

5-2006

The Initium RJS Screensaver: Part 1, MS Windows

Douglas A. Lyon

Fairfield University, dlyon@fairfield.edu

Francisco Castellanos

Follow this and additional works at: <https://digitalcommons.fairfield.edu/engineering-facultypubs>

Copyright 2006 Journal of Object Technology

Archived with permission from the copyright holder.

Peer Reviewed

Repository Citation

Lyon, Douglas A. and Castellanos, Francisco, "The Initium RJS Screensaver: Part 1, MS Windows" (2006). *Engineering Faculty Publications*. 55.

<https://digitalcommons.fairfield.edu/engineering-facultypubs/55>

Published Citation

Douglas A. Lyon, Francisco Castellanos, "The Initium RJS Screensaver: Part 1, MS Windows", *Journal of Object Technology*, Volume 5, no. 4 (May 2006), pp. 7-16

This item has been accepted for inclusion in DigitalCommons@Fairfield by an authorized administrator of DigitalCommons@Fairfield. It is brought to you by DigitalCommons@Fairfield with permission from the rights-holder(s) and is protected by copyright and/or related rights. **You are free to use this item in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses, you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.** For more information, please contact digitalcommons@fairfield.edu.

The Initium RJS Screensaver: Part 1, MS Windows

By **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 Microsoft Windows. 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.

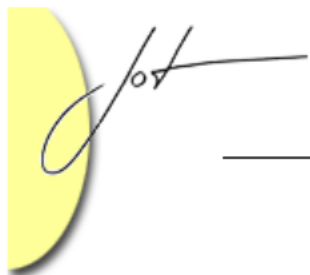
A screensaver is a program that activates during a period of user-computer quiescence. Detection of this quiet time enables the use of otherwise wasted CPU cycles. When the period of user-computer quiescence ceases, the screensaver terminates any currently running compute jobs, releasing the computer 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. Screen-saver based *cycle scavenging* improves 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 1 of a 5 part series on Java-based screensavers. Part 1 addresses 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 integration of the screensavers 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 screen-saver based grid computing available the Java development community.



1 INTRODUCTION

We are interested in screen-saver technologies, in Java, in order to facilitate a minimally invasive computing service able to make use of otherwise unused computational resources. There is little written on the subject of screen-saver based grid computing, in Java.

Screen-saver based grid computing systems are not new [Boinc] but their use for Java computing is. Also, Java-based screensavers have, in the past, been restricted to MS Windows and Xwindows (UNIX)-based systems. The idea for using screensavers to accelerate Java grid computing has been mentioned in literature, but implementations have not been forthcoming [SaverScience].

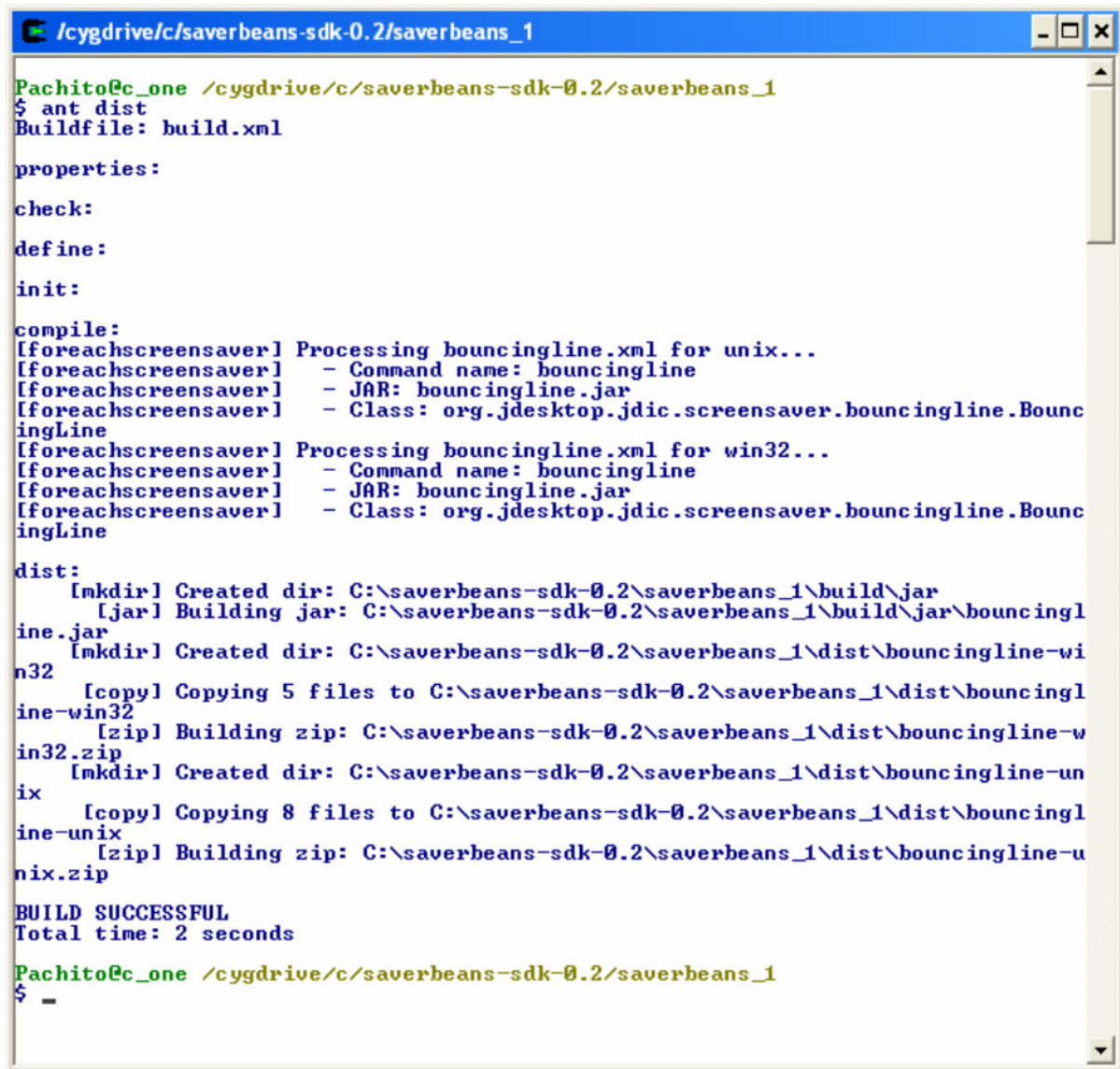
We theorize that the creation of a Java-based screensaver that is both cross-platform and automatically installed will help in the promotion of Java as a grid-based computing platform. This paper shows how to create a screensaver using an existing framework called *SaverBeans*. The *SaverBeans* development kit is an open-source, freely-available framework consisting of both C/C++ code and Java code.

2 A JAVA-BASED SCREENSAVER

The *SaverBeans Screensaver SDK* project, under the *Java.net* group, provides a set of native subroutines that invokes Java methods in the screensaver. The *SaverBeans* SDK has its roots in the JDIC project (**J**Desktop **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]. The kit is available from [JDIC2] as an open-source distribution.

2.1. Building the SaverBeans SDK

Once the development kit has been downloaded, create a copy of the *saverbeans_startup* directory and rename it to *saverbeans_1*. Figure 2.1-1 shows the contents of the *SaverBeans* startup directory.



```
/cygdrive/c/saverbeans-sdk-0.2/saverbeans_1
Pachito@c_one /cygdrive/c/saverbeans-sdk-0.2/saverbeans_1
$ ant dist
Buildfile: build.xml

properties:

check:

define:

init:

compile:
[foreachscreensaver] Processing bouncingline.xml for unix...
[foreachscreensaver]   - Command name: bouncingline
[foreachscreensaver]   - JAR: bouncingline.jar
[foreachscreensaver]   - Class: org.jdesktop.jdic.screensaver.bouncingline.Bounc
ingLine
[foreachscreensaver] Processing bouncingline.xml for win32...
[foreachscreensaver]   - Command name: bouncingline
[foreachscreensaver]   - JAR: bouncingline.jar
[foreachscreensaver]   - Class: org.jdesktop.jdic.screensaver.bouncingline.Bounc
ingLine

dist:
  [mkdir] Created dir: C:\saverbeans-sdk-0.2\saverbeans_1\build\jar
  [jar] Building jar: C:\saverbeans-sdk-0.2\saverbeans_1\build\jar\bouncingl
ine.jar
  [mkdir] Created dir: C:\saverbeans-sdk-0.2\saverbeans_1\dist\bouncingline-wi
n32
  [copy] Copying 5 files to C:\saverbeans-sdk-0.2\saverbeans_1\dist\bouncingl
ine-win32
  [zip] Building zip: C:\saverbeans-sdk-0.2\saverbeans_1\dist\bouncingline-w
in32.zip
  [mkdir] Created dir: C:\saverbeans-sdk-0.2\saverbeans_1\dist\bouncingline-un
ix
  [copy] Copying 8 files to C:\saverbeans-sdk-0.2\saverbeans_1\dist\bouncingl
ine-unix
  [zip] Building zip: C:\saverbeans-sdk-0.2\saverbeans_1\dist\bouncingline-u
nix.zip

BUILD SUCCESSFUL
Total time: 2 seconds

Pachito@c_one /cygdrive/c/saverbeans-sdk-0.2/saverbeans_1
$ -
```

Figure 2.1-1. The Contents of the SaverBeans Startup Directory

The *build directory* contains the libraries and platform specific files needed in the building process. The *src directory* contains the documentation, packages, and Java code used by the screensaver.

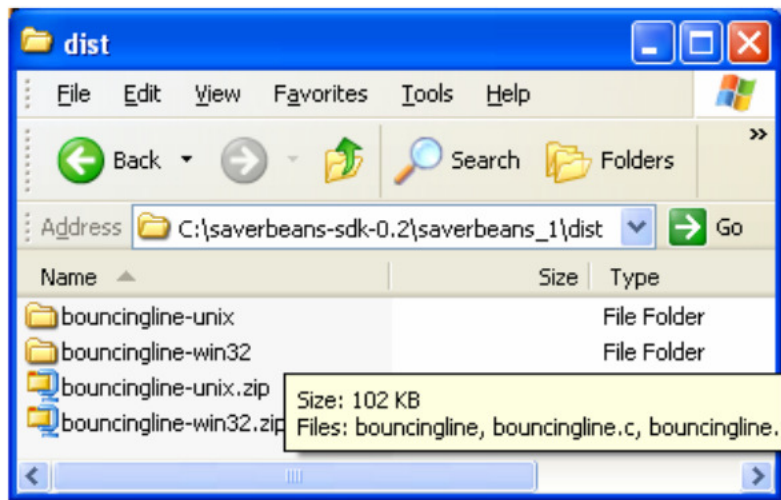


Figure 2.1-2. The Dist Directory is created during the Ant build.

The *dist* directory is created automatically during the compilation and construction process, as shown in Figure 2.1-2. Platform specific files of the screensaver are placed in this directory during the construction process. Copy the *building.properties.sample* file to a new file called *build.properties*. This file contains the SDK home property, and this must be set correctly. For example:

```
sdk.home=C:\\j2sdk1.4.2_04
```

The *build.xml* file contains the *ant* build code. In order to perform a correct ant build, you must set the *saverbeans.path* in the *build.xml* file. To enable ant compilation, use:

```
saverbeans.path=C:/saverbeans-sdk-0.2
```

2.2 Compiling, Debugging and Deploying

Under windows, we install the Cygwin system [Cygwin]. Using the command prompt, change directory to *saverbeans_1*. Type *ant clean* in order to remove anything left over from the last build. Type *ant debug* in order to compile and run the project. A frame will open, displaying the demo screensaver (a bouncing line).

In order to create a distribution, type *ant dist*. This step creates the *dist* directory containing windows specific files, ready for installation.

Install the screensaver by changing to the *bouncingline-win32* folder. This directory contains three files of interest: *bouncingline.jar*, *bouncingline.src*, *SaverBeans-api.jar*. Copy these files into the Windows system directory. The exact location is a function of the windows version:

For Windows XP, the location is: windows/system32

For Windows 98, the location is: windows/system



For Windows NT, the location is: winnt/system
Figure 2.2-1 shows the files after deployment.

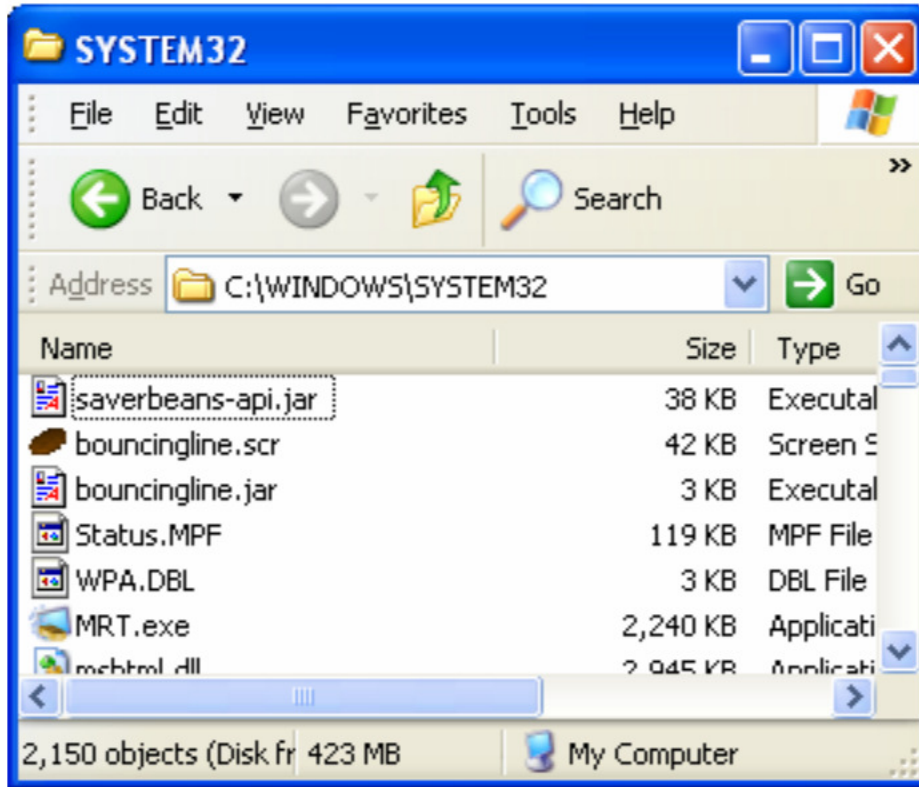


Figure 2.2-1. A Sample Windows XP Deployment

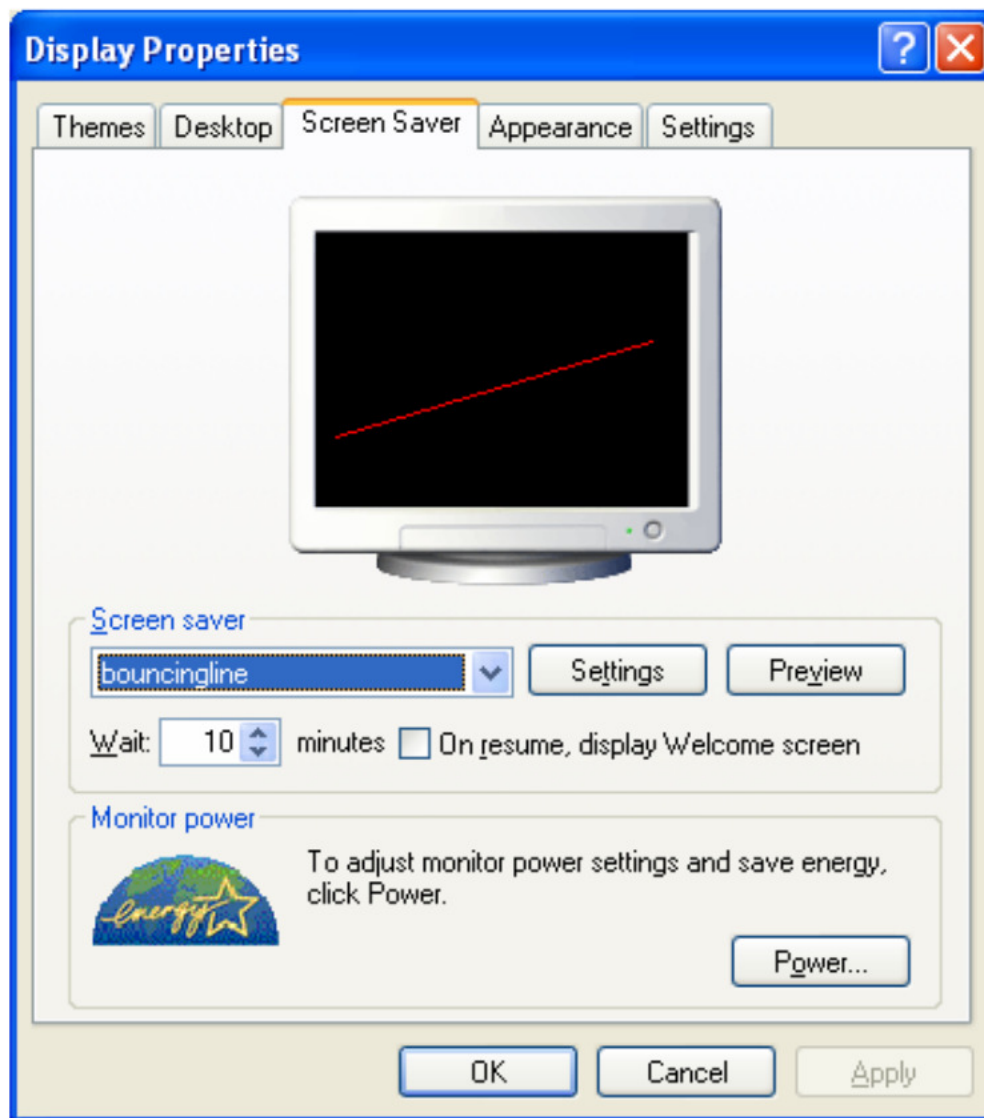


Figure 2.2-2. Setting the Screensaver

The last step is to set the screensaver to the *bouncingline* demo. Open the control panel and select the *bouncingline* screensaver in the screensaver tab. We set the time for “Wait” and apply the changes. The *bouncingline* screensaver is triggered automatically after the entered time has passed (given an idle machine, as shown in Figure 2.2-2).

The installation of a screensaver, in windows, requires that the user have write permission to the windows system directories. Typically, this is a non-issue, for a single users’ machine. However, in an industrial setting, this can be a showstopper.



3 IMPLEMENTATION DETAILS

3.1 Screensaver class

As part of the startup package, the code of the *bouncingline* screensaver is included. The code is found in the *BouncingLine* class located in *src* directory. The complete path is `<startup project location>\src\java\org\jdesktop\jdic\screensaver\bouncingline`.

```
package org.jdesktop.jdic.screensaver.bouncingline;

public class BouncingLine extends SimpleScreensaver {
    public void init() {}
    public void paint( Graphics g ) {}
    public void destroy() {}
}
}
```

There are a few points to notice about this class. The *BouncingLine* extends *SimpleScreensaver*, an abstract class that is part of the *SaverBeans* API. Developers should extend either *SimpleScreensaver* or *JOGLScreensaver* (an OpenGL screensaver that typically makes use of 3D graphics).

SimpleScreensaver declares the abstract paint method. The frame is rendered by a regular callback to the *paint* method. In the *BouncingLine* class, the *paint* method erases the previous painted line and draws the new line. For example:

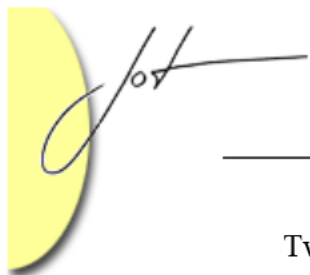
```
public void paint( Graphics g ) {
    Component c = getContext().getComponent();
    int width = c.getWidth();
    int height = c.getHeight();

    // Erase old line:
    g.setColor( c.getBackground() );
    g.drawLine( p1.x, p1.y, p2.x, p2.y );

    // Move points and bounce off walls:
    bounce( p1, dir1, width, height );
    bounce( p2, dir2, width, height );

    // Draw new line:
    g.setColor( lineColor );
    g.drawLine( p1.x, p1.y, p2.x, p2.y );
}
```

The *SimpleScreensaver* class extends the abstract *ScreensaverBase* class [SaverBeans]. *SimpleScreensaver* implements the *renderFrame* method, which is used as a call-back method from the *SaverBeans* framework.



Two other callback methods include *init* and *destroy*. These are called when the screensaver starts and stops. The *init* method is called once, and only once, after the screensaver starts. It will not be called, for example, if screen resolutions change. Our implementation of the *init* method follows:

```
public void init(){
    ScreensaverSettings settings =
getContext().getSettings();
    Component c = getContext().getComponent();
    int width = c.getWidth();
    int height = c.getHeight();
    randomizePoint( p1, width, height );
    randomizePoint( p2, width, height );
    dir1 = new Point( randomVector(), randomVector() );
    dir2 = new Point( randomVector(), randomVector() );
    String colorOption = settings.getProperty( "color" );
```

The second useful method defined in *ScreensaverBase* is the *destroy* method, which we will not need now.

3.2 Screensaver settings

The *SaverBeans* framework provides an XML file that is used to store screensaver properties. In the case of the *bouncingLine* screensaver, the file contains the following XML, located in *src/bouncingline.xml*:

```
<screensaver name="bouncingline" _label="Bouncing Line">
  <command arg="-root"/>
  <command arg="-jar bouncingline.jar"/>
  <command arg="-class
    org.jdesktop.jdic.screensaver.bouncingline.BouncingLine"/>

  <file id="jdkhome" _label="Java Home (blank=auto)" arg="-
    jdkhome %" />

  <select id="color">
    <option id="blue" _label="Blue Line" /> <!-- default -->
    <option id="green" _label="Green Line" arg-set="-color
      #00ff00" />
    <option id="red" _label="Red Line" arg-set="-color
      #ff0000" />
  </select>

  <_description> □.</_description>
</screensaver>
```

While the use of an XML file to establish these properties seems cumbersome, it is required because of the framework. These XML files do not need to be altered, once they are established for a screensaver with a stable class name.



4 CONCLUSION

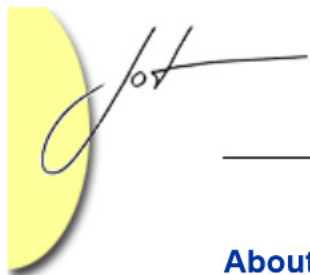
Screensavers in a heterogeneous computing environment are an enabling technology for grid computing based on cycle scavenging. The screensaver software development kit provides a framework for the construction of screensavers in Java.

The nature of the deployment and construction of the screensaver is labor intensive, error-prone and tedious. In our next article we will describe the deployment and construction of the screensaver in an Xwindows environment, under UNIX. A follow-on article will show how to create a Java-based screensaver for the Macintosh operating system using the Apple Quartz interface (as opposed to running X on the Mac). After the three platforms are described, we show how the installation of screensavers can be made nearly automatic, using Webstart technologies.

A basic limitation of the screensaver is the requirement that the user have write access to the windows systems directory. The question of how to overcome this limitation remains a topic of future work.

5 REFERENCES

- [Boinc] <http://boinc.berkeley.edu/> Last accessed March 14, 2005.
- [Cygwin] <http://www.cygwin.com> Last accessed March 14, 2005.
- [JDIC1] Java.net : “JDIC project home”, <https://jdic.dev.java.net/> Last accessed March 14, 2005.
- [JDIC2] <https://jdic.dev.java.net/documentation/incubator/screensaver/index.html> Last accessed March 14, 2005.
- [SaverBeans] <https://jdic.dev.java.net/documentation/incubator/screensaver/index.html> 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.



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 Castellanos. 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.