HDCOOKBOOK README

This directory contains two folders that are used to produce a set of applications (also called “xlets”) on the disc included with the HD Cookbook, and a third folder that contains documentation. There is a great deal of sharing of source code between the three xlets, and between the xlets and some of the tools that are used to produce the final xlet bundles. For this reason, the source code for everything is stored in a single directory hierarchy, organized by Java package name. Each application picks and chooses the packages that it needs as part of the compilation process.

To see how to build the source, skip down to the “scripts” directory.

You can get an updated copy of Blu-ray source from the repository at https://hdcookbook.dev.java.net.

src
This directory contains all of the source assets used to produce both the applications used during compilation, and the final xlet applications themselves. It also includes LICENSE.html, which contains the rather generous licensing terms that cover the source code in the subdirectories.

com/hdcookbook/tools/bdjo
This directory contains to tool that can read and write BD-J Object, or BDJO files. A BDJO file tells the player where to look for the xlets and other resources that control disc playback. The BDJO tool in this directory will read a XML file containing a representation of the BDJO structure, and emit a binary .bdjo file suitable for inclusion on a disc. The code in this directory is stand-alone, but it does require JDK 1.6 to run.

com/hdcookbook/gunbunny (and subdirectory)
This contains the “Gun Bunny” game xlet. The java code is in this directory and the “util” subdirectory, whereas the images are in the assets subdirectory. Gun Bunny is a stand-alone xlet; it does not depend on other packages in com.hdcookbook. However, Gun Bunny uses the same “Lisa” font that the menu xlet uses. That font is stored with the menu xlet’s assets.

com/hdcookbook/bookmenu
This directory contains one interface definition that is shared between the monitor xlet and the menu xlet.

com/hdcookbook/bookmenu/bdjo

This contains the XML definition of the BDJO file on our disc. It is compiled by com.hdcookbook.tools.bdjo into a binary BDJO file by one of the build scripts.

com/hdcookbook/bookmenu/monitor
This directory contains the code for the monitor xlet. The monitor xlet is a small xlet that manages the other two xlets. In our disc, the monitor xlet launches the game and terminates the menu when the menu xlet requests that it do so. When the game ends, it re-launches the menu xlet. The monitor xlet uses a small utility class from GRIN (com.hdcookbook.grin.util.Debug). In this directory you'll also find the permission request file (PRF) for the monitor xlet. This is used in the codesigning process.

com/hdcookbook/bookmenu/assets
This directory contains the non-code assets used by the menu xlet. The file "menu.txt" contains the GRIN script that defines most of the visual user interface of the menu, and much of its behavior. This file is parsed at xlet startup by the GRIN framework. Also in this directory are the main graphics in the Graphics subfolder, the sound effects in sound.bdmv, and the font used by the menu and by Gun Bunny, in Font. The images in the Graphics directory are compiled by the *mosaic builder* program in GRIN to form two image mosaics that are used in the original disc image. The actual images themselves are not present in the disc image.

In this directory you'll also find a directory called "FakeNetwork." This is used to simulate loading an updated bio for Gun Bunny on players that don't have a network connection.

com/hdcookbook/bookmenu/menu (and subdirectories except "test_assets")
In this directory, you'll find the Java code for the menu xlet, as well as a permission request file (PRF) used for code signing. This xlet relies heavily on the GRIN framework to present graphical and sound assets, and to handle remote control and mouse input.

com/hdcookbook/bookmenu/menu/test_assets
This contains a PNG image that can be used when testing the menu's GRIN script. This PNG is not copied the disc image. It's used with a GUI emulator tool that's part of the GRIN framework. That tool can produce a snapshot of a simulation of the UI state of the xlet at a given point of time. That snapshot looks better if you feed the tool a PNG image that simulates background video; that's what's in this directory.

com/hdcookbook/grin (and subdirectories except “build” and “test”)
This directory contains code an documentation for GRIN, a framework for graphical interactivity. GRIN is really the heart of the menu xlet. GRIN is described in the file doc-files/index.html.

com/hdcookbook/grin/build/mosaic
This directory contains a program that runs on JDK 1.5 (or higher) that automatically combines images into an image mosaic. It reads one or more GRIN show files and locates all of the images used in that file. It then creates an efficient set of image matrixes along with a companion binary metadata file that can be used to reproduce the contents of the original source images. The GRIN framework reads both the matrix and the metadata file at xlet start up time. The desktop java mosaic builder program uses the GRIN framework itself fairly extensively, so when this tool is compiled it includes all of GRIN.

com/hdcookbook/grin/test (and “assets” subdirectory)
These two directories contain the the GRIN show file and image assets that were used in the first xlet to use GRIN. It was called “Ryan’s Life,” and was part of a demo developed by Sun and Technicolor. Ryan’s Life is not used in any way in the Blu-ray disc image in the HD cookbook, but it’s of some historical interest with GRIN, and it’s useful for testing changes to GRIN. However, the menu xlet in com.hdcookbook.bookmenu.menu is a much better guide to how to create a GRIN show than the original Ryan’s Life demo contained here.

com/hdcookbook/grin/test/bigjdk
This directory contains three programs (and assorted other classes) used to test and exercise the GRIN framework, and to test and debug a GRIN show. They run on desktop java (JDK 1.5 or higher). Desktop Java is affectionately known as “big JDK” in some circles. The first program, GrinTestRyan, is of mostly historical interest. It runs a simulation of “Ryan’s Life” on big DJK. The second program is “GenericMain”. It lets you read and display any show file. It shows the show file in a Frame, but the interface to control the show is a command-line interface, so you have to run this from a terminal window.

The most interesting program in this directory is GuiGenericMain. It’s like GenericMain, but it adds a GUI interface to control a show file. With GuiGenericMain, you can read in any valid GRIN show. You’re presented with a tree widget that shows all of the segments in the show. By double-clicking a segment, you tell GRIN to move

the show to the given segment. In this way, you can simulate the behavior of a GRIN show file on desktop Java, and see what it looks like. This program also lets you set the frame rate, step through a GRIN animation frame-by-frame, or even take a snapshot of the UI's state and output a PNG image.

scripts

This directory contains shell scripts that build the source code presented above into elements that can be put on the disc. The choice of shell scripts is a little non-traditional, and yes, I'm dating myself :-) However, an implementation of a common Unix shell is easy to get for Windows, and is already there on OS/X and Linux. To build the source code using these shell scripts, you launch a terminal window running bash, and execute the shell script from this directory (with a command like "./build_xlets.sh"). For windows, I've had good luck with zsh from http://unxutils.sourceforge.net/. Note that the "sh.exe" that comes with unxutils is actually zsh, so it reads its startup commands from ~/.zshrc (where ~ is c:/Documents and Settings/userid). For anything other than casual use, you'll probably want to set up a real build environment using your IDE of choice.

vars.sh

This script contains various variable definitions that need to be set to point to the relevant directories in your repository. HDC_REPOSITORY needs to point to the HD Cookbook repository, that is, the directory that contains the "src" directory given above. HDC_BUILD_DIR should point to a temporary directory somewhere. Incremental build results are stored here, and copied to the BD disc image by the build scripts. HDC_DISC_BDMV points to the BDMV file of your disc image; this is where everything gets copied to. HDC_BDJ_PLATFORM_CLASSES needs to point to a file that contains .class definitions for the BD-J platform. Sorry, we can't provide that on this disc; we hope it will be available from the BDA by the time this book is published. Please check http://hdcookbook.com for news. Finally, HDC_MOUNT_POINT is there for a safety check. Often, the directory that contains the BDMV will be on a separate computer that is mounted. Where appropriate, build scripts check that this directory exists, so that if you forget to mount it the script will fail, rather than copy a bunch of files to your local machine in a place you probably don't want them to go. If you don't mount a filesystem like this, just set HDC_MOUNT_POINT to any directory that exists.

WARNING: classes.zip that is referenced in vars.sh was not available to the public at the time this was written. It is a necessary component to compile a BD-J Xlet. Some have found a set of

usable classfiles to compile against as a ZIP or JAR file within a PC player implementation, though a classes.zip from this source might contain extra API definitions.

The existing vars.sh points to paths that work on my Mac with a PC filesystem mounted. In case it's helpful, here are a set vars.sh that I used to test on the PC:

```
HDC_REPOSITORY=c:/tmp/hdcookbook
HDC_BUILD_DIR=c:/tmp/build
HDC_DISC_BDMV=c:/shared/BookDiscBD/BDMV
HDC_BDJ_PLATFORM_CLASSES=c:/tmp/bdj/classes.zip
HDC_MOUNT_POINT=c:/shared
```

build_bdjo.sh
This script builds the BDJO file on the disc, using the BDJO building tool and the BDJO xml source under the src directory. This script requires JDK 1.6, which you should have installed already if you're on a PC. If you're on a Mac, as of this writing you had to get JDK 1.6 from http://developer.apple.com/java/ (under "Java SE 6.0 Release 1 Developer Preview 6"). This didn't have a JIT for PowerPC, so it wasn't really suitable as the default JRE for OS/X. For this reason, build_bdjo.sh does a bit of shell script magic to look for JDK 1.6 in the place it gets installed on a Mac, even if it's not the default JDK. This bit of the build script should be harmless on PC.

build_xlets.sh
This script builds the three xlets from scratch, makes the image mosaic, and puts the compiled results in JAR files in the target disc image. There is an easy modification you can make to this script to turn off mosaic generation, but a better solution is to make a real build environment in an IDE. This script doesn't sign the xlets, because we don't know of a publicly available BD-J jarsigner. We expect to have one by the time this book is published; please check http://hdcookbook.com for news. If the menu xlet and the monitor xlet aren't signed, you won't be able to launch the game. For our disc image, we manually used Scenarist's jarsigner, by the way.

build_book_menu.sh
This script is only used for debugging; it's not a part of build a disc image. It builds the "GuiGenericMain" program discussed above, under com/hdcookbook/grin/test/bigjdk. This builds the program used by run_book.menu.sh.

run_book_menu.sh

This copies the bookmenu GRIN script file out com/hdcookbook/bookmenu/ assets to the build directory, then runs the GuiGenericMain program compiled by build_book_menu.sh on that script. This allows you to make a change to the GRIN script for the main menu, and see the results almost immediately on in a desktop Java emulation.

www

This contains the website at https://hdcookbook.dev.java.net. That website also contains a SVN (subversion) repository that contains the latest version of the src director. It will either contain the “scripts” directory, or something better, or both :-) Under this directory, you can find the generated javadocs of GRIN, which can be handy.