =  
 **new** AbsoluteLayout.LayoutParams(**width1**, **height1**, **x1**, **y1**);  
 **super**.setLayoutParams(params);  
 }  
  
  
   
 @Override  
 **public void** setOnMouseDown(**final** Runnable eventHandler) {  
 setOnTouchListener(**new** OnTouchListener() {  
 @Override  
 **public boolean** onTouch(View v, MotionEvent event) {  
 **if** (event.getAction() == MotionEvent.***ACTION\_DOWN***) {  
 eventHandler.run();  
 }  
 **return true**;  
 }  
 });  
 }  
  
 @Override  
 **public void** setOnDrag(**final** Runnable eventHandler) {  
 setOnDragListener(**new** OnDragListener() {  
 @Override  
 **public boolean** onDrag(View v, DragEvent event) {  
 **if** (event.getAction() == DragEvent.***ACTION\_DRAG\_STARTED***)  
 eventHandler.run();  
 **return true**;  
 }  
 });  
 }  
  
 @Override  
 **public void** setOnDragDrop(**final** Runnable eventHandler) {  
 setOnDragListener(**new** OnDragListener() {  
 @Override  
 **public boolean** onDrag(View v, DragEvent event) {  
 **if** (event.getAction() == DragEvent.***ACTION\_DROP***)  
 eventHandler.run();  
 **return true**;  
 }  
 });  
 }  
  
   
 @Override  
 **public boolean** onTouchEvent(MotionEvent ev) {  
 **mScaleDetector**.onTouchEvent(ev);  
 **float** xEvent = ev.getX();  
 **float** yEvent = ev.getY();  
 **int** w1=width();  
 **int** h1=height();  
 **if** (**mScaleDetector**.isInProgress()) {  
 **this**.setPivotX(xEvent);  
 **this**.setPivotY(yEvent);  
 **this**.setScaleX(**mScaleFactor**);  
 **this**.setScaleY(**mScaleFactor**);Log.*d*(**"mScaleFactor= "**, **""**+ **mScaleFactor**);  
  
 }  
 **else** {  
 **this**.setScaleX(1);  
 **this**.setScaleY(1);}  
 **return true**;  
 }  
  
  
 **private class** ScaleListener **extends** ScaleGestureDetector.SimpleOnScaleGestureListener { *//zoom* @Override  
 **public boolean** onScale(ScaleGestureDetector detector) {  
 **mScaleFactor** \*= detector.getScaleFactor();  
 *// Don't let the object get too small or too large.* **mScaleFactor** = Math.*max*(0.3f, Math.*min*(**mScaleFactor**, 5.0f));  
  
 invalidate();  
 **return true**;  
 }  
 }

…  
  
}