

HOWTO: Create, run and deploy Java 3D applications using Excelsior JET

Article ID: 000027

Last Revised On: 03-Feb-2006

The information in this article applies to:

- Excelsior JET version 4.1 and above
- Java 3D 1.3.2

The similar article for Excelsior JET 4.5 can be found here [000028](#).

SUMMARY

The **Java 3D(tm) API** is an application programming interface used for writing three-dimensional graphics applications. It has been originally developed by Sun Microsystems and later turned into a community source project. This article describes how you can use Excelsior JET to optimize your Java application that utilizes the Java3D API and create an installation package to deploy it to end-user systems. The names and locations of Java 3D files are given for Java 3D 1.3.2.

Note: This article describes only the essential details of optimizing and deploying a Java 3D application with Excelsior JET. If you are a first time Excelsior JET user, please refer to the [Excelsior JET online tutorials](#) guiding you through the basics of application optimization and deployment.

DETAILS

Java 3D is a so called *optional package*, defined in the Java platform documentation as “*package of Java classes and associated native code that application developers can use to extend the functionality of the core platform.*” An optional package consists of one or more jar files, and may also contain native method libraries, property files and other auxiliary files.

Optional packages are normally installed right into the JRE installation directory. Their jar files are placed in the `lib/ext` subdirectory, or right into the `lib` subdirectory containing the core platform jar files. Jars from these directories are automatically added to the classpath of any application that runs on that JRE.

Similarly, the native method libraries or an optional package are placed alongside their core platform counterparts. On Windows, that is the `bin` subdirectory of the JRE, where the `java` launcher resides. On Linux, native method libraries are located in the `lib/i386` subdirectory of the JRE.

JAVA 3D FILES

The runtime part of the Java 3D optional package consists of three jar files:

```
lib/ext/j3dcore.jar
lib/ext/j3dutils.jar
lib/ext/vecmath.jar
```

and a few native method libraries:

Windows:

```
bin\j3dutils.dll
bin\j3dcore-d3d.dll
bin\j3dcore-ogl.dll
```

Linux:

```
lib/i386/libj3dutils.so
lib/i386/libj3dcore-ogl.so
```

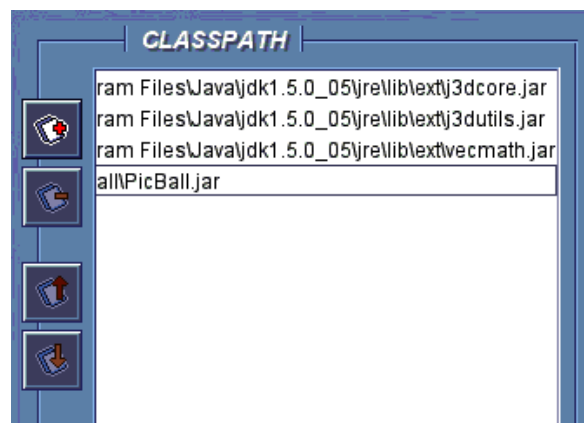
You have to compile the jar files into the application executable and make the native method libraries available at run time.

BUILDING A JAVA 3D APPLICATION

Start page

Excelsior JET does not depend on a JRE installation and is unable to detect where the Java 3D package is installed on any of the JREs present on the system. Therefore you have to include the Java 3D jars into the compilation set manually.

On the **Start** page of the JET Control Panel, add the Java 3D jars to the beginning of the classpath:



The import dependencies specific to the Java 3D API will be resolved as soon as these files are added to the classpath.

You also have to ensure that the resulting executable will be able to find the Java 3D native libraries on startup. On Windows, you may simply copy those libraries into the directory where the application EXE resides. If you do not want to do that for any reason, or if you are targeting Linux, you must specify the pathname of the directory containing the Java 3D native method libraries using the `java.library.path` system property.

Examples:

Suppose you will copy the Java 3D native libraries into the `./lib` subdirectory relative to the resulting executable. Specify the `java.library.path` system property in the following way:

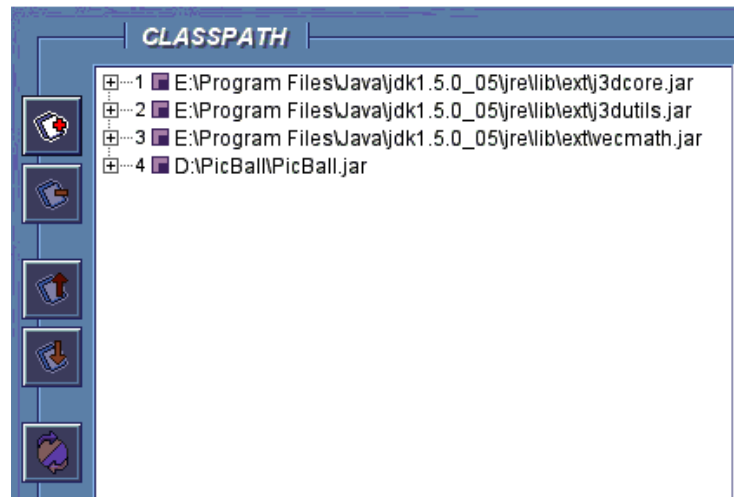
```
java.library.path=lib
```

Suppose Java 3D is installed into the JRE located in `/usr/java/jdk1.5.0_05/jre` and you do not want to copy the native method libraries. Set `java.library.path` as follows:

```
java.library.path=/usr/java/jdk1.5.0_05/jre/lib/i386
```

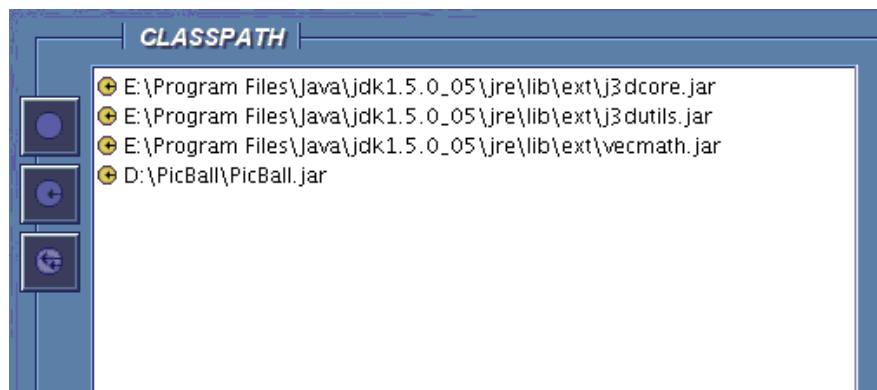
Classes page

On the **Classes** page, you may choose whether to include the entire jars in the compilation set or let the import scanner exclude the classes that are not explicitly imported. By default, all classes from each jar file are included into the compilation set. We strongly recommend that you include the Java 3D jars entirely, since it is not possible to determine which classes of a third-party jar may be loaded dynamically at application run time.



Resources page

On this page, you choose the resource packing mode. By default, resources from Java 3D jars are packed into the executable, and it is recommended to leave this default setting intact.



Your actions on further steps do not differ from those you perform when compiling a usual Java application with Excelsior JET. However, we recommend that you do **not** disable the console window when building your application on Windows for the first time so as to be able to track down the run time errors (if any).

RUNNING THE COMPILED JAVA 3D APPLICATION

Before you run the compiled application, make sure that the Java 3D native method libraries can be found either:

- in the directory you have specified in the `java.library.path` system property, or

- in the directory where the application executable resides (Windows only)

The pathnames of the required libraries relative to the JRE root directory are as follows:

Windows:

```
bin\j3dutils.dll
bin\j3dcore-d3d.dll
bin\j3dcore-ogl.dll
```

Linux:

```
lib/i386/libj3dutils.so
lib/i386/libj3dcore-ogl.so
```

In the next section, we describe how to create an installation package for a Java 3D application compiled with Excelsior JET.

CREATING AN INSTALLATION PACKAGE

Your application needs the Java 3D native method libraries to run, so you must include these libraries into the installation package and ensure that they can be found at run time.

Files page

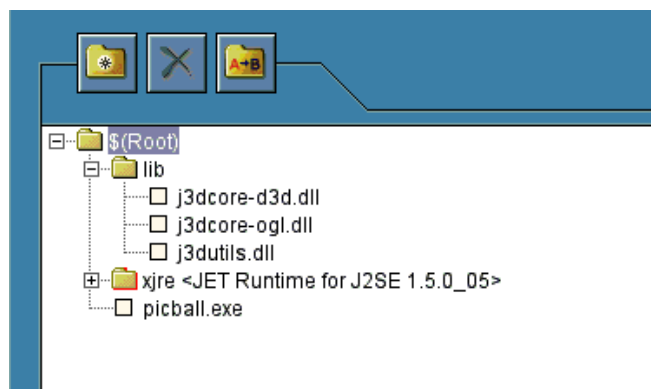
Add the main application executable to the installation package, then add the following native method libraries from the JRE that has Java 3D installed:

Windows:

```
bin\j3dutils.dll
bin\j3dcore-d3d.dll and/or bin\j3dcore-ogl.dll
```

Linux:

```
lib/i386/libj3dcore-ogl.so
lib/i386/libj3dutils.so
```



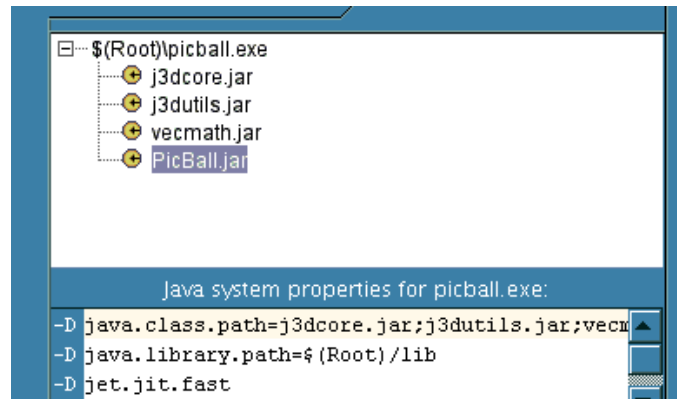
Note: On Windows, you may choose whether to use the Direct3D- or OpenGL-based implementation of the Java 3D optional package. On Linux, only the OpenGL version is available.

Resources page

On Windows, no Java 3D specific actions are required if you have placed the Java 3D native libraries alongside the application executable. Otherwise, you must specify the `java.library.path` system property for that executable. Suppose your installation has Java 3D native libraries placed in the

`$(Root)/lib` subdirectory. Select the executable of your Java 3D application and specify the said property in the **Java System Properties** field as follows:

```
java.library.path=$(Root)/lib
```



No Java 3D specific actions are required on the further steps of creating the installation package for your compiled application.

Note: We strongly recommend you to perform a trial run before building the installation package in order to check if the package is complete and the application works properly.

BUILDING A MULTI-COMPONENT JAVA 3D APPLICATION

In this section we describe how to build a multi-component Java 3D application with Excelsior JET, compiling the Java 3D jars into a separate dynamic library and the jars of your application into an executable file.

We recommend you to create a multi-component application from the jars that could be flawlessly compiled into a single-component application with Excelsior JET.

Note: Having created a dynamic library from Java 3D jars, you will still need the appropriate Java 3D native method libraries (`j3dcore.dll`, etc.) for your application to run.

Step 1

On this step we define the content of the dynamic library containing Java 3D classes.

The compilation set should contain all Java 3D jars (`j3dcore.jar`, `j3dutils.jar`, and `vecmath.jar`) to enable Excelsior JET to compile them into a library.

In order to create a library from the compilation set, you should create a project file and compile it using the command line Excelsior JET compiler, `jc`.

Create an empty directory called, for instance, `MultiCompBuild`. In the directory, create an Excelsior JET project file (plain text file with the extension `.prj`) using the following template:

```
-OUTPUTNAME=j3d          % library name without extension
-GENDLL+
-PACKRESOURCES+
-IGNOREMEMBERABSENCE+

-LOOKUP=* .obj= ./obj_$(OUTPUTNAME)
```

```
-LOOKUP=*.jar=C:\Program Files\java\jdk1.5.0_05\jre\lib\ext\

% Java 3D jars
!module j3dcore.jar
!module j3dutils.jar
!module vecmath.jar
```

Make sure to edit the `-LOOKUP=*.jar=` line to point to the location of Java 3D jars on your system.

Step 2

On this step, you create a project file for the main part of your application.

In the same directory as on a previous step, create another project file using the following template:

```
-OUTPUTNAME=testMulti          % EXE-name without extension
-PACKRESOURCES+
-IGNOREMEMBERABSENCE+

-LOOKUP=*.sym=./sym_$(OUTPUTNAME)
-LOOKUP=*.bod=./bod_$(OUTPUTNAME)
-LOOKUP=*.obj=./obj_$(OUTPUTNAME)

-LOOKUP=*.jar=D:\Picball      % path to application jars
-main=PictureBall           % main class name

% Application jars
!module PicBall.jar
```

Edit `-OUTPUTNAME=`, `-LOOKUP=*.jar=`, `-MAIN=`, and `!module` settings as appropriate for your application.

Step 3

On this step, you compile the project files you have created with the Excelsior JET command-line compiler. Set `MultiCompBuild` as the current directory and compile these projects using the commands:

```
jc =p library-project
jc =p executable-project
```

Now you can run the resulting executable.

For more comprehensive and detailed instructions on how to create multi-component Java applications with Excelsior JET, please refer to the [Excelsior JET User's Guide](#), Chapter "Dynamic Linking", section "Multi-component applications".

REFERENCES

1. Java 3D home page: <http://java3d.dev.java.net>
2. Excelsior JET User's Guide: <http://www.excelsior-usa.com/jetdoc.html>