Dietrich Boles
University of Oldenburg (Germany)

Droidfoot – Porting Greenfoot-Scenarios to Android Devices

Version 2.0

1 Introduction

Greenfoot (www.greenfoot.org) is an interactive Java development environment that has been developed primarily for educational purposes. It allows for the easy development of two-dimensional graphical applications such as simulations and games. Programs developed with Greenfoot (called scenarios) can be run as JavaSE applications on the PC as well as in web browsers in form of java applets.

Droidfoot (http://www.programmierkurs-java.de/droidfoot) is a project that makes it possible to run Greenfoot scenarios as Android apps on Android devices. Droidfoot consists of the Greenfoot class library that has been ported to Android, as well as a player with which the scenarios can be run on the Android device. Currently Android 4.0 and up is supported.

1.1 Changes in version 2.0 compared with version 1.0

Compared with version 1.0 there are the following changes in version 2.0 of Droidfoot:

- Greenfoot class *GreenfootSound* has been implemented (see chapter 2.6).
- Greenfoot class *UserInfo* has been implemented (see chapter 2.7).

- Now the rotation of actors is incorporated in the implementation of the collision detection methods.
- Now it is possible to configure the Android Greenfoot Player (see chapter 5.4).
- The examples have slightly been modified (see chapter 4).

1.2 Structure oft he manual

In this document the changes of the Android Greenfoot class library in contrast to the original Greenfoot class library are described in Section 2. Chapter 3 introduces the basic structure and handling of the Android Greenfoot player. Chapter 4 is devoted to the installation of Droidfoot if you want to implement your own Greenfoot scenarios as Android apps. Finally, Chapter 5 presents the basic procedure to implement already finished Greenfoot scenarios as Android apps.

2 Greenfoot class library for Android

The Greenfoot class library consists of the following classes:

- *Actor*
- *Greenfoot*
- *GreenfootImage*
- *GreenfootSound*
- *MouseInfo*
- *UserInfo*
- *World*

In version 2.0 of Droidfoot all classes are supported.

The JavaSE package *jawa.awt* is not accessible in Android. In exchange for that package a package *greenfoot.awt* has been defined, which implements substitute classes *Color* and *Font*.

2.1 Greenfoot

The following methods of class *Greenfoot* are not supported by Droidfoot:

- *getMicLevel*
- *getKey*
- *isKeyDown*
- *mouseMoved*
- *mouseClicked*

The method *mousePressed* returns *true* due to a touch event.

2.2 World

The methods with class *java.awt.Color* in their signature are adapted to class *greenfoot.awt.Color*.

Defining your own world class please note

- that you have to implement a constructor without parameters and
- that you have to call one of the *setBackground* methods in each constructor.

2.3 Actor

Defining an actor class please note

- that you have to call one of the *setImage* methods in each constructor.

2.4 GreenfootImage

The following methods of class *GreenfootImage* are not supported by Droidfoot:

- *drawShape*
- *fillShape*

The methods with class *java.awt.Color* or class *java.awt.Font* in their signature are adapted to class *greenfoot.awt.Color* and *greenfoot.awt.Font*.

The return type of method *getAwtImage* is *android.graphics.Bitmap*.

2.5 MouseInfo

All methods of class *MouseInfo* are supported by Droidfoot. However, Droidfoot does not support double clicks. So method *getClickCount* will always return 1.

2.6 GreenfootSound

All methods of class *GreenfootSound* are supported by Droidfoot. The implementation is done using the Android media player, so all audio formats which are supported by the Android media player are supported by Droidfoot too (see http://developer.android.com/guide/appendix/media-formats.html).

2.7 UserInfo

All methods of class *UserInfo* are supported by Droidfoot. The storage and management of data is not done on a server but locally on the Android device, so it behaves like executing a scenario in Greenfoot itself. The user name can be changed in the Android Greenfoot player using the menu item "settings". The method *isStorageAvailable* returns *false* if and only if the user name is empty.

2.8 Class greenfoot.awt.Color

Class *greenfoot.awt.Color* is a substitute of class *java.awt.Color*. The class defines nearly the same methods as class *java.awt.Color*.

In Android color information are stored in an *int* value (see class *android.graphics.Color*). The methods *getColor* and *getRGB* return that value.

2.9 Class greenfoot.awt.Font

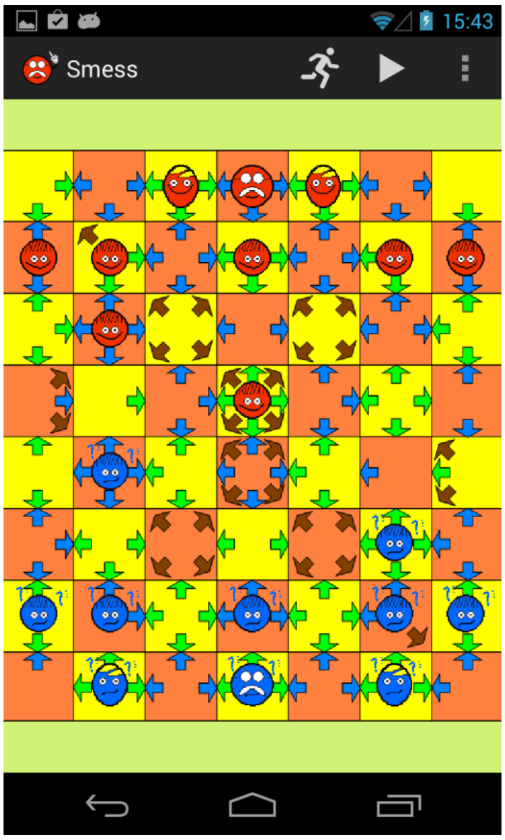
Class *greenfoot.awt.Font* is a substitute of class *java.awt.Font*. Fonts are represented differently in Android and in JavaSE-AWT. Internally the new class represents fonts by an instance of class *android.graphics.Typespace* and an *int* value for the font size.

Please be careful if you use class *greenfoot.awt.Font*. If you use fonts excessively in your Greenfoot scenarios you should consider converting your implementation to class *android.graphics.Typespace* and use the corresponding constructor of class *greenfoot.awt.Font*.

3 Greenfoot player for Android

The Android Greenfoot player requires an Android device with Android 4.0 and up. Reason for that is the usage of the Android action bar (see http://developer.android.com/guide/topics/ui/actionbar.html). In the action bar the buttons (run, pause, reset, …) and the speed slider are placed.

The following figure shows the structure and design of the Android Greenfoot player.



The whole screen is reserved for displaying the Greenfoot world. In the action bar above the controls are placed. In the left of the action bar you can see the app icon and the app name. In the right of the action bar or in the overflow area the action buttons and action items are located. The following actions exist if not configured otherwise (see chapter 5.4):

- Speed (runner symbol): By touching the runner button a slider appears which can be used to change the speed of the Greenfoot scenario. By touching the app icon the slider disappears.
- Start/Pause (start/pause symbol): By touching the start/pause button you can start or pause the Greenfoot scenario (equipollent to the Greenfoot start/pause button)
- Reset (reset symbol): By touching the reset button the Greenfoot scenario is resetted (equipollent to the Greenfoot reset button).
- Help: By touching the help menu item a web browser with useful information will be opened.
- Settings: By touching the settings menu item a screen with the following settings will be opened:
    - Scale: If the checkbox is activated the world will always cover the whole screen. If the checkbox is not activated it will be possible that the world only covers a part of the screen.

  - Screen orientation: The user can choose the screen orientation: always portrait modus, always landscape modus or automatic change between portrait and landscape modus due to device rotations.
  - Username: Clicking this item a dialogbox will be opened in which the username can be changed.

4 Installation

Prerequisite for the use of Droidfoot is the installation of Eclipse and the Android SDK. Please have a look at the Android Developer website (http://developer.android.com/sdk/index.html) and follow the instructions given there. It is definitely an advantage if you already have developed an own Android app.

Download the file *droidfoot-2.0.zip* from the Droidfoot website and unzip the file. The included directory *eclipse-workspace-droidfoot* can directly be used as Eclipse workspace. It contains the following projects:

- *Droidfoot*: An Android library project that contains the Android Greenfoot class library und the Android Greenfoot player (inclusive source code und resources). That Android library has to be included in all Android Greenfot projects.
- *DFTemplate*: An Android application project which provides a template for new Android Greenfoot scenarios.
- *DFBalloons*: A sample project which contains an implementation of the well-known Greenfoot balloon scenario.
- *DFSnake*: A sample project which contains an implementation of the well-known snake game (see http://www.greenfoot.org/scenarios/9260).
- *DFLights*: A sample project which contains an implementation of the well-known lights game (see http://www.greenfoot.org/scenarios/1380).
- *DFSmess*: A sample project which contains an implementation of a chess like game called *smess* (see http://www.greenfoot.org/scenarios/354).

Please start Eclipse and switch to the workspace *eclipse-workspace-droidfoot* (> File > Switch Workspace)

5 Basic procedure

In this chapter the general procedure porting a Greenfoot scenario to Android is described. Please note the differences between the original and the Android Greenfoot class library (see Chapter 2). For example, the Android Greenfoot class library does not support keyboard input.

5.1 Default way

As an example the porting of a Greenfoot scenario called *Snake* is used, which implements the well-known snake game.

1. Copy-paste the eclipse android project *DFTemplate* to a new eclipse project called *Snake* (in this example).

2. Please check that the eclipse Android project Droidfoot is included as library (> Properties > Android), what should be true by default performing step 1 before.
3. Open the file *strings.xml* in folder *res/values* and replace the name *DFTemplate* by the name of your scenario (e.g. *Snake*).
4. Rename the package *org.droidfoot.dftemplate* in folder *src* (e.g. *org.droidfoot.snake*) (> Refactor > Rename).
5. Rename the file/class *DFTemplateActivity* in folder *src* in the package being renamed in step 4 (bspw. *SnakeActivity*)
6. Open the file *AndroidManifest.xml* and rename the package-attribute (second line) corresponding to the renaming in step 4 (e.g. *package=“org.droidfoot.snake“*)
7. Copy the java-files of your Greenfoot scenario into the package being renamed in step 4. Please note that all java-file have to define the corresponding package-statement as the first statement (*package org.droidfoot.snake;*).
8. Eliminate all errors in your java-files which can occur due to the differences between the original and the Android Greenfoot class library (see chapter 2).
9. Open the activity file being renamed in step 5, remove the comment in front the statement in method *onCreate* and assign the class object of your world class to the variable *DroidfootActivity.worldClass* (*DroidfootActivity.worldClass = SnakeWorld.class;*)
10. Copy the folders *images* and *sounds* which contain the images and sounds of your Greenfoot scenario into the folder *assets*.
11. Run your project as Android application (> Run as > Android Application)!

5.2 Customized launcher icons

By default, the launcher icons of the Android library Droidfoot are used. However, you can use customized icons for your app. You can do that by copying the corresponding icon files (png or gif) into the appropriate *drawable*-subfolders of the folder *res*. Please note that the icon files must have the name *droidfoot_launcher* (with the extension .png or .gif)!

5.3 Customized help file

By default, a page "no help available" appears when the user touches the help menu item in the action bar. However, you can create a customized help page for your Android Greenfoot scenario. Simply create an html-file with the name *help.html* and place it in the folder *res/raw*. Following the localization rules of Android you can also place localized help files in the corresponding subfolders of folder *res*. For example you could place a german help file named *help.html* in the folder *res/raw-de*.

5.4 Modifications of the Android Greenfoot players

Since version 2.0 of Droidfoot it is possible to make adjustments to the Android Greenfoot player in a simple manner. For this purpose, certain attributes must be set accordingly in the same activity file where the attribute *DroidfootActivity.worldClass* has to be set. If you have used the project DFTemplate as a starting point for your project, as described in section 5.1, you can find the following instructions in the activity file:

```
// help item available in the menu (true/false)?
Settings.help = true;

// speed item and speed slider available in the action bar (true/false)?
Settings.speed = true;
```

```
// background color of the droidfoot activity view
Settings.backgroundColor = Color.rgb(207, 241, 116);

// scale option available in the settings (true/false)?
Settings.scale = true;
// if true the world will always cover the whole screen
Settings.scaleDefault = true;

// orientation option available in the settings (true/false)?
Settings.orientation = true;
// if LANDSCAPE always landscape modus
// if PORTRAIT always portrait modus
// if SWITCH automatic change between portrait and landscape modus due
// to device rotations
Settings.orientationDefault = Settings.SWITCH; // LANDSCAPE, PORTRAIT, SWITCH

// username option available in the settings (true/false)?
Settings.username = true;
// name for the highscore table
Settings.usernameDefault = "Default";
```

The following adjustments are possible:

- **Help**: Shall a help symbol be placed into the action bar or its overflow menu (attribute *Settings.help*)?
- **Speed**: Shall a speed button respectively a speed slider be placed into the action bar (attribute *Settings.speed*)? If the user shall not be allowed to change the speed of the scenario, the attribute has to be set to *false*.
- **Background color**: Via this attribute you can change the background color of the view which contains the Greenfoot world (attribute *Settings.backgroundColor*).
- **Scaling**: Shall the item *Scale* be placed into the settings menu (attribute *Settings.scale*)?
- **Default value of scaling**: Via this attribute you can define if the world shall always cover the whole screen (attribute *Settings.scaleDefault*)?
- **Screen orientation**: Shall the item *Screen orientation* be placed into the settings menu (attribute *Settings.orientation*)?
- **Default value of screen orientation**: Via this attribute you can define the default value of the screen orientation: always portrait modus, always landscape modus or automatic change between portrait and landscape modus due to device rotations (attribute *Settings.orientationDefault*)?
- **Username** Shall the item *Username* be placed into the settings menu (attribute *Settings.username*)?
- **Default value of username**: Via this attribute you can define the default value of the username (attribute *Settings.usernameDefault*)?

5.5 Further modifications

Further modifications are possible by performing appropriate modifications in the eclipse project Droidfoot. It should be noted that the implementation of Droidfoot in this first version 2.0 is not very “clean” in the technical sense. That will change in one of the next versions. However, Droidfoot was extensively tested and should not contain major bugs.

6 Contact

If you have questions, problems or suggestions for improvement fell free to contact me: Dietrich Boles, Universität Oldenburg, Email: boles@informatik.uni-oldenburg.de