JOURNAL OF OBJECT TECHNOLOGY

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

Vol. 5. No. 7, September-October 2006

# Initium RJS: A Macintosh Screensaver in Java, Part 3

*By Douglas A. Lyon, Pawel Krepsztul and Francisco Castellanos*

**Abstract**

We describe how to create a Java-based screensaver for a Macintosh. A screensaver is a program that automatically runs when the computer enters a quiescent state. Screensaver frameworks enable CPU scavenging. CPU scavenging enables the use of otherwise wasted CPU cycles.

Screensavers are a minimally-invasive technology for volunteering CPU services. Computers typically have 23% utilization (40 out of 168 hours, per week) or less. Screensaver-based *cycle scavenging* improves utilization dramatically.

This paper is part 3 of a 5 part series on Java-based screensavers. Parts 1 and 2 addressed the creation of screensavers on Ms Windows and XWindows platforms. These screensavers are a part of the Initium Remote Job Submission system (Initium RJS). Initium RJS is a joint project between DocJava, Inc. and Fairfield University. The goal of the Initium RJS system is to make grid-based computing, in Java, a little easier.

## 1 INTRODUCTION

This paper is the 3rd in a series on screensavers in Java and describes the application of our technology to the Macintosh platform. Our previous papers covered the Windows and Unix platforms. With this paper, we have covered Java-based screensavers on three major platforms. Our goal is to make use of these screensavers in the *Initium RJS system*, our grid-computing framework.

We show how to create a screensaver using a custom framework. Our past work described an existing framework, called the SaverBeans development kit, an open-source, freely-available framework consisting of both C and Java code. The kit is available for both the MS 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. Our impression is that this is an idiosyncratic use of the Macintosh, and users prefer a solution that makes use of the native window manager (quartz) of the Macintosh.

## 2 A JAVA SCREENSAVER FRAMEWORK

The Macintosh has a development IDE available for simultaneous creation of both Objective C and Java programs. The IDE is freely available, as a part of the *XCode* distribution and is called *ProjectBuilder*.

The basic idea behind our screensaver is that it will run a Java Web Start Application. This enables deployment of updates to our screensaver without having to reinstall it.

We start with an Objective C program (a file with a *.m* suffix), inspired by [Christensen]:

```
//
//  ScreenView.m
//  ScreenSaver
//

#import "ScreenView.h"


@implementation ScreenView

int i = 0;
- (void)animateOneFrame {
      //- (void)startAnimation{
      NSBezierPath *path;
      NSRect rect;
      NSSize size;
      NSColor *color;size = [self bounds].size;

      if(i==0){
       NSLog(@" First time   %d      SS start now", i);
       //Call to java class
       [NSClassFromString(@"RunCS")
       newWithSignature:@"(Ljava/lang/String;)",@"start"];
}

rect.size=NSMakeSize(SSRandomFloatBetween(size.width/100,size.
       width/10),
SSRandomFloatBetween(size.height/100,size.height/10));
rect.origin = SSRandomPointForSizeWithinRect(rect.size,[self
       bounds]);
if (SSRandomIntBetween( 0, 1 ) == 0) {
    path = [NSBezierPath bezierPathWithRect:rect];
} else {
    path = [NSBezierPath bezierPathWithOvalInRect:rect];
}
color = [NSColor colorWithCalibratedRed:(SSRandomFloatBetween(
       0.0, 255.0)/ 255.0)
         green:(SSRandomFloatBetween( 0.0, 255.0 ) / 255.0)
         blue:(SSRandomFloatBetween( 0.0, 255.0 ) / 255.0)
         alpha:(SSRandomFloatBetween( 0.0, 255.0 ) /255.0)];
[color set];
i++;
[path fill];
}

- (void)stopAnimation{
      //Call to java class to stop dhry.main.app
    [NSClassFromString(@"RunCS")
      newWithSignature:@"(Ljava/lang/String;)",@"stop"];
      NSLog(@"SS  stop now %d  ", i);
}


@end
```

If the screensaver is started for the first time, the *counter* (i=0) is zero. Events are logged to the console using *NSLog*. The *stopAnimation* method is invoked when the screensaver terminates. The *RunCS* code follows:

```
//
//  RunCS.java
//  ScreenSaver
//

import com.apple.cocoa.foundation.*;
import com.apple.cocoa.application.*;
import java.io.IOException;
import java.util.Properties;
import java.io.File;
import java.io.FileOutputStream;

public class RunCS {
    private static String tmpDir =
       System.getProperty("java.io.tmpdir");
    private static String fileSep =
       System.getProperty("file.separator");
    public final static File killFile = new File(tmpDir +
                            fileSep +
                            "killcs");

    private static void startCs()  {
        if (killFile.exists())
            killFile.delete();

        final  String wsMacLocation = fileSep +
          "Applications" +
          fileSep +
          "Utilities" +
          fileSep +
          "Java" +
          fileSep +
          "Java Web Start.app" +
          fileSep +
          "Contents" +
          fileSep +
          "MacOS" +
          fileSep +
          "Java Web Start";
        String url = "http://show.docjava.com:8086/" +
          "book/cgij/code/jnlp/net.rmi.rjs.pk.main.CsMain.jnlp";

        System.out.println("webstart is here:"+ wsMacLocation);
```

```
            String args[] = {
                wsMacLocation,
                url
            };
            Runtime rt = Runtime.getRuntime();
            try {
                rt.exec(args, null, null);
            } catch (IOException e) {

            }
            System.out.println("finished!");
        }
        public void stopCs(){
          killFile.mkdir();
          System.out.println("Stoping the CS");
        }

        public RunCS (String cmd) {
          if (cmd.equals("start")){
              System.out.println("start");
              startCs();
          } else stopCs();
        }
    }
```

The screensaver runs a computation server using the *RunCS* class. The*RunCS* constructor is created with a *String* argument. If the argument is equal to “start”, then the *startCs* method is invoked. The method checks to see if the semaphore file, *killcs* exists, and deletes it, if it does. A thread checks the file, and if it exists, the computation server is terminated. The semaphore file is stored in a temporary directory. If the argument to the constructor is “stop” then the *stopCs* method is invoked. The *stopCs* method creates the *killcs* file to trigger termination. The screensaver (written in Objective C) invokes the Java program using an objective C to Java bridge [Lyon and Huntley] [Monitzer].

## 2.1. Installing The Screensaver

We have created a web start method for automatically deploying and installing the screensaver to a Mac. The URL is available at http://show.docjava.com:8086/book/cgij/code/jnlp/net.rmi.rjs.MacScreenSaverUtils.jnlp and provides for an installation using a technique we call *beaming over* the files. The basic idea is that the screensaver files are transferred from the web server to the local disk. There they are uncompressed and placed into the proper location for user screensavers (~/Library/Screensavers/). The code for this type of beam over operation follows:

```
public class MacScreenSaverUtils {

    private static String screenSaverDirectoryName =
            SystemUtils.getUserHome() +
            "/Library/Screensavers/" ;
    private static File outputJarFile = new File(
            screenSaverDirectoryName+
            "screenSaver.jar");
    //download the screensaver in:
```

```
//
    http://show.docjava.com:8086/book/cgij/code/jnlp/libs/mac
    /screenSaver.jar

public static void downloadScreenSaverJar()
        throws IOException {

    URL screenSaverUrl = getResourceUrl();
    UrlUtils.getUrl(screenSaverUrl,outputJarFile);

}
public static void testStartRunCheckThread(){
    new CheckForDeathJobProperties();
    guiKillCS();
}

private static void guiKillCS() {
    while(In.getBoolean(("keep cs running?"))){
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            In.message(e);

        }
    }
    putPrefPropToDie();
}

private static void putPrefPropToDie() {
    Preferences p = Preferences.systemRoot();
    p.put(csKillKey, "true");
}

private static final String csKillKey = "timeToKillCS";
public static void startRunCheckThread(){
    final Preferences p = Preferences.systemRoot();
    p.put(csKillKey, "false");
    new RunJob(1){
        public void run(){
            final Preferences p = Preferences.systemRoot();
            String value = p.get(csKillKey, "false");
            if (value == null) return;
            if (value.equals("false")) return;
            killCS();//kill the CS
        }
    };
}
private static class RunCheckThread extends Thread {

    public void run() {
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
        }
        final Preferences p = Preferences.systemRoot();
        String value = p.get(csKillKey, "false");
        if (value == null) return;
        if (value.equals("false")) return;
        killCS();//kill the CS
    }
}
```

```
private static void killCS() {
    System.out.println("cs is dead");
    System.exit(0);
}

private static URL getResourceUrl() throws
   MalformedURLException {
    URL screenSaverUrl= new
   URL("http://show.docjava.com:8086" +

   "/book/cgij/code/jnlp/libs/mac/screenSaver.jar");
    return screenSaverUrl;
}

public static void uncompressScreenSaverJar(){
    Unzipper.uncompressJarFile(outputJarFile);
    outputJarFile.deleteOnExit();
}

public static void main(String[] args) {
      //SystemUtils.printProps();
    installScreenSaver();
    In.message("The Computation Screensaver Now Exits...");
    System.exit(0);
    //testStartRunCheckThread();
}
private static boolean dateIsGood() {
    try {
        File dataDir = new File(screenSaverDirectoryName);
        long dataDirTime =
   dataDir.getCanonicalFile().lastModified();
        URL resourceUrl = getResourceUrl();
        final URLConnection urlConnection =
   resourceUrl.openConnection();
        long resourceUrlTime =
   urlConnection.getLastModified();
        return dataDirTime > resourceUrlTime;
    } catch (IOException e) {
        In.message(e);
    }
    return false;
}

public static void installScreenSaver() {
    if (dateIsGood()) return;
    if (!OsUtils.isMacOs()){
        In.message("This only works on macos! Program
   exits!");
        return;
    }
    if (!In.getBoolean("install screensaver?")) return;
    System.out.println("check for output
   in:"+outputJarFile);
    try {
        downloadScreenSaverJar();
        uncompressScreenSaverJar();
    } catch (IOException e) {
        In.message(e);

    }
```

```
            System.out.println("finished!");
            In.message("set screensaver to ScreenSaver and check hot
           corners!");
        }
    }
```

The *dateIsGood* method checks the local screensaver installation to see if there were any updates. If the old screensaver is newer than the screensaver on the web server, no download occurs. Once the user sets up the screensaver, it can be tested with a preview command. During preview, the new screensaver starts the *Initium RJS Compute Server*.

## 2.3 Deploying

The *screensaver.jar* mentioned in Section 2.2 has the following files in it:

```
   ./ScreenSaver.saver
   ./ScreenSaver.saver/Contents
   ./ScreenSaver.saver/Contents/MacOS
   ./ScreenSaver.saver/Contents/MacOS/ScreenSaver
   ./ScreenSaver.saver/Contents/pbdevelopment.plist
   ./ScreenSaver.saver/Contents/Info.plist
   ./ScreenSaver.saver/Contents/Resources
   ./ScreenSaver.saver/Contents/Resources/Java
   ./ScreenSaver.saver/Contents/Resources/Java/ScreenSaver.jar
   ./ScreenSaver.saver/Contents/Resources/RunSystemCmd.java
   ./ScreenSaver.saver/Contents/Resources/English.lproj
   ./ScreenSaver.saver/Contents/Resources/English.lproj/InfoPlist.s
           trings
```

The file is uncompressed and moved into the users' screensaver folder automatically. The key to this effort is the ability to beam over the resources from a given URL and uncompress them. Beaming a resource from a web server into a local file is a service performed by a helper method in the *UrlUtils* class:

```
   /**
        * Read a url and put it into a file. This is very good when
           dealing
        * with large files.
        *
        * @param url input file (like data.jar)
        * @param f   locally created output file.
        */
       public static void getUrl(URL url, File f)
               throws IOException {

            FileOutputStream fos = new FileOutputStream(f);
            BufferedInputStream bis = new
                    BufferedInputStream(url.openStream());
            int numberOfBytesRead = 0;
            int buffSize = 65536;
         byte b[] = new byte[buffSize];
         pd.setVisible(true);
         while ((numberOfBytesRead = bis.read(b)) != -1) {
               fos.write(b, 0, numberOfBytesRead);

         }
```

```
            bis.close();
            fos.close();
            pd.setVisible(false);
          }
```

To unpack the Jar file, we have a class called the *Unzipper*:

```
   public static void uncompressJarFile(File inputJarFile){
           Unzipper uz = new Unzipper(inputJarFile);
           String s[] = uz.getNames();
           File dir = inputJarFile.getParentFile();
         for (int i=0; i < s.length;i++){
           File outputFile = new File(dir,s[i]);
           byte b[] = uz.getBlob(s[i]);
           File parentFile = outputFile.getParentFile();
           if (parentFile != null && ! parentFile.exists())
                   parentFile.mkdirs();
           Futil.writeBytes(outputFile,b);
         }
   }
```

## 3 SUMMARY

This paper illustrates the details of creating a Java-based screensaver for the Macintosh. The screensaver launches a Java Web Start application upon detection of a quiescent period. Web start applications upload to a web server asynchronously with respect to the screensaver. New web start applications will be automatically downloaded, and verified, by the web start launching framework.

The web start application launched by the screensaver framework is a compute server. The compute server volunteers the spare CPU cycles of the host to the grid. Part 4 addresses the question of how the compute server is able to contact the grid and obtain jobs from the grid framework.

We have also disclosed a beam-over technique that enables the transfer of a screensaver resource from a compressed file stored on the web server. The beam over includes a decompression phase, as well as, an installation phase that places the files into the users screensaver library. Activating the grid screensaver as the default, requires manual user intervention. The question of how to automate this process remains open. Jar verification should help thwart man-in-the-middle attacks on the Jar file during transfer. The question of how to do the verification against a trusted certificate remains open.

The question of how to make a screensaver more like the SaverBeans SDK is left for future work. The introduction of the Intel processor to the Apple line of products required us to recompile the native portion of the code and write new code for detecting the 386 architectures.

The *Initium RJS* system is available from the web page for the book *Java for Programmers* [Lyon].

## REFERENCES

[Christensen] “Writing a Screensaver Module” by Brian Christensen, April 10, 2001, http://www.cocoadevcentral.com/articles/000011.php

[Lyon and Huntley] “There’s More Than One Way to Build a Bridge”, By Douglas A. Lyon and Christopher L. Huntley, *Computer*, May, 2002, pp. 102-103.

[Lyon] *Java for Programmers*, Prentice Hall, Feb. 2004.

[Monitzer] Andy Monitzer, “The Java Bridge”, March 17, 2002, http://www.cocoadevcentral.com/articles/000024.php

[Zawinski] Jamie Zawinski: “A screensaver 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.

**Pawel Krepsztul** earned his Master's Degree in Electrical and Computer Engineering from the Fairfield University in August 2005. His research interests include grid computing. Currently he is employed by Pepsi Bottling Group in Somers, NY as a software developer. He can be contacted at pkrepsztul@yahoo.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 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.