7-2006

The Initium RJS ScreenSaver: Part 2, UNIX

Douglas A. Lyon
Fairfield University, dlyon@fairfield.edu

Francisco Castellanos

Published Citation

Douglas A. Lyon, Francisco Castellanos, “The Initium RJS ScreenSaver: Part 2, UNIX”, Journal of Object Technology, Volume 5, no. 6 (July 2006), pp. 7-15

JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2006**

Vol. 5. No. 6. July - August 2006

The Initium RJS ScreenSaver: Part 2, UNIX

Douglas A. Lyon and **Francisco Castellanos**

Abstract

This paper describes a Java-based screensaver technology for the Initium Remote Job Submission (RJS) system running on UNIX *XWindows*. Initium RJS is a Java Web Start (JAWS) technology that enables Java-based grid computing. The Initium RJS system uses screensavers to enable CPU scavenging.

A screensaver is a program that activates during a period of user-computer quiescence. Detection of this quiet time enables the use of otherwise wasted CPU cycles. When the period of user-computer quiescence ceases, the screensaver terminates any currently running compute jobs, releasing the computer back for general use. Such a program constitutes a first step toward utilizing otherwise idle compute resources in a grid computing system.

We are motivated to study screen-savers because they represent a minimally invasive technology for volunteering CPU services. Typically, computers are used between 40 and 60 hours out of a 168-hour week. This represents approximately 35% utilization. Our theory is that a screen-saver based *cycle scavenging* will improve this number dramatically.

We are motivated to provide a Java-based environment in order to capitalize on Java's inherent heterogeneity. This makes a larger universe of grid-compute servers available, without requiring changes to the computational program.

This paper is part 2 of a 5 part series on Java-based screensavers. Part 1 addressed the creation of screensavers on MS Windows platform systems. Parts 2 and 3 address the Linux and Macintosh-based screensavers. Part 4 addresses the automatic deployment and installation of the screensavers. Part 5 speaks to the problem of screensaver integration with the Initium RJS system.

Initium RJS is a joint project between DocJava, Inc. and Fairfield University. The goal of the Initium RJS system is to improve the accessibility of grid computing to Java developers.

Cite this column as follows: Douglas Lyon and Francisco Castellanos: "The Initium RJS ScreenSaver: Part 2, UNIX", in *Journal of Object Technology*, vol. 5, no. 6, July-August 2006, pp. 7-15 http://www.jot.fm/issues/issue_2006_07/column1

1 INTRODUCTION

This paper describes the application of our technology to the UNIX platform. Our previous paper covered the Windows platform, and our next paper will cover the Macintosh platform. Our goal is to make use of these screensavers in the *Initium RJS system*, our grid-computing framework. While we acknowledge that the form of the screensaver installation, as presented in this article, is tedious and error-prone, we will (in a future paper) describe the automatic installation of a compute-serving screensaver. It is our hope that our system will help in the promotion of Java as a grid-based computing platform.

This paper shows how to create a screensaver using an existing framework called *SaverBeans*. The *SaverBeans* development kit is an open-source, freely-available framework consisting of both C/C++ and Java code. The kit is available for both the Windows and Linux systems. However, it is not available for the Macintosh. The alternative to creating a Macintosh-based screensaver is to run X-windows under the Macintosh (an atypical use of the Macintosh).

This paper introduces the *SaverBeans SDK* for UNIX with an *XWindows* GUI. The idea of using screensavers for Java-based grid computing is not new [SaverScience]. However, the work was not continued and present implementations do not make use of screensavers for grid computing [George].

2 SAVERBEANS – A JAVA SCREENSAVER FRAMEWORK

The *SaverBeans Screensaver SDK* is a project of the *Java.net* group. It provides a set of native subroutines that invoke Java methods when a screensaver activates and deactivates (i.e., experiences a change of state). The *SaverBeans* SDK has its roots in the JDIC project (**JD**esktop **I**ntegration **C**omponents). The **JDIC** project aims to make Java™ technology-based applications ("Java applications") first-class citizens of current desktop platforms without sacrificing platform independence. Its mission is to enable seamless desktop/Java integration [JDIC1].

2.1. Building the SaverBeans SDK

In order to install the screensaver in a UNIX/*XWindows* workstation, you must have the standard *xscreensaver* installed. While most workstation installations come with the screensaver already installed, most installations do not include the source code, that is needed for the native method builds. The *xscreensaver* installation includes a daemon that detects quiescence [Zawinski]. To determine the version of the *xscreensaver* type:

```
xscreensaver &
xscreensaver-command -version
```

The computer responds with:

```
XScreenSaver 4 21
```

The '&' in the first line place the *xscreensaver* in the background.

The method for installation of the *xscreensaver* may vary from platform to platform. Download http://www.jwz.org/xscreensaver/xscreensaver-4.23.tar.gz and uncompress and untar the *xscreensaver* file and cd into that directory, then run configure and make. Type:

```
tar -vxzf xscreensaver-4 23 tar gz
cd xscreensaver-4 23/
 /configure
make
make install
```

Even if the *xscreensaver* binaries are already installed, you may still need the source code to compile the *SaverBeans* SDK. This has been bundled in a special distribution available at http://www.docjava.com. As an alternative, you can download the *SaverBeans* SDK from https://jdic.dev.java.net/files/documents/880/12349/saverbeans-sdk-0.2-beta.zip and unzip. Type:

```
cd saverbeans-sdk-0 2-beta
unzip saverbeans-startup zip
cp build properties sample build properties
```

Alter the *SaverBeans* path in the *build.properties* directory to reflect the installation location of the SDK. For example:

```
saverbeans path=/opt/saverbeans
```

becomes:

```
saverbeans path=
       saverbeans path=/home/lyon/current/ssbeta/saverbeans-
       sdk-0 2-beta/
```

Make sure that the Java virtual machine is in the *PATH* and that the *JAVA_HOME* is set correctly:

```
show docjava com[lyon]113  which java
/usr/java/jdk1 5 0_04/bin/java
show docjava com[lyon]114  echo $JAVA_HOME
/usr/java/jdk1 5 0_04/
```

Finally build the project using:

```
ant dist
```

This will generate a directory called *dist*. The *dist* directory contains the distributable files. Edit the *Makefile* in the *dist/bouncingline-unix* directory altering *jdkhome* and *xscreensaverhome* to valid directories. For example:

```
jdkhome=/usr/java/jdk1_5_0_04
xscreensaverhome=/home/lyon/current/ssbeta/saverbeans-sdk-0_2-
        beta/xscreensaver-4_23
```

To compile the native methods for the current platform, type:

```
Make
```

Notice the files generated: *bouncingline-bin* and *bouncingline.o*.

The following section describes how to deploy the screensaver to an *XWindows* system under UNIX.

2.3 Deploying

Now that the binaries have been generated, these files can be used to deploy the screensaver into an *XWindow* platform. With the native method framework in place, a wide variety of different screensavers can be authored, in Java, without having to rebuild the native methods.

The *.xscreensaver* file needs to be modified to include the *bouncingline* screensaver. The *xscreensaver-demo* program generates this file and places it into the users' home. Type:

```
xscreensaver-demo
```

Look into the users' home directory to verify the existence of the *.xscreensaver* file. To inform the *xscreensaver* program that you have a new screensaver, you should edit the *.xscreensaver* file in your home directory. To add the *bouncingline* screensaver to the *.xscreensaver* file, use:

```
programs
        'Bouncingline  java '/home/lyon/ss/bouncingline -root -
        jdkhome /usr/java/jdk1_5_0_04
…
```

Now execute the screensaver by typing:

```
xscreensaver-demo
```

Figure 2.3-1 shows the screensaver dialog box, with the *bouncingline* screensaver selected.

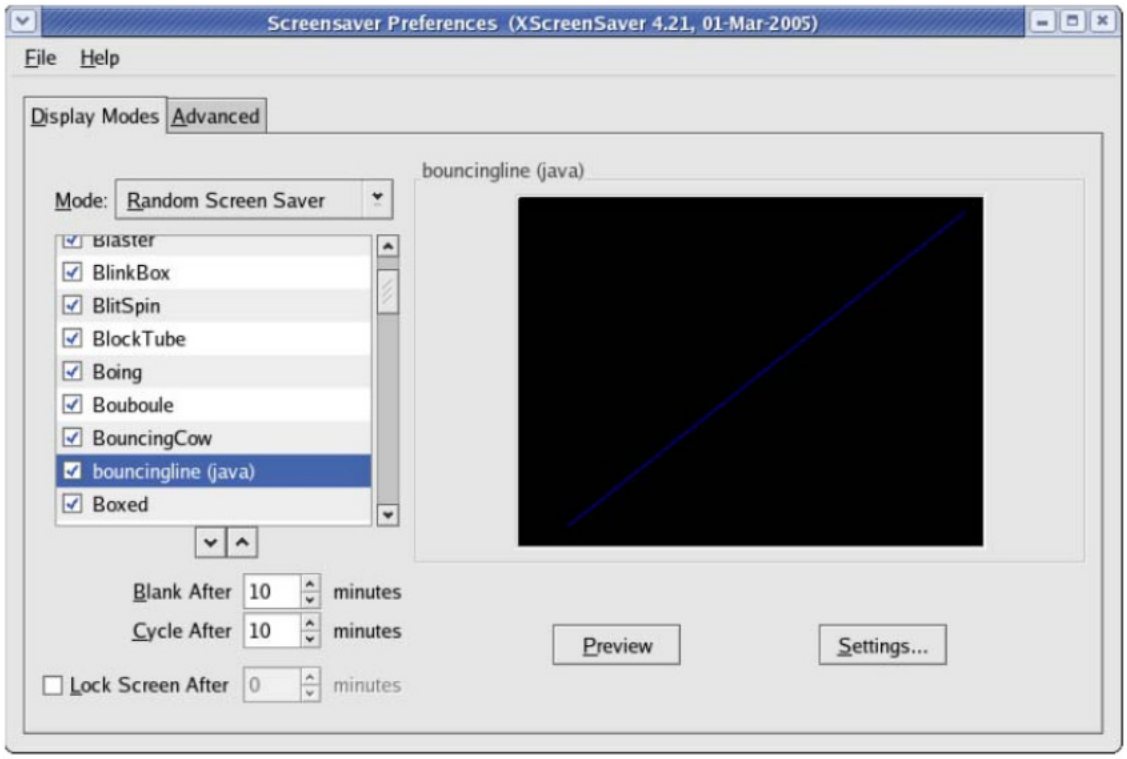

Figure 2.3-1. The Screensaver Dialog

For ease of use, the files required to run the screensaver are placed in a single directory called *ss*. No root permissions are required to install custom screensavers that reside in the home directory.

3 MAKEFILE SYNTHESIS

Ant is a multi-platform, java-based, make-like utility. The *ScreenSaver* SDK uses *ant* as well as custom *ant* tasks, stored in a jar file called *saverbeans-ant.jar*. In the course of running the *ant dist* a make file is synthesized. This is created for UNIX and Windows. The contents of the *dist* directory include:

```
bouncingline-unix/     bouncingline-win32/
bouncingline-unix zip  bouncingline-win32 zip
```

The *bouncingline-unix* directory contains the files:

```
bouncingline          bouncingline jar   COPYING      saverbeans-
      api jar
bouncingline-bin*  bouncingline o     Makefile
bouncingline c     bouncingline xml   README txt
```

The *Makefile* is generated from a template. We have altered this template in order to generate a file that is somewhat more automatic in its installation. For example:

```
# Set this to your Java home directory
jdkhome=${JAVA_HOME}

#  Set  this  to  where  the  xscreensaver  source  bundle  is
      installed
xscreensaverhome=  /  /xscreensaver-4 23
```

Further, we have installed a version of the *xscreensaver* that makes compilation somewhat more automatic. In order to accomplish these changes, we altered the *Makefile.template* file in:

```
saverbeans-
     ant/org/jdesktop/jdic/screensaver/autogen/resources/unix
```

The template contains a file with variables that to help drive the *Makefile* synthesizer. The creation of *Makefiles* in this way is unique, as far as we know.

4 WRITING A SCREENSAVER

The following code shows how to write a screensaver by subclassing the *SimpleScreenSaver* class:

```
public class Test1
        extends SimpleScreensaver {

    public void paint Graphics g) {
        Component c = getContext ) getComponent ),
        int x = 0;
        int y = c getHeight ) / 2,
        g setColor Color WHITE);
        g setFont new Font 'Dialog', Font BOLD, 30)),
        g drawString 'Initium RJS see
               http //www docjava com', x, y);
    }


    public static void main String ] args) {
        new ScreensaverFrame new Test1 )) setVisible true),
    }
}
```

The *main* method makes an instance of a *ScreensaverFrame*, used for testing. The *ScreensaverFrame* is a subclass of the *JFrame* and sets the *context* of the *SimpleScreensaver*. This context is an instance of a *Component* class. In the case of a *JFrame* it is also an instance of a *Container*. Knowing this, we are at liberty to establish a layout with standard swing components as a part of our screensaver. We need to obtain the *Container* of our *Component* via focus traversal in the *init* method. For example:

```
public class Test2
        extends SimpleScreensaver {
    JPanel buttonControlPanel = getButtonControlPanel ();

    private JPanel getButtonControlPanel () {
        JPanel jp = new JPanel ();
        jp setLayout new FlowLayout ());
        jp add new RunButton 'ok'( {
            public void run () {
                System out println getText ());
            }
        });
        jp add new RunButton 'cancel'( {
            public void run () {
                System out println getText ());
            }
        });
        return jp;
    }

    public void init () {
        Container c =
        super getContext () getComponent () getFocusCycleRootAnc
        esto
r ();
        c add buttonControlPanel();
    }

    public void paint Graphics g( {
        Component c = getContext () getComponent ();
        c paint g();
    }


    public static void main String [] args( {
        new ScreensaverFrame new Test2 ()) setVisible true();
    }
}
```

The *getFocusCycleRootAncestor* enables the addition of an arbitrary swing panel to the display. This is useful for creating GUIs needed for controlling the compute server.

5 SUMMARY

This paper addressed the issue of implementing a Java-based screensaver under the X-Window system, as well as providing a solution to the automation of installation and

deployment for these systems. Focus traversal techniques helped with the swing programming. We find that focus traversal works for screensavers written for either MS Windows or XWindows. Unlike the MS Windows screen saver, no system administration privileges were required for installation. Further, unlike MS Windows, Linux does not need to run a windows server and therefore may not have a screen saver. In such a case, no screen-saver based system can take advantage of the machine.

Part 3 will describe how to implement a Java-based screen saver on the Macintosh operating system. Screensaver integration with the grid-based middleware and automatic screen saver deployment are the topics of Parts 4 and 5 of the Initium RJS paper sequence.

REFERENCES

[George] Private communications with William L. George, Ph.D., National Institute of Standards and Technology, 100 Bureau Dr. Stop 8911, Gaithersburg, MD 20899-8911, email: wgeorge@nist.gov, March 15, 2006.

[JDIC1] Java.net: "JDIC project home", https://jdic.dev.java.net/ Last accessed March 14, 2005.

[SaverScience] William L. George and Jacob Scott, "Screen Saver Science: Realizing Distributed Parallel Computing with Jini and JavaSpaces", in *2002 Conference on Parallel Architectures and Compilation Techniques* (PACT2002), Charlottesville, VA, September 22-25, 2002.

[Zawinski] Jamie Zawinski: "A screen saver and locker for the X Window System" http://www.jwz.org/xscreensaver/

About the authors

After receiving his Ph.D. from Rensselaer Polytechnic Institute, **Dr. Lyon** worked at AT&T Bell Laboratories. He has also worked for the Jet Propulsion Laboratory at the California Institute of Technology. He is currently the Chairman of the Computer Engineering Department at Fairfield University, a senior member of the IEEE and President of DocJava, Inc., a consulting firm in Connecticut. E-mail Dr. Lyon at Lyon@DocJava.com. His website is http://www.DocJava.com.

Francisco Catellanos. Earned his bachelors degree with honors in Computer Science at Western Connecticut State University. Francisco Castellanos worked at Pepsi Bottling Group in Somers, NY as a software developer. Currently he is working on a thesis to complete his Master's Degree in Computer Engineering from the Fairfield University. His research interests include grid computing. Francisco Castellanos is also employed by Access Worldwide in Boca Raton, FL as a software developer. He can be contacted at fsophisco@yahoo.com.