

# **JavaOne Tokyo Hands on lab**

## **Project Looking Glass ( LG3D ) プログラミング**

このテキストは JavaOne Tokyo 2005 で行われた Project Looking Glass (LG3D) Hands on Lab の内容に修正・加筆を行い、LG3D Release 1.0に対応したものです。

このテキストでは LG3Dを利用するための準備、LG3Dのインストール方法、起動方法 および NetBeans 5.5 によるプログラミング方法を解説しています。

### **1. グラフィックス環境のセットアップ**

- 1. [Solaris 10 x86 編](#)**
- 2. [Linux & NVIDIA 編](#)**
- 3. [Linux & ATI 編](#)**

### **2. LG3D のインストールと実行**

- 1. [インストール \( Solaris 10 x86 編 \)](#)**
- 2. [インストール \( Linux 編 \)](#)**
- 3. [実行方法](#)**
- 4. [インストールと実行方法 \( Windows XP 編 \)](#)**

### **3. NetBeans による LG3D プログラミング ( 準備 )**

- 1. [NetBeans 5.5 のダウンロードおよびインストール](#)**
- 2. [NetBeans の設定](#)**

### **4. NetBeans による LG3D プログラミング ( 実践 )**

- 1. [プロジェクトの作成](#)**
- 2. [シンプルな 3D オブジェクトの作成](#)**
- 3. [JARベースのアプリケーションの作成と実行](#)**
- 4. [テクスチャを使う](#)**
- 5. [サムネイルの作成](#)**
- 6. [マウスイベント・アニメーションの作成](#)**
- 7. [Swingを利用する \(JFrame 編\)](#)**
- 8. [Swingを利用する \(SwingNode 編\)](#)**

### **5. [応用問題](#)**

JavaOne Tokyo 2005 の LG3D HoL で課題が終了した方に  
お渡ししたものです。解答は用意していません。

- [参考文献, URL 等](#)**

**注意:**

このテキストでは Solaris 10 x86 の Java Desktop System (JDS) 環境を基本にしています。  
また、ユーザ名やディレクトリ(フォルダ)などのユーザ環境は下記の通りです。  
それぞれの環境にあわせて、適宜読み替えてください。

ユーザ名	duke
ホームディレクトリ	/home/duke
LG3D ディレクトリ	/home/duke/lg3d
JDK ディレクトリ	/home/duke/lg3d/usr/share/lg3d-jdk
データディレクトリ	/home/duke/Documents
NetBeans ディレクトリ	/opt/netbeans-5.5
画面解像度	1280x1024
LG3D ウィンドウサイズ	1152x864

テキスト内で使用している画像ファイル等は [all\\_contents.zip](#)  
[\(https://lg3d.dev.java.net/ja/lg3d\\_hol/html/all\\_contents.zip\)](https://lg3d.dev.java.net/ja/lg3d_hol/html/all_contents.zip) にまとめています。  
ダウンロードしてから、/home/duke/Documents に解凍してください。

# 1.1 グラフィックス環境のセットアップ (Solaris x86 編)

Solaris 10 x86 で LG3D 1.0 を動かすためには下記の環境が必要になります。

- Pentium 4, Athlon 64 以降の比較的新しい CPU ( 1.x GHz 以上推奨 )
- 512MB 以上のメモリ ( 利用環境によっては 1GB 以上推奨 )
- NVIDIA Quadro シリーズ ( GeForce FX シリーズも利用できます )
- テスト済みのビデオカードについては、

**Getting started with the Project Looking Glass Developer's Release の 1.**

**Platform Requirements** を参照してください。

URL : <https://lg3d.dev.java.net/lg3d-getting-started.html>

**補足:**

Solaris Express(次期 Solaris の β 版) には NVIDIA 社製グラフィックスドライバが統合されているため、下記のグラフィックスドライバのインストールは不要です。

また、グラフィックスコントローラを内蔵している Intel 社製チップセットを搭載した環境 (Intel 915, Intel 945GM(Centrino プロセッサテクノロジ環境)など) でも LG3D を動かせます。

---

## グラフィックスドライバの確認

システムに NVIDIA 社製グラフィックスドライバが入っているかどうかは pkginfo コマンドを用いて調べることができます。

以下のように、実行した際にも表示されなければ NVIDIA 社製グラフィックスドライバはインストールされていません。

```
# pkginfo | grep NVDA
```

---

## グラフィックスドライバのインストール

NVIDIA のサイト(<http://www.nvidia.com/object/unix.html>) からグラフィックスドライバをダウンロードします。

ドキュメント記述時(2007/3/1)の最新版のドライバは 1.0-9746 (NVIDIA-Solaris-x86-1.0-9746.run)です。

グラフィックスドライバはインストールはルートユーザで行います。 インストール方法は次の通りです。

```
# sh NVIDIA-Solaris-x86-1.0-9746.run
```

```
A sample xorg configuration file has been copied to /etc/X11/xorg.conf.nvidia
Copyright 2005 by NVIDIA Corporation. All rights reserved.
Use is subject to license terms.
```

```
Installation of <NVDAgraphicsr> was successful.
Copyright 2005 by NVIDIA Corporation. All rights reserved.
Use is subject to license terms.
```

```
Installation of <NVDAgraphics> was successful.
```

インストールを終了するとシステムに2つのパッケージ(NVDAGraphics,NVDAGraphicsr)が追加されます。

以下のように pkginfo コマンドを使って確認できます。

```
# pkginfo | grep NVDA
system    NVDAGraphics          NVIDIA Graphics System Software
system    NVDAGraphicsr         NVIDIA Graphics System Device Driver
```

グラフィックスドライバのインストールが終了したら X-Window 環境の設定を行います。(このままで X-Window は起動しません)

NVIDIA 社製グラフィックスドライバをインストールすると /etc/X11/xorg.conf.nvidia というファイルがインストールされますので、これを利用します。

このファイルは NVIDIA 社製グラフィックスドライバ環境での xorg.conf のサンプルファイルです。ほとんどの環境ではこのファイルを /etc/X11/xorg.conf にコピーするだけです。

```
# cp /etc/X11/xorg.conf.nvidia /etc/X11/xorg.conf
```

次に xorg.conf を編集し、AllowGLXWithComposite オプションを有効にします。

LG3D には lg3d-dev, lg3d-app, lg3d-session の3つのコマンドが用意されていますが、lg3d-session コマンドを利用するためには必要なオプションです。

(このテキストでは lg3d-session を利用しないので、設定しなくても構いません。)

```
# Device configured by xorgconfig:

Section "Device"
    Identifier "*/ NVIDIA (generic)"           [nv]"
    Driver     "nvidia"
    #VideoRam  131072
    # Insert Clocks lines here if appropriate
    Option     "AllowGLXWithComposite" "yes"
EndSection
```

#### 補足:

UXGA(1600x1200)を利用したい場合などは /usr/X11/bin/xorgconfig を用いて /etc/X11/xorg.conf を生成します。その後 xorg.conf を編集し、NVIDIA 社製グラフィックスドライバ用の設定 (Driver "nv" を Driver "nvidia" に書き換える) を行ってください。

インストール後、リブートします。

```
# init 6
```

NVIDIA Quadro シリーズを利用している場合は NVIDIA のロゴが表示され、ログイン画面が表示されると思います。グラフィックスドライバのインストールおよび設定は終了です。

NVIDIA GeForce FX シリーズを利用している場合は GUI の表示に失敗します。

NVIDIA 社製グラフィックスドライバのインストール時点では NVIDIA GeForce FX シリーズのカード情報が Solaris に登録されないのが原因です。

NVIDIA 社製グラフィックスドライバを有効にするためには グラフィックスカード情報を登録します。

登録方法は次の通りです。

- /usr/X11/bin/scanpci コマンドを実行します。 PCI 情報を表示されるので、 グラフィックスカードの vendor, device の値を確認します (下図の青字(下線)の部分)。  
vendor, device の値のうち「 0x 」と 4 衔の数値の最初の「 0 」は無視します。  
つまり、例では vendor = 10de , device = 322 となります。
- update\_drv コマンドでグラフィックスカード情報を登録します。
- リコンフィグオプション付きでリブートします。

```
# /usr/X11/bin/scanpci

...
pci bus 0x0000 cardnum 0x1f function 0x05: vendor 0x8086 device 0x24d5
Intel Corp. 82801EB AC'97 Audio Controller

pci bus 0x0001 cardnum 0x00 function 0x00: vendor 0x10de device 0x0322
nVidia Corporation NV34 [GeForce FX 5200]

pci bus 0x0002 cardnum 0x01 function 0x00: vendor 0x8086 device 0x1019
Intel Corp. 82547EI Gigabit Ethernet Controller (LOM)

...
# update_drv -a -i "pci10de,322" nvidia
# reboot --r
```

## 1.2 グラフィックス環境のセットアップ(Linux & NVIDIA 編)

LG3D を動かすためには 24bit color, OpenGL 1.3 以上をサポートした環境が必要です。Linux 上でこの環境を構築するには以下のグラフィックスチップを搭載した PC が必要です(NVIDIA 社製グラフィックスチップを搭載した環境の場合)。

- NVIDIA GeForce FX シリーズ
- NVIDIA Quadro シリーズ

### グラフィックスドライバのインストール

NVIDIA のサイト(<http://www.nvidia.com/object/unix.html>) から NVIDIA 社製のグラフィックスドライバをダウンロードします。

ドキュメント記述時(2007/3)の最新版のドライバは 1.0-9746 (NVIDIA-Linux-x86-1.0-9746-pkg1.run) です。

ここではファイルを /tmp にダウンロードしたと仮定します。

グラフィックスドライバはインストールはルートユーザで行います。

インストール方法は次の通りです。 (ユーザーの入力したコマンドは赤(太字)で示します)

1. ルートユーザになりコンソールモードに変更する。 コンソールモードに変わったら再度ログインし、インストーラを起動します。  
対話式のインストーラが表示されるので、指示に従いインストールします。

```
> su -
Password: ←パスワードを入力
# init 3

console login: root
Password: ←パスワードを入力
...
# sh NVIDIA-Linux-x86-1.0-9746-pkg1.run
```

#### 補足:

Fedora 7/ Fedora Core 6 等を利用している場合、

「**Fedora Core 6 で 3D GUI 環境 Compiz を使うには (NVIDIA 編)**」

(<http://www.atmarkit.co.jp/flinux/rensai/linuxtips/960compiznvidia.html>)

を参照し、ドライバーをインストールしてください。

2. ドライバのインストールが終了したら X-Window 環境の設定を行います。以下のように /etc/X11/xorg.conf を編集します。

赤(太字)は変更部分に関するコメントです。これを参考に変更してください。

```
...(略)...  
Section "Module"  
    Load      "glx"          [モジュー ルの設定 ]  
    # Load     "dri"          [ない場合は追加 する]  
    # Load     "GLcore"        [ある場合はコメ ントアウト (文の先頭に # )]  
EndSection  
...(略)...  
Section "InputDevice"          [キーボー ド設定 ]  
    Identifier "Keyboard1"  
    Driver     "kbd"          ["keyboard" の場合、 kbd に変更 ]  
    Option    "AutoRepeat" "500 30"  
    Option    "XkbRules"   "xfree86"  
    Option    "XkbModel"   "jp106"  
    Option    "XkbLayout"  "jp"  
EndSection  
...(略)...  
Section "Device"              [グラフィ ックスドライ バの設定 ]  
    Identifier "NVIDIA"  
    Driver     "nvidia"        ["nv" から変更 ]  
EndSection  
...(略)...  
Section "Screen"               [スクリーン設定 ]  
    Identifier "Screen0"  
    Device     "NVIDIA"  
    Monitor   "Monitor0"  
    DefaultDepth 24           [無い場合 は追加、 24 以外の場合は 24 に変更 ]  
    Subsection "Display"  
        Depth    24  
        Modes   "1280x1024"  
        ViewPort 0 0 # initial origin if mode is smaller than desktop  
    EndSubsection  
EndSection
```

3. X を再起動します。

```
# init 5
```

### 1.3 グラフィックス環境のセットアップ(Linux & ATI 編)

LG3D を動かすには 24bit color, OpenGL 1.3 以上をサポートした環境が必要です。Linux の場合、以下の環境が必要となります。

- ATI Radeon 8500 以降

## ドライバのインストール

AMD社のATIブランドのグラフィックスドライバは AMD社のATI製品のウェブサイト（<http://ati.amd.com/jp/index.html>）で公開されています。

トップページから「ドライバ&ソフトウェア」→「Linux x86」→「Radeon」→「対象となるグラフィックチップを選択」→「Go」とリンクをたどっていくと見つかります。ドキュメント記述時(2007/8)の最新版のドライバは 8.39.4 です。

Radeon 8500 をサポートしているドライバは 8.28.8 が最終リリースとなります。そのため、Radeon 8500 の場合は 8.28.8 を利用してください。

ドライバはインストールはルートユーザで行います。

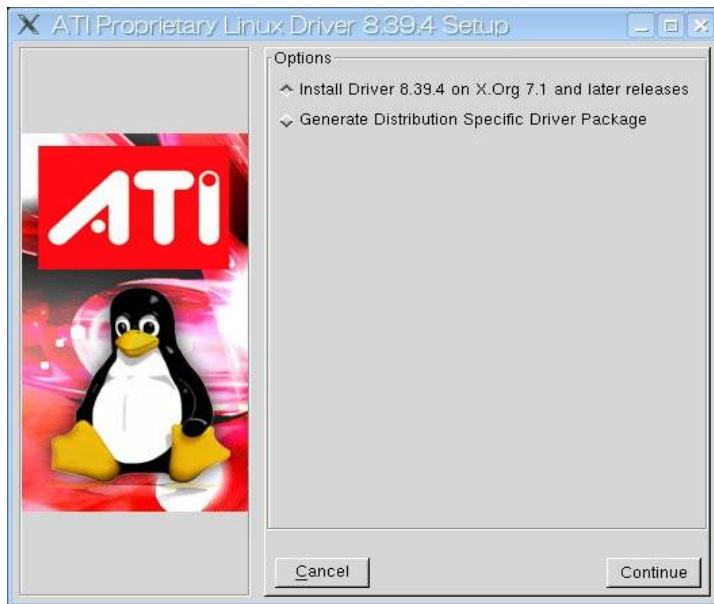
インストール方法は次の通りです。（ユーザーの入力したコマンドは赤（太字）で示します）

1. ルートユーザになる。

```
> su -  
Password: ← パスワードを入力  
#
```

2. グラフィックスドライバのインストーラを起動します。

上記のようにコマンドを実行すると、下図のようなインストール用の GUI が表示されますので、指示にしたがって グラフィックスドライバをインストールします。



3. 下記のように aticonfig コマンドを用いて xorg.conf を作成します。  
aticonfig は自動的に xorg.conf を作成するコマンドです。

```
# aticonfig --initial --input=/etc/X11/xorg.conf
```

4. X を再起動します。

```
# init 3  
# init 5
```

5. X の再起動後、fglrxinfo コマンドで OpenGL 設定が正しく行われたことを確認します。

```
# fglrxinfo  
display: :0.0 screen: 0  
OpenGL vendor string: ATI Technologies Inc.  
OpenGL renderer string: ATI MOBILITY RADEON 9600/9700 Series  
OpenGL version string: 2.0.6650 (8.39.4)
```

#### 補足:

Fedora Core 6/Fedora 7 の Xorg では Composite Extension がデフォルトで有効になっています。 fglrx ドライバでは、DRI と Composite Extension の同時利用がおこなえず、Composite Extension が優先されます。そのため、この状態では LG3D がちゃんと動きません。 ですので、/etc/X11/xorg.conf に次の行を加えて Composite Extension を無効化します。

```
Section "Extensions"  
Option "Composite" "false"  
EndSection
```

## 2.1 LG3D のインストール ( Solaris 10 x86 編 )

---

### 準備

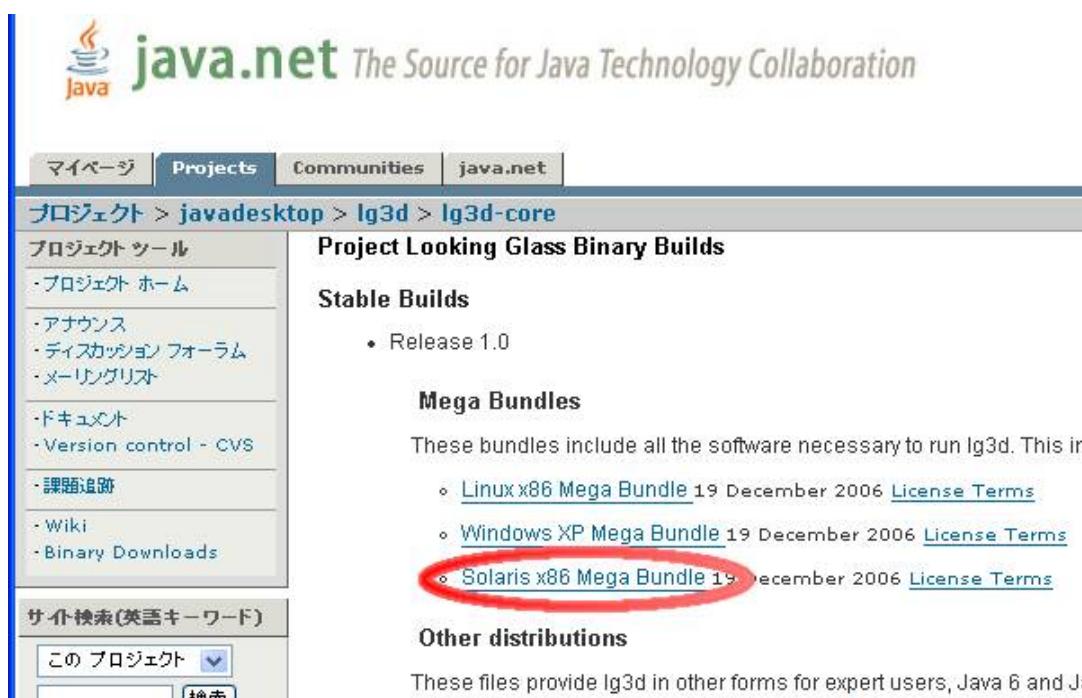
Solaris 10 で LG3D 1.0 を実行するためのすべてのコンポーネントは Solaris 10 x86 専用のインストーラーにまとめられています。(LG3D Javadoc は除く)

**LG3D のウェブサイト からインストーラをダウンロードし、実行するだけで LG3D をインストールすることができます。**

インストーラはできるだけ最新のものをダウンロードしてください。ダウンロードしたファイルは /tmp ディレクトリ(フォルダ)にあるとします。

**LG3D API Javadoc のインストール**については「[LG3D Javadoc のインストール](#)」を参照してください。

- ・ ウェブページ URL <https://lg3d-core.dev.java.net/binary-builds.html>
- ・ インストーラ名 **Solaris x86 Mega Bundle**
- ・ ウェブページのスクリーンショット ↓



## LG3D Release 1.0 のインストール

「準備」の項目でダウンロードしたファイルは /tmp にあるものと仮定します。

- LG3D Release 1.0 をインストールします。

```
~ $ cd  
~ $ chmod u+x /tmp/lg3d--1-0-0-solaris-i86pc-0612190950.bin  
~ $ sh /tmp/lg3d--1-0-0-solaris-i86pc-0612190950.bin
```

This software consists of 3 components, each component with a distinct license represented below. By selecting the "accept" button, you are accepting the terms of all the licenses for the software.

- 1) JRE License
- 2) Java 3D License
- 3) Project Looking Glass License

Pre-Release Software Evaluation Agreement

SUN MICROSYSTEMS, INC. ("SUN") IS WILLING TO LICENSE THE JAVA SE DEVELOPMENT KIT (JDK), VERSION 6 PRE-RELEASE SOFTWARE TO LICENSEE ONLY UPON THE CONDITION THAT LICENSEE ACCEPTS ALL OF THE TERMS CONTAINED IN THIS LICENSE AGREEMENT ("AGREEMENT"). PLEASE READ THE TERMS AND CONDITIONS OF THIS AGREEMENT CAREFULLY. BY DOWNLOADING OR INSTALLING THIS SOFTWARE, LICENSEE ACCEPTS THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT. INDICATE ACCEPTANCE BY SELECTING THE "ACCEPT" BUTTON AT THE BOTTOM OF THIS AGREEMENT. IF LICENSEE IS NOT WILLING TO BE BOUND BY ALL THE TERMS, SELECT THE "DECLINE" BUTTON AT THE BOTTOM OF THE AGREEMENT AND THE DOWNLOAD OR INSTALL PROCESS WILL NOT CONTINUE.

....

Do you agree to the above license terms? [yes or no]

**yes**

Unpacking...

....

```
inflating: lg3d/usr/share/lg3d-jdk/include/jdwpTransport.h  
inflating: lg3d/usr/share/lg3d-jdk/include/classfile_constants.h  
inflating: lg3d/usr/share/lg3d-jdk/COPYRIGHT
```

Done.

~ \$

これで LG3D 1.0 のインストールは終了です。

ユーザのホームディレクトリには **lg3d** というディレクトリが作成されています。

JDK 6 など、LG3D 1.0 に必要なすべてのコンポーネントはこのディレクトリに入っています。

補足:

- Daily Builds の場合は、JDK 6, Java3D 1.5 を別途ダウンロードする必要があります。
- LG3D 1.0 では JAI は不要です。

## LG3D Javadoc のインストール

LG3D 0.7.1 から Javadoc は本体からはずされました。

LG3Dを利用するだけであれば Javadoc は必要ありませんが、 LG3D プログラミングを行う場合には Javadoc は有用なのでインストールします。

まず、 **Project Looking Glass Binary Builds** (<https://lg3d-core.dev.java.net/binary-builds.html>) から Javadoc をダウンロードしてください。

Javadoc は Stable Builds ではなく、 Daily Builds にあります。

Daily Builds からできるだけ最新の javadoc をダウンロードしてください。

ダウンロードしたファイルは /tmp にあるとします。

執筆時点(2007/3/1)での最新版は `javadoc-lg3d-daily-dev-1-0-1-linux-i686-0702080130.tar.gz` です。

ダウンロード後、 ファイルを解凍します。 解凍時に警告が出ますが無視してかまいません。

解凍後にホームディレクトリに **lg3d/docs/javadoc** というディレクトリが作られます。

解凍後 `@LongLink` という不要なファイルを削除します。

```
~ $ cd
~ $ gunzip -c /tmp/javadoc-lg3d-daily-dev-1-0-1-linux-i686-0702080130.tar.gz | tar xf -
tar: ./@LongLink: タイプフラグ 'L' を認識できません。通常のファイルに変換しています
...
~ $ rm @LongLink
```

また、以下の URL からも Javadoc を参照することも可能です。

Javadoc : Project Looking Glass (LG3D) API (英語)

<http://javadesktop.org/lg3d/javadoc/rel-1-0-0/lg3d/docs/javadoc/api/index.html>

Javadoc : LG3D SceneManager API (英語)

<http://javadesktop.org/lg3d/javadoc/rel-1-0-0/lg3d/docs/javadoc/scenemanager-api/index.html>

## 2.2 LG3D のインストール (Linux 編)

### 準備

Linux で LG3D 1.0 を実行するためのすべてのコンポーネントは Linux 専用のインストーラにまとめられています。(LG3D Javadoc を除く)

**LG3D のウェブページ からインストーラをダウンロードして実行するだけでインストールを行えます。**

インストーラはできるだけ最新のものをダウンロードしてください。ダウンロードしたファイルは /tmp ディレクトリにあるとします。LG3D Javadoc のインストールについては、「[LG3D Javadoc のインストール](#)」を参照してください。

- ウェブページ URL <https://lg3d-core.dev.java.net/binary-builds.html>
- インストーラ名 **Linux x86 Mega Bundle**
- ウェブページのスクリーンショット ↓

The screenshot shows the 'Project Looking Glass Binary Builds' page. On the left, there's a sidebar with links like 'プロジェクト シール', 'プロジェクト ホーム', 'アナウンス', 'ディスカッション フォーラム', 'メーリングリスト', 'ドキュメント', 'Version control - CVS', '課題追跡', 'Wiki', and 'Binary Downloads'. The main content area has a heading 'Project Looking Glass Binary Builds' and a sub-section 'Stable Builds' with a link to 'Release 1.0'. Below that is a section titled 'Mega Bundles' which contains a list of bundles: 'Linux x86 Mega Bundle' (circled in red), 'Windows XP Mega Bundle', and 'Solaris x86 Mega Bundle'. At the bottom, there's a section titled 'Other distributions'.

### LG3D Release 1.0 のインストール

「準備」の項目でダウンロードしたファイルは /tmp にあるものと仮定します。

- LG3D Release 1.0 をインストールします。

```
~ $ cd  
~ $ chmod u+x /tmp/lg3d--1-0-0-linux-i686-0612190943.bin  
~ $ sh /tmp/lg3d--1-0-0-linux-i686-0612190943.bin
```

This software consists of 3 components, each component with a distinct license represented below. By selecting the "accept" button, you are accepting the terms of all the licenses for the software.

1) JRE License

- 2) Java 3D License
- 3) Project Looking Glass License

#### Pre-Release Software Evaluation Agreement

SUN MICROSYSTEMS, INC. ("SUN") IS WILLING TO LICENSE THE JAVA SE DEVELOPMENT KIT (JDK), VERSION 6 PRE-RELEASE SOFTWARE TO LICENSEE ONLY UPON THE CONDITION THAT LICENSEE ACCEPTS ALL OF THE TERMS CONTAINED IN THIS LICENSE AGREEMENT ("AGREEMENT"). PLEASE READ THE TERMS AND CONDITIONS OF THIS AGREEMENT CAREFULLY. BY DOWNLOADING OR INSTALLING THIS SOFTWARE, LICENSEE ACCEPTS THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT. INDICATE ACCEPTANCE BY SELECTING THE "ACCEPT" BUTTON AT THE BOTTOM OF THIS AGREEMENT. IF LICENSEE IS NOT WILLING TO BE BOUND BY ALL THE TERMS, SELECT THE "DECLINE" BUTTON AT THE BOTTOM OF THE AGREEMENT AND THE DOWNLOAD OR INSTALL PROCESS WILL NOT CONTINUE.

....

Do you agree to the above license terms? [yes or no]

**yes**

Unpacking...

....

inflating: lg3d/usr/share/lg3d-jdk/include/jdwpTransport.h

inflating: lg3d/usr/share/lg3d-jdk/include/classfile\_constants.h

inflating: lg3d/usr/share/lg3d-jdk/COPYRIGHT

Done.

~ \$

これで LG3D 1.0 のインストールは終了です。

ユーザのホームディレクトリに **lg3d** というディレクトリが作られています。

LG3D 1.0 に必要なすべてのコンポーネント (JDK 6.0 など) がこのディレクトリに入っています。

#### 補足 1:

- Daily Builds 版の LG3D をインストールする場合は、JDK 6 および Java3D 1.5 を別途ダウンロードする必要があります。

#### 補足 2:

- root 権限で LG3D を /usr/share にインストールした場合、root で以下のコマンドを実行することで LG3D をウインドウマネージャ (A gdm session type) にすることができます (Linuxのみ)。

```
# cd /usr/share/lg3d/bin  
# ./postinstall
```

## JDK の日本語フォント設定

最近の Linux ディストリビューションでは Java を使用したアプリケーションが文字化け（日本語が□□化）する場合があります。このような場合、以下の作業を行ってフォントを正しく設定してください。

JDK 内の jre/lib/fonts ディレクトリに fallback ディレクトリを作成します。  
この中に TrueType フォントのコピーまたはシンボリックリンクを作成します。

- /usr/java/jdk1.6.0\_01/に JDK をインストールしている場合の例

```
fallback ディレクトリを作成
# mkdir /usr/java/jdk1.6.0_01/jre/lib/fonts/fallback
fallback ディレクトリへ移動
# cd /usr/java/jdk1.6.0_01/jre/lib/fonts/fallback
sazanami-gothic.ttf へのシンボリックリンクを作成
# ln -s /usr/share/fonts/japanese/TrueType/sazanami-gothic.ttf
sazanami-mincho.ttf へのシンボリックリンクを作成
# ln -s /usr/share/fonts/japanese/TrueType/sazanami-mincho.ttf
```

### 注意：

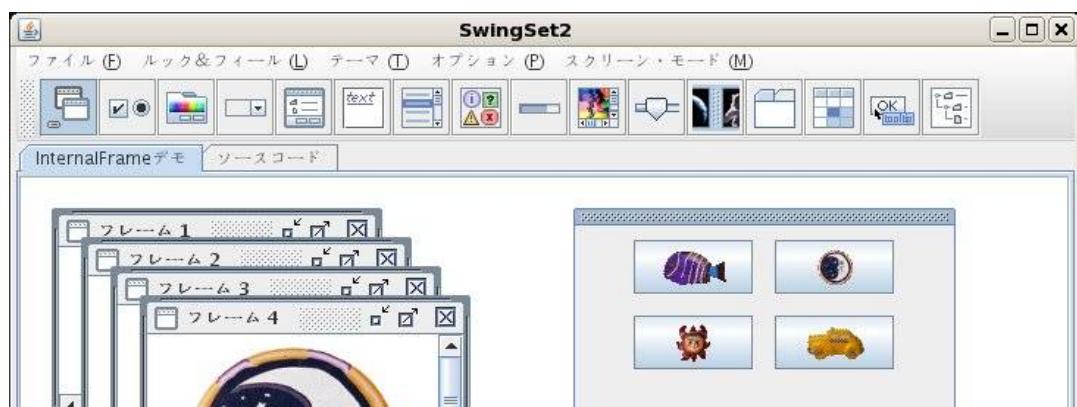
上記の sazanami~ の部分をそれぞれ使用しているフォントに置き換えて設定してください。（例：M+ & IPA Font を使用している場合 sazanami-\*.ttf が、M+\*.ttf になります。）

上記の設定を行った後、JDK 付属のデモなどで動作確認してみてください。

例えば、SwingSet2 を起動する場合は以下のようになります。

```
# /usr/java/jdk1.6.0_01/bin/java \
> -jar /usr/java/jdk1.6.0_01/demo/jfc/SwingSet2/SwingSet2.jar
```

下図のように日本語が正しく表示されているか確認してください。



上記の設定で解決できない場合、Java SE のページや各ディストリビューションのページを参考に設定を行ってください。

## 2.3 LG3D の実行

### LG3D の 3 つの実行コマンド

LG3D をインストールすると lg3d/bin 内に lg3d-dev, lg3d-app, lg3d-app-full という 3 つのコマンドができます。これらが LG3D を実行するためのコマンドです。その他のファイルは無視して構いません。

lg3d-dev, lg3d-app はウィンドウアプリケーションとして LG3D を実行するためのコマンドです。開発者用のコマンドと位置づけられています。

lg3d-dev と lg3d-app の違いは X-Window アプリケーションが実行できるかどうかです。

lg3d-dev は LG3D アプリケーションのみ実行でき、lg3d-app は X-Window アプリケーションの実行も行えます。

lg3d-app-full は lg3d-app のフルスクリーンモード（フレームなし）です。lg3d-app -f を実行するのと同じです。

各環境により利用できるコマンドは異なります。各環境で利用できるコマンドは次の通りです。

	lg3d-dev	lg3d-app	lg3d-app-full
<b>Solaris 10 x86 + NVIDIA</b>	○	○	○
<b>Linux + NVIDIA</b>	○	○	○
<b>Linux + ATI</b>	○	○	○
<b>Windows</b>	○	×	×

### 開発者向け モード (lg3d-dev/lg3d-app) の実行

lg3d-dev, lg3d-app の実行方法は次の通りです。

(インストールしたユーザーのホームディレクトリを /home/duke とします。)

```
# cd lg3d  
# bin/lg3d-dev ( bin/lg3d-app, bin/lg3d-app-full )
```

#### 補足:

Fedora Core 4 などの一部の Linux ディストリビューションの標準状態では lg3d-app コマンドが動かない場合があります。

その場合、/etc/X11/gdm/gdm.conf に DisallowTCP=false を追加し、X-Window の再起動を行ってください。

下図は Solaris 10 x86 JDS 上で Ig3d-dev を実行したものです。



### 補足:

LG3D のインストールディレクトリにある etc/Ig3d/displayconfig/j3d1x1 ファイル (ここでは /home/duke/Ig3d/etc/Ig3d/displayconfig/j3d1x1 ) の設定を変更することにより LG3D ウィンドウの大きさを変更できます。

下の例では画面サイズを 1152x864 に設定しています。画面サイズの代わりに NoBorderFullScreen と指定すればフルスクリーン表示も可能です。これは Ig3d-app-full を実行するのと同じです  
変更箇所を **赤(太字)** で示します。

```
/*
*****
*
* Java 3D Calibration file for single-screen desktop configuration with
* neither head tracking nor 6DOF sensor tracking.
*
*****
*/

```

```

// Create a new screen object and associate it with a logical name and a
// number. This number is used as an index to retrieve the AWT GraphicsDevice
// from the array that GraphicsEnvironment.getScreenDevices() returns.
//
(NewScreen center 0)

// Set the available image area for a full screen. This is important when
// precise scaling between objects in the virtual world and their projections
// into the physical world is desired through use of an explicit ScreenScale
// view attribute. The defaults are 0.365 meters for width and 0.292 meters
// for height.
//
(ScreenAttribute center PhysicalScreenWidth 0.360)
(ScreenAttribute center PhysicalScreenHeight 0.288)
//(ScreenAttribute center      WindowSize      NoBorderFullScreen)
//(ScreenAttribute center      WindowSize      (800 600))
(ScreenAttribute center WindowSize      (1152 864))

// Create a view using the defined screen.
//
(NewView view0)
(ViewAttribute view0 Screen center)
(ViewAttribute   view0  FrontClipDistance    0.01)
(ViewAttribute   view0  BackClipDistance     10.0)
(ViewAttribute view0 WindowEyepointPolicy RELATIVE_TO_COEXISTENCE)
(ViewAttribute view0 WindowMovementPolicy VIRTUAL_WORLD)
(ViewAttribute view0 WindowResizePolicy VIRTUAL_WORLD)
(ViewAttribute view0 ScreenScalePolicy SCALE_EXPLICIT)

// For debugging this will give us the standard scale world and view
// but in a window. Obviously if the window is reduced in size less of
// the world is visible.
(ViewAttribute view0 CoexistenceCenteringEnable true)
(ViewAttribute view0 WindowEyepointPolicy RELATIVE_TO_WINDOW)
(ViewAttribute view0 WindowMovementPolicy PHYSICAL_WORLD)

// Enable stereo viewing if desired
// (ViewAttribute view0 StereoEnable      true)

```

## 2.4 LG3D のインストールと実行方法 (Windows XP 編)

### 準備

Windows XP で LG3D 1.0 を実行するためのすべてのコンポーネントは Windows XP 専用のインストーラにまとめられています。

LG3D のウェブページからインストーラをダウンロードし、実行するだけでインストールを行えます。

- ・ ウェブページ URL <https://lg3d-core.dev.java.net/binary-builds.html>
- ・ インストーラ名 **Windows XP Mega Bundle**
- ・ ウェブページのスクリーンショット ↓



#### 補足:

LG3D 0.7.1 から Javadoc が独立したパッケージになりました。 (javadoc-lg3d-fcs-rel-0-7-1-generic-0510281819.tar.gz)

ただ、現状の配布形式は tar.gz 形式のため、Windows 環境で解凍するためには 別途 tar.gz に対応した解凍ツールが必要になります。

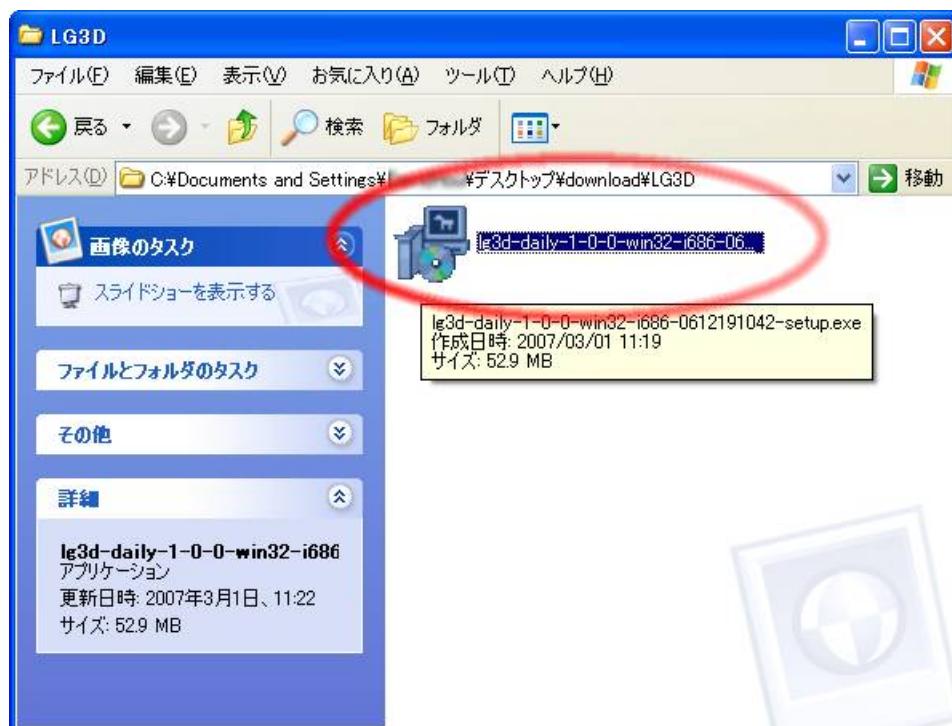
そのため、このテキストでは Javadoc のインストール部分は解説していません。

## LG3D Release 1.0 のインストール

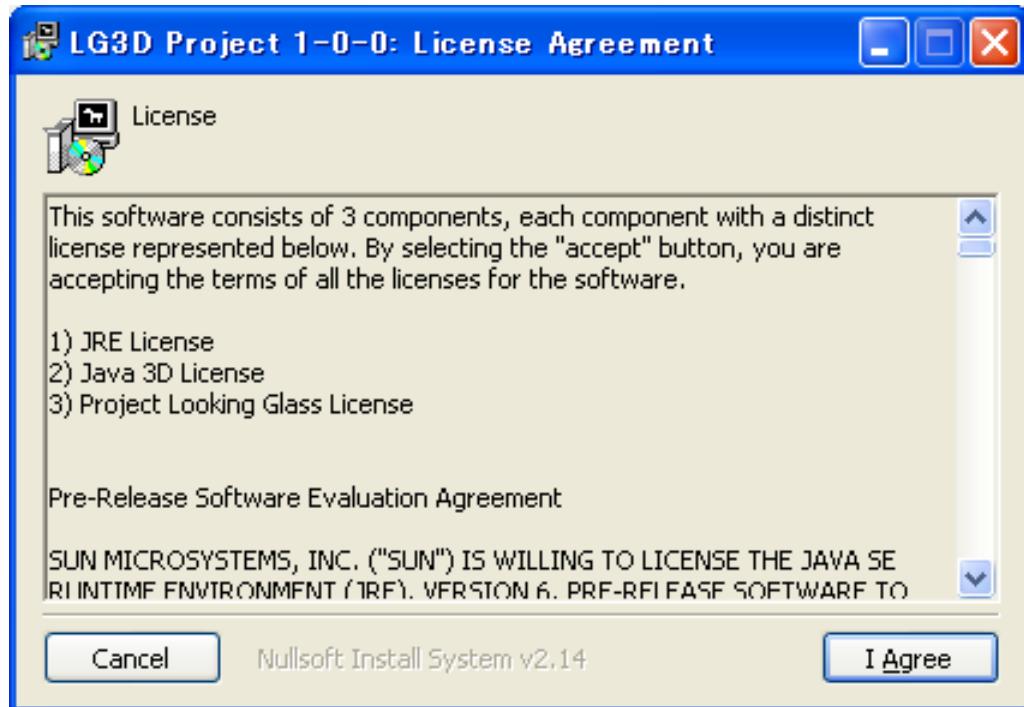
LG3Dのインストールは"Administrator"権限を持つユーザで行う必要があります。ここではデスクトップ上に **download\lg3d** というディレクトリを作成し、Windows XP 専用のインストーラをダウンロードしたとします。



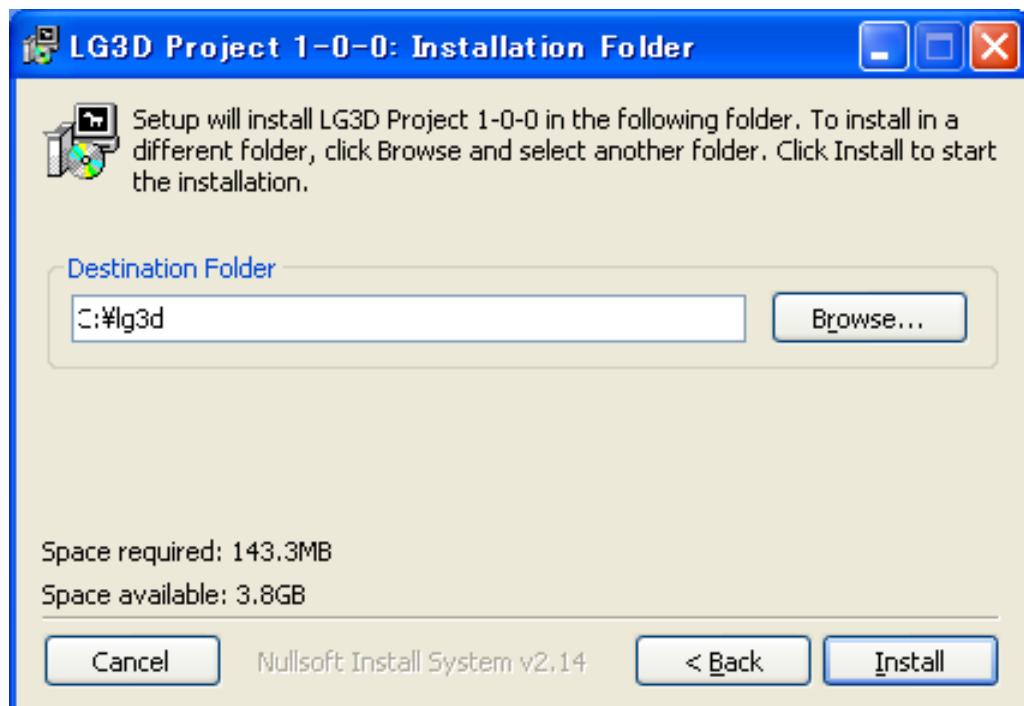
- **lg3d-daily-1-0-0-win32-i686-0612191042-setup.exe** をダブルクリックし、インストーラを起動します。



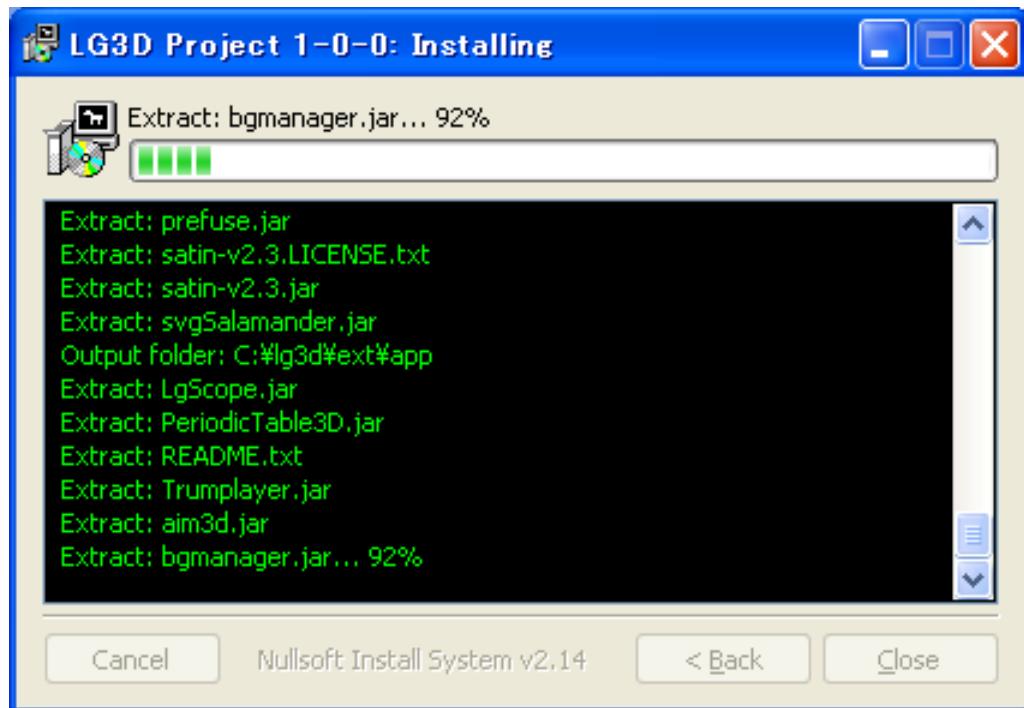
- しばらく待つとインストーラが起動します。  
ライセンスが表示されるので確認してください。  
インストールを継続する場合は「**I Agree**」をクリックします。



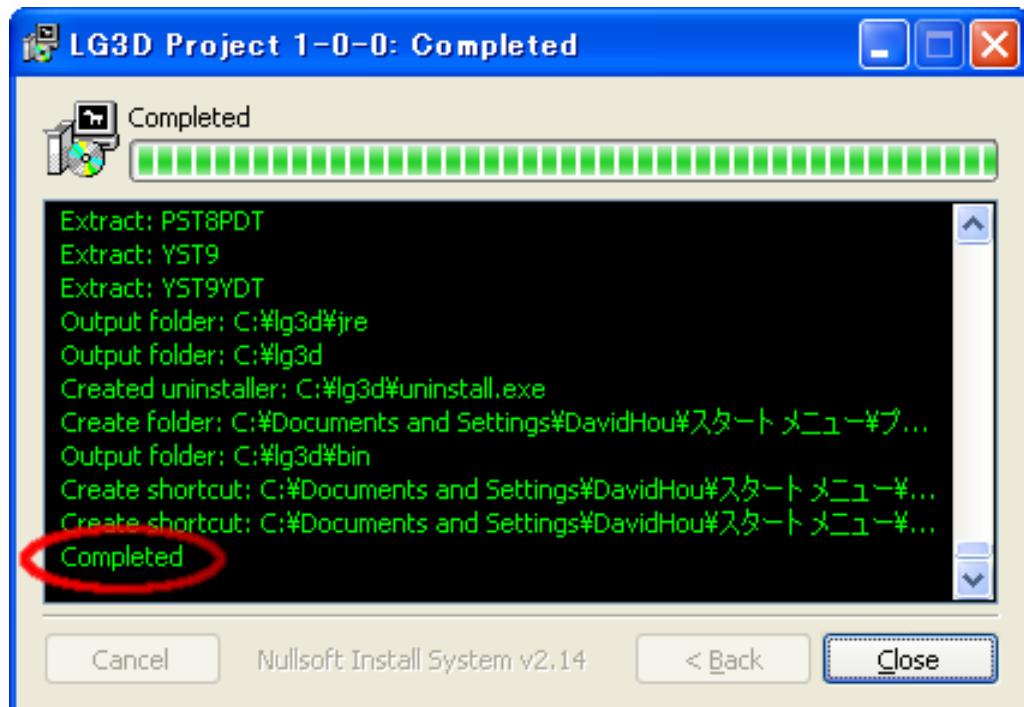
- インストールフォルダの選択に移ります。  
Destination Folder にはインストール先のディレクトリを入力します。  
この場合、インストール先は「**C:\lg3d\**」になります。  
続けるには「**Install**」ボタンをクリックします。



- LG3D Release 1.0がインストールされます。



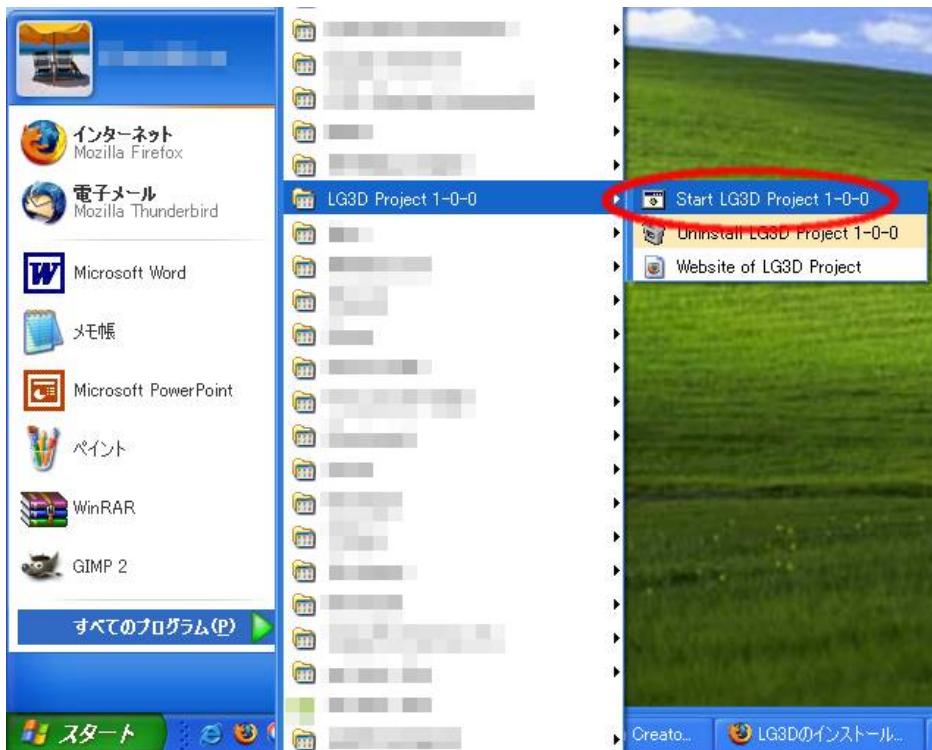
- LG3D Release 1.0のインストールの「Completed」が表示されるので、「Close」をクリックして、インストーラを終了します。



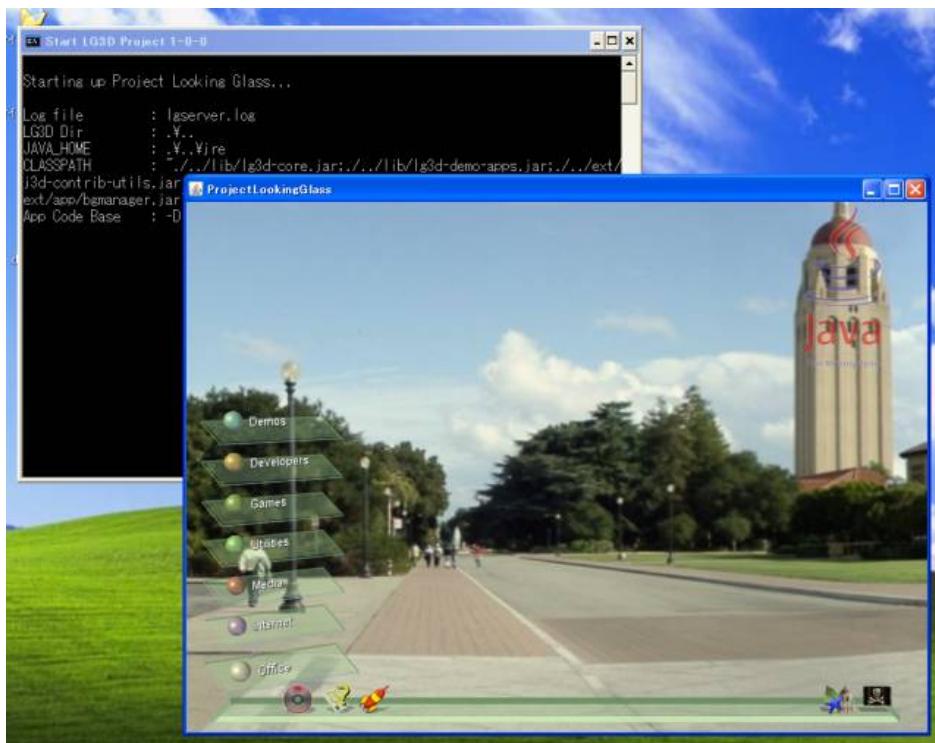
## LG3D の実行

LG3Dの実行はスタートメニューから行います。実行手順は以下の通りです。

- ・ 「スタートメニュー」から「すべてのプログラム」⇒「LG3D Project 1-0-0」⇒「Start LG3D Project 1-0-0」を選択します。



- 下図は実行画面です。



## 補足

インストールディレクトリの etc\lg3d\displayconfig\j3d1x1 ファイルの設定を変更することにより LG3D ウィンドウの大きさを変更することができます。（ここでは c:\lg3d\etc\lg3d\displayconfig\j3d1x1 になります）  
下の例では画面サイズを 1152x864 にしています。画面サイズの代わりに NoBorderFullScreen と指定することでフルスクリーン表示も可能です。  
変更する箇所を **赤(太字)** で示します。

```
/*
*****
*
* Java 3D Calibration file for single-screen desktop configuration with
* neither head tracking nor 6DOF sensor tracking.
*
*****
*/
// Create a new screen object and associate it with a logical name and a
// number. This number is used as an index to retrieve the AWT GraphicsDevice
// from the array that GraphicsEnvironment.getScreenDevices() returns.
(NewScreen ProjectLookingGlass 0)

// Set the available image area for a full screen. This is important when
// precise scaling between objects in the virtual world and their projections
// into the physical world is desired through use of an explicit ScreenScale
// view attribute. The defaults are 0.365 meters for width and 0.292 meters
// for height.
(ScreenAttribute ProjectLookingGlass PhysicalScreenWidth 0.360)
(ScreenAttribute ProjectLookingGlass PhysicalScreenHeight 0.288)
//(ScreenAttribute ProjectLookingGlass WindowSize      NoBorderFullScreen)
//(ScreenAttribute ProjectLookingGlass WindowSize      (800 600))
(ScreenAttribute ProjectLookingGlass WindowSize      (1152x864))

// Create a view using the defined screen.
//
(NewView view0)
(ViewAttribute view0 Screen ProjectLookingGlass)
(ViewAttribute view0 FrontClipDistance 0.01)
(ViewAttribute view0 BackClipDistance 10.0)
(ViewAttribute view0 WindowEyepointPolicy RELATIVE_TO_COEXISTENCE)
(ViewAttribute view0 WindowMovementPolicy VIRTUAL_WORLD)
(ViewAttribute view0 WindowResizePolicy VIRTUAL_WORLD)
(ViewAttribute view0 ScreenScalePolicy SCALE_EXPLICIT)

// For debugging this will give us the standard scale world and view
// but in a window. Obviously if the window is reduced in size less of
// the world is visible.
(ViewAttribute view0 CoexistenceCenteringEnable true)
(ViewAttribute view0 WindowEyepointPolicy RELATIVE_TO_WINDOW)
(ViewAttribute view0 WindowMovementPolicy PHYSICAL_WORLD)

// Enable stereo viewing if desired
// (ViewAttribute view0 StereoEnable      true)
```

## JDK と Java3D のインストールと環境変数の設定

Windows 版 LG3D は実行のための JRE(Java Runtime Environment) が組み込まれていますが、開発環境である JDK は含まれていません。LG3D アプリケーションの開発を行うためには JDK 6 と Java3D 1.5 をインストールする必要があります。

まず、JDK6 のインストーラと Java3D 1.5 のバイナリをダウンロードします。

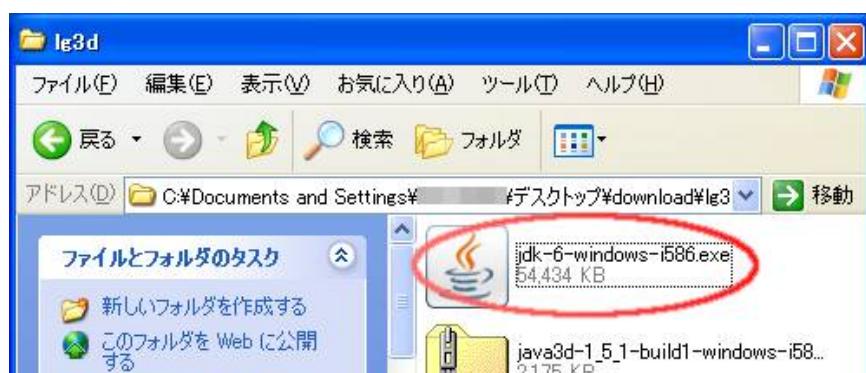
以下のファイルは執筆時点(2007/3/1)での最新版です。できるだけ最新版を利用してください。ダウンロードしたファイルは デスクトップ\download\lg3d にあるとします。

コンポーネント名	ファイル名	ダウンロード元
JDK 6.0	jdk-6-windows-i586.exe	<a href="http://java.sun.com/javase/ja/6/download.html">http://java.sun.com/javase/ja/6/download.html</a>
Java 3D 1.5	java3d-1_5_1-build1-windows-i586.zip	<a href="https://java3d.dev.java.net/binary-builds-pre.html">https://java3d.dev.java.net/binary-builds-pre.html</a>

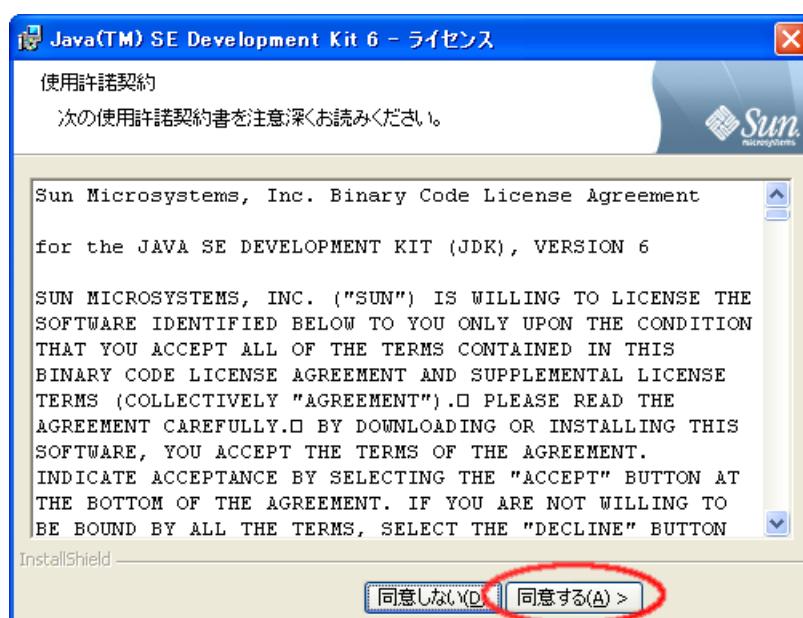
### ・ JDK のインストール

インストールは "Administrator" 権限を持つユーザで行う必要があります。

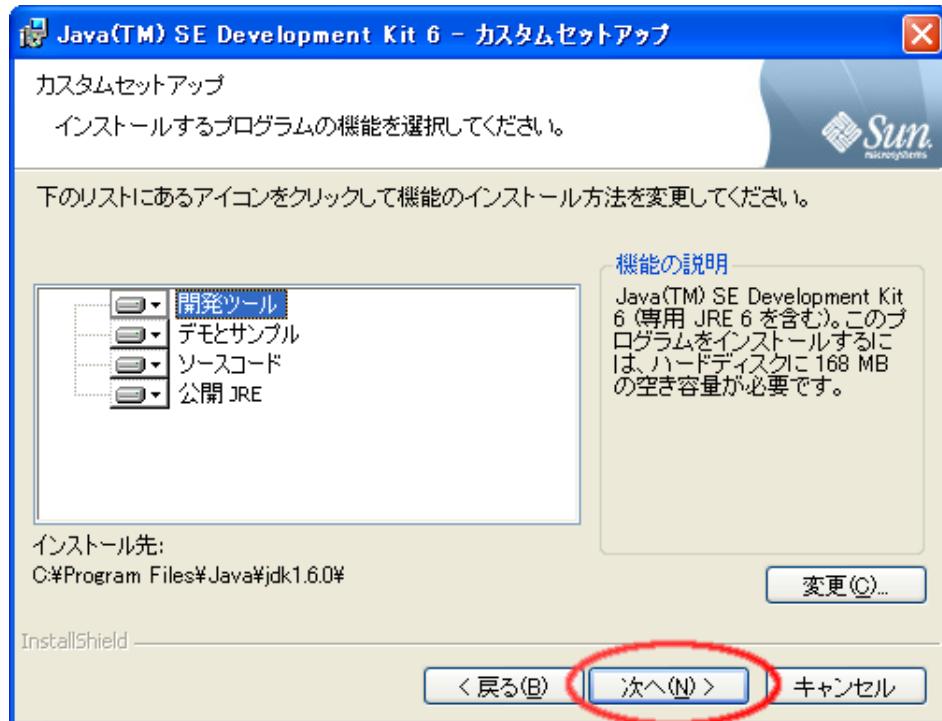
1. jdk-6-windows-i586.exe をダブルクリックし、インストーラを起動します。



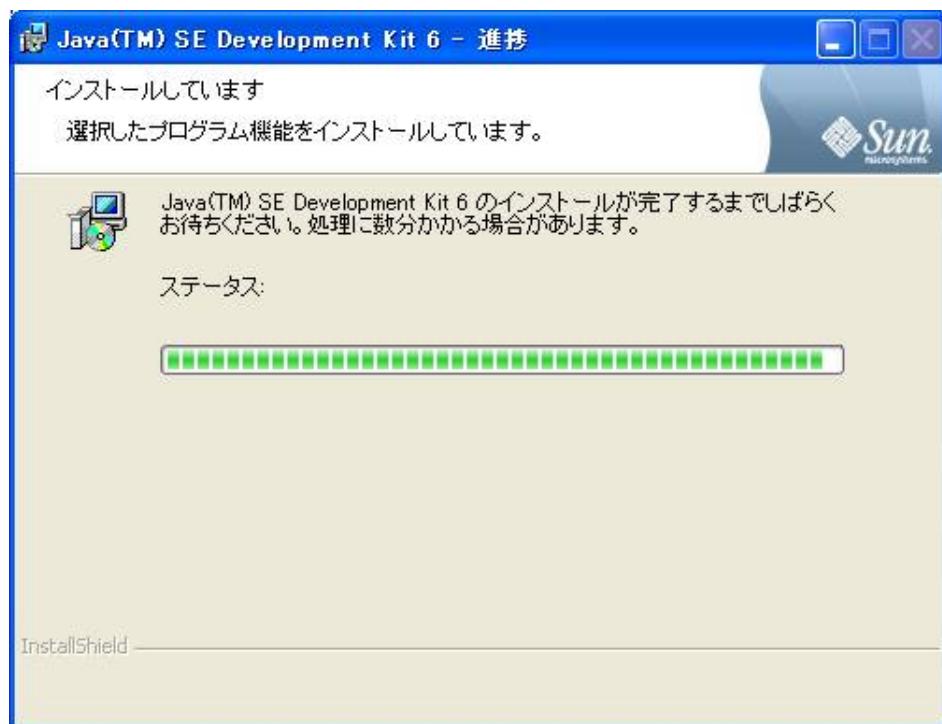
2. しばらく待つとインストーラが起動します。ライセンスが表示されるので確認してください。インストールを継続する場合は「同意する (A) >」をクリックします。



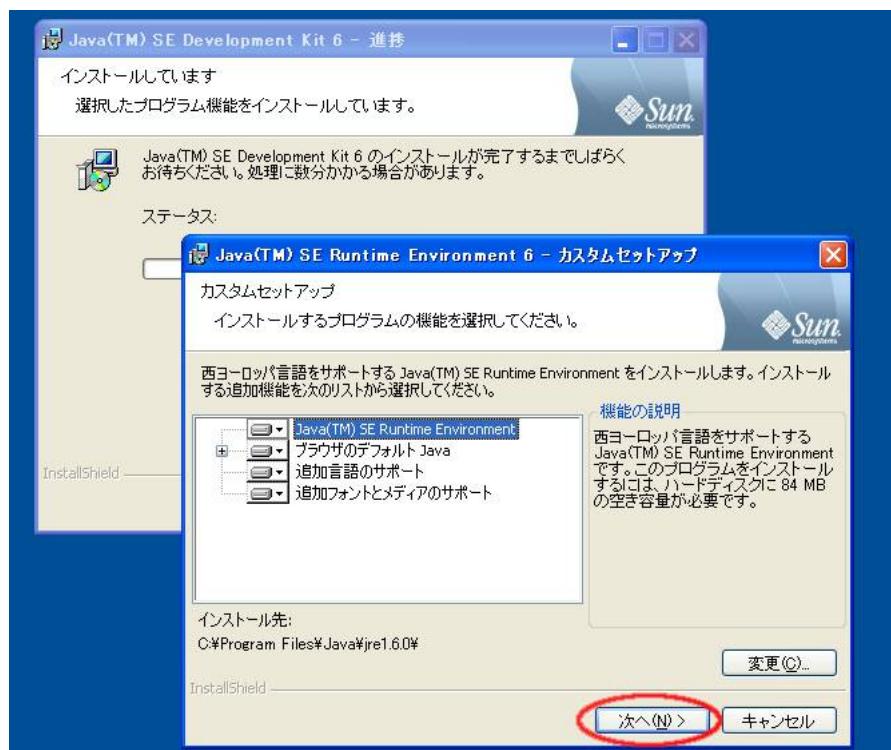
3. カスタムセットアップの項目に移ります。ここではデフォルト設定のままインストールします。インストール先は「C:\Program Files\Java\jdk1.6.0」になります。続けるには「次へ」をクリックします。



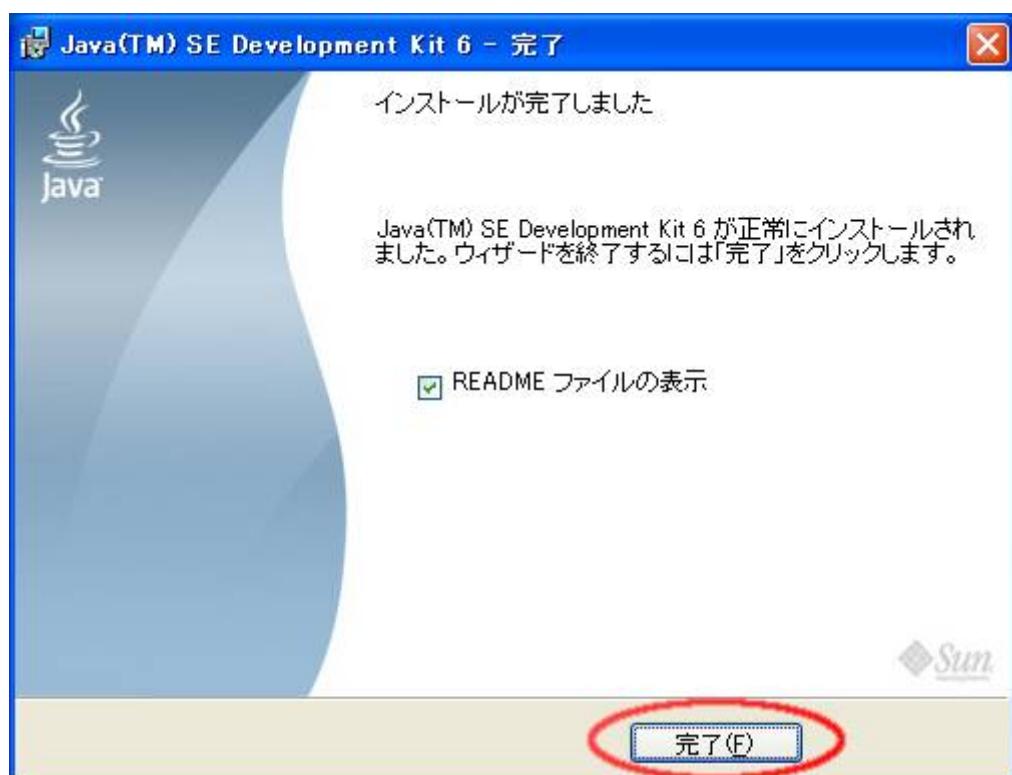
4. JDK がインストールされます。



5. JDK のインストール途中で、JRE のインストールダイアログが表示されます。JRE のカスタムセットアップの項目になります。ここではデフォルト設定のままインストールします。「**次へ**」をクリックします。



6. しばらく待つと、JDK のインストールの終了が表示されます。ここで「README」ファイルを表示する必要が無い場合は「 **README ファイルの表示**」のチェックボックスを外します。「**完了**」をクリックして、インストーラを終了します。



- Java 3D のインストール

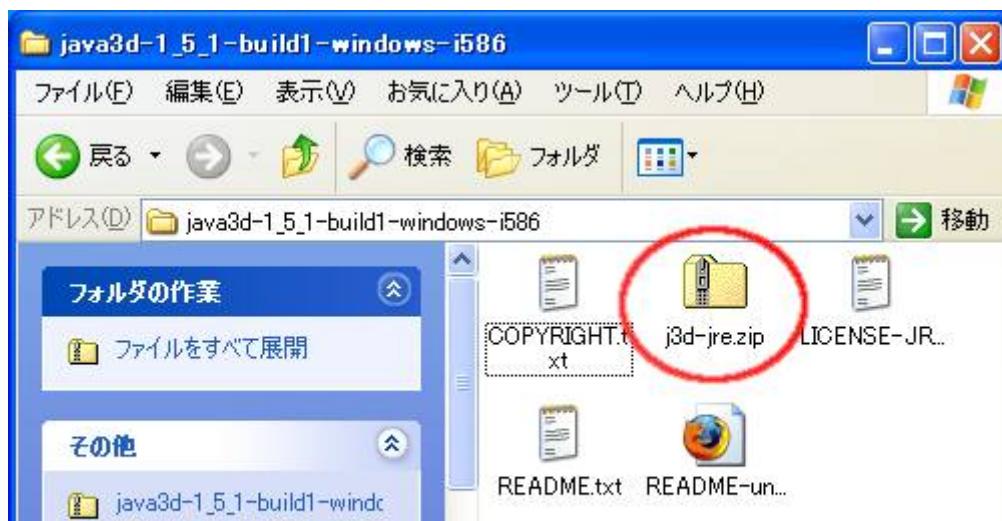
1. **java3d-1\_5\_1-build1-windows-i586.zip** をダブルクリックします。



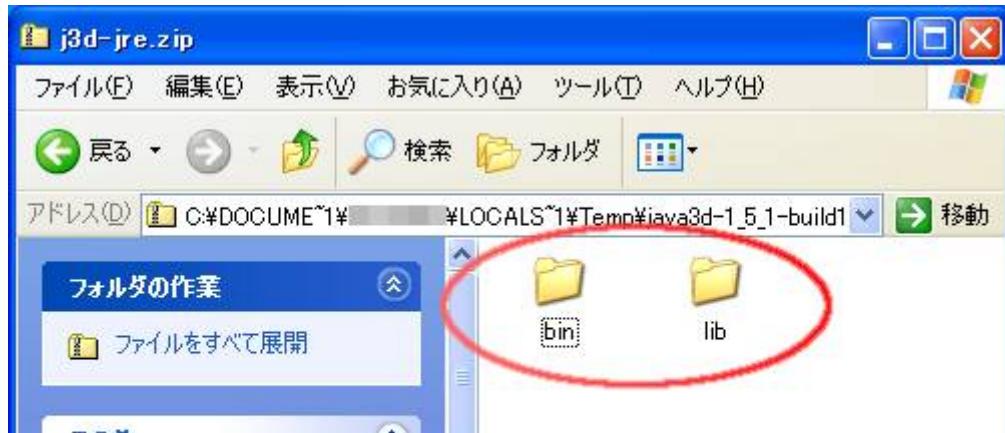
2. zip ファイル内のフォルダが表示されます。フォルダをダブルクリックします。



3. **j3d-jre.zip** 圧縮フォルダをダブルクリックします。



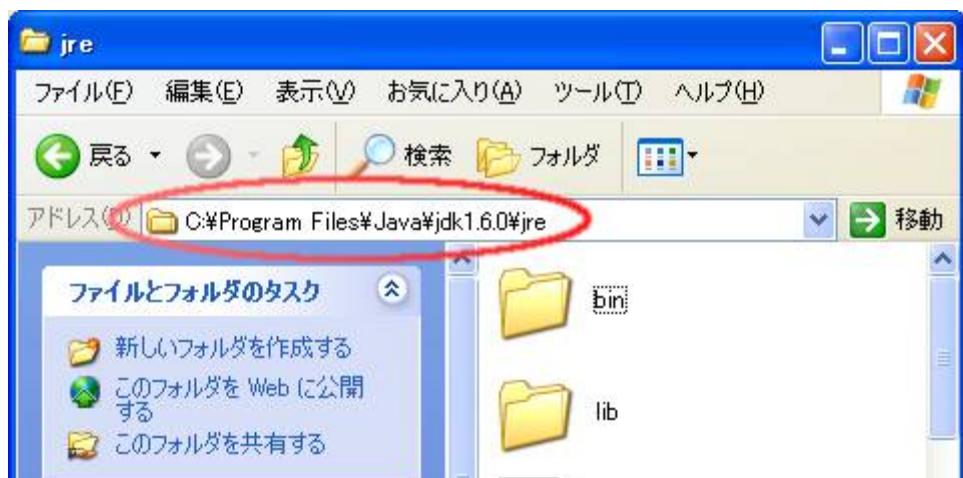
4. **bin** と **lib** が表示されるので、両方選択します。



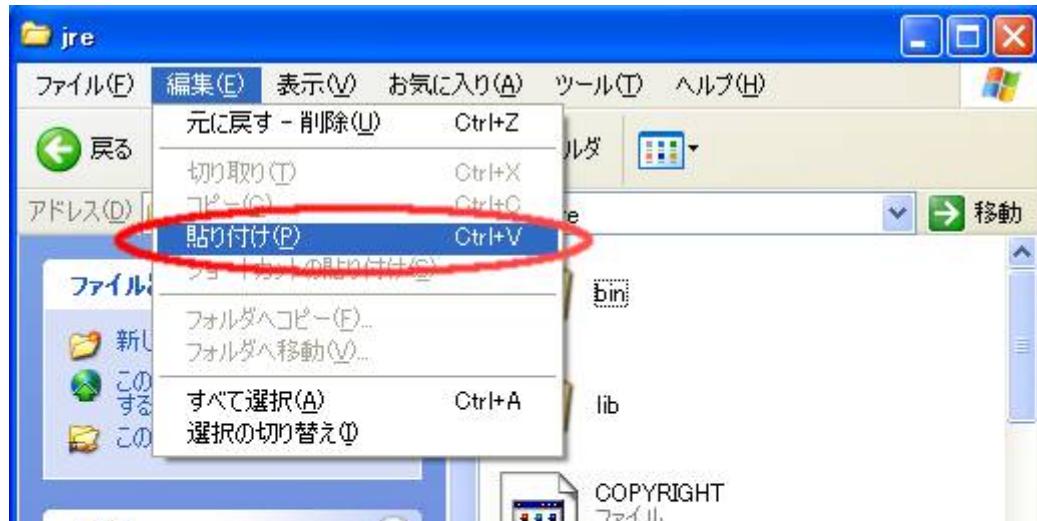
5. 「**編集**」 → 「**コピー**」を選択します。



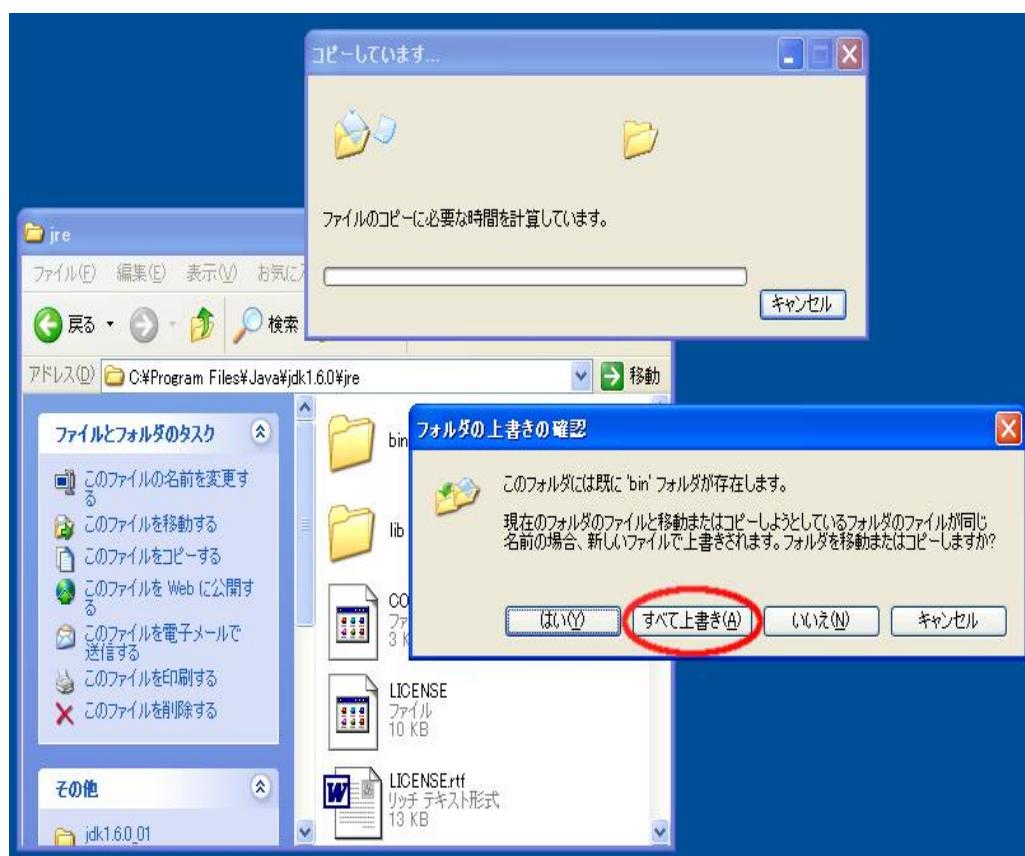
6. JDKをインストールしたディレクトリにある jre ディレクトリ (**C:\Program Files\Java\jdk1.6.0\jre**) をエクスプローラで開きます。



7. 「編集」→「貼り付け」を選択します。



8. フォルダの上書き確認が出ますので、「すべて上書き」を選択します。



これで JDK と Java3D のインストールは完了です。

### 3 NetBeans による LG3D プログラミング (準備編)

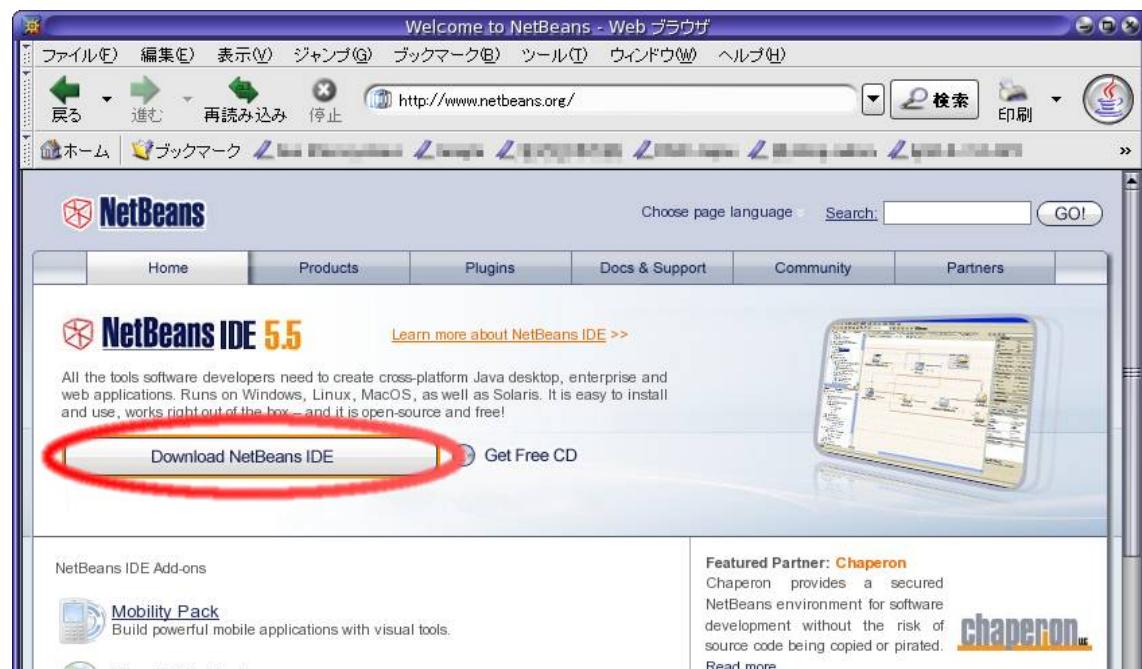
#### 3.1 NetBeans のインストール

##### NetBeans のダウンロード

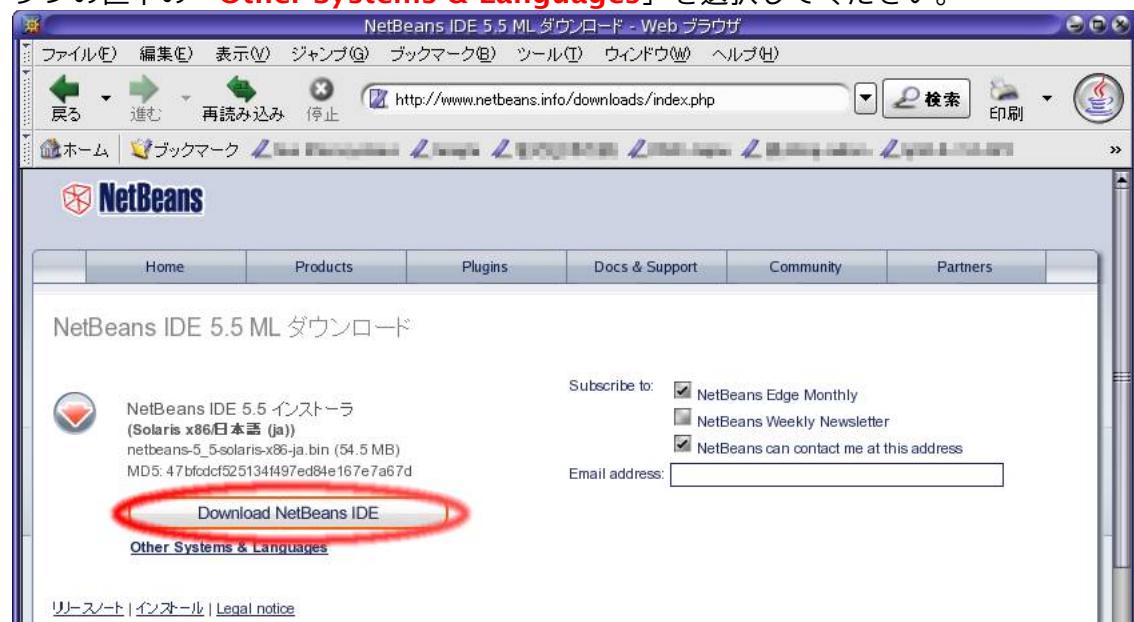
NetBeans の最新版を <http://www.netbeans.org/> からダウンロードします。

NetBeans 5.5 のダウンロードの手順は次の通りです。

1. ブラウザで <http://www.netbeans.org/> を開きます。  
「Download NetBeans IDE」をクリックします。



2. 次に、ご使用の環境と対応言語が自動に選択されたダウンロードページに進みます。  
ボタン「Download NetBeans IDE」をクリックして、ダウンロードが開始します。  
ほかの環境と言語に対応した NetBeans IDE をダウンロードする場合は、ダウンロードボタンの直下の「Other Systems & Languages」を選択してください。



## NetBeans のインストール

NetBeans 5.5のインストール手順は以下の通りです。

ルートユーザでインストーラを起動した場合 /opt がデフォルトのインストール先になります。

一般ユーザで起動した場合はホームディレクトリがデフォルトのインストール先になります。

ここではルートユーザでインストールを行います。

1. ダウンロードしたファイル(**netbeans-5\_5-solaris-x86-ja.bin**)の実行権限を変更し、実行します。

```
# chmod 755 netbeans-5_5-solaris-x86-ja.bin  
# ./netbeans-5_5-solaris-x86-ja.bin
```

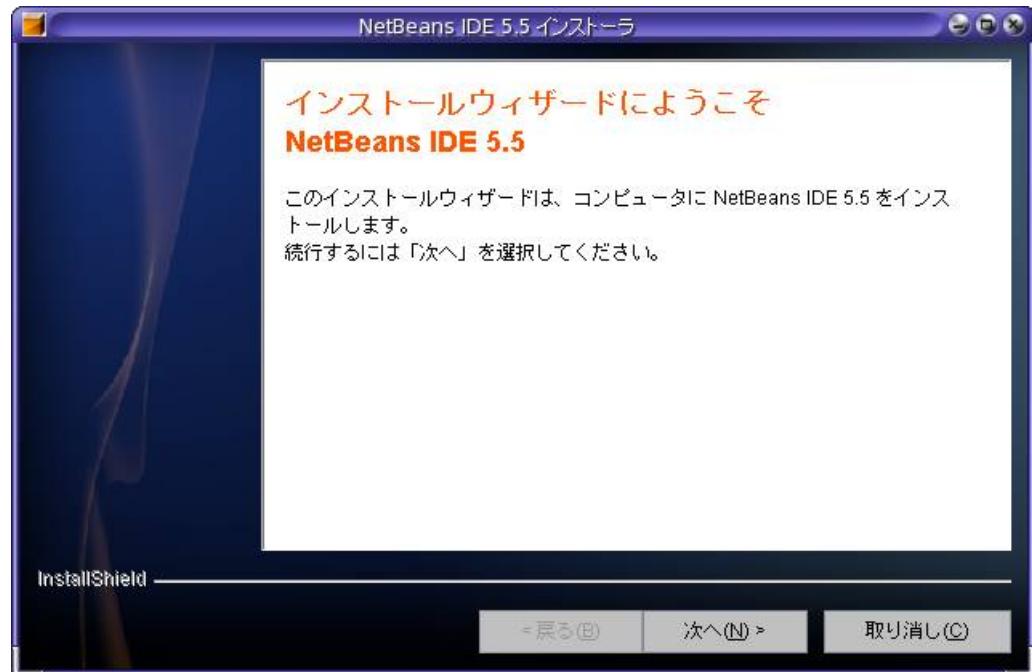
InstallShield Wizard

InstallShield Wizard を初期化中です...

Java(tm) 仮想マシンを検索中です...

.....

2. インストーラが起動しますので、「**次へ**」をクリックします。

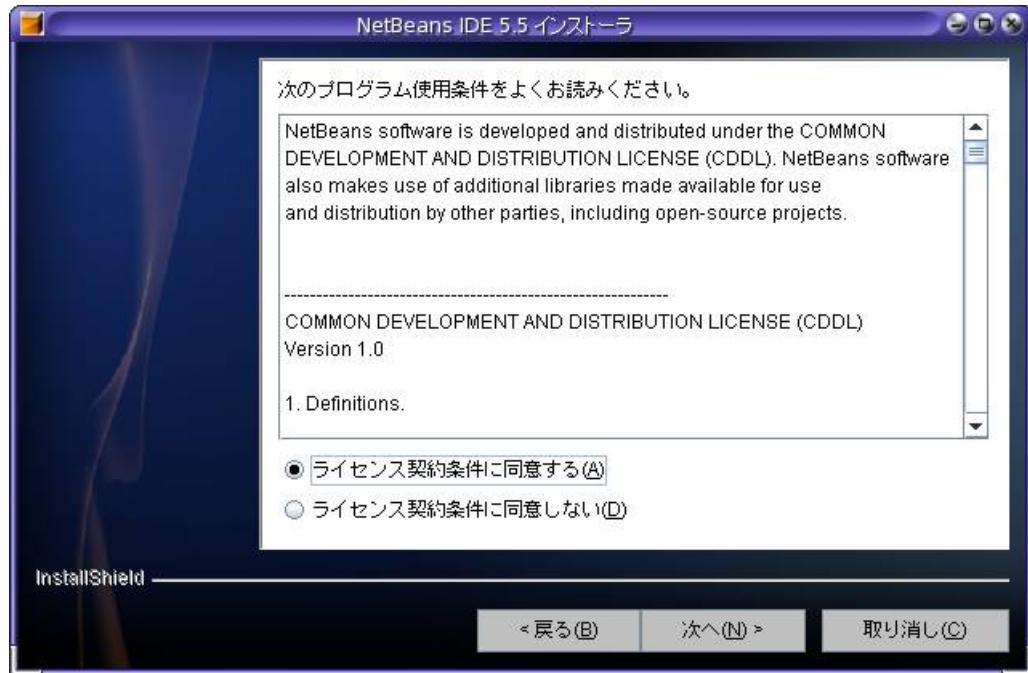


※Linuxの場合インストーラが文字化け（日本語が口口化）する事があります。

「[LG3Dのインストール（Linux編）](#)」の JDKの日本語フォント設定を参考にして設定を行ってください。

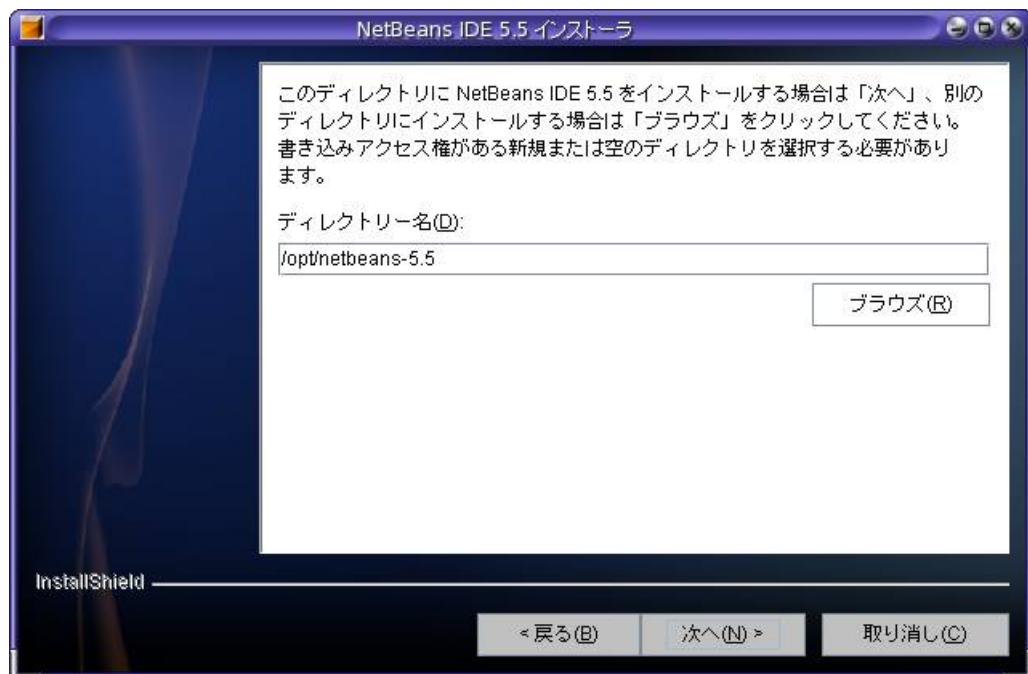
3. ライセンス条項を確認します。

「**ライセンス契約 条件に同意する**」のチェックボックスを選択後、「次へ」をクリックします。

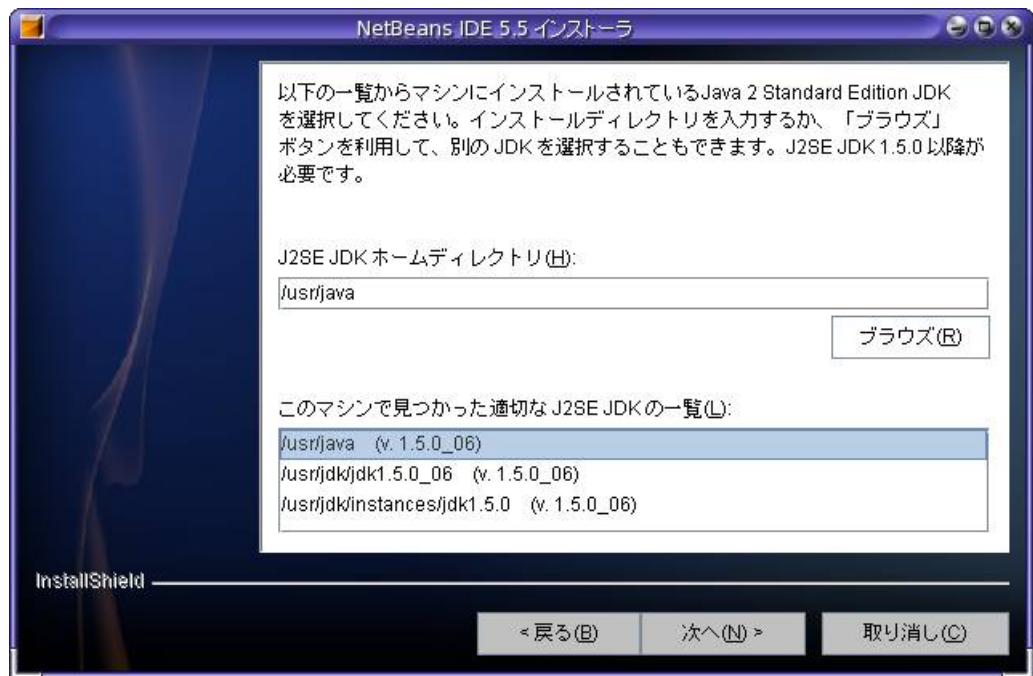


4. インストールするディレクトリを入力します。ここではデフォルト(**/opt/netbeans-5.5**)をインストール先とします。

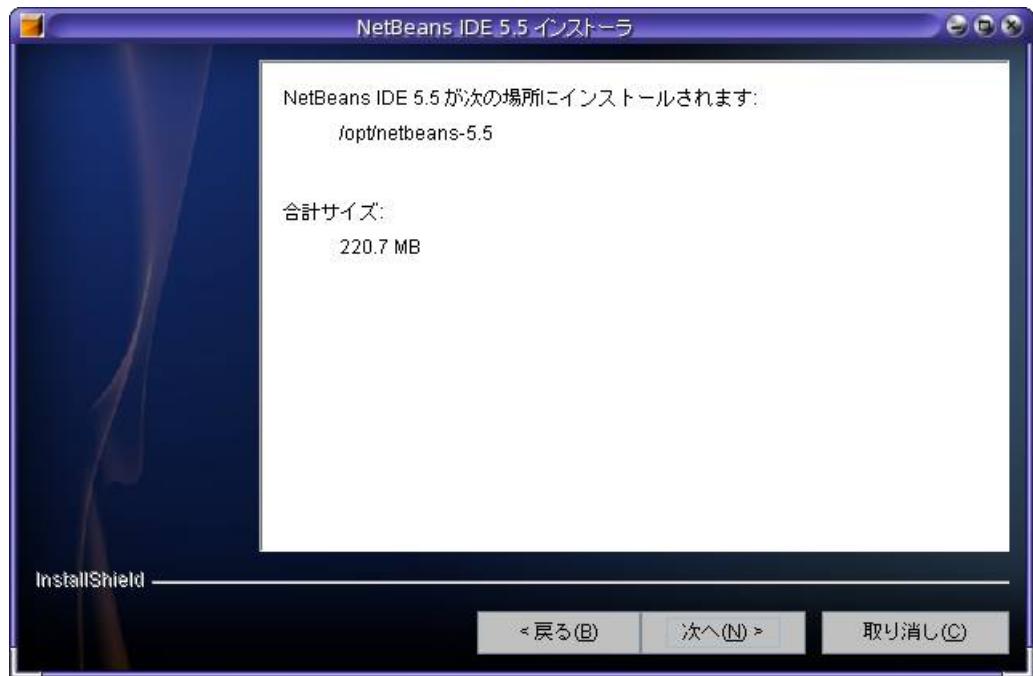
「次へ」をクリックします。



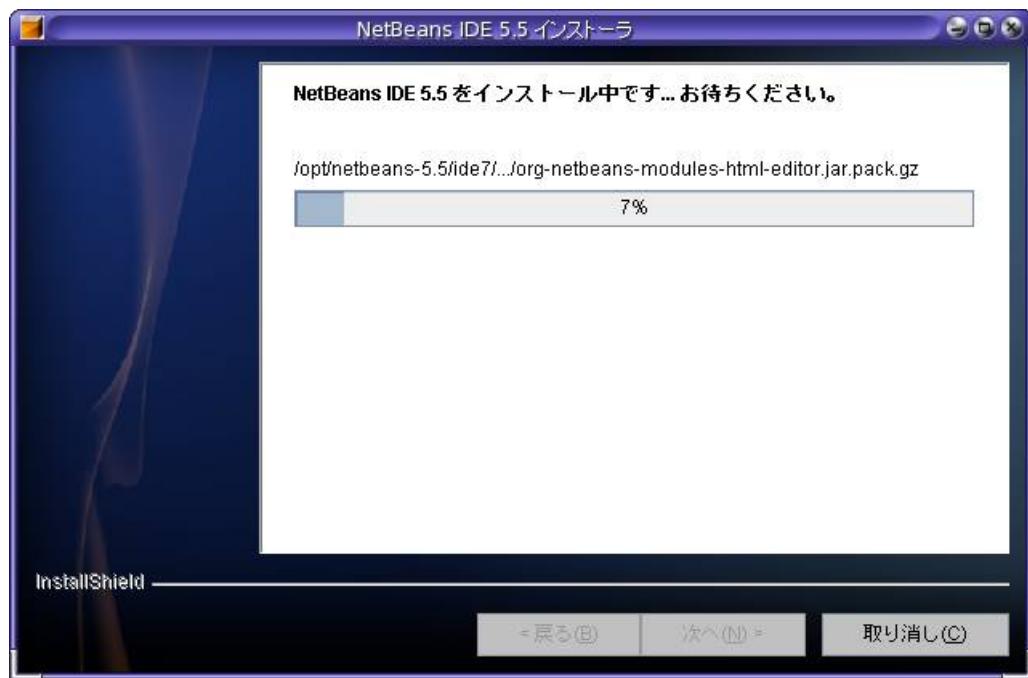
5. 使用する JDK を指定します。 選択後「次へ」をクリックします。



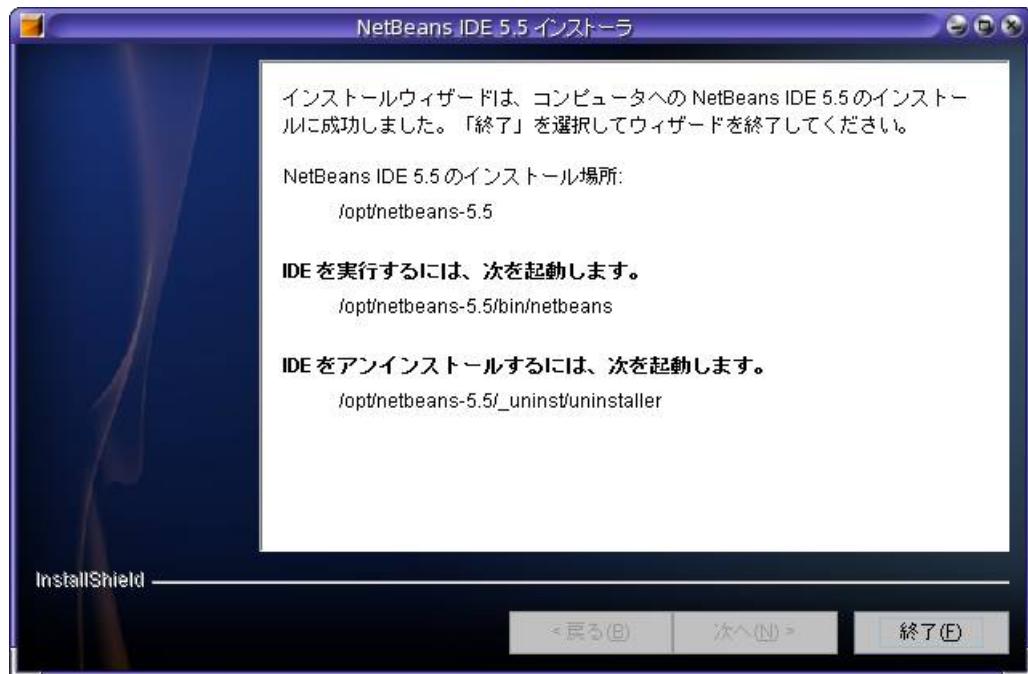
6. インストールの確認画面が表示されるので、「次へ」をクリックします。



7. インストール中の画面です。



8. 「終了」をクリックし、インストーラを終了します。



## 3.2 NetBeans のセットアップ

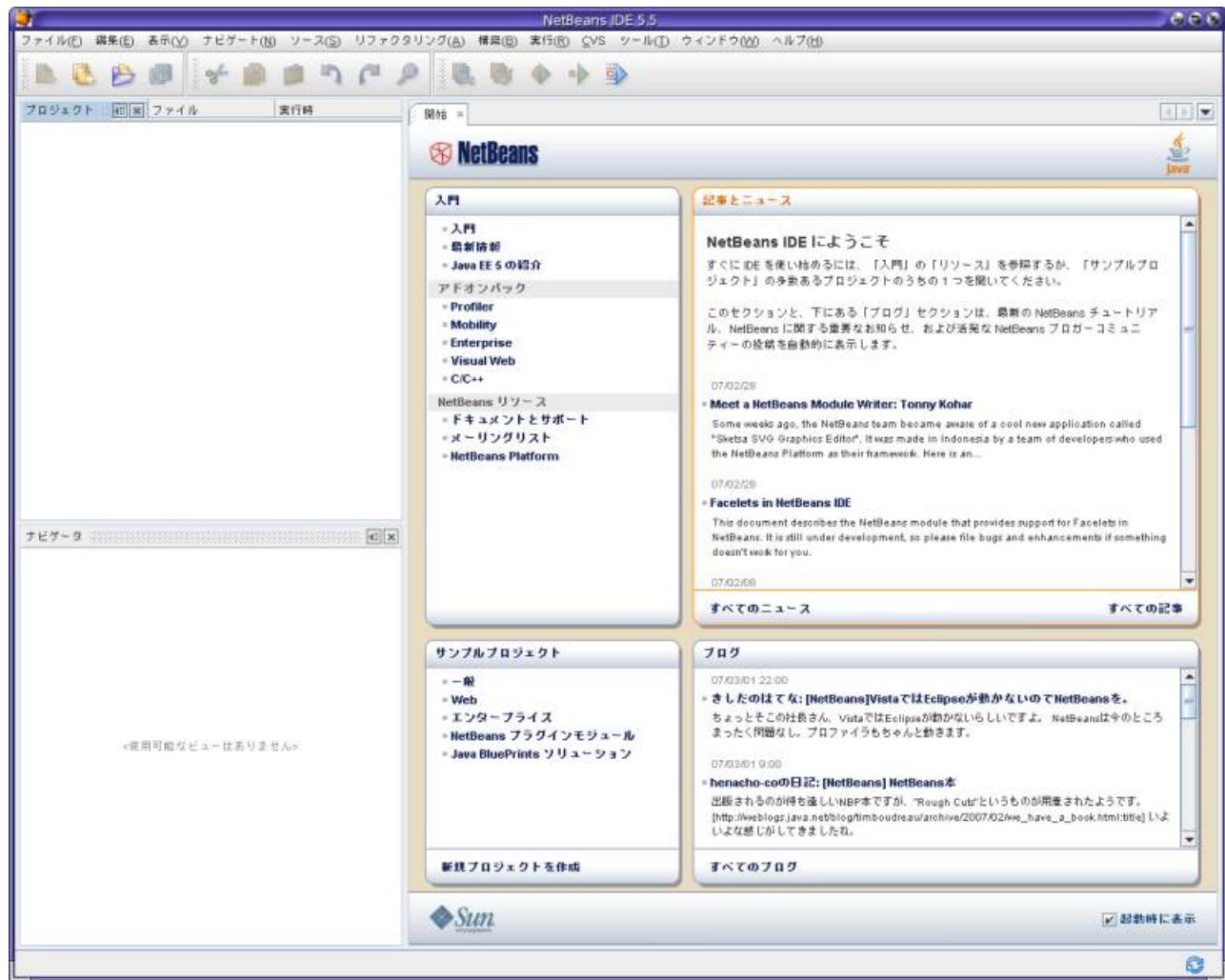
NetBeans のインストールが終了したら、次に LG3D でプログラミングを行うための準備を行います。準備の内容は「[LG3D 用のコンパイラー環境の登録](#)」と「[LG3D API の登録](#)」です。

### NetBeans の起動

NetBeans の起動を行います。端末エミュレータ上で次のコマンドを実行します。

```
# /opt/netbeans-5.5/bin/netbeans
```

コマンドを実行すると NetBeans 5.5 の起動画面が表示されます。しばらく待つと IDE が起動します。

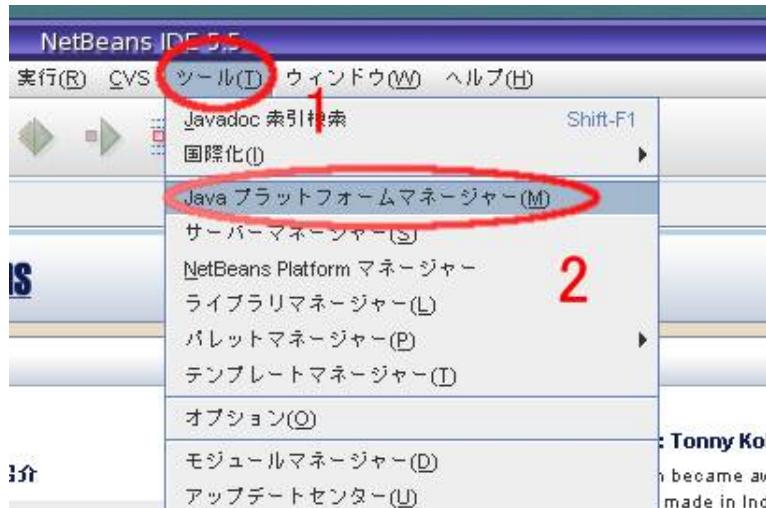


## Platform Manager

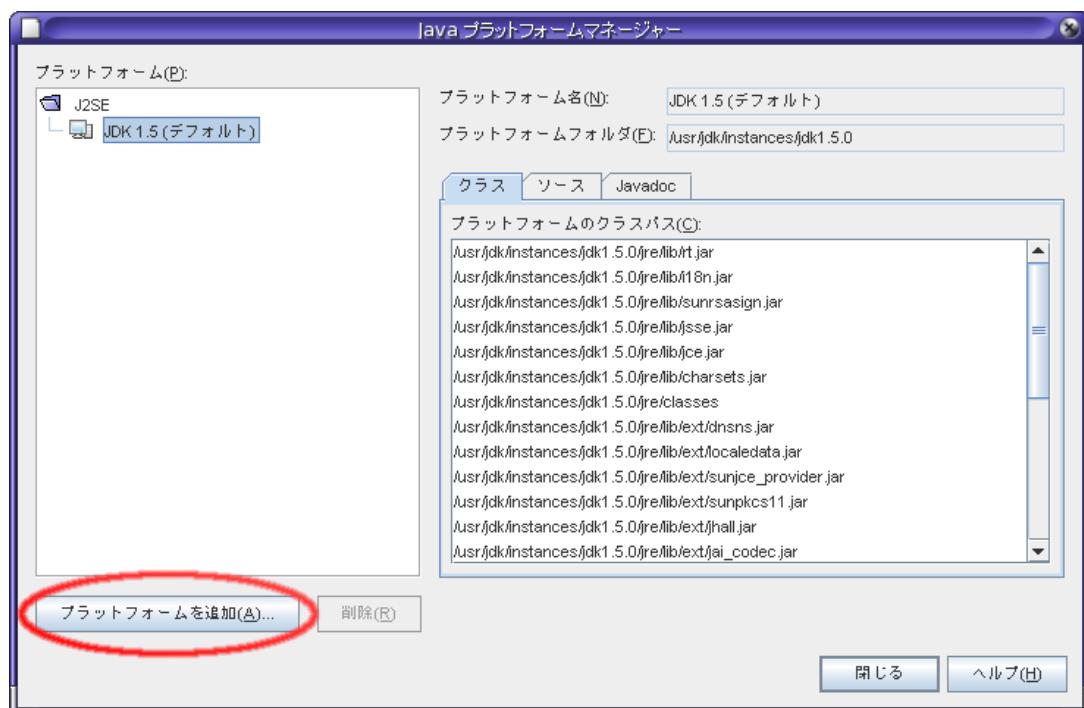
Platform Manager は JDK 環境を管理しています。

ここでは、システム標準の JDK とは別に、あらかじめインストールしておいた LG3D 専用の JDK(Java3D)環境を登録します。登録手順は次のようにになります。

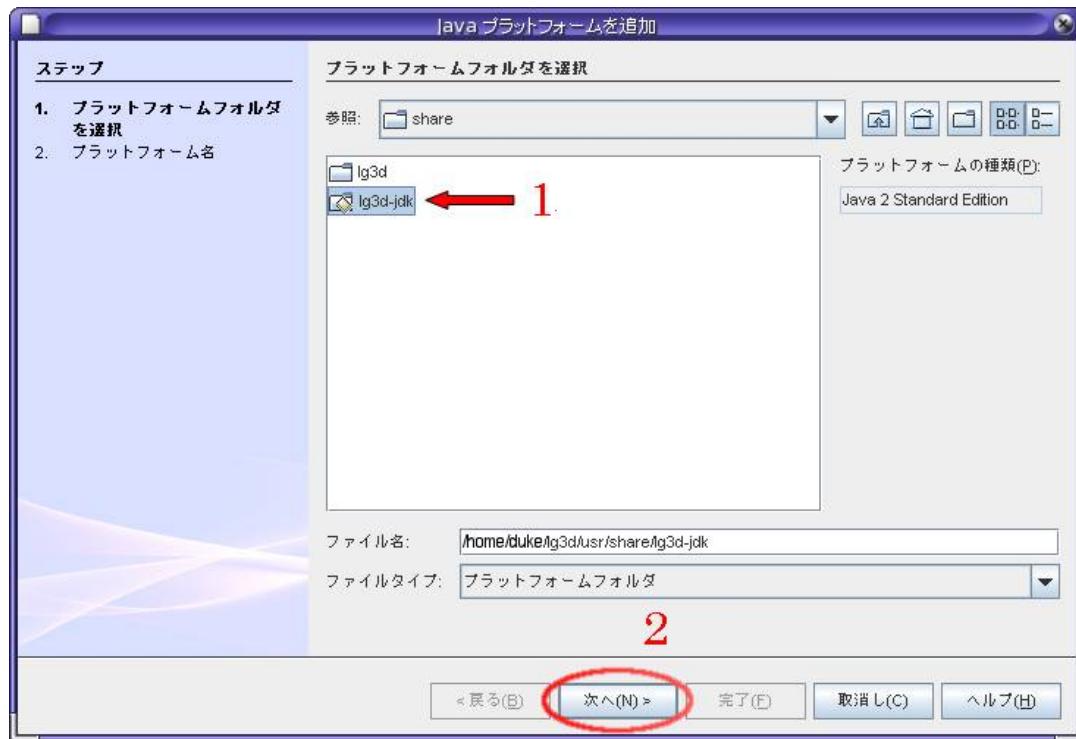
1. NetBeans のメニューから 「ツール」 → 「Java プラットフォームマネージャ」 を選択します。



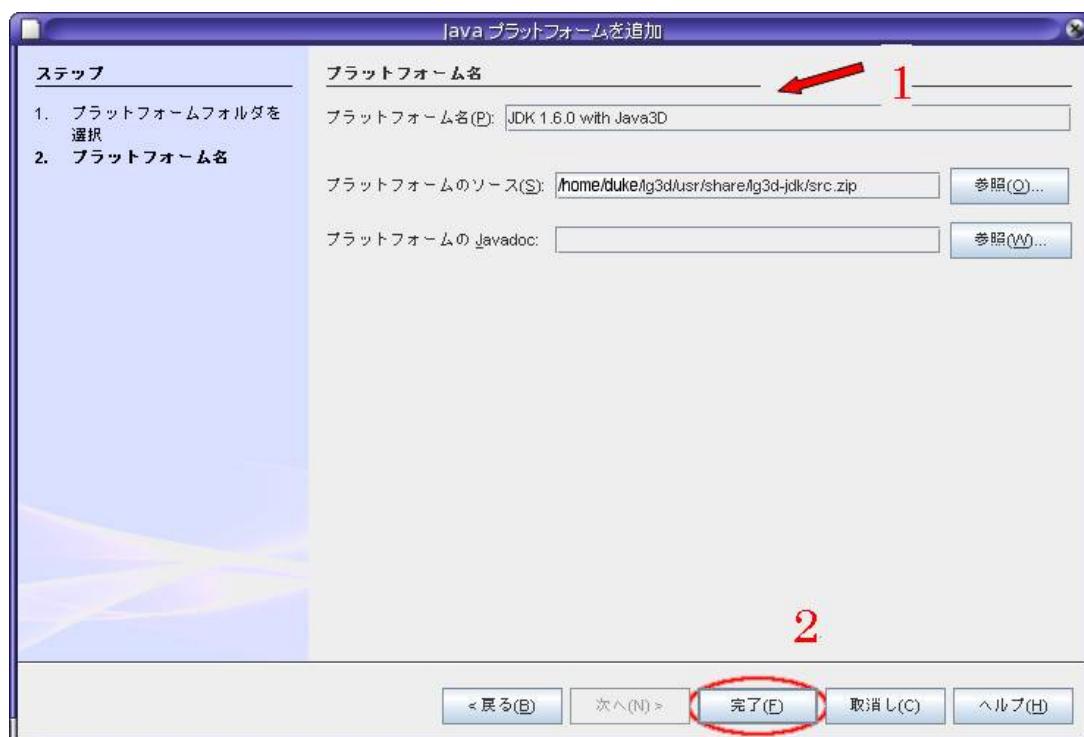
2. Java プラットフォームマネージャダイアログが表示されるので、「**プラットフォームを追加**」をクリックします。



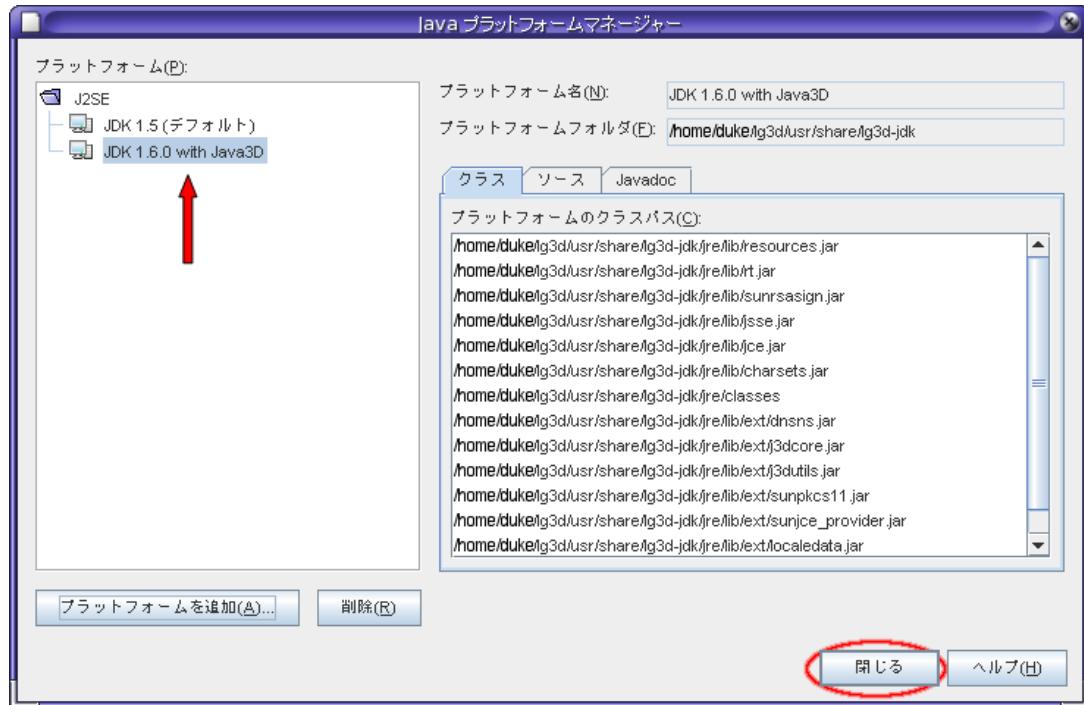
3. ファイラが表示されるので JDK のフォルダ(ディレクトリ)を選択します。  
ここでは **/home/duke/lg3d/usr/share/lg3d-jdk** となります。  
Windows の場合、**C:\Program Files\Java\jdk1.6.0** を指定します。  
JDK のディレクトリを選択後「次へ」をクリックします。



4. プラットフォーム名の確認が出ますので適宜変更します。  
ここでは「**JDK 1.6.0 with Java3D**」とします。  
プラットフォーム名を入力後「完了」を押します。



5. Java プラットフォームマネージャに追加されていることを確認し、ウィンドウを閉じます。



## Library Manager

Library Manager は外部の API(ライブラリ)の管理を行うためのツールです。

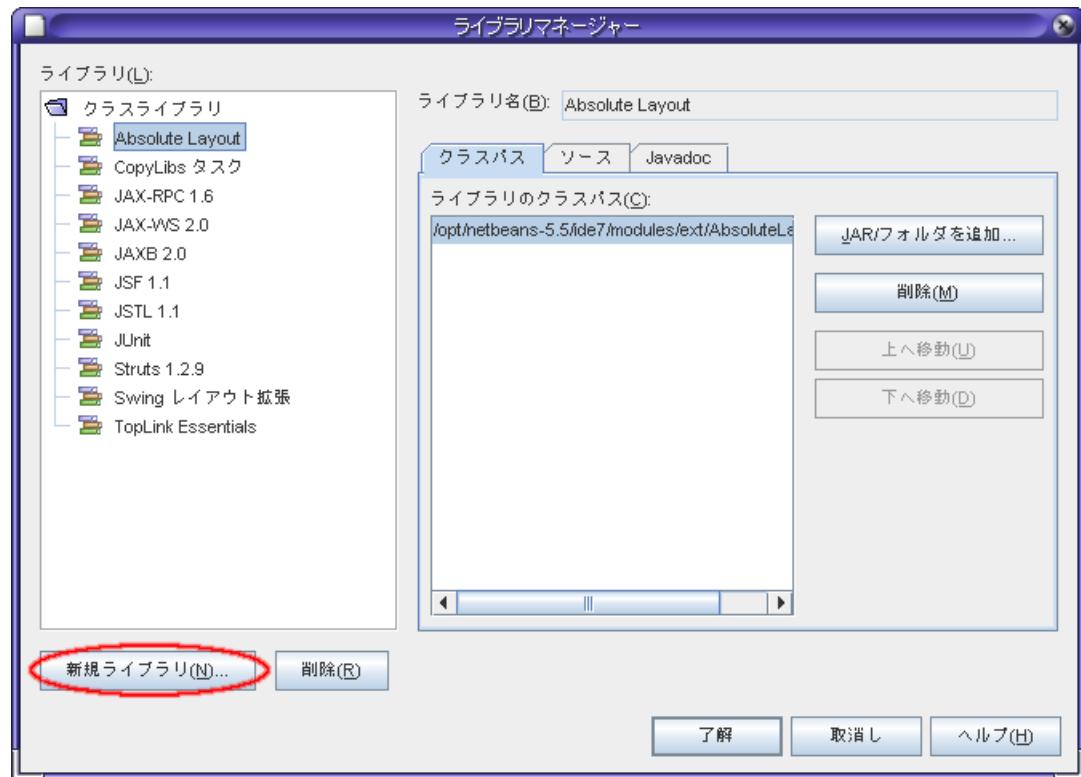
ここでは、LG3D プログラミングを行うために lg3d-core.jar(lg3d-core の API 群) を登録します。

登録手順は次の通りです。

1. NetBeans のメニューから 「ツール」 → 「ライブラリマネージャ」を選択します。



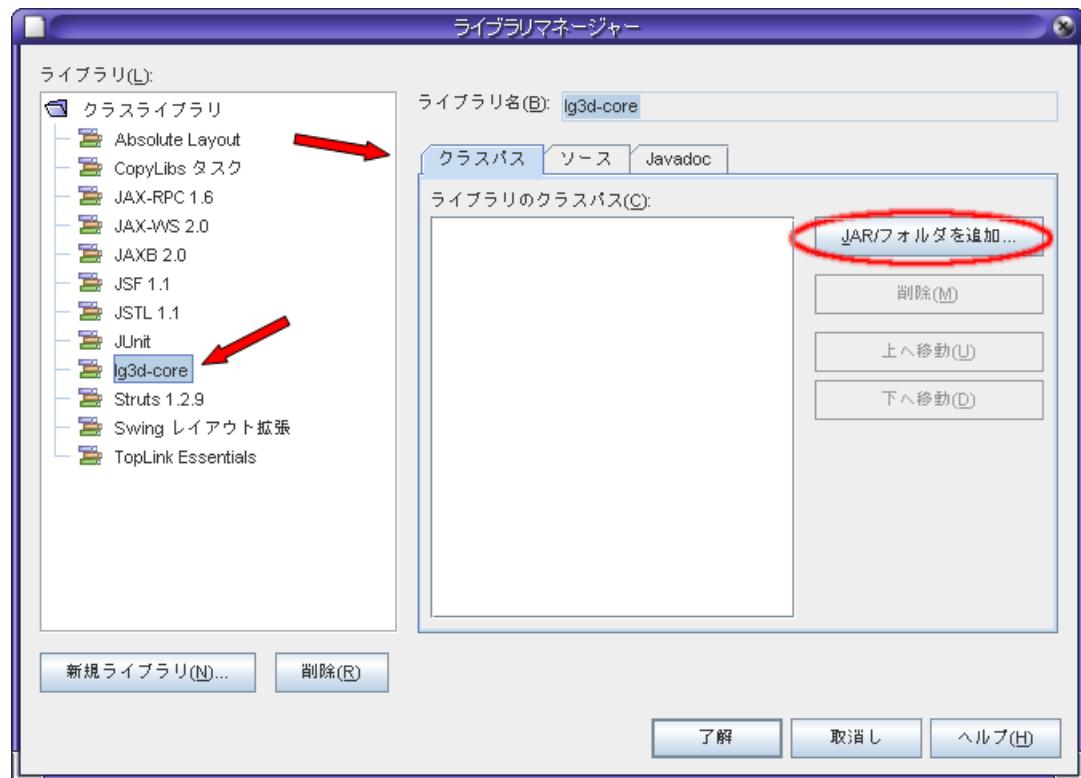
2. ライブラリマネージャダイアログが表示されますので、「**新規ライブラリ**」を選択します。



3. 新規ライブラリダイアログが表示されるのでライブラリ名を入力します。ここでは**lg3d -core**にします。  
また、ライブラリの種類を「**クラスライブラリ**」にします。  
入力後「**了解**」をクリックします。



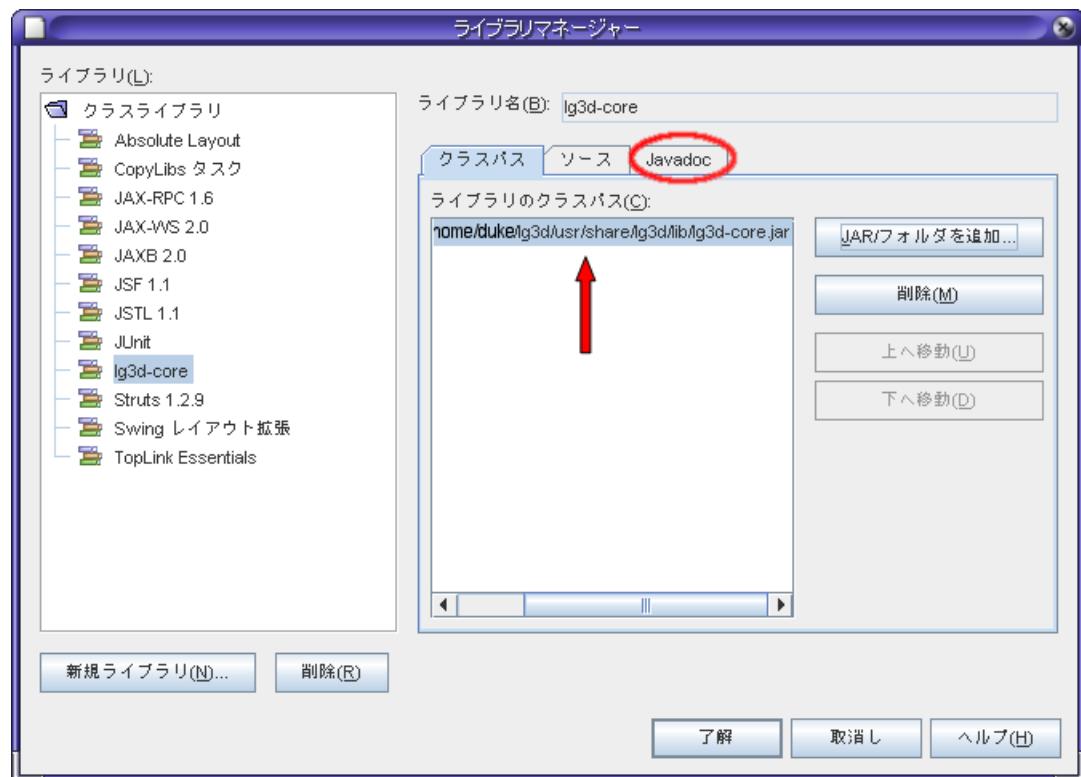
4. ライブラリマネージャダイアログに「lg3d-core」があることを確認し、それを選択します。右側が「**クラスライブラリ**」となっていることを確認し、「**JAR/フォルダの追加**」を選択します。



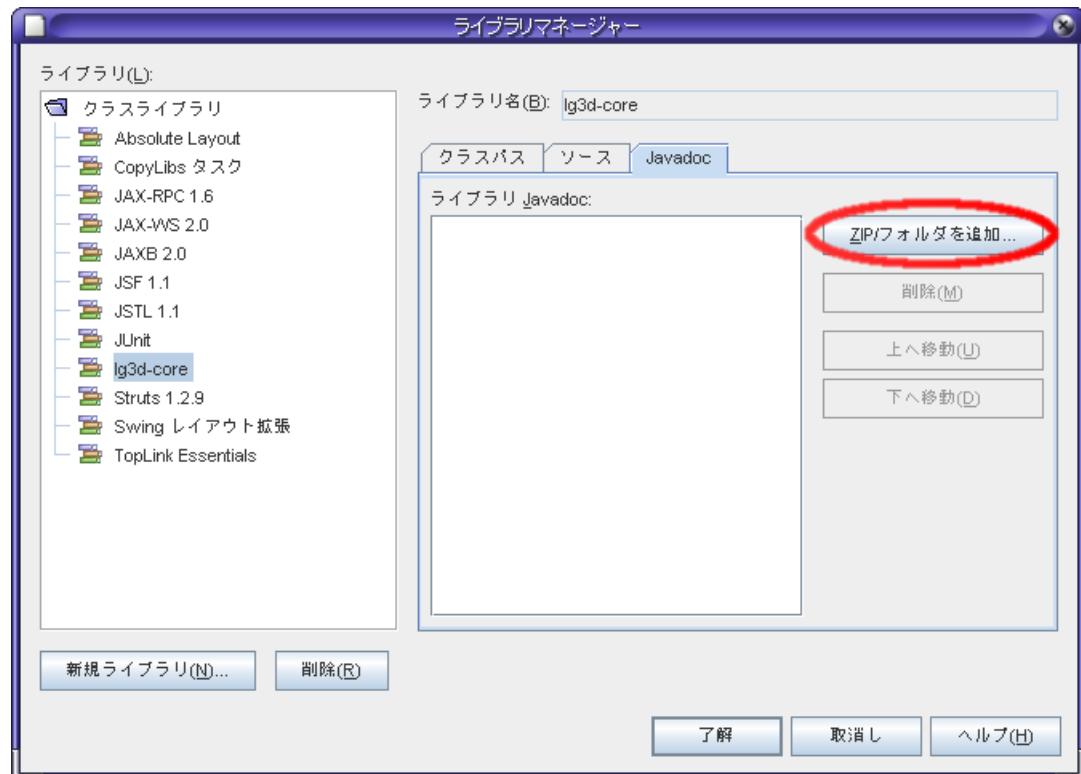
5. ファイラが表示されるので、LG3Dをインストールしたディレクトリ  
(/home/duke/lg3d/usr/share/lg3d/lib/)に移動します。  
**lg3d-core.jar**を選択し、「**JAR/フォルダの追加**」をクリックします。



6. ライブラリマネージャに JAR ファイルが追加されたことを確認します。  
確認後、「**Javadoc**」を選択して、クラスパスの表示から Javadoc の表示に切替えます。

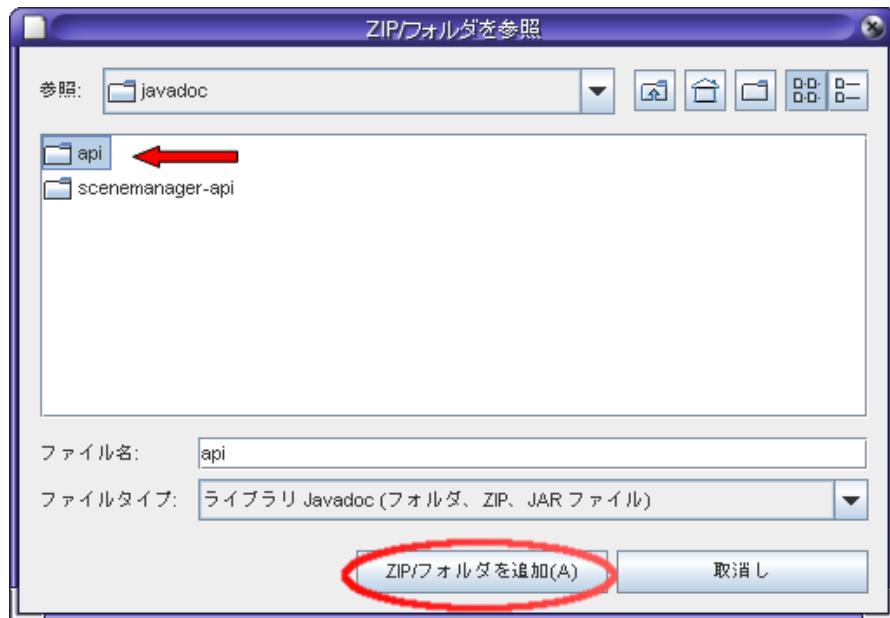


7. 「ZIP/フォルダを追加 する」をクリックします。



8. LG3D の API ドキュメント(</home/duke/java-docs/api>) を選択し、「ZIP/フォルダを追加」をクリックします。

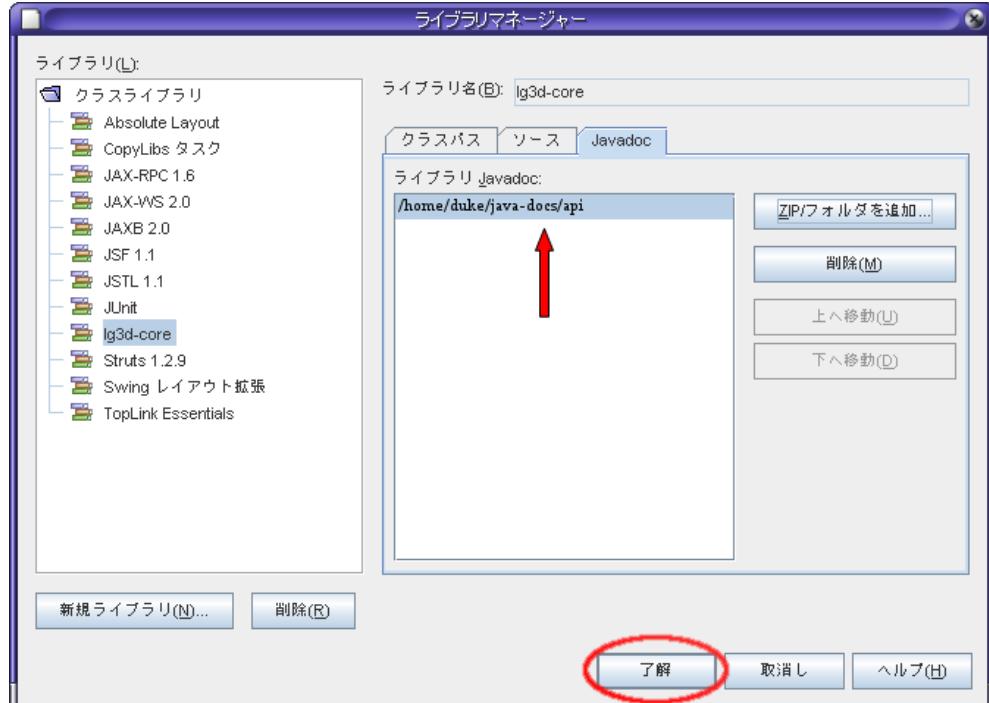
Javadoc の場所については「LG3D のインストール ( Solaris 10 x86 編 )」の後半にある「[Javadoc のインストール](#)」を参照してください。



9. Javadoc にフォルダ(ディレクトリ)が登録されたことを確認します。確認後、「了解」をクリックします。

**補足:**

[/home/duke/java-docs/scenemanager-api](#) に含まれる内容は LG3D のシステムレベルの開発 (シーンマネージャの開発) で用います。  
通常の LG3D アプリケーションの開発では 使わないので [登録する必要はありません](#)。



# 4 NetBeans による LG3D プログラミング (実践編)

## 4.1 プロジェクトの作成

この章で行うこと :

NetBeans を用いて LG3D アプリケーションを作成するために 以下のことを行います。

- LG3D アプリケーションを作成するためのプロジェクトの作成
- プログラミング環境のセットアップ
  - Java プラットフォームの設定
  - ライブラリの設定

### プロジェクトの作成

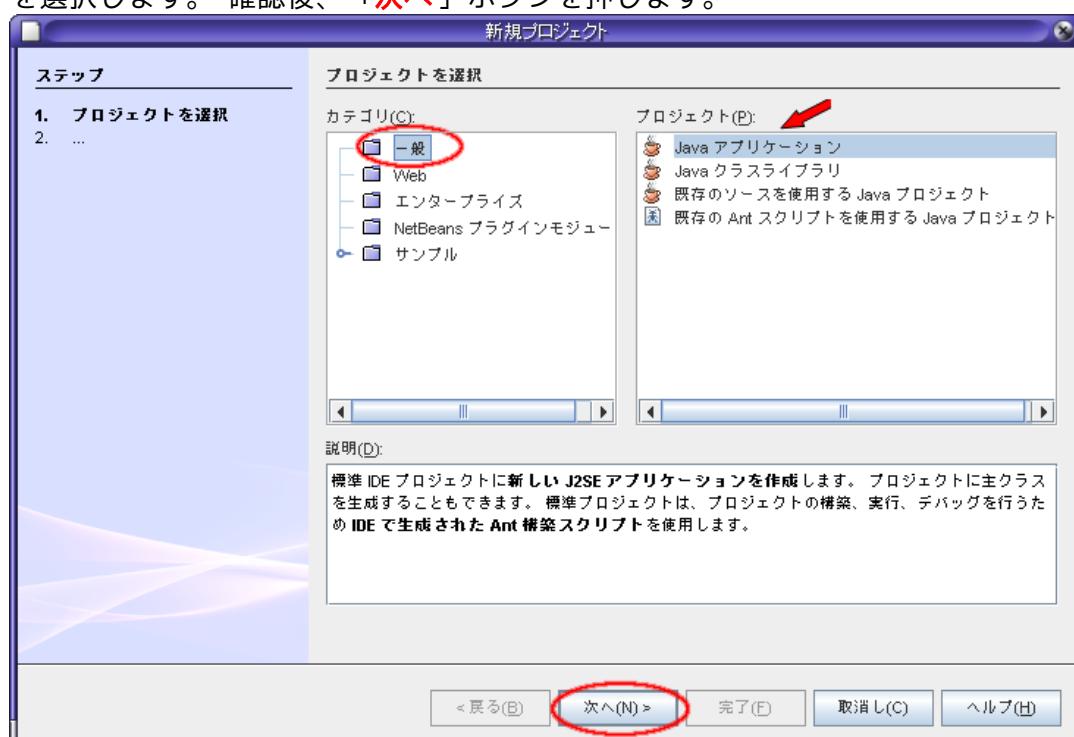
1. NetBeans 5.5 のメニューから、「**ファイル**」→「**新規プロジェクト**」を選択します。



2. プロジェクト作成用のダイアログが表示されますので、

- カテゴリ : **一般**
- プロジェクト : **Java アプリケーション**

を選択します。確認後、「次へ」ボタンを押します。



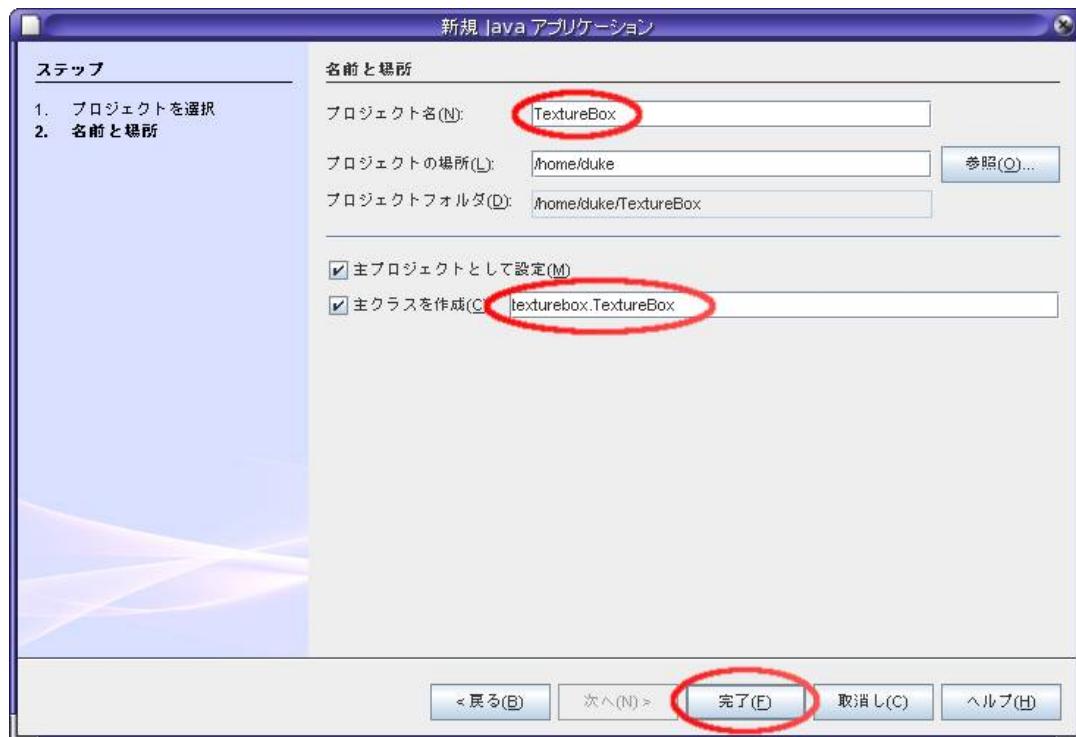
3. プロジェクトの情報を入力を求められますので、

- ・ プロジェクト名 : **TextureBox**
- ・ 主クラスを作成 : **texturebox.TextureBox**

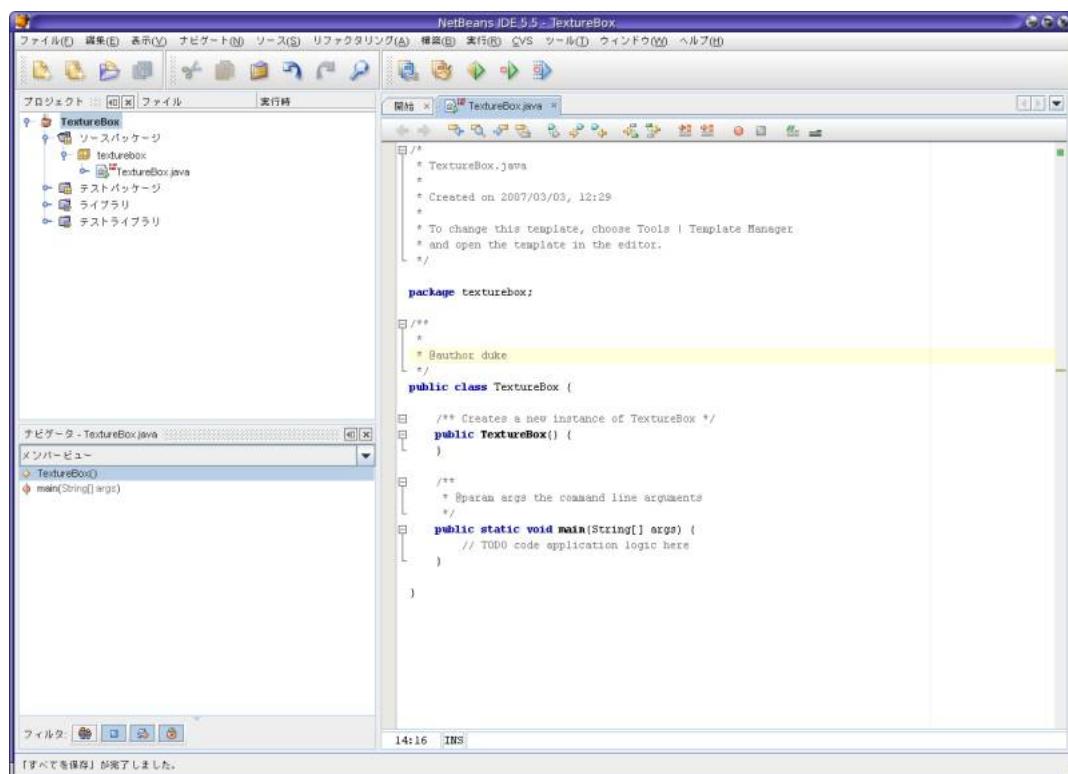
に変更します。

パッケージ名はデフォルトの `texturebox` をそのまま利用します。

入力が終わったら、「完了」を押します。

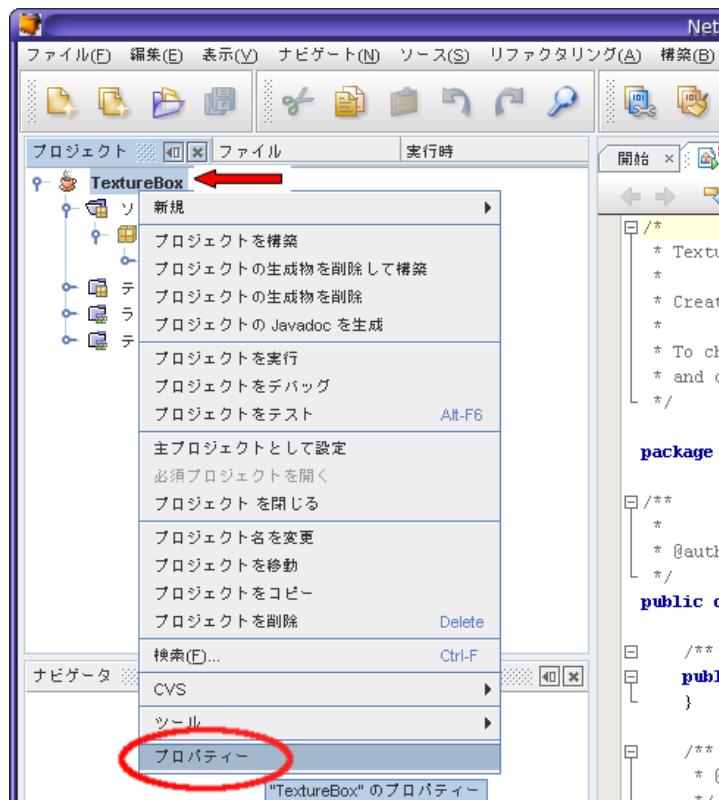


4. クラスなどを含むプロジェクトが作成され、下図のようになります。



## プログラミング環境、実行環境の設定

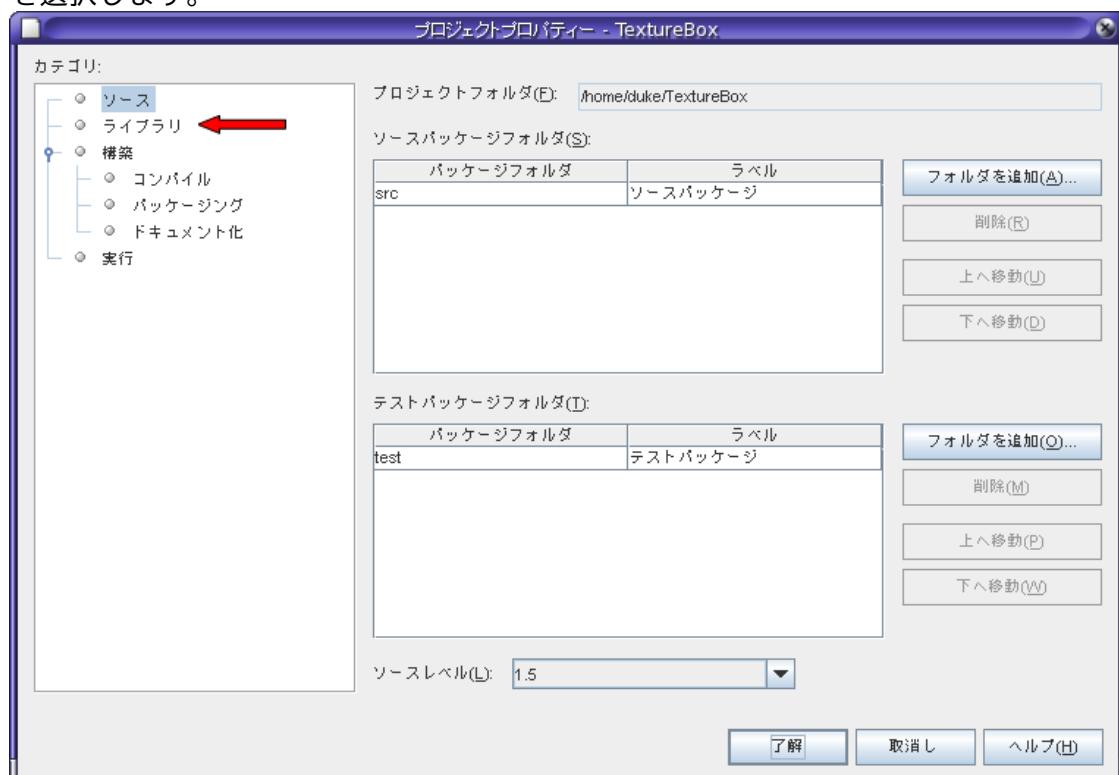
- IDE の左上部にプロジェクトの情報が表示されていることを確認し 「**TextureBox**」を選択します。選択後、右クリックを押すことでメニューが表示されますので 「**プロパティ**」を選択します。



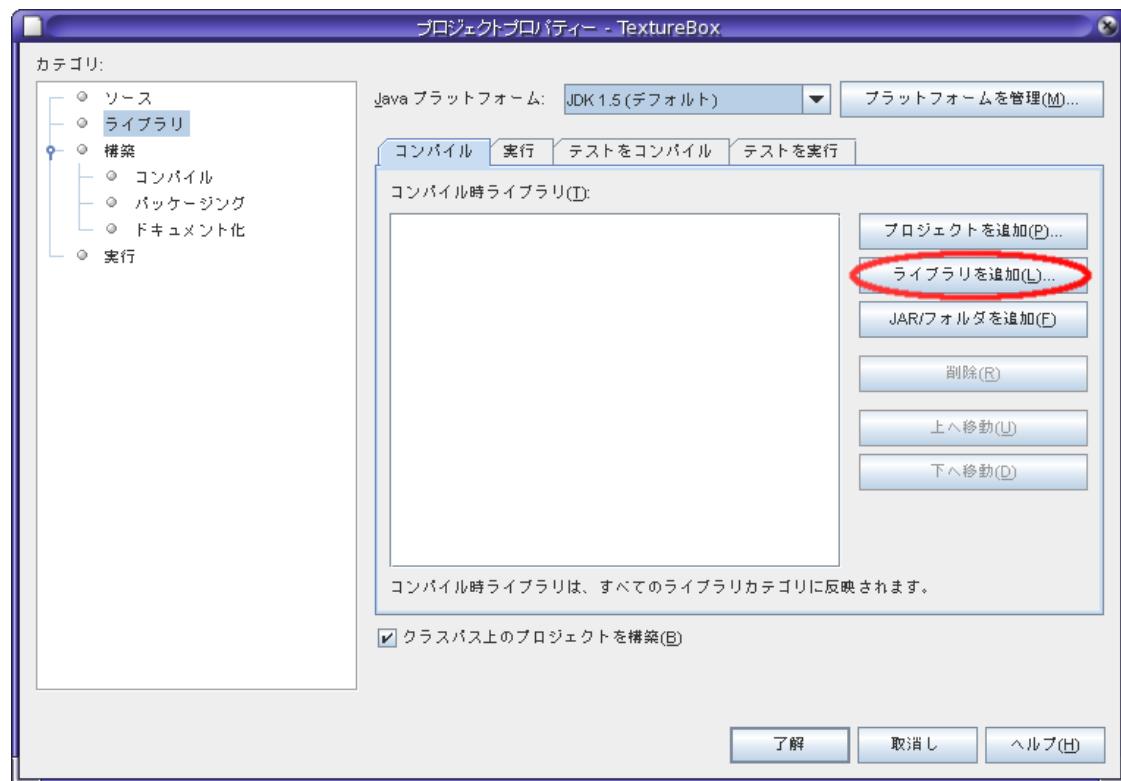
- プロジェクトプロパティの設定用ウィンドウが表示されますので、

- ・ カテゴリ : **ライブラリ**

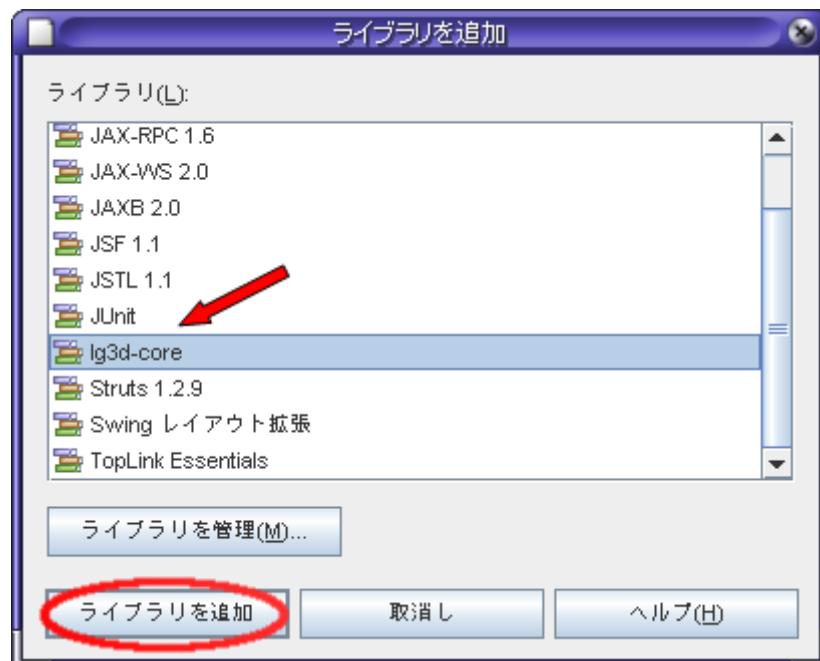
を選択します。



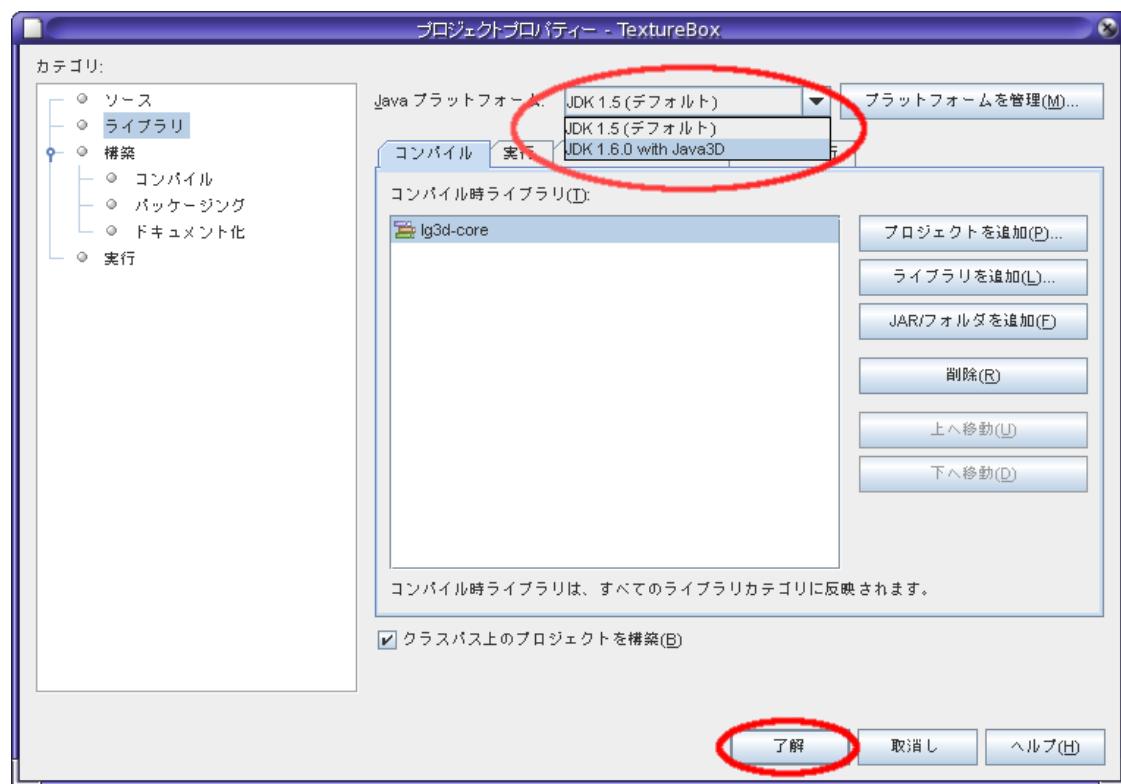
3. ウィンドウの右側の「コンパイル」が選択されていることを確認し、「ライブラリの追加」をクリックします。



4. 「ライブラリの追加」という新しいウィンドウが表示されますので、「lg3d-core」というライブラリを選択し、「ライブラリの追加」をクリックします。



5. プロジェクトプロパティの設定用ウィンドウの「コンパイル時ライブラリ」のリストに  
「**lg3d-core**」が追加されていることを確認します。  
確認後、上部の「Java プラットフォーム」の項目を  
• Java プラットフォーム : **JDK 1.6.0 with Java3D**  
に変更します。  
そして、下側にある「**了解**」をクリックします。  
これで設定は終了です。



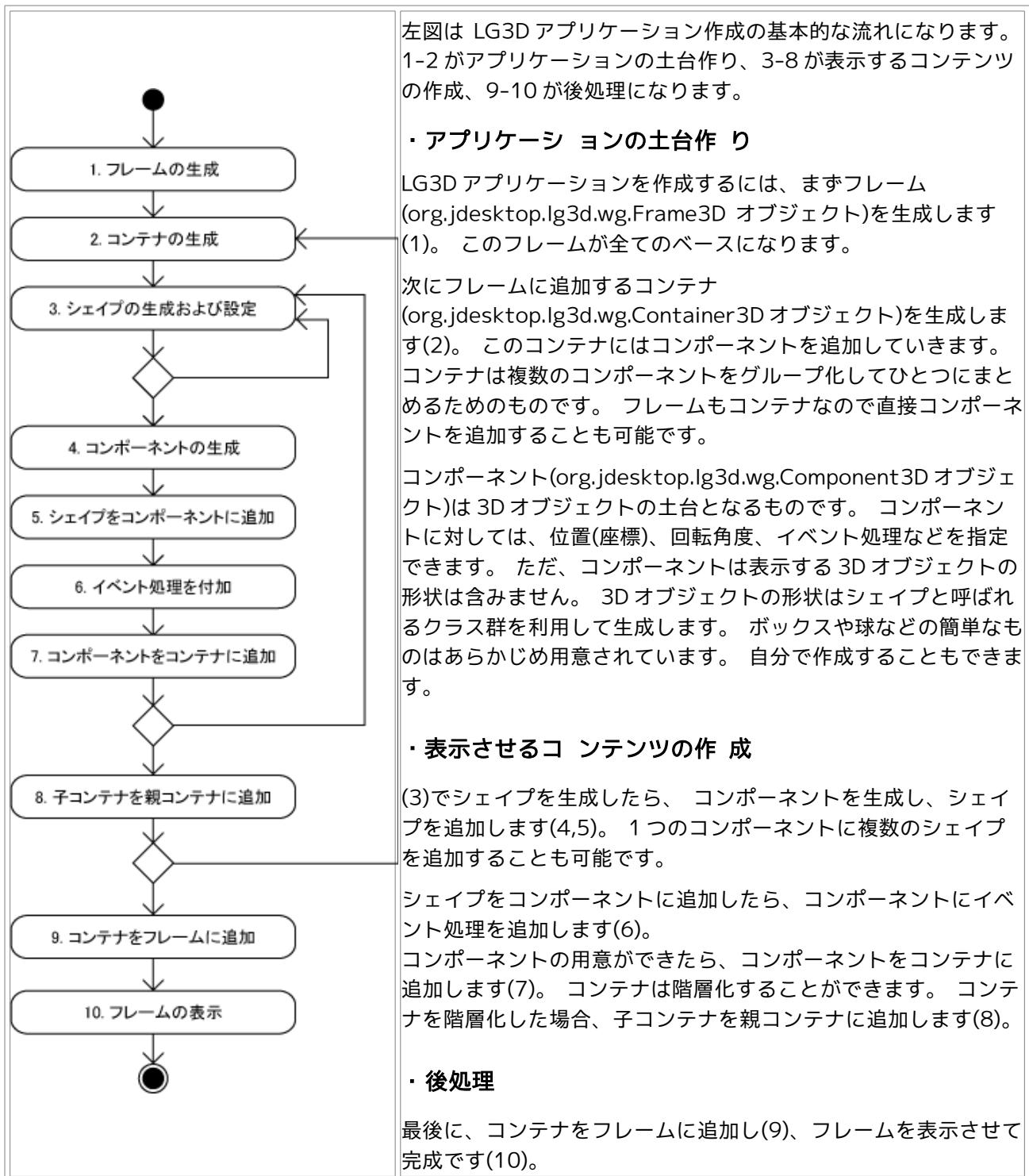
## 4.2 シンプルな 3D オブジェクトの作成

### この章で行うこと：

NetBeans を用いて以下のことを行います。

- ・簡単な LG3D アプリケーションの作成
- ・コンパイルおよび実行

### LG3D アプリケーションの作成の流れについて



## シンプルな 3D オブジェクトの作成

この章では LG3D アプリケーション作成の手始めとしてボックスを作成します。LG3D にあらかじめ用意してあるシェイプである org.jdesktop.lg3d.utils.shape.Box クラスを 利用します。このボックスを作成する時、形状を示すために x, y ,z 座標 (すべて float) と アピアランスと呼ばれるものが必要になります。作成されるボックスは (0,0,0)を中心とし、各辺の長さが 2x, 2y, 2z となります。また、アピアランスは作成する 3D アプリケーション(ここではボックス)の 表面の色や光の反射率を表すものです。

ここでは簡単なアピアランスを作成することができます

org.jdesktop.lg3d.utils.shape.SimpleAppearance クラスを用いて、 不透明な薄い青色を示すアピアランスを作成しています。ボックスのシェイプはコンポーネントに追加し、コンポーネントをフレームに追加しています。（今回コンテナは作成せず、フレームに直接コンポーネントを追加しています。）

1. 右側のパネルの **TextureBox.java** 下のリストのように書き換えます。

赤(太字)の部分が元のプログラムから変更・追加したところです。

```
/*
 * TextureBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package texturebox;

/**
 *
 * @author duke
 */

import javax.vecmath.Vector3f;
import org.jdesktop.lg3d.sg.Appearance;
import org.jdesktop.lg3d.utils.shape.Box;
import org.jdesktop.lg3d.utils.shape.SimpleAppearance;
import org.jdesktop.lg3d.wg.Component3D;
import org.jdesktop.lg3d.wg.Frame3D;

public class TextureBox {

    /** Creates a new instance of TextureBox */
    public TextureBox() {

        // フレームの生成
    }
}
```

```
Frame3D frame = new Frame3D();

// シェイプの生成 および設定

// 色を指定した Appearance の設定
// 引数は R,G,B,アルファ (透過度) で、0-1 までの float 値
Appearance appearance
    = new SimpleAppearance(0.6f, 0.6f, 1.0f, 1.0f);

// Box の生成
// 引数は、x、y、z 座標、アピアラ ンス
// 座標の単位はメートル (float 値)
// 作成されるのは 横 0.20f, 縦 0.16f, 奥行き 0.12f のボックス
Box box = new Box(0.10f, 0.08f, 0.06f, appearance);

// コンポーネント の生成
Component3D component = new Component3D();

// シェイプをコンポーネントに追加
component.addChild(box);

// コンポーネント をコンテナに追加
// ここではコンテナを使用してないので、
// 直接フレームに追加
frame.addChild(component);

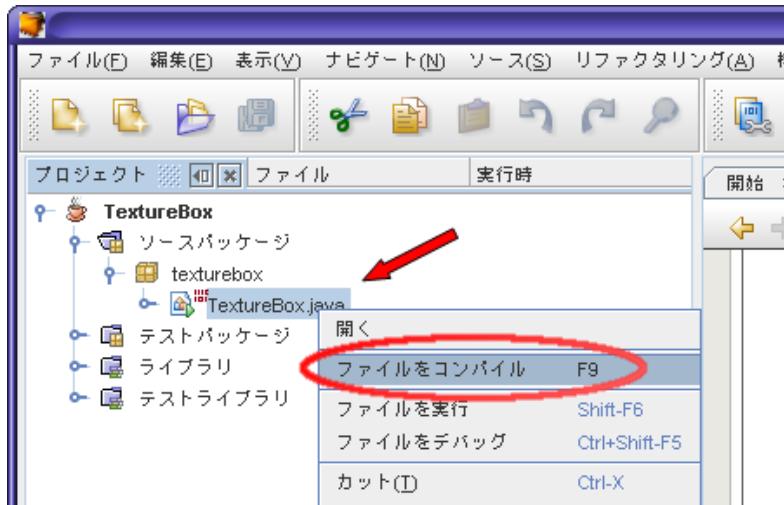
// フレームの表示
// フレームの大きさを設定
// 引数は、Java3D のクラス
// javax.vecmath.Vector3f(float x,float y,float z)
// x = 幅、y = 高さ , z = 奥行き (単位はメートル)
frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));

// フレームの表示
frame.changeEnabled(true);
frame.changeVisible(true);

}

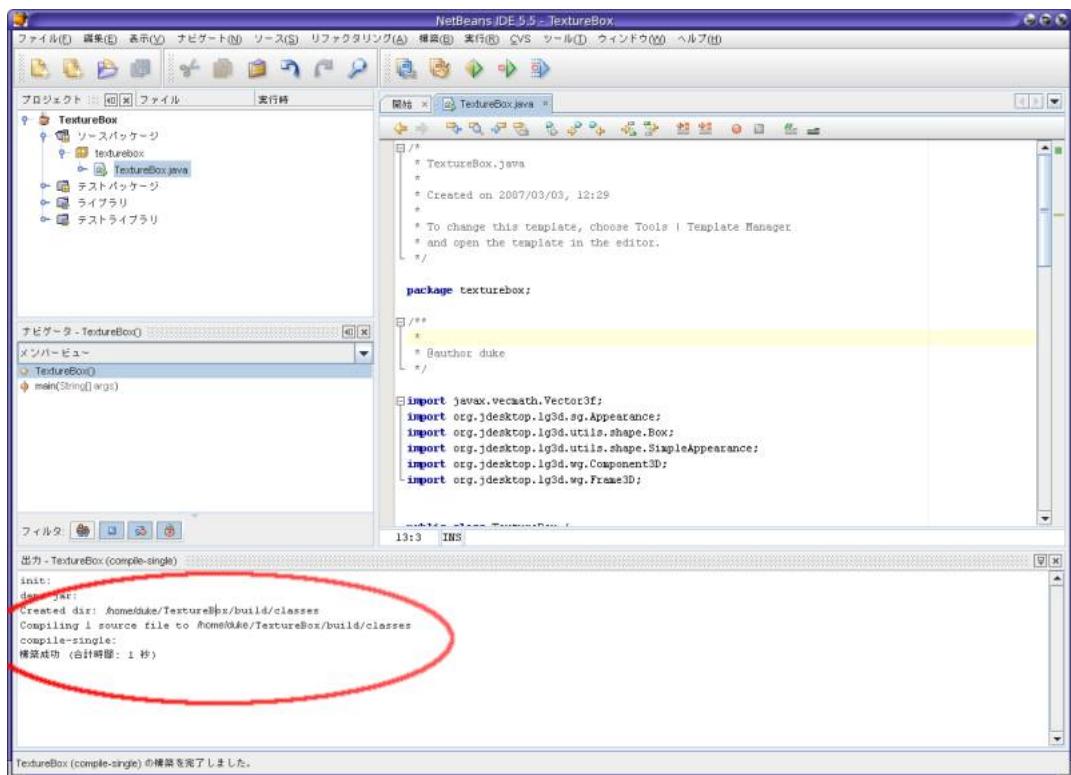
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    new TextureBox();
}
}
```

2. プログラムの記述が終わったら、左側のパネルの **TextBox.java** を選択し、マウスの右クリックをします。下図のようなメニューが表示されるので「**ファイルをコンパイル**」を選択します。



2. コンパイルが成功すると「構築成功」というメッセージが表示されます（下図の最下部を参照）。

コンパイルに失敗した場合、この部分に **エラーとその内容が表示されます**。  
その場合は、プログラムを修正し、再度コンパイル作業を行ってください。



## 4.3 Jarベースアプリケーションの作成と実行

この節では、Jarベースアプリケーションの作成と実行について説明します。

作成したアプリケーションを実行するためには、Jarパッケージを作成し、

\$LG3DHOME/usr/share/lg3d/ext/app/に配置して実行する必要があります。

LG3D 1.0 用の Jar ファイルを作るには、以下のファイルを用意します。

- **Manifest File** : JAR ファイルの内容(バージョン、メインクラスなど)について記述するためのファイル
- **.lgcfg File** : アプリケーションを LG3D のメインメニューのどこに配置するかを記述するためのファイル
- **Icon-File** : タスクバー、あるいはメニューに表示するアイコンファイル

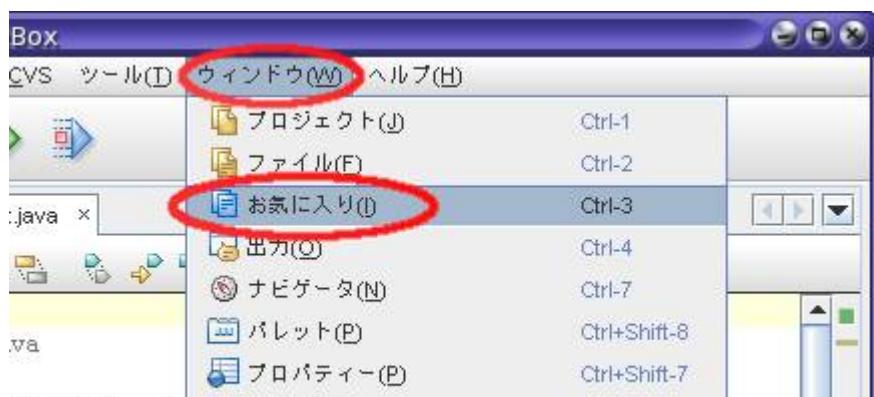
### アイコンファイルのプロジェクトへの取り込み

アプリケーションのアイコンファイル(画像ファイル)はプロジェクトには含まれていないため、外部から取り込む必要があります。

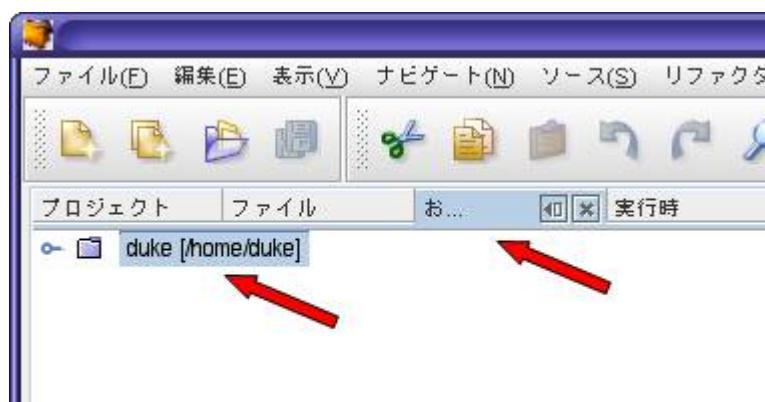
以下の手順で外部ディレクトリ /home/duke/Documents を読み込めるように設定し、画像ファイルをプロジェクトに取り込みます。ハンズオンで利用した画像ファイル等は [all\\_contents.zip](https://lg3d.dev.java.net/ja/lg3d_hol/html/all_contents.zip) ([https://lg3d.dev.java.net/ja/lg3d\\_hol/html/all\\_contents.zip](https://lg3d.dev.java.net/ja/lg3d_hol/html/all_contents.zip)) にまとめています。

/home/duke/Documents に解凍してください。

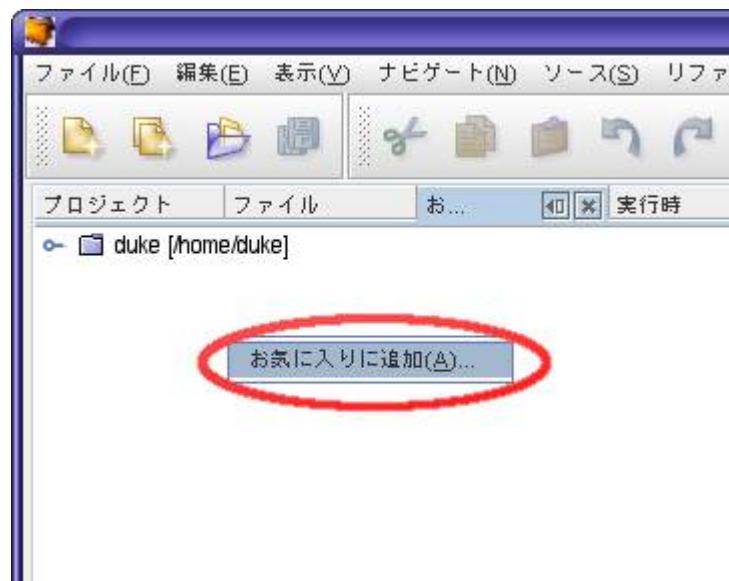
1. メニューから「ウィンドウ」→「お気に入り」を選択します。



2. IDE の左側のパネルに「お気に入り」が追加されていることを確認します。



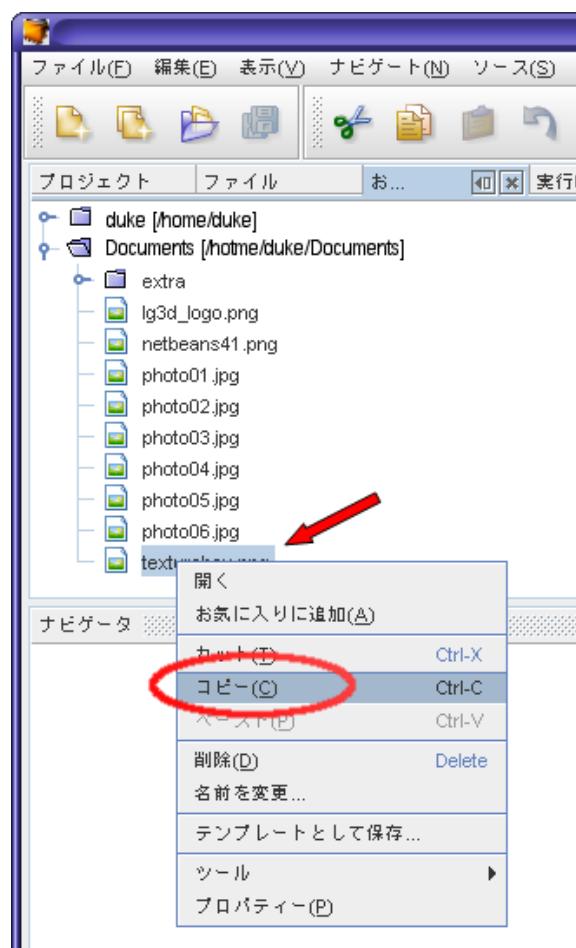
3. 「お気に入り」を右クリックするとメニューが表示されるで「お気に入りを追加」を選択します。



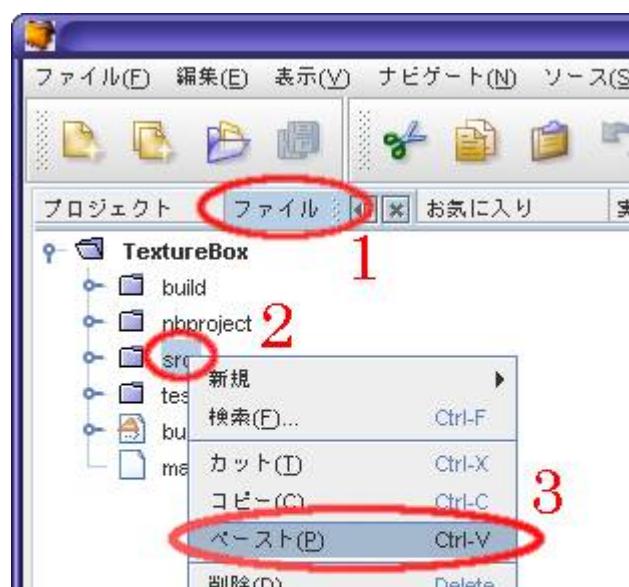
4. ファイル選択ダイアログが表示されるので「Documents」を選択し、「追加」をクリックします。



5. 左側のパネルに「Documents」 というフォルダ(ディレクトリ)が追加されます。  
その中を開くと **texturebox.png** というファイルが表示されますので、これを選択します。選択後、右クリックするとメニューが表示されますので「**コピー**」を選択します。



6. 左上のタブから「**ファイル**」を選択します。  
プロジェクト作成時に生成されたファイル、フォルダ(ディレクトリ)の一覧が表示されます。この中から「**src**」(ソースファイルのフォルダ(ディレクトリ))を選択し、右クリックします。メニューが表示されますので「**ペースト**」を選択します。



7. 下図のように画像ファイルが追加されたことを確認します。

これでプロジェクトへのファイルの追加は完成です。

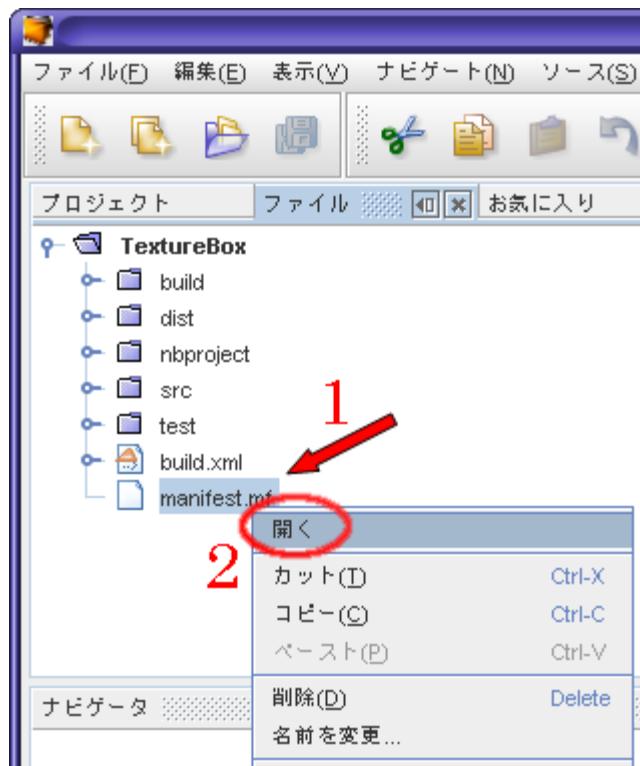
プロジェクトを構築する際に、これらのファイルが自動的に JAR アプリケーションに追加されます。



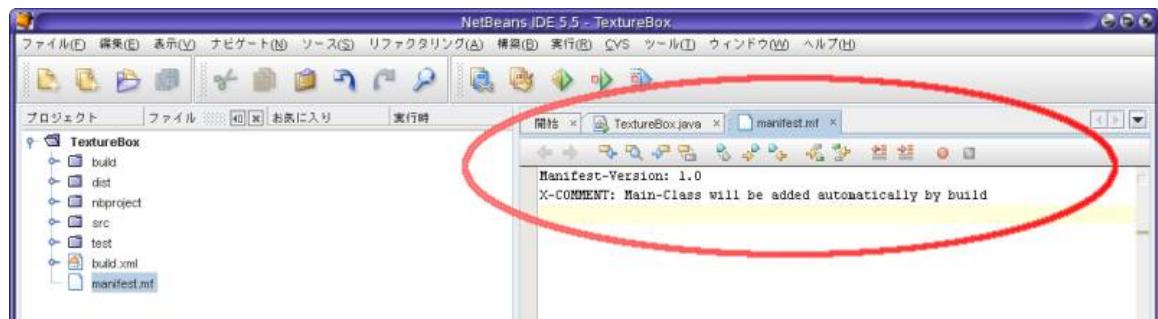
## JAR ベースのア プリケ ショ ン の作成

1. 左側のパネルの最下部にある「**manifest.mf**」を選択します。

右クリックを押すとメニューが出ますので「開く」を選択します。



2. 右側のソースコードエディタに「**manifest.mf**」が表示され、編集可能な状態になります。



3. manifest.mf を編集し、

- **Config-File (.lgcfg ファイル)**

を追加します。

追加部分は**赤(太字)**で示します。

Manifest-Version: 1.0

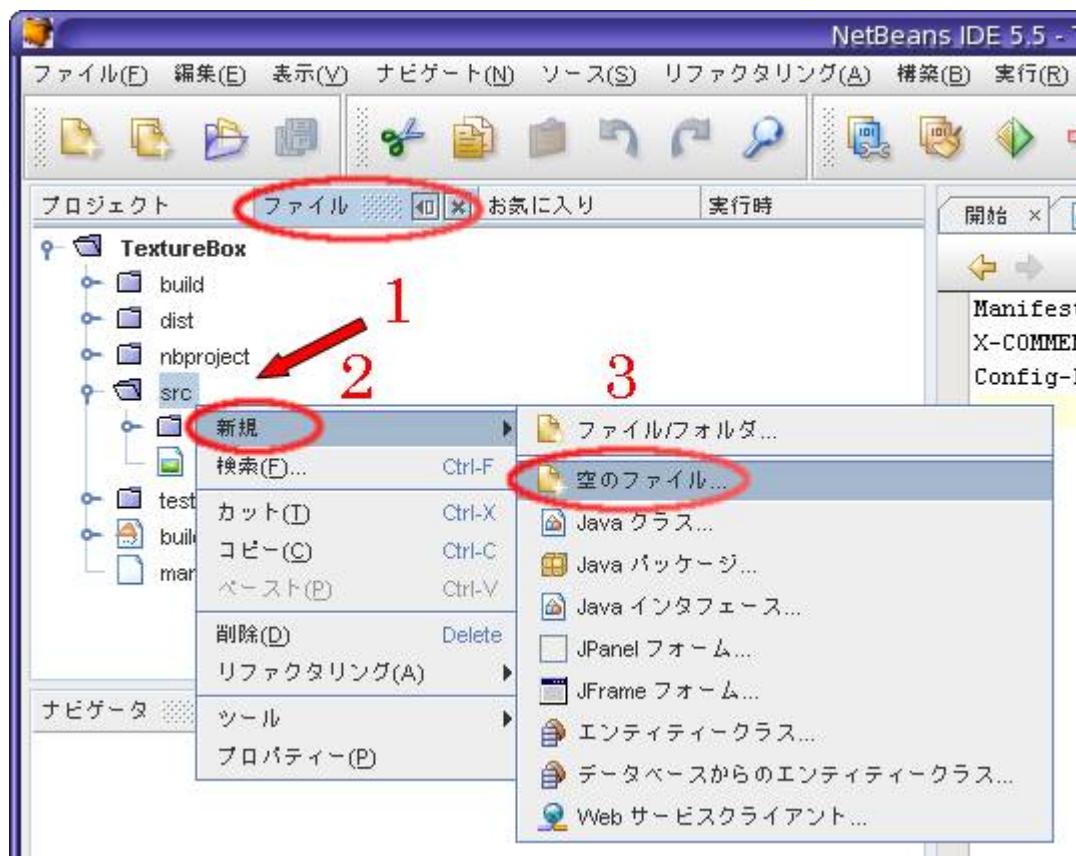
X-COMMENT: Main-Class will be added automatically by build

**Config-File: TextureBox.lgcfg ← アプリケーションと同じ名前にしています。**

**注**：最後に空行を入れます。

4. .lgcfg ファイルを追加

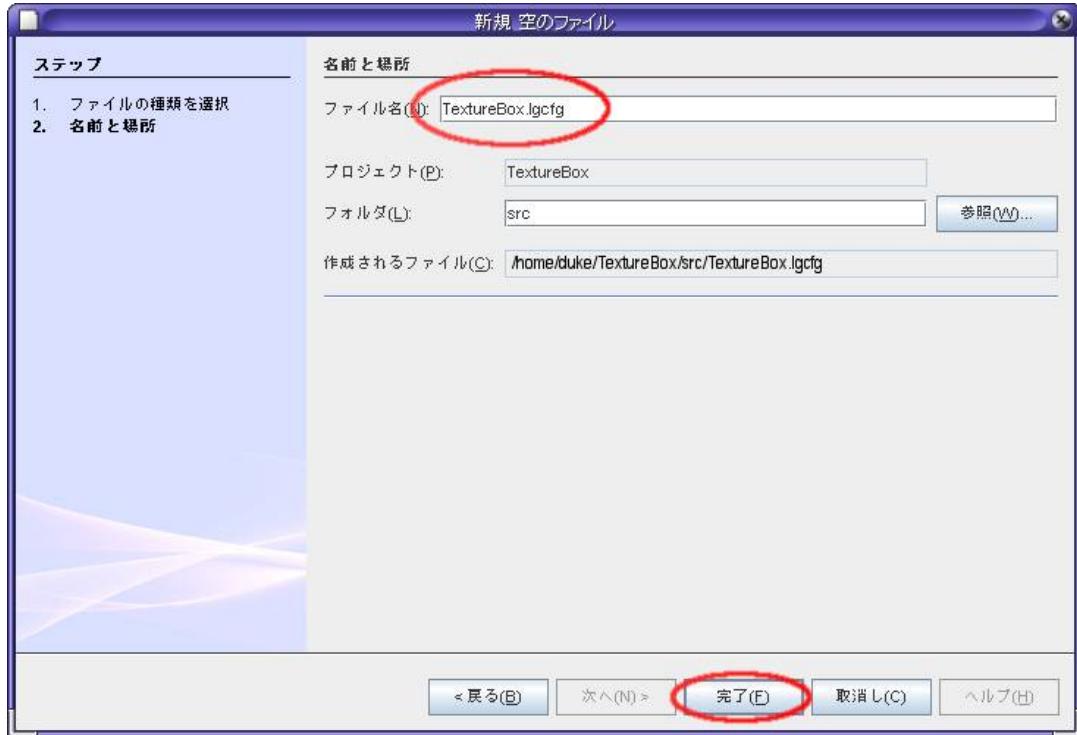
「ファイル」タブの「src」を右クリックして、メニューが表示されます  
そして、「新規」→「空のファイル」を選びます。



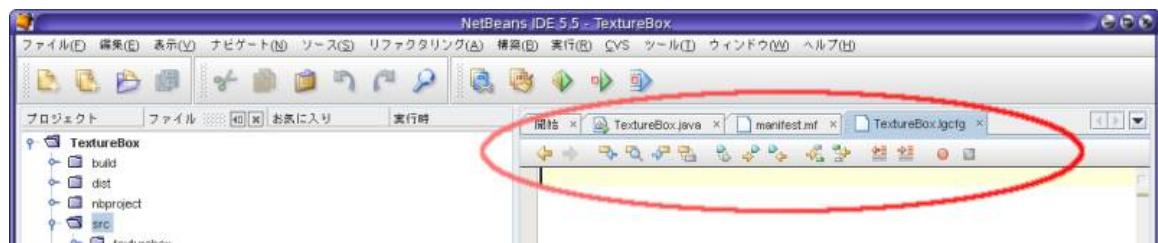
## 5. .lgcfg ファイル名の入力

「新規 空のファイル」というタイトルのウィンドウが表示されます。

「ファイル名」に「**TextBox.lgcfg**」を入力して、「完了」をクリックします。



## 6. 右側のソースコードエディタに「**TextBox.lgcfg**」が表示され、編集可能な状態になります。



## 7. .lgcfg ファイルの編集

アプリケーションをタスクバーに登録する場合、以下の内容（赤（太字））を TextBox.lgcfg に追加して保存します。

メニューにプログラムを登録する場合、このページの最後にある [「アプリケーションをメインメニューに登録するには」](#) を参照してください。

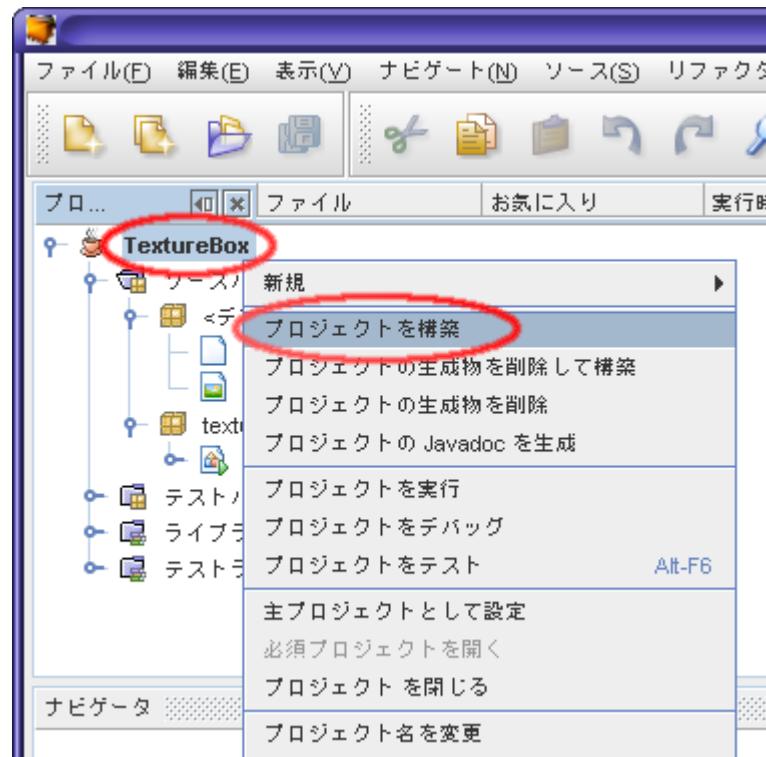
※Windows の場合、タスクバーへの登録はうまく動作しないことがあります。  
その場合は「[アプリケーションをメインメニューに登録するには](#)」の項目を行ってください。

```

<?xml version="1.0" encoding="UTF-8"?>
<java version="1.5.0" class="java.beans.XMLDecoder">
<object
class="org.jdesktop.swingx.scenemanager.config.ApplicationDescription">
  <void property="exec">
    <string>java texturebox.TextureBox</string>
      ↑ 実行コマンド(メインクラスを 指定)
  </void>
  <void property="iconFilename">
    <string>texturebox.png</string>
      ↑ アプリケーションのアイコン ファイル名
  </void>
  <void property="name">
    <string>TextureBox</string> ← アプリケーションの名 前
  </void>
  <void property="classpathJars">
    <string>TextureBox.jar</string>
      ↑ アプリケーションが入っている Jar ファイルの名前
  </void>
</object>
</java>

```

8. 左側のパネルのタブから「プロジェクト」を選択し、「プロジェクトの構築」を行います。



JARベースのアプリケーションの作成はこれで終わりです。  
アプリケーションは プロジェクトのフォルダ(ディレクトリ)内の「**dist**」に置かれます。

今回の場合は **/home/duke/TextureBox/dist/texturebox.jar** が作成したアプリケーションです。

## JARベースのアプリケーションの実行

作成したJARベースアプリケーションをLG3Dに登録します。

LG3D本体のフォルダ(ディレクトリ)にある ext/app フォルダ(ディレクトリ)が 作成したアプリケーションの登録フォルダになります。

Linuxの場合 **/home/duke/lg3d/usr/share/lg3d/ext/app** になります。

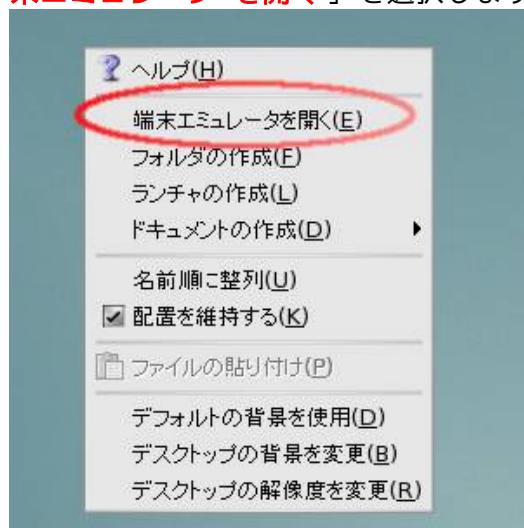
Windowsの場合 **C:\lg3d\ext\app** になります。

ここに作成したJARベースアプリケーションをコピーすれば登録完了です。

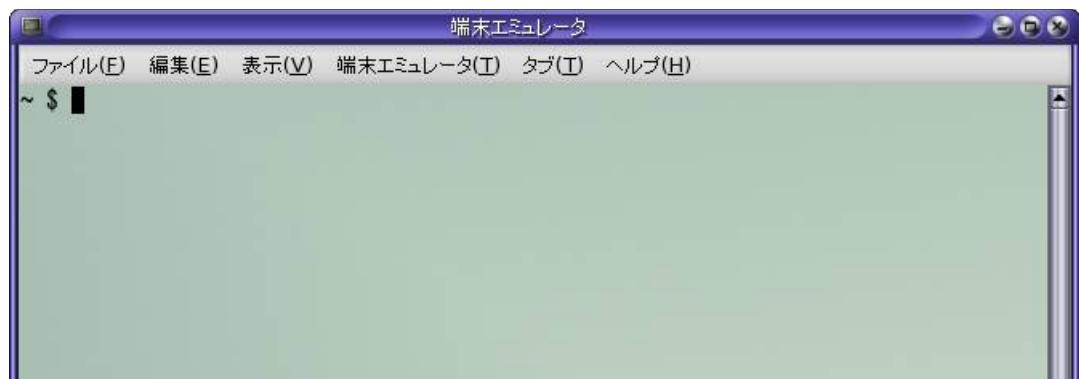
アプリケーションの登録がうまくいけば、LG3Dのタスクバーにアプリケーションの起動アイコンが追加されます。

以下に、作成したアプリケーションを登録して、実行する方法を示します。

1. Solaris 10のJDS環境では背景で右クリックするとメニューが表示されますので、「**端末エミュレータを開く**」を選択します。



2. 次のような端末が表示されますので、その上で作業を行います。



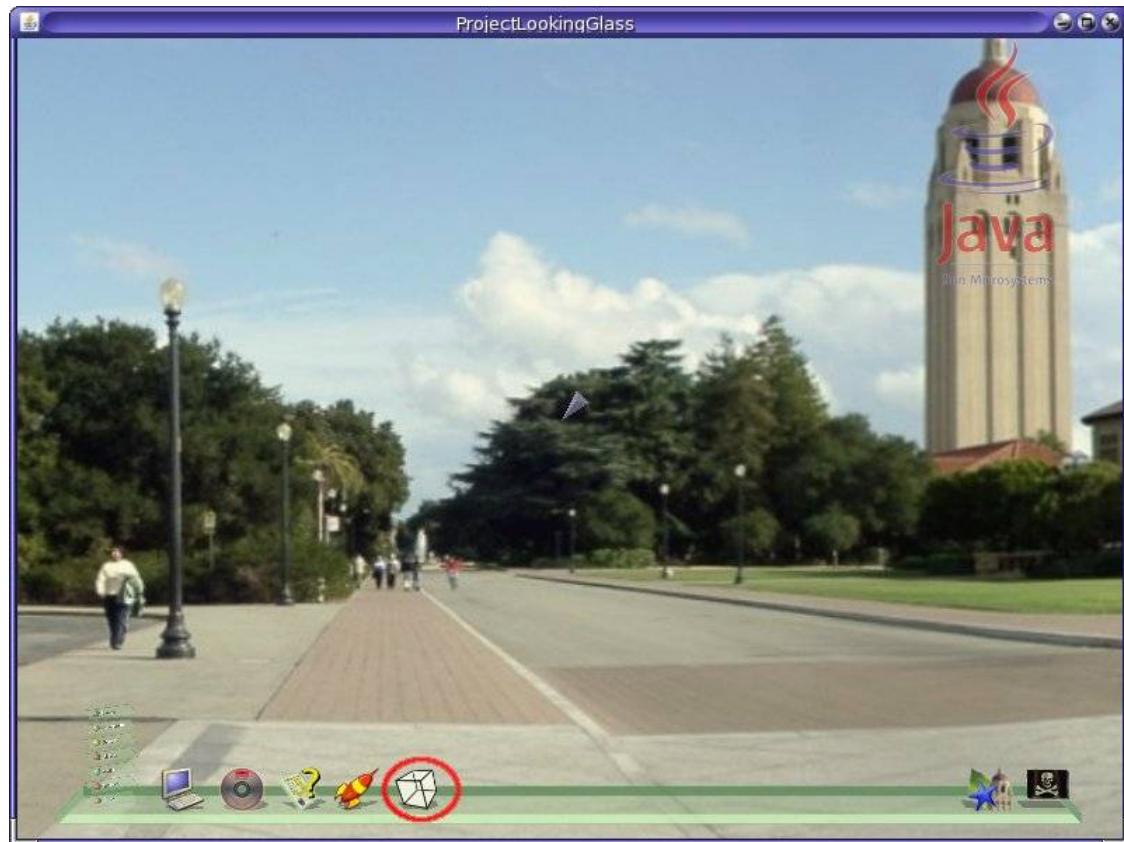
3. 次のコマンドを実行して、JARベースアプリケーションを LG3Dのディレクトリにコピーします。（#を入力する必要はありません。）

```
# cp ~/TextureBox/dist/TextureBox.jar ~/lg3d/usr/share/lg3d/ext/app/
```

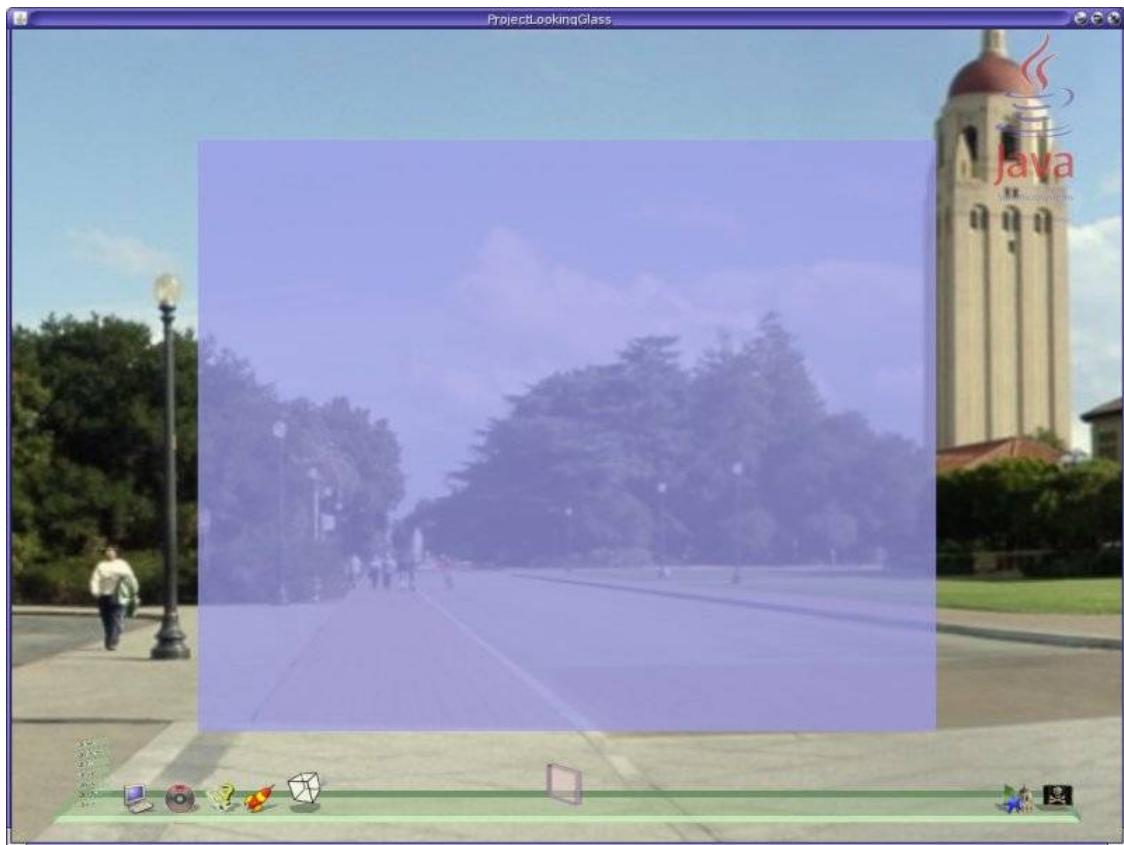
4. 次のコマンドを実行し、LG3Dを起動します。

```
# cd  
# lg3d/bin/lg3d-dev
```

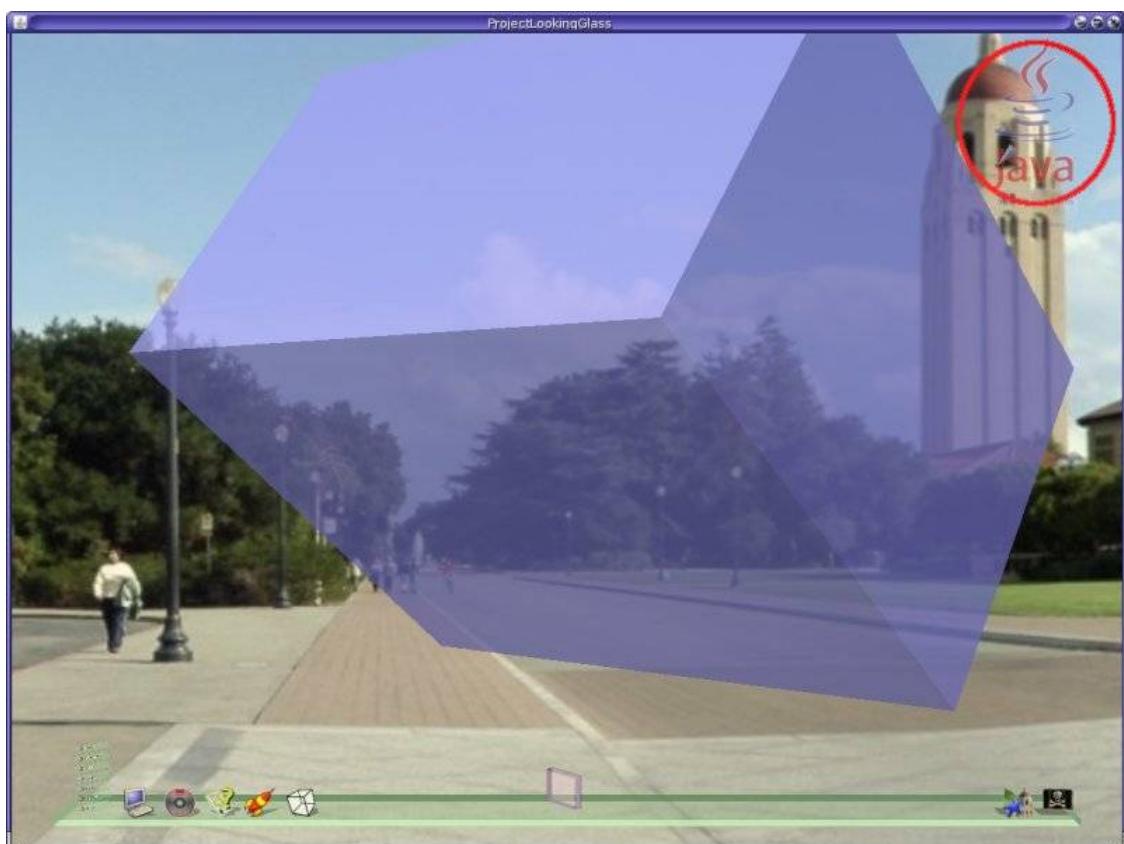
5. 下図のように JARベースアプリケーションのアイコンが表示されます。クリックして実行します。



6. 下図はアイコンをクリックしてアプリケーションを起動した後の画面です。



7. LG3Dにボックスが表示されたら、ウィンドウの右上にある Java ロゴをドラッグ (Java ロゴをクリックしたままマウスを動かす) してみてください。下図のようにボックスを動かすことができます。ドラッグをやめるとボックスは元に戻ります。



## アプリケーションをメインメニューに登録するには

作成した JAR ベースのアプリケーションを LG3D のメインメニューに登録するには、.lgcfg ファイルを修正する必要があります。「メインメニュー」の「Demos」サブメニューに登録するための .lgcfg ファイルは以下のように修正します。

- .lgcfg ファイルの編集

以下の内容（赤（太字））を TextBox.lgcfg に追加して保存します。

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.5.0" class="java.beans.XMLDecoder">

<!-- Taskbar item -->
<!-- ←タスクバー登録用の記述をコメントアウト
<object class="org.jdesktop.swingx.JXTaskBar$TaskItem">
  <void property="label">
    <string>TextureBox</string>
  </void>
  <void property="iconImage">
    <string>texturebox.png</string>
  </void>
  <void property="name">
    <string>TextureBox</string>
  </void>
  <void property="classpathJars">
    <string>TextureBox.jar</string>
  </void>
</object>
--> ←タスクバー登録用の記述をコメントアウト

<!-- Start Menu item -->
<object class="org.jdesktop.swingx.JXTaskBar$TaskItem">
  <void property="command">
    <string>java texturebox.TextureBox</string> ←実行コマンド(メインクラスを指定)
  </void>
  <void property="displayResourceType">
    <string>ICON</string> ←表示用アイテムのタイプにアイコンを指定
  </void>
  <void property="displayResourceUrlName">
    <string>resource:///texturebox.png</string> ←アプリケーションのアイコンファイル名
  </void>
  <void property="menuGroup">
    <string>Demos</string> ←「Demos」サブメニューに登録
  </void>
  <void property="name">
    <string>TextureBox</string> ←アプリケーションの名前
  </void>
```

```
<void property="classpathJars">
<string>TextureBox.jar</string> ← アプリケーションが入っている Jar ファイル の名前
</void>
</object>

</java>
```

- 正しく登録されると、「Demos」サブメニューはこのように変わります。



## 4.4 テクスチャを使う

### この章で行うこと：

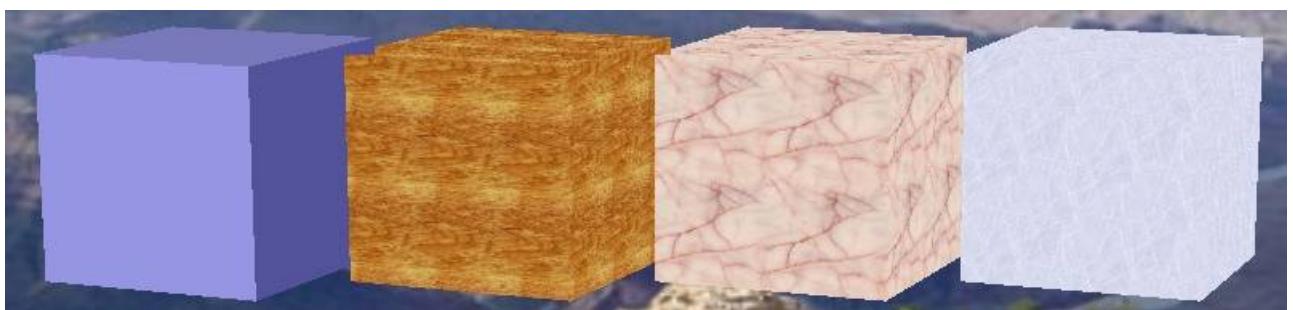
ここまでで作成したボックスを拡張します。

以下のようなボックスを作成してみます。

- ・ 同じ画像をすべての面に貼り付けたボックス
- ・ すべての面に異なる画像を貼り付けたボックス

### テクスチャとは？

3Dコンピュータグラフィックスにおいて、3Dオブジェクトの表面に貼り付ける画像のことをテクスチャと呼びます。テクスチャを利用することにより3Dオブジェクトの質感などを簡単に表現することができます。下図は前章で作成したボックスにテクスチャを貼ったした例です。左から順に、テクスチャ無し、木のテクスチャ、ピンクの大理石のテクスチャ、氷のテクスチャです。



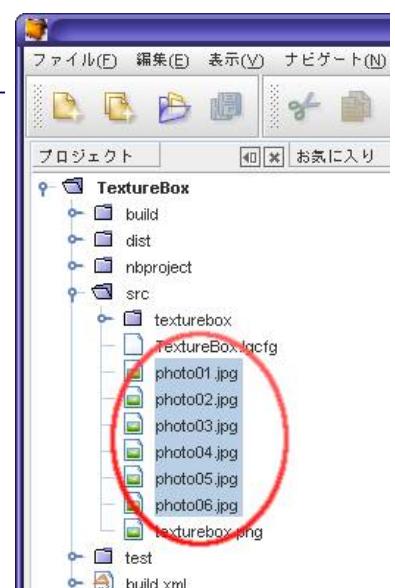
### 画像ファイルをプロジェクトに取り込む

テクスチャを利用するためには、まず画像を用意します。

ここではあらかじめ用意された画像(photo01.jpg - photo06.jpg)を利用します。

画像は作成するJARアプリケーションとは別に用意しても構いませんが、簡単化のために作成するJARアプリケーション内部に取り込むことにします。

1. 「4.3 Jarベースのアプリケーションの作成と実行」にある[「アイコンファイルのプロジェクトへの取り込み」](#)を参考にして、画像(photo01.jpg - photo06.jpg)をプロジェクトのソースフォルダに取り込みます。  
(シフトキーを押しながらマウスで選択すると、複数選択が行えます)
2. 画像ファイルの取り込みが終わったら、右図のように画像ファイルが追加されたことを確認します。これでプロジェクトへのファイルの追加は完成です。  
プロジェクトを構築する際に、これらのファイルが自動的にJARアプリケーションに追加されます。



## テクスチャを使う(その1)

前章で作成したボックスにテクスチャを貼ってみます。まずは1つの画像をすべての面に表示させてみます。

テクスチャを用いるためにプログラムに以下の変更を行います。

- アピアランスの生成部分について

前章では3Dオブジェクトの表面の情報を指定するためにアピアランスを使用することを学びました。テクスチャは3Dオブジェクトの表面に貼り付ける画像ですので、その指定にも同じアピアランスを用います。

具体的には、org.jdesktop.lg3d.utils.shape.SimpleAppearanceクラスの引数の「R,G,B,アルファ(透過度)」の部分を「**テクスチャとして利用する画像ファイルのURL**」に変更します。(JPEG, GIF, PNGの3つ画像ファイル形式に対応しています。)

- 画像ファイルのURLについて

画像ファイルのURLは以下の形式で指定します。

- \* World Wide Web上のファイルの場合:  
**"http://www.foo.com/image.jpg"**
- ローカルディスク上のファイルの場合:  
**"file:///jarFileFullPath/imageFileName.jpg"**
- Jarファイルに取り込んだ画像を利用する場合:  
**"jar:http:jarFileFullPath/jarFileName.jar!/image.jpg"**  
(Web上にあるJarファイル)  
**"jar:file:jarFileFullPath/jarFileName.jar!/image.jpg"**  
(ローカルにあるJarファイル)

今回の場合、画像ファイルはすべてJarファイル内にありますので、URLは  
**jar:file:/home/duke/lg3d/usr/share/lg3d/ext/app/TextureBox.jar!/image.jpg**  
のようになります。しかし、上記のように**フルパスをプログラム内に記述すると開発者と異なる環境では正しく動作しません。**

その問題を解決するために、ここでは**ClassLoader**を利用し、URLを取得します。

以下のように**ClassLoader**を使って、Jarファイルにある画像ファイル、image.jpgのURLを取得します。

```
ClassLoader loader = this.getClass().getClassLoader();
url = loader.getResource("image.jpg");
```

- 例外処理について

アピアランス生成時に指定された画像ファイルの読み込みに失敗した場合にIOExceptionを発生します。

そのため、例外処理を加える必要があります。

- org.jdesktop.lg3d.utils.shape.Boxオブジェクトの生成部分について

ボックスにテクスチャを貼るためににはBox.GENERATE\_TEXTURE\_COORDSというフラグを立てる必要があります。

このフラグはBoxオブジェクトがテクスチャを貼り付けるときに必要となるテクスチャ座標の生成を行うというものです。

1. TextureBox.java を以下のように変更します。

変更箇所は **赤(太字)** で示します。

```
/*
 * TextureBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package texturebox;

/**
 *
 * @author duke
 */

import java.io.IOException;
import java.net.URL;
import java.lang.ClassLoader;
import javax.vecmath.Vector3f;
import org.jdesktop.lg3d.sg.Appearance;
import org.jdesktop.lg3d.utils.shape.Box;
import org.jdesktop.lg3d.utils.shape.SimpleAppearance;
import org.jdesktop.lg3d.wg.Component3D;
import org.jdesktop.lg3d.wg.Frame3D;

public class TextureBox {

    /** Creates a new instance of TextureBox */

    public TextureBox(){

        // フレームの生成
        Frame3D frame = new Frame3D();

        // シェイプの生成および設定
        // 画像を指定した Appearance の設定
        Appearance appearance = null;
        URL url = null;
        ClassLoader loader = this.getClass().getClassLoader();

        // 読み込みに失敗した場合 IOException
        // を発生させますので、それに対応するために Exception 処理を追加
        try {
```

```
        url = loader.getResource("photo01.jpg");
        appearance = new SimpleAppearance( url );
    } catch (IOException ex) {
        System.err.println("画像ファイルの 読み込みに失 敗しました。 ");
        ex.printStackTrace();
    }

    // Box の生成
    // 引数は、幅(x)、高さ(y)、奥行き(z)、フラグ、アピアランス
    // Box.GENERATE_TEXTURE_COORDS は
    // Box オブジェクトでテクスチャを利用するためには必要なフラグです
    Box box = new Box(0.10f, 0.08f, 0.06f,
                      Box.GENERATE_TEXTURE_COORDS, appearance);

    // コンポーネントの生成
    Component3D component = new Component3D();

    // シェイプをコンポーネントに追加
    component.addChild(box);

    // コンポーネントをコンテナに追加
    // ここではコンテナを使用していないので、
    // 直接フレームに追加
    frame.addChild(component);

    // フレームの表示
    // フレームの大きさを設定
    frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));

    // フレームの表示
    frame.changeEnabled(true);
    frame.changeVisible(true);

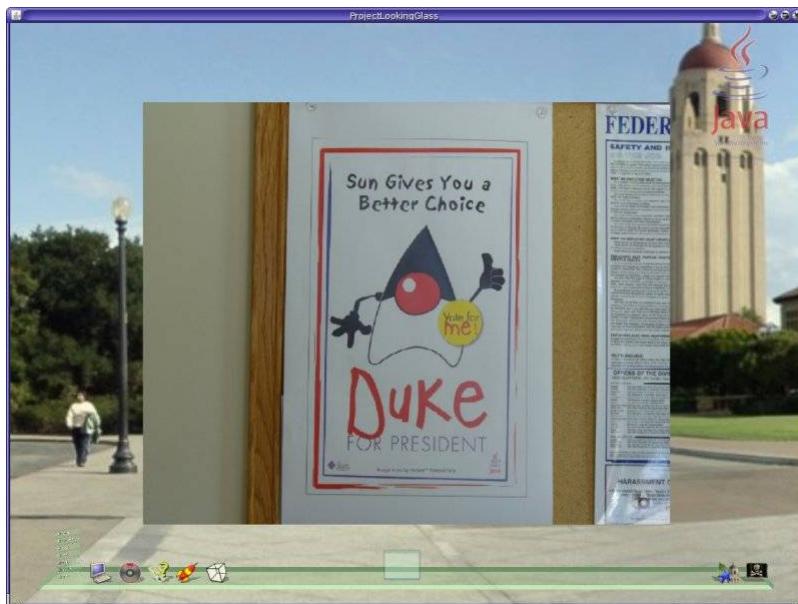
}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    new TextureBox();
}

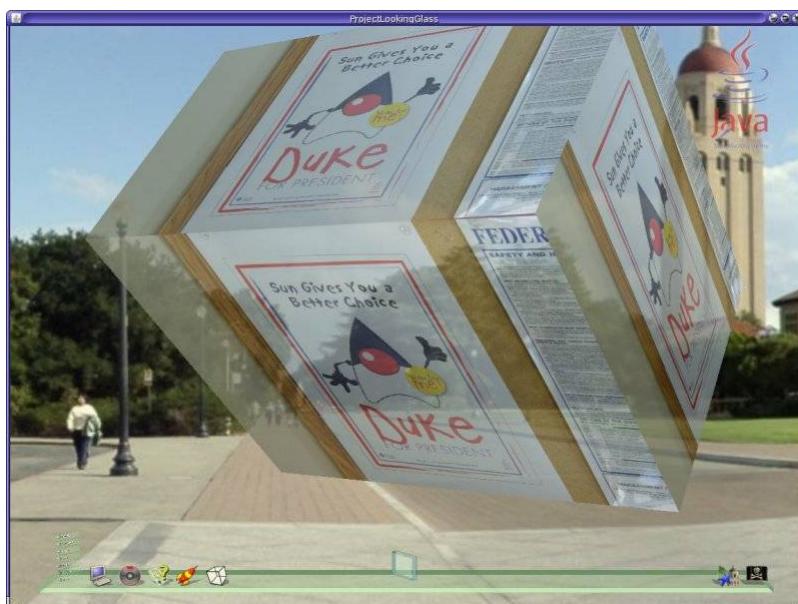
}
```

2. 「4.3 Jarベースのアプリケーションの作成と実行」にある [「JARベースのアプリケーションの作成」](#) の [「ステップ8」](#) を参照し、プロジェクトを構築します。同ページにある [「JARベースのアプリケーションの実行」](#) を参照し、アプリケーションの実行を行います。

実行画面は下図のようになります。



3. Java ロゴをドラッグしてボックスを回転させてみて下さい。全ての面に同じ画像が表示されていることがわかります。  
確認後、右下のドクロマークで LG3D を終了します。



## テクスチャを使う(その2)

上記では1つの画像をすべての面に表示させる処理を行いましたが、次は6面すべてに違う画像を表示するようにプログラムを変更します。

変更点は次の通りです。

- org.jdesktop.lg3d.utils.shape.Box オブジェクトの生成部分
  - アピアランスを null にする。  
各面に異なるテクスチャを貼るために、ここではアピアランスは指定しない。
- ボックスの各面に異なる画像を貼り付ける
  - Box オブジェクトから各面のシェイプ(org.jdesktop.lg3d.sg.Shape3D オブジェクト)を取得。
  - Shape3D.setAppearance() メソッドを利用し、ボックスの各面にアピアランスを設定する。  
各面のアピアランスは SimpleAppearance クラスを利用し、別々のテクスチャを使ったアピアランスを生成する。

1. TextureBox.java を次のように書き換えます。

変更部分は赤(太字)の部分です。

```
/*
 * TextureBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */
package texturebox;

/**
 *
 * @author duke
 */
import java.io.IOException;
import java.net.URL;
import java.lang.ClassLoader;
import javax.vecmath.Vector3f;
import org.jdesktop.lg3d.sg.Appearance;
import org.jdesktop.lg3d.sg.Shape3D;
import org.jdesktop.lg3d.utils.shape.Box;
import org.jdesktop.lg3d.utils.shape.SimpleAppearance;
import org.jdesktop.lg3d.wg.Component3D;
import org.jdesktop.lg3d.wg.Frame3D;

public class TextureBox {
    /** Creates a new instance of TextureBox */
    public TextureBox() {
```

```

// フレームの生成
Frame3D frame = new Frame3D();

// シェイプの生成および設定
// ボックス 6 面の画像を指定した Appearance の設定
Appearance appearanceFront = null;
Appearance appearanceRight = null;
Appearance appearanceLeft = null;
Appearance appearanceTop = null;
Appearance appearanceBottom = null;
Appearance appearanceBack = null;

URL url = null;
ClassLoader loader = this.getClass().getClassLoader();

// SimpleAppearance オブジェクトを作成する際に、
// テクスチャファイルの読み込みに失敗した場合 IOException
// を発生させますので、それに対応するために Exception 処理を追加
try {
    url = loader.getResource("photo01.jpg");
    appearanceFront = new SimpleAppearance( url );
    url = loader.getResource("photo02.jpg");
    appearanceRight = new SimpleAppearance( url );
    url = loader.getResource("photo03.jpg");
    appearanceLeft = new SimpleAppearance( url );
    url = loader.getResource("photo04.jpg");
    appearanceTop = new SimpleAppearance( url );
    url = loader.getResource("photo05.jpg");
    appearanceBottom = new SimpleAppearance( url );
    url = loader.getResource("photo06.jpg");
    appearanceBack = new SimpleAppearance( url );
} catch (IOException ex) {
    System.err.println("画像ファイルの読み込みに失敗しました。");
    ex.printStackTrace();
}
// Box の生成
// 引数は、幅(x)、高さ(y)、奥行き(z)、フラグ、アピアランス
// Box.GENERATE_TEXTURE_COORDS は
// Box オブジェクトで テクスチャを利用するため必要なフラグです
// 6 面の Appearance を貼り付けるため、とりあえず Box の Appearance を null
// にします。
Box box = new Box(0.10f,0.08f,0.06f, Box.GENERATE_TEXTURE_COORDS,
null);
// 正面
Shape3D shape = box.getShape(Box.FRONT);
shape.setAppearance(appearanceFront);

```

```
// 右側
shape = box.getShape(Box.RIGHT);
shape.setAppearance(appearanceRight);

// 左側
shape = box.getShape(Box.LEFT);
shape.setAppearance(appearanceLeft);

// 上
shape = box.getShape(Box.TOP);
shape.setAppearance(appearanceTop);

// 下(底面)
shape = box.getShape(Box.BOTTOM);
shape.setAppearance(appearanceBottom);

// 後
shape = box.getShape(Box.BACK);
shape.setAppearance(appearanceBack);

// コンポーネントの生成
Component3D component = new Component3D();

// シェイプをコンポーネントに追加
component.addChild(box);

// コンポーネントをコンテナに追加
// ここではコンテナを使用していないので、
// 直接フレームに追加
frame.addChild(component);

// フレームの表示
// フレームの大きさを設定
frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));

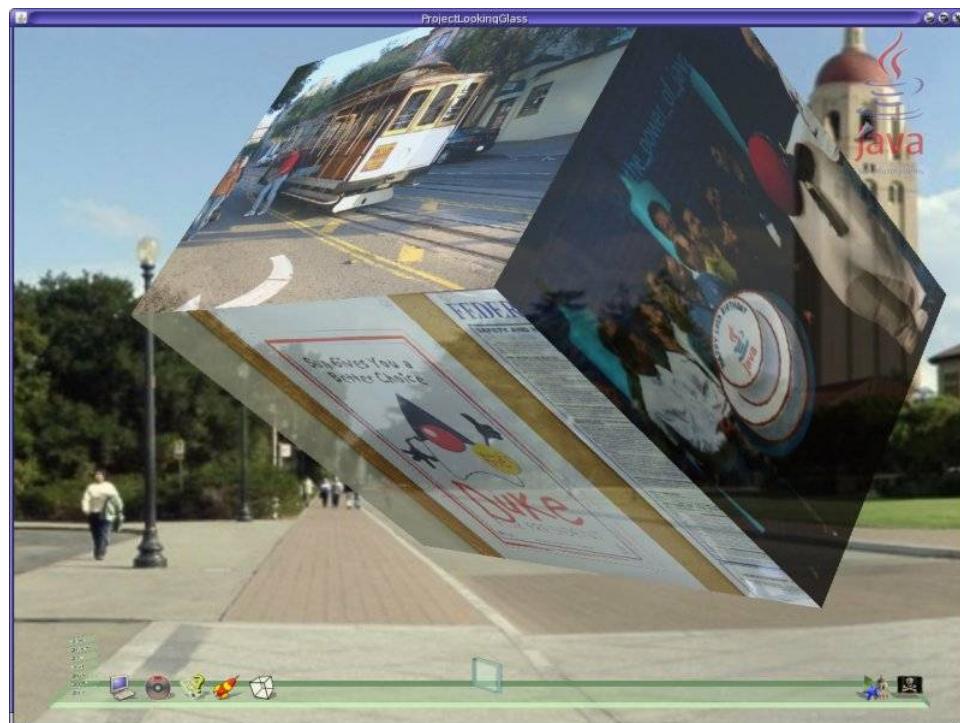
// フレームの表示
frame.changeEnabled(true);
frame.changeVisible(true);

}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    new TextureBox();
}
```

2. 「4.3 Jar ベースのアプリ ケーションの 作成と実行」 にある [「JARベースのアプリケーションの作成」 の ステップ8](#) を参照し、プロジェクトを構築します。同ページにある [「JARベースのアプリケーションの実行」](#) を参照し、アプリケーションの実行を行います。

LG3Dの実行ウィンドウが表示されますので、Java 口ゴをドラッグし、各面に違う画像が表示されているか確認します。



## 4.5 サムネイルの作成

### この章で行うこと：

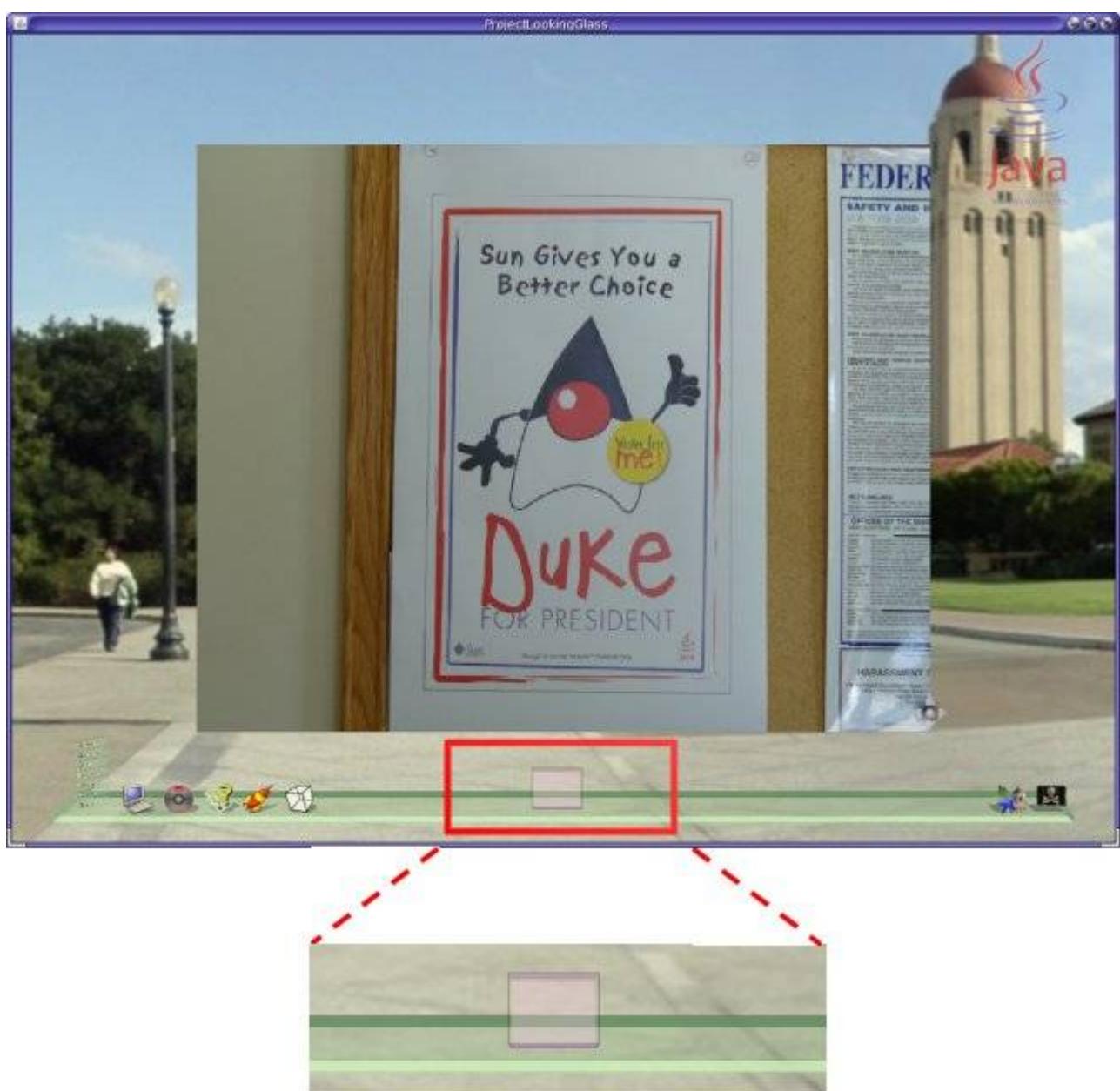
この章では、前章までで作成したボックスと同じ形をしたサムネイルの作成を行います。

LG3Dを起動すると下部にタスクバーが表示されます。

このタスクバーには「アプリケーションを起動するためのアイコン」と「現在起動しているアプリケーションの情報を示すサムネイル」が表示されます。

デフォルトのサムネイルは下図のような単純なものですですが、LG3Dではこのサムネイルも自由に作成することができます。

ここでは、作成したボックスと同じ形のサムネイルを作成します。



## サムネイルの作成

org.jdesktop.desktop.lg3d.wg.Thumbnail クラスを利用してサムネイルを作成します  
Thumbnail クラスは Component3D の派生クラスで、コンポーネントと同様に

- Thumbnail オブジェクトにコンポーネントを追加する

ことによりサムネイルを作成します。

ここでは、前章で作成したテクスチャを貼ったボックスと同じ形状のサムネイルを作成します。  
なお、フレームにコンポーネントを追加する場合 Frame3D.addChild() を使用しましたが、サムネイルを追加する場合は Frame3D.setThumbnail() になります。

1. TextureBox.java を変更し、テクスチャを貼る部分を別のメソッドにします。

変更箇所は赤(太字)で示します。

サムネイルの作成方法はコンポーネントと同様なので、テクスチャの生成および貼り付け作業をコンポーネントとサムネイルの 2 度行うことになります。ここでは簡単化のために **setTextures(Box box)** というテクスチャ貼り付け用のメソッドを作成します。

### 補足:

LG3D では 1 つの 3D オブジェクトを複数のコンポーネント/コンテナに追加することができません。そのため、見栄えが同じオブジェクトが 2 つ必要な場合でも、それぞれ別々に作成する必要があります。

```
/*
 * TextureBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package texturebox;

/**
 *
 * @author duke
 */

import java.io.IOException;
import java.net.URL;
import java.lang.ClassLoader;
import javax.vecmath.Vector3f;
import org.jdesktop.desktop.lg3d.sg.Appearance;
import org.jdesktop.desktop.lg3d.sg.Shape3D;
import org.jdesktop.desktop.lg3d.utils.shape.Box;
import org.jdesktop.desktop.lg3d.utils.shape.SimpleAppearance;
import org.jdesktop.desktop.lg3d.wg.Component3D;
import org.jdesktop.desktop.lg3d.wg.Frame3D;
```

```
public class TextureBox {  
    /** Creates a new instance of TextureBox */  
    public TextureBox() {  
  
        // フレームの生成  
        Frame3D frame = new Frame3D();  
  
        // Box の生成  
        // 引数は、x、y、z 座標、アピアランス  
        // 座標の単位はメートル(float 値)  
        // 作成されるのは 横 0.20f, 縦 0.16f, 奥行き 0.12f のボックス  
        Box box = new Box(0.10f, 0.08f, 0.06f,  
                          Box.GENERATE_TEXTURE_COORDS, null);  
  
        setTextures(box);  
  
        // コンポーネントの生成  
        Component3D component = new Component3D();  
  
        // シェイプをコンポーネントに追加  
        component.addChild(box);  
  
        // コンポーネントをコンテナに追加  
        // ここではコンテナを使用していないので、  
        // 直接フレームに追加  
        frame.addChild(component);  
  
        // フレームの表示  
        // フレームの大きさを設定  
        frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));  
  
        // フレームの表示  
        frame.changeEnabled(true);  
        frame.changeVisible(true);  
    }  
  
    private void setTextures(Box box){  
        URL url = null;  
        ClassLoader loader = this.getClass().getClassLoader();  
  
        // シェイプ の生成および 設定  
        // ボックス 6面の画像を 指定した Appearance の設定  
        Appearance appearanceFront = null;  
        Appearance appearanceRight = null;  
        Appearance appearanceLeft = null;  
        Appearance appearanceTop = null;  
        Appearance appearanceBottom = null;
```

```
Appearance appearanceBack = null;

// SimpleAppearance オブジェクトを作成する際に、
// テクスチャファイルの読み込みに失敗した場合 IOException
// を発生させますので、それに対応するため Exception 处理を追加
try {
    url = loader.getResource("photo01.jpg");
    appearanceFront = new SimpleAppearance( url );
    url = loader.getResource("photo02.jpg");
    appearanceRight = new SimpleAppearance( url );
    url = loader.getResource("photo03.jpg");
    appearanceLeft = new SimpleAppearance( url );
    url = loader.getResource("photo04.jpg");
    appearanceTop = new SimpleAppearance( url );
    url = loader.getResource("photo05.jpg");
    appearanceBottom = new SimpleAppearance( url );
    url = loader.getResource("photo06.jpg");
    appearanceBack = new SimpleAppearance( url );
} catch (IOException ex) {
    System.err.println("画像ファイルの読み込みに失敗しました。");
    ex.printStackTrace();
}
// 正面
Shape3D shape = box.getShape(Box.FRONT);
shape.setAppearance(appearanceFront);

// 右側
shape = box.getShape(Box.RIGHT);
shape.setAppearance(appearanceRight);

// 左側
shape = box.getShape(Box.LEFT);
shape.setAppearance(appearanceLeft);

// 上
shape = box.getShape(Box.TOP);
shape.setAppearance(appearanceTop);

// 下(底面)
shape = box.getShape(Box.BOTTOM);
shape.setAppearance(appearanceBottom);

// 後
shape = box.getShape(Box.BACK);
shape.setAppearance(appearanceBack);
}
```

```
/**  
 * @param args the command line arguments  
 */  
public static void main(String[] args) {  
    // TODO code application logic here  
    new TextureBox();  
  
}  
}
```

2. Texture.java を変更し、サムネイルを表示するようにします。

サムネイルの作成方法は以下のようになります

- ・ シェイプを作成する
  - ・ シェイプにテクスチャを貼る
  - ・ シェイプを含むコンポーネントを作成する
  - ・ コンポーネントをサムネイルに格納する
  - ・ サムネイルをフレームに格納する

また、LG3D1.0 では、サムネイルは自動に縮小表示されなくて、サムネイルを作成する際、コンポーネントのサイズより小さく設定する必要があります。  
ここでは、コンポーネントのサイズの 10 分の 1 とします。

```
/*
 * TextureBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */
package texturebox;

/**
 *
 * @author duke
 */
import java.io.IOException;
import java.net.URL;
import java.lang.ClassLoader;
import javax.vecmath.Vector3f;
import org.jdesktop.desktop.lq3d.sq.Appearance;
```

```
import org.jdesktop.lwjgl3d.sg.Shape3D;
import org.jdesktop.lwjgl3d.utils.shape.Box;
import org.jdesktop.lwjgl3d.utils.shape.SimpleAppearance;
import org.jdesktop.lwjgl3d.wg.Component3D;
import org.jdesktop.lwjgl3d.wg.Frame3D;
import org.jdesktop.lwjgl3d.wg.Thumbnail;

public class TextureBox {

    /** Creates a new instance of TextureBox */
    public TextureBox() {

        // フレームの生成
        Frame3D frame = new Frame3D();

        // Box の生成
        // 引数は、x、y、z 座標、アピアランス
        // 座標の単位はメートル(float 値)
        // 作成されるのは 横 0.20f, 縦 0.16f, 奥行き 0.12f のボックス
        Box box = new Box(0.10f, 0.08f, 0.06f,
                           Box.GENERATE_TEXTURE_COORDS, null);

        setTextures(box);

        // コンポーネントの生成
        Component3D component = new Component3D();

        // シェイプをコンポーネントに追加
        component.addChild(box);

        // サムネイル用の Box(シェイプ)の生成
        // サイズは コンポーネントのサイズの 10 分の 1 にします。
        Box thumbBox = new Box(0.010f, 0.008f, 0.006f,
                               Box.GENERATE_TEXTURE_COORDS, null);

        // サムネイル用の Box(シェイプ)にテクスチャを 貼る
        setTextures(thumbBox);

        // コンポーネントを生成 し、Box(シェイプ)を格納する
        Component3D thumbComponent = new Component3D();
        thumbComponent.addChild(thumbBox);

        // サムネイルの生成
        Thumbnail thumbnail = new Thumbnail();

        // サムネイルにコンポーネントを格納
        thumbnail.addChild(thumbComponent);
```

```

// サムネイルのサイズを 設定
// このサイズもコンポーネントのサイズの 10 分の 1 にします。
thumbnail.setPreferredSize(new Vector3f(0.010f, 0.008f, 0.006f));

// サムネイルの追加
// サムネイルの追加は Frame3D.setThumbnail() を使用します
frame.setThumbnail(thumbnail);

// コンポーネントをコンテナに追加
// ここではコンテナを使用していないので、
// 直接フレームに追加
frame.addChild(component);

// フレームの表示
// フレームの大きさを設定
frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));

// フレームの表示
frame.changeEnabled(true);
frame.changeVisible(true);

}

private void setTextures(Box box){
    URL url = null;
    ClassLoader loader = this.getClass().getClassLoader();

    // シェイプの生成および設定
    // ボックス 6 面の画像を指定した Appearance の設定
    Appearance appearanceFront = null;
    Appearance appearanceRight = null;
    Appearance appearanceLeft = null;
    Appearance appearanceTop = null;
    Appearance appearanceBottom = null;
    Appearance appearanceBack = null;

    // SimpleAppearance オブジェクトを作成する際に、
    // テクスチャファイルの読み込みに失敗した場合 IOException
    // を発生させますので、それに対応するために Exception 処理を追加
    try {
        url = loader.getResource("photo01.jpg");
        appearanceFront = new SimpleAppearance( url );
        url = loader.getResource("photo02.jpg");
        appearanceRight = new SimpleAppearance( url );
        url = loader.getResource("photo03.jpg");
    }
}

```

```
appearanceLeft = new SimpleAppearance( url );
url = loader.getResource("photo04.jpg");
appearanceTop = new SimpleAppearance( url );
url = loader.getResource("photo05.jpg");
appearanceBottom = new SimpleAppearance( url );
url = loader.getResource("photo06.jpg");
appearanceBack = new SimpleAppearance( url );
} catch (IOException ex) {
    System.err.println("画像ファイルの読み込みに失敗しました。");
    ex.printStackTrace();
}

// 正面
Shape3D shape = box.getShape(Box.FRONT);
shape.setAppearance(appearanceFront);

// 右側
shape = box.getShape(Box.RIGHT);
shape.setAppearance(appearanceRight);

// 左側
shape = box.getShape(Box.LEFT);
shape.setAppearance(appearanceLeft);

// 上
shape = box.getShape(Box.TOP);
shape.setAppearance(appearanceTop);

// 下(底面)
shape = box.getShape(Box.BOTTOM);
shape.setAppearance(appearanceBottom);

// 後
shape = box.getShape(Box.BACK);
shape.setAppearance(appearanceBack);
}

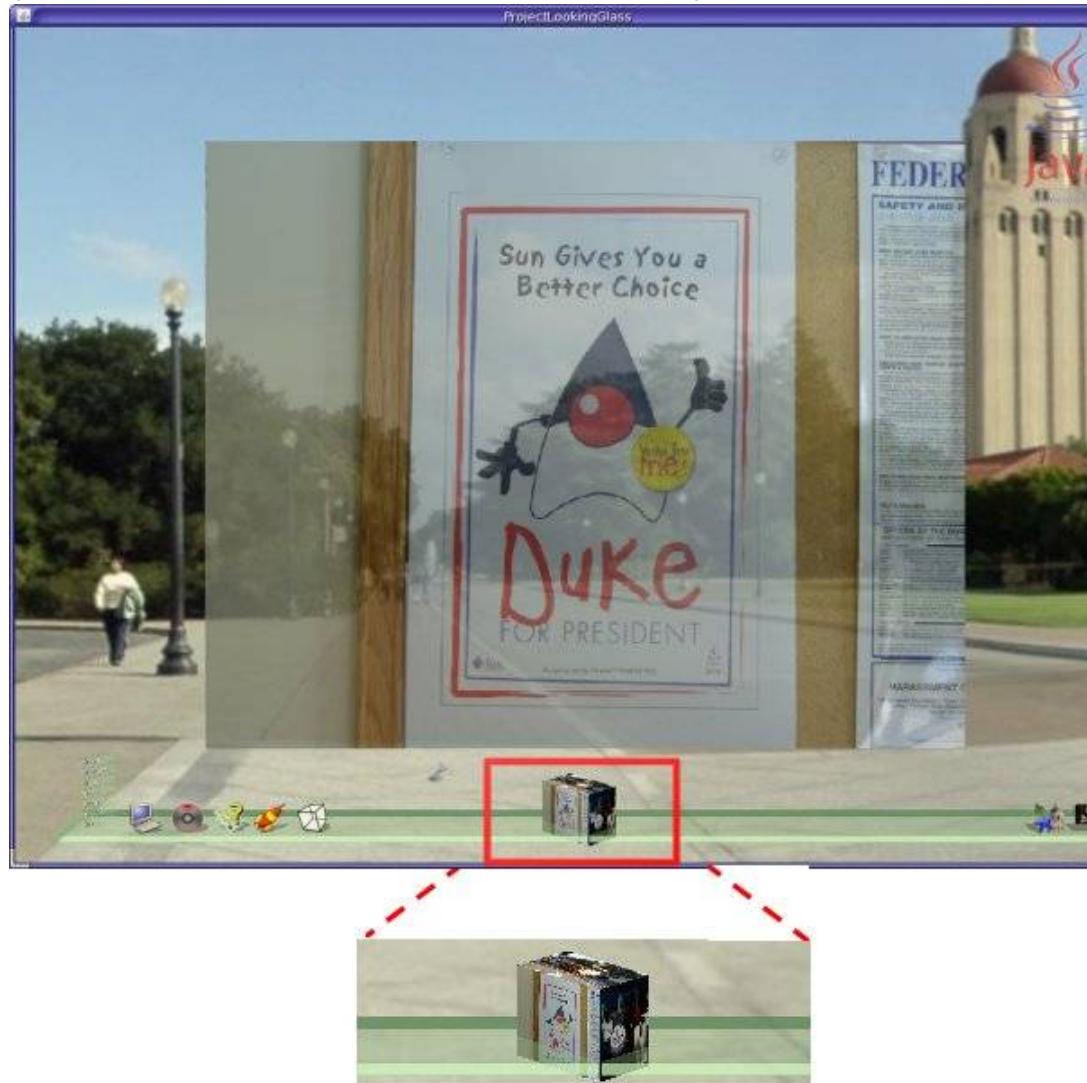
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    new TextureBox();

}
```

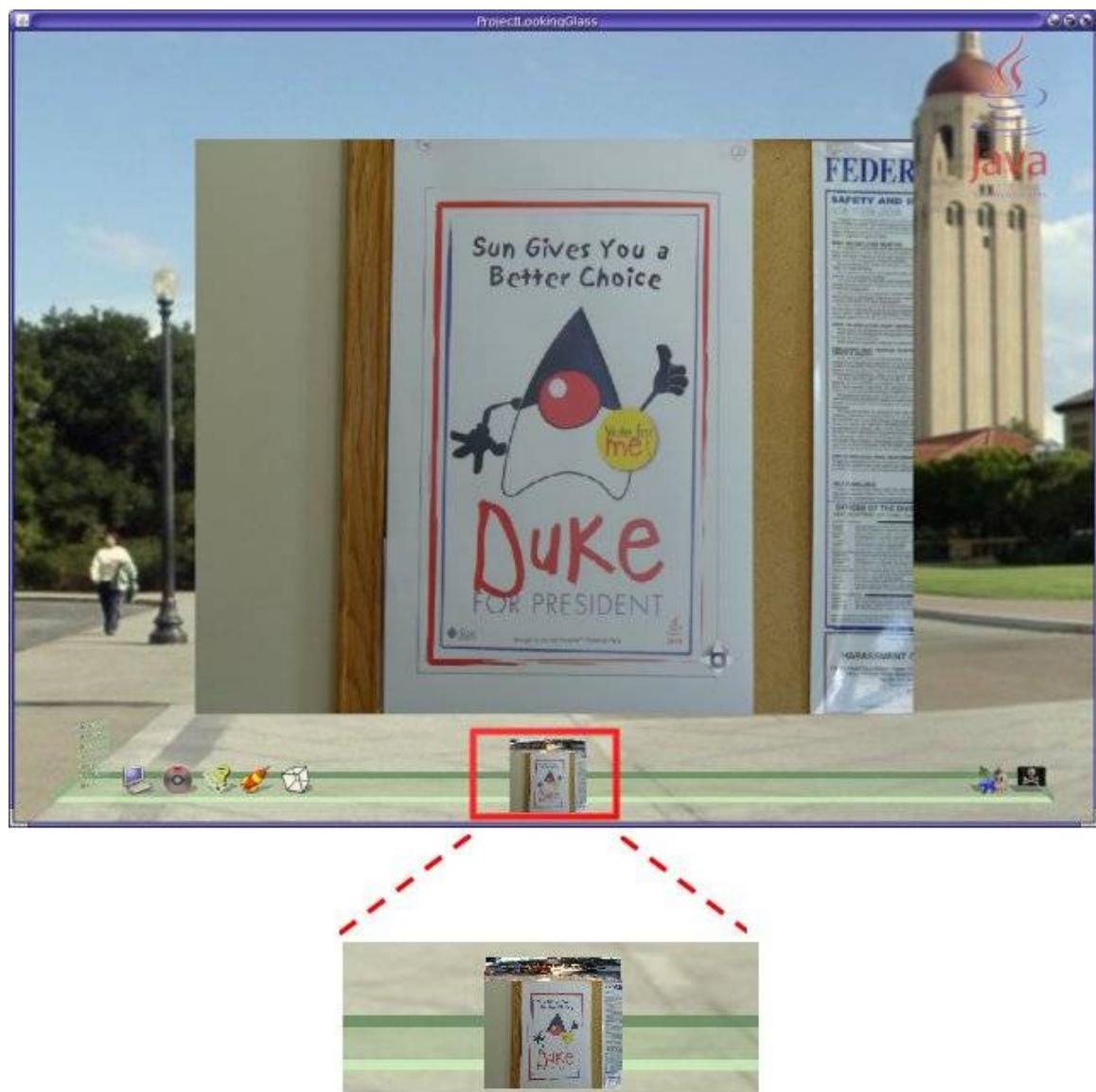
3. 「4.3 Jar ベースのアプリ ケーションの 作成と実行」 にある [「JARベースのアプリケーションの作成」](#) の [ステップ8](#) を参照し、プロジェクトを構築します。  
同ページにある [「JARベースのアプリケーションの実行」](#) を参照し、アプリケーションの実行を行います。

実行画面をキャプチャしたものが下図になります。

(マウスカーソルがアプリケーションの外にある場合)



(マウスカーソルがアプリケーション内にある場合)



## 4.6 マウスイベント・アクションの作成

### この章で行うこと：

作成したボックスを拡張してマウスイベントおよびアクションを作成します。

以下の2種類を実装してみます。

1. マウスボタンを押している間、ボックスが回転する
  - LG3D 標準のアクションを利用します
2. マウスの左クリックでボックスを 90 度回転させ、右クリックでボックスを逆方向に 90 度回転させる
  - アクションを新規に作成します

---

### マウスイベント・アクションの概要

ここまで行ってきた内容は LG3D アプリケーションの表示の部分です。

この章ではイベント処理の実装を行います。

LG3D でイベント処理を行うためには **イベントアダプタ** と **アクション** を用意します。

イベントアダプタ	イベントアダプタはイベントが発生したときに呼び出されるクラスです。イベントアダプタはイベントから情報を取得し、イベントアダプタに登録されているアクションの performAction() メソッドを呼び出します。イベントアダプタは org.jdesktop.j3d.utils.eventadapter パッケージで定義されており EventAdapter インタフェースを実装しています。
アクション	アクションはイベントが発生した際に行う処理を定義するためのクラスです。 org.jdesktop.j3d.utils.action.Action インタフェースを親とした派生インターフェース群が提供されているので、これらのインターフェースのいずれかを実装したクラスでイベント処理を定義します。

イベントアダプタとアクションを利用したイベント処理の作成方法は次のようになります。

- シェイプを含んだコンポーネントを作成する
- イベント発生時に行う動作(アクション)を定義する
- アクションを含むイベントアダプタを作成する
- コンポーネントにイベントアダプタを登録する

LG3D では、基本的に簡単なアクションをいくつか提供していますが、自分で作成することもできます。

ここでは、まず標準で提供されているアクションを利用してマウスイベントの作成を行います。その後、オリジナルのアクションを作成してみます。

---

## 既存のアクションを利用したアニメーション

ここでは、作成したコンポーネント上でマウスのボタンを押している続けている間にゆっくりと1回転し、マウスのボタンをはなすと元に戻るというイベント処理を作成してみます。このイベント処理は次のイベントアダプタおよびアクションを用いて実装します。

イベント アダプタ	org.jdesktop.swingx.event.adapter.MousePressedEventAdapter クラス。マウスのボタンを押している間のイベント処理を行うためのイベントアダプタ。このイベントアダプタは次に示す3種類のインターフェースのいずれかを実装したアクションを設定することができます。 <ul style="list-style-type: none"><li>• org.jdesktop.swingx.event.action.ActionBoolean</li><li>• org.jdesktop.swingx.event.action.ActionBooleanFloat2</li><li>• org.jdesktop.swingx.event.action.ActionBooleanFloat3</li></ul>
アクション	org.jdesktop.swingx.event.action.RotateActionBoolean クラス。イベント発生時にコンポーネントを回転させるためのアクションです。コンストラクタの引数として、回転させたいオブジェクト、回転角度(ラジアン)、回転にかかる時間(ミリ秒)を取ります。

1. TextureBox.javaを以下のように変更します。

変更箇所は**赤(太字)**で示します。

adapter と thumbAdapter という2つのイベントアダプタ(MousePressedEventAdapter オブジェクト)を作成します。作成したイベントアダプタは両方とも component (ボックスのコンポーネントのオブジェクト)に登録します。これにより、ボックス上でマウスのボタンを押し続けることにより、ボックスと同時にサムネイルも回転します。ボックスとサムネイルは 10秒間で 1回転するようにしています。

イベント処理を追加する前に、ボックスのコンポーネントとサムネイルの setAnimation() メソッドを呼び出しています ( component.setAnimation() , thumbComponent.setAnimation() )。これはアニメーション時にどのような動きをするかを定義したものです。プログラム中に出てくる設定は滑らかなアニメーションをするための設定です。アニメーションをさせたいオブジェクトに対するお約束の処理と考えてください。

```
/*
 * TextureBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */
package texturebox;
```



```

// サムネイル用の Box(シェイプ)にテクスチャを貼る
setTextures(thumbBox);

// コンポーネントを生成し、Box(シェイプ)を格納する
Component3D thumbComponent = new Component3D();
thumbComponent.addChild(thumbBox);

// サムネイルの生成
Thumbnail thumbnail = new Thumbnail();

// サムネイルにコンポーネントを格納
thumbnail.addChild(thumbComponent);

// サムネイルのサイズを設定
// このサイズもコンポーネントのサイズの 10 分の 1 にします。
thumbnail.setPreferredSize(new Vector3f(0.010f, 0.008f, 0.006f));

// 滑らかに動かす ためのお約束
component.setAnimation(new NaturalMotionAnimation(1000));
// マウスボタンを 押していると きのイベント 处理を
// 行うため のイベントアダプタ
MousePressedEventAdapter adapter = new MousePressedEventAdapter(
    new RotateActionBoolean(component,
        (float)(Math.PI * 2.0), 10000));
// Y 軸を回転軸と 設定します
component.setRotationAxis(0, 1, 0);
// コンポーネント にイベントアダプタを登録
component.addListener(adapter);

// コンポーネントの上でマウスボタンを押した場合に
// サムネイルも 同じようにアニメーションをさせるため のイベントアダプタ
thumbComponent.setAnimation(new NaturalMotionAnimation(1000));
MousePressedEventAdapter thumbAdapter =
    new MousePressedEventAdapter(
        new RotateActionBoolean(thumbComponent,
            (float)(Math.PI * 2.0), 10000));
// サムネイルも Y 軸を回転軸と 設定します
thumbComponent.setRotationAxis(0, 1, 0);

// イベントをコンポーネントに 登録する
// コンポーネント (ボックス)上でマウスボタンを押している時に
// サムネイルも動くようにする ためにコンポーネントに登録しています
// サムネイル (thumbComponent)にイベントアダプタを
// 登録しないでください
component.addListener(thumbAdapter);

```

```
// コンポーネントにイベントリスナーを追加し た際に必要な お約束
component.setMouseEventPropagatable(true);

// サムネイルの追加
// サムネイルの追加は Frame3D.setThumbnail() を使用します
frame.setThumbnail(thumbnail);

// コンポーネントをコンテナに追加
// ここではコンテナを使用していないので、
// 直接フレームに追加
frame.addChild(component);

// フレームの表示
// フレームの大きさを設定
frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));

// フレームの表示
frame.changeEnabled(true);
frame.changeVisible(true);

}

private void setTextures(Box box){
    URL url = null;
    ClassLoader loader = this.getClass().getClassLoader();

    // シェイプの生成および設定
    // ボックス 6 面の画像を指定した Appearance の設定
    Appearance appearanceFront = null;
    Appearance appearanceRight = null;
    Appearance appearanceLeft = null;
    Appearance appearanceTop = null;
    Appearance appearanceBottom = null;
    Appearance appearanceBack = null;

    // SimpleAppearance オブジェクトを作成する際に、
    // テクスチャファイルの読み込みに失敗した場合 IOException
    // を発生させますので、それに対応するために Exception 処理を追加
    try {
        url = loader.getResource("photo01.jpg");
        appearanceFront = new SimpleAppearance( url );
        url = loader.getResource("photo02.jpg");
        appearanceRight = new SimpleAppearance( url );
        url = loader.getResource("photo03.jpg");
        appearanceLeft = new SimpleAppearance( url );
        url = loader.getResource("photo04.jpg");
    }
}
```

```
appearanceTop = new SimpleAppearance( url );
url = loader.getResource("photo05.jpg");
appearanceBottom = new SimpleAppearance( url );
url = loader.getResource("photo06.jpg");
appearanceBack = new SimpleAppearance( url );
} catch (IOException ex) {
    System.err.println("画像ファイルの読み込みに失敗しました。");
    ex.printStackTrace();
}

// 正面
Shape3D shape = box.getShape(Box.FRONT);
shape.setAppearance(appearanceFront);

// 右側
shape = box.getShape(Box.RIGHT);
shape.setAppearance(appearanceRight);

// 左側
shape = box.getShape(Box.LEFT);
shape.setAppearance(appearanceLeft);

// 上
shape = box.getShape(Box.TOP);
shape.setAppearance(appearanceTop);

// 下(底面)
shape = box.getShape(Box.BOTTOM);
shape.setAppearance(appearanceBottom);

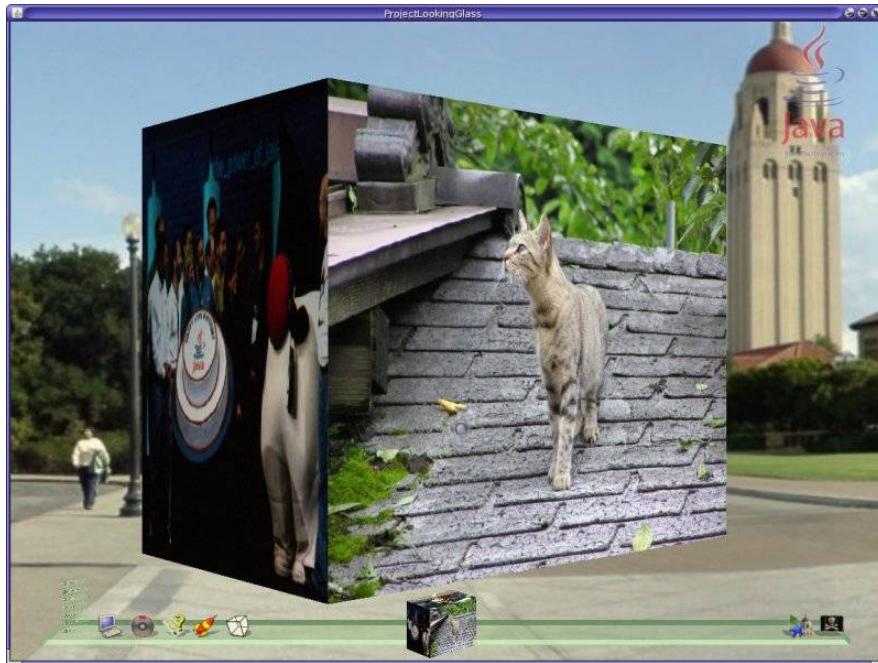
// 後
shape = box.getShape(Box.BACK);
shape.setAppearance(appearanceBack);
}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    new TextureBox();

}

}
```

2. 「4.3 Jar ベースのアプリ ケーションの 作成と実行」 にある [「JARベースのアプリケーションの作成」 の ステップ8](#) を参照し、プロジェクトを構築します。同ページにある [「JARベースのアプリケーションの実行」](#) を参照し、アプリケーションの実行を行います。  
実行後、ボックス上で**マウスのボタン を押し続ける** とゆっくりと ボックスとサムネイルが回転します。 また、マウスのボタンを離すと元に戻ります。



---

## アクションの作成

次はマウスをクリックしたときに 90 度づつ回転するようにプログラムを変更します。  
マウスをクリックした場合のイベントアダプタは  
org.jdesktop.desktop.lg3d.utils.eventadapter.MouseClickedEventAdapter クラスです。  
MouseClickedEventAdapter クラスはコンストラクタの引数として、次のインターフェースのいずれかを実装したアダプタが必要です。

- org.jdesktop.desktop.lg3d.utils.action.ActionNoArg
- org.jdesktop.desktop.lg3d.utils.action.ActionFloat2
- org.jdesktop.desktop.lg3d.utils.action.ActionFloat3

ここでは ActionNoArg インタフェースを実装したクラスを作成します。

### 補足：

アクションを作成するために利用するインターフェースの名前は イベント発生時に呼び出される performAction() メソッドの引数に対応しています。 (引数のうち、先頭の org.jdesktop.desktop.lg3d.wg.event.LgEventSource を除いたもの)

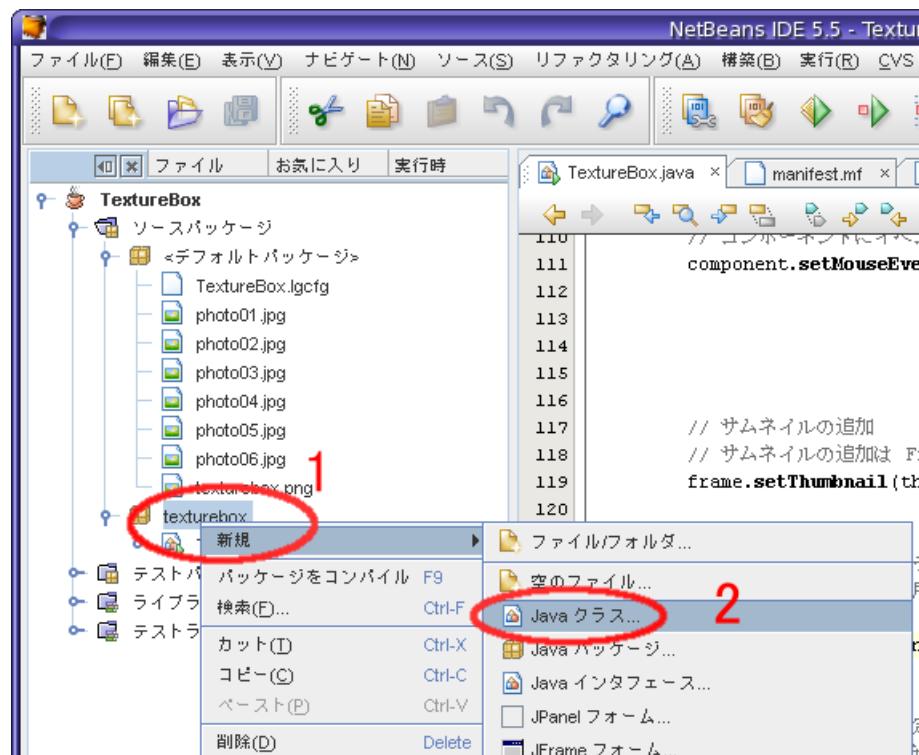
上記のインターフェースの performAction メソッドは次のようになっています。

- ActionNoArg.performAction(LgEventSource source)
- ActionNoArg.performAction(LgEventSource source, float x, float y)
- ActionNoArg.performAction(LgEventSource source, float x, float y, float z)

performAction()メソッド実行時に渡される LgEventSource オブジェクトはイベントの発生元です。

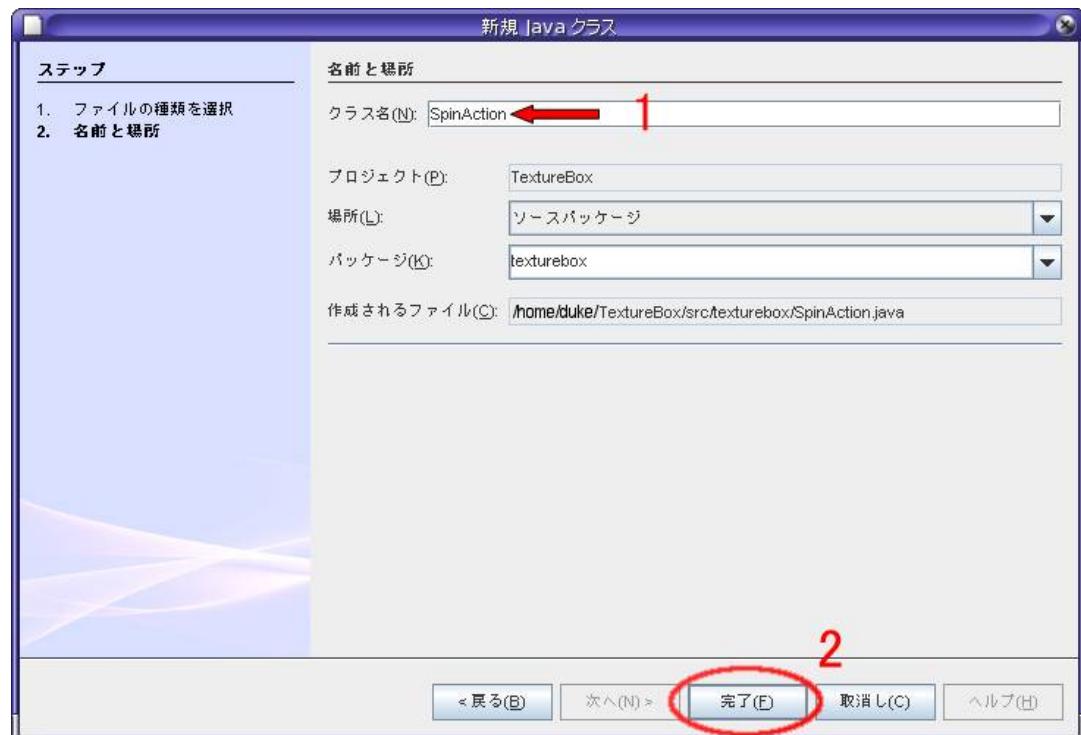
1. アクションを作成するために新しいクラスを作成します。

左側のパネルが「プロジェクト」となっていることを確認し、TextureBox プロジェクト内の「ソースパッケージ」→「texturebox」を選択します。選択後、右クリックでメニューが表示されるので「新規」→「Java クラス」を選択します。

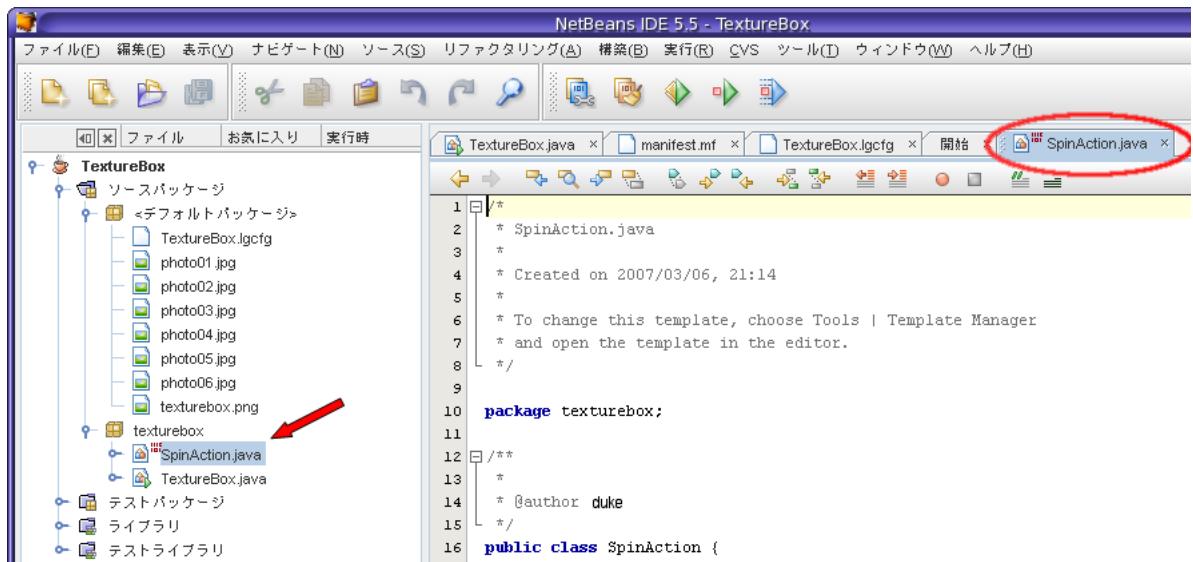


2. クラス情報を入力するためのダイアログが表示されます。

- ・ クラス名 : **SpinAction**  
と入力し、「完了」を選択します。



3. 左側のパネルに「**SpinAction**」が追加されます。  
 また、右側のソースコードエディタに「**SpinAction**」が表示されます。



4. SpinAction.java を以下のように変更します。

org.jdesktop.lg3d.utils.action.Action の派生インターフェースは イベント発生時に呼び出される performAction メソッドが必要とする変数が応じたものが用意されています。  
 ここでは引数と特に必要としない org.jdesktop.lg3d.utils.action.ActionNoArg インターフェースを実装しています。

SpinActionクラスのメソッドはコンストラクタと performAction() の 2 つになります。  
 コンストラクタでは、

- 回転させたいコンポーネント
- 回転する角度
- 回転にかかる時間

を引数にしています。

performAction() はイベント発生時に実際に使うアクションを実装するためのメソッドです。 ここではコンポーネントを 90 度回転させています。

引数の org.jdesktop.lg3d.event.LgEventSource オブジェクトはイベント発生元で、イベント発生時にイベントアダプタからアクションに渡されます。

```

/*
 * SpinAction.java
 *
 * Created on 2005/10/09, 14:47
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */
package texturebox;

```

```

import org.jdesktop.lg3d.utils.action.ActionNoArg;
import org.jdesktop.lg3d.wg.Component3D;
import org.jdesktop.lg3d.wg.event.LgEventSource;

/**
 *
 * @author duke
 */

public class SpinAction implements ActionNoArg{

    private Component3D target;
    private float diff;
    private int duration;
    private float angle;

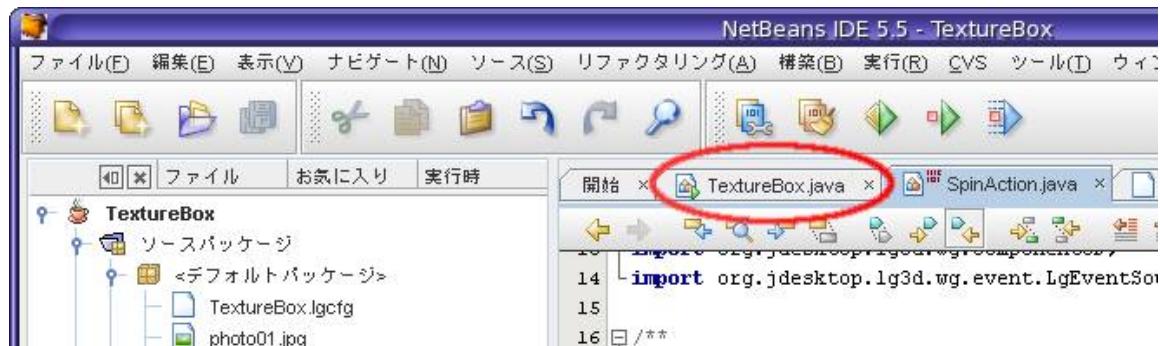
    // コンストラクタ
    // target 回転させたいコンポーネント
    // diff 回転させる角度(ラジアン)
    // duration 回転に要する時間(ミリ秒)
    public SpinAction(Component3D target, float diff, int duration) {
        if (target == null) {
            throw new IllegalArgumentException
                ("target オブジェクトが指定されていません");
        }

        this.target = target;
        this.diff = diff;
        this.duration = duration;
    }

    // イベント処理をおこなうためのメソッド
    public void performAction(LgEventSource source) {
        // target.getFinalRotationAngle() は現在の回転角度(ラジアン)を
        // 取得するためのメソッドです。
        // このメソッドに diff(回転させる角度) を足して、
        // 新しい回転させる角度を定義します。
        angle = target.getFinalRotationAngle() + diff;
        // angle ... 最終的な回転角度
        // duration ... 回転に要する時間
        // です。実際に回転する角度は diff
        // (最終的な回転角度 angle - 現在の回転角度 getFinalRotationAngle())
        // になります
        target.changeRotationAngle(angle, duration);
    }
}

```

5. 右側のソースコードエディタから「**TextureBox**」を選択し、 **TextBox.java** の編集画面に切り替えます。



6. **TextBox.java** を以下のように変更します。

イベントアダプタとして

`org.jdesktop.lg3d.utils.eventadapter.MouseClickedEventAdapter`

クラスを利用しています。イベントアダプタ生成時のコンストラクタの第 1 引数はマウスのボタンを定義しています。

マウスのボタンは `org.jdesktop.lg3d.wg.event.MouseEvent3D` クラス内の `Enum` 変数 `ButtonId` に定義されており、

- `ButtonId.BUTTON1` = 左ボタン
- `ButtonId.BUTTON2` = 中ボタン
- `ButtonId.BUTTON3` = 右ボタン

を示しています。

ここでは、マウスの左クリックと右クリックの際のイベント処理の作成をしており、 左クリックで 90 度回転、右クリックで逆方向に 90 度回転します。

また、登録するアクションは先ほど作成した `SpinAction` クラスを利用しています。

```
/*
 * TextBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */
package texturebox;

/**
 *
 * @author duke
 */

import java.io.IOException;
import java.net.URL;
import java.lang.ClassLoader;
```

```
import javax.vecmath.Vector3f;
import org.jdesktop.j3d_sg.Appearance;
import org.jdesktop.j3d_sg.Shape3D;
import org.jdesktop.j3d_utils.shape.Box;
import org.jdesktop.j3d_utils.shape.SimpleAppearance;
import org.jdesktop.j3d_wg.Component3D;
import org.jdesktop.j3d_wg.Frame3D;
import org.jdesktop.j3d_wg.Thumbnail;
import org.jdesktop.j3d_utils.c3d_animation.NaturalMotionAnimation;
import org.jdesktop.j3d_utils.eventadapter.MouseClickedEventAdapter;
import org.jdesktop.j3d_wg.event.MouseEvent3D.ButtonId;

public class TextureBox {

    /** Creates a new instance of TextureBox */
    public TextureBox() {

        // フレームの生成
        Frame3D frame = new Frame3D();

        // Box の生成
        // 引数は、 x、 y、 z 座標、 アピアランス
        // 座標の単位はメートル(float 値)
        // 作成されるのは 横 0.20f, 縦 0.16f, 奥行き 0.12f のボックス
        Box box = new Box(0.10f, 0.08f, 0.06f,
                          Box.GENERATE_TEXTURE_COORDS, null);

        setTextures(box);

        // コンポーネントの生成
        Component3D component = new Component3D();

        // シェイプをコンポーネントに追加
        component.addChild(box);

        // サムネイル用の Box(シェイプ) の生成
        // サイズはコンポーネントのサイズの 10 分の 1 にします。
        Box thumbBox = new Box(0.010f, 0.008f, 0.006f,
                               Box.GENERATE_TEXTURE_COORDS, null);

        // サムネイル用の Box(シェイプ)にテクスチャを貼る
        setTextures(thumbBox);

        // コンポーネントを生成し、 Box(シェイプ)を格納する
        Component3D thumbComponent = new Component3D();
        thumbComponent.addChild(thumbBox);

        // サムネイルの生成
    }
}
```

```
Thumbnail thumbnail = new Thumbnail();

// サムネイルにコンポーネントを格納
thumbnail.addChild(thumbComponent);

// サムネイルのサイズを設定
// このサイズもコンポーネントのサイズの 10 分の 1 にします。
thumbnail.setPreferredSize(new Vector3f(0.010f, 0.008f, 0.006f));

// 滑らかに動かすためのお約束
component.setAnimation(new NaturalMotionAnimation(1000));
// Y 軸を回転軸と設定します
component.setRotationAxis(0, 1, 0);

// 左クリックで回転
// 回転角度は 90 度
MouseClickedEventAdapter leftClickedAdapter =
    new MouseClickedEventAdapter(ButtonId.BUTTON1,
        new SpinAction(component, (float)(Math.PI / 2.0), 1000));

// 右クリックで逆回転
// 回転角度は -90 度
MouseClickedEventAdapter rightClickedAdapter =
    new MouseClickedEventAdapter(ButtonId.BUTTON3,
        new SpinAction(component, - (float)(Math.PI / 2.0), 1000));

// コンポーネントにイベントアダプタを登録
component.addListener(leftClickedAdapter);
component.addListener(rightClickedAdapter);

// イベント発生時にサムネイルも同じアクションを行うようにするため、
// サムネイルにも同じアクションを設定する
thumbComponent.setAnimation(new NaturalMotionAnimation(1000));
// サムネイルも Y 軸を回転軸と設定します
thumbComponent.setRotationAxis(0, 1, 0);

// 左クリックで回転
MouseClickedEventAdapter thumbLeftClickedAdapter =
    new MouseClickedEventAdapter(ButtonId.BUTTON1,
        new SpinAction(thumbComponent, (float)(Math.PI / 2.0), 1000));

// 右クリックで逆回転
MouseClickedEventAdapter thumbRightClickedAdapter =
    new MouseClickedEventAdapter(ButtonId.BUTTON3,
        new SpinAction(thumbComponent, - (float)(Math.PI / 2.0), 1000));
```

```
// コンポーネントがクリックされた時にサムネイルも回転させるために
// イベントアダプタをコンポーネントに登録します
component.addListener(thumbLeftClickedAdapter);
component.addListener(thumbRightClickedAdapter);

// コンポーネントにイベントリスナーを追加した際に必要なお約束
component.setMouseEventPropagatable(true);

// サムネイルの追加
// サムネイルの追加は Frame3D.setThumbnail() を使用します
frame.setThumbnail(thumbnail);

// コンポーネントをコンテナに追加
// ここではコンテナを使用していないので、
// 直接フレームに追加
frame.addChild(component);

// フレームの表示
// フレームの大きさを設定
frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));

// フレームの表示
frame.changeEnabled(true);
frame.changeVisible(true);

}

private void setTextures(Box box){
    URL url = null;
    ClassLoader loader = this.getClass().getClassLoader();

    // シェイプの生成および設定
    // ボックス 6 面の画像を指定した Appearance の設定
    Appearance appearanceFront = null;
    Appearance appearanceRight = null;
    Appearance appearanceLeft = null;
    Appearance appearanceTop = null;
    Appearance appearanceBottom = null;
    Appearance appearanceBack = null;

    // SimpleAppearance オブジェクトを作成する際に、
    // テクスチャファイルの読み込みに失敗した場合 IOException
    // を発生させますので、それに対応するために Exception 処理を追加
    try {
        url = loader.getResource("photo01.jpg");
    }
}
```

```
appearanceFront = new SimpleAppearance( url );
url = loader.getResource("photo02.jpg");
appearanceRight = new SimpleAppearance( url );
url = loader.getResource("photo03.jpg");
appearanceLeft = new SimpleAppearance( url );
url = loader.getResource("photo04.jpg");
appearanceTop = new SimpleAppearance( url );
url = loader.getResource("photo05.jpg");
appearanceBottom = new SimpleAppearance( url );
url = loader.getResource("photo06.jpg");
appearanceBack = new SimpleAppearance( url );
} catch (IOException ex) {
    System.err.println("画像ファイルの読み込みに失敗しました。");
    ex.printStackTrace();
}

// 正面
Shape3D shape = box.getShape(Box.FRONT);
shape.setAppearance(appearanceFront);

// 右側
shape = box.getShape(Box.RIGHT);
shape.setAppearance(appearanceRight);

// 左側
shape = box.getShape(Box.LEFT);
shape.setAppearance(appearanceLeft);

// 上
shape = box.getShape(Box.TOP);
shape.setAppearance(appearanceTop);

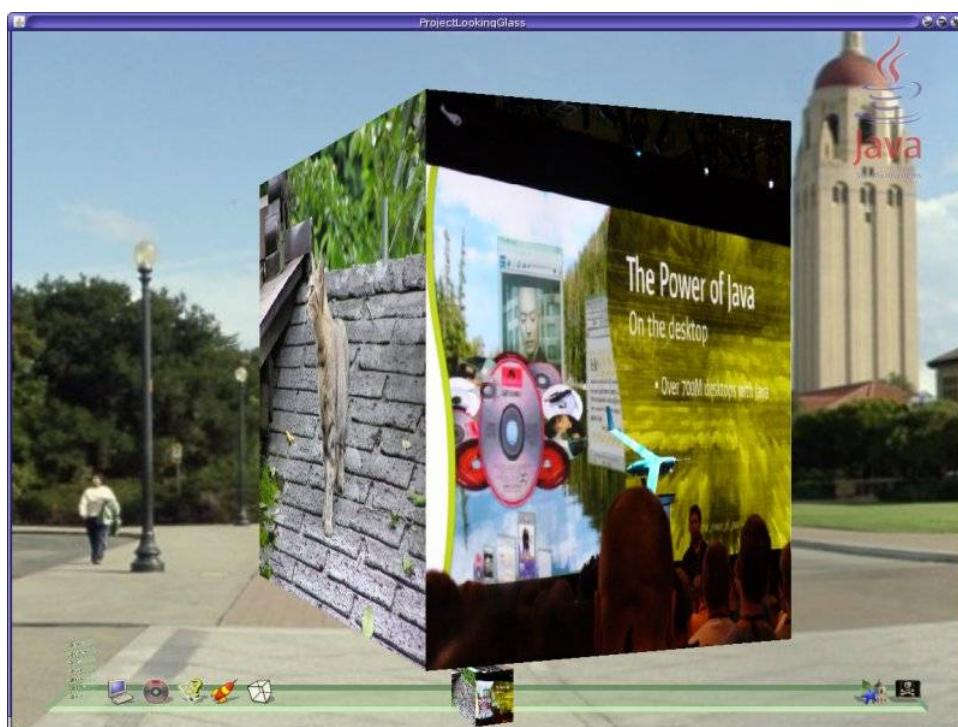
// 下(底面)
shape = box.getShape(Box.BOTTOM);
shape.setAppearance(appearanceBottom);

// 後
shape = box.getShape(Box.BACK);
shape.setAppearance(appearanceBack);
}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
}
```

```
new TextureBox();  
}  
}  
}
```

7. 「4.3 Jar ベースのアプリ ケーションの 作成と実行」 にある [「JARベースのアプリケーションの作成」](#) の [「ステップ8」](#) を参照し、プロジェクトを構築します。  
同ページにある [「JARベースのアプリケーションの実行」](#) を参照し、アプリケーションの実行を行います。  
実行画面は下図のようになります。ボックスをクリックまたは右クリックしてください。



## 4.7 Swing を利用する (JFrame 編)

この章で行うこと：

swing コンポーネントである JFrame を LG3D 上で利用します。

### LG3D 上で Swing を利用する方法

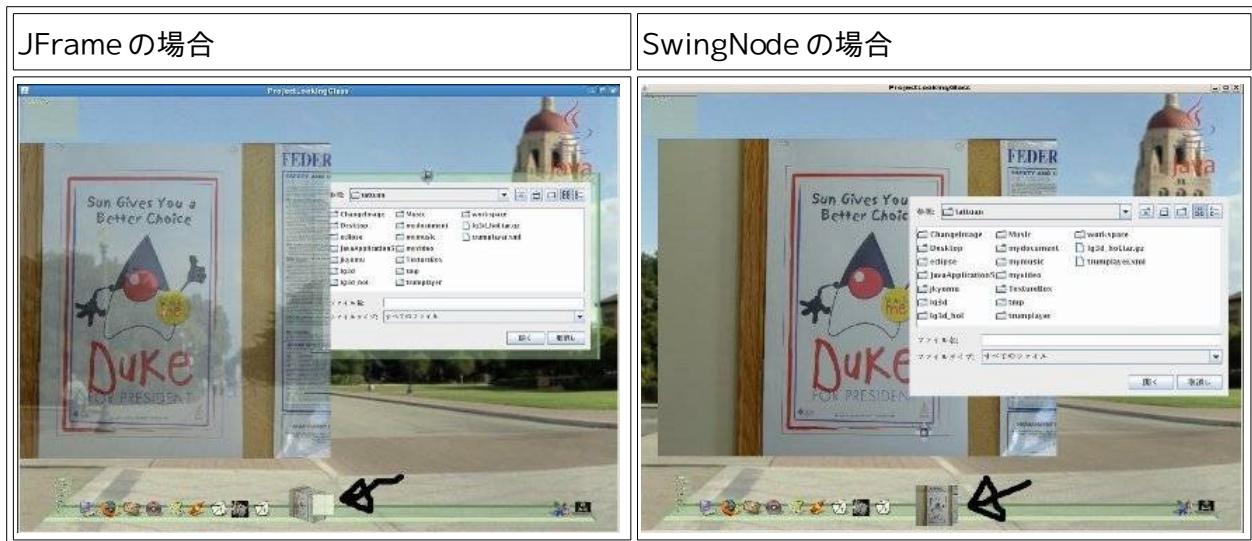
LG3D ではファイルチューラやテキストボックスのような 2D コンポーネントは用意されていません。LG3D は Swing をサポートしており、Swing を用いてこれらの機能を利用出来ます。

LG3D での Swing の使用方法は 2 種類あります。Swing の JFrame を利用する方法と、LG3D で提供されている SwingNode (org.jdesktop.lg3d.wg.SwingNode) を利用する方法があります。

前者は通常の Swing プログラミングと同様の方法でプログラミングを行うことが出来ます。表示はシーンマネージャ側で制御しており、新しいフレームとして扱われるためサムネイルが表示されます。

後者の場合、JFrame 利用時のような枠などが自動的に生成されませんが、Swing コンポーネントを LG3D コンポーネントの 1 つとして扱うことができます。

そのため、LG3D で提供されている機能を利用し、拡大・縮小・回転などを行うことができます。また、サムネイルも表示されません。



### JFrame を利用する

JFrame を利用する場合、特別なことをする必要ありません。

通常の Swing プログラムと同様に JFrame を継承した GUI を作成します。

```
public class ChooserFrame extends javax.swing.JFrame {  
    //通常の Swing プログラムの Frame 部分  
    ...  
}
```

呼び出しも

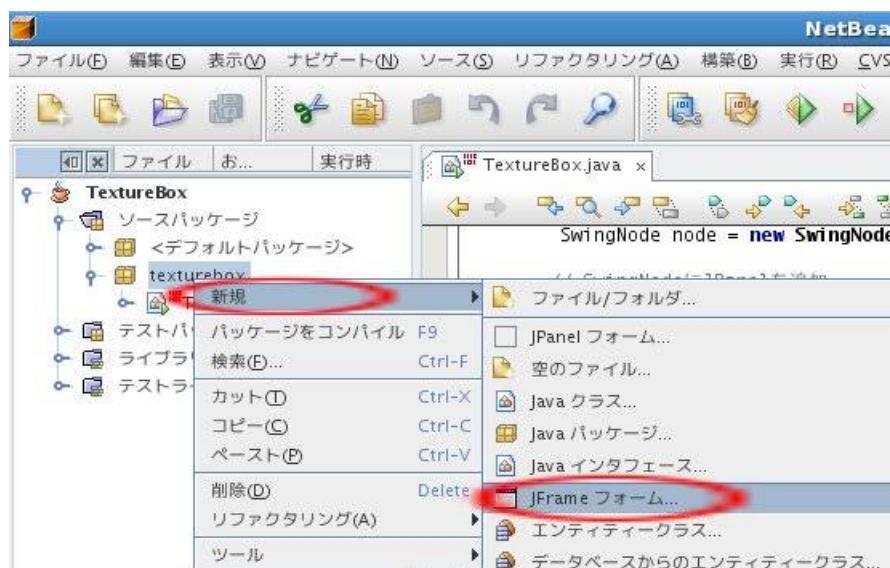
```
new ChooserFrame();
```

と通常の Swing プログラミングと同様にして呼び出せます。

1. JFrame を継承した GUI を作成する SwingFrame.java を作成します。

NetBeans では、Swing の作成用の GUI エディタを標準装備しているので、これを利用して作成します。（NetBeans 以外の環境を利用している場合は後述のソースコードを参照してください。）

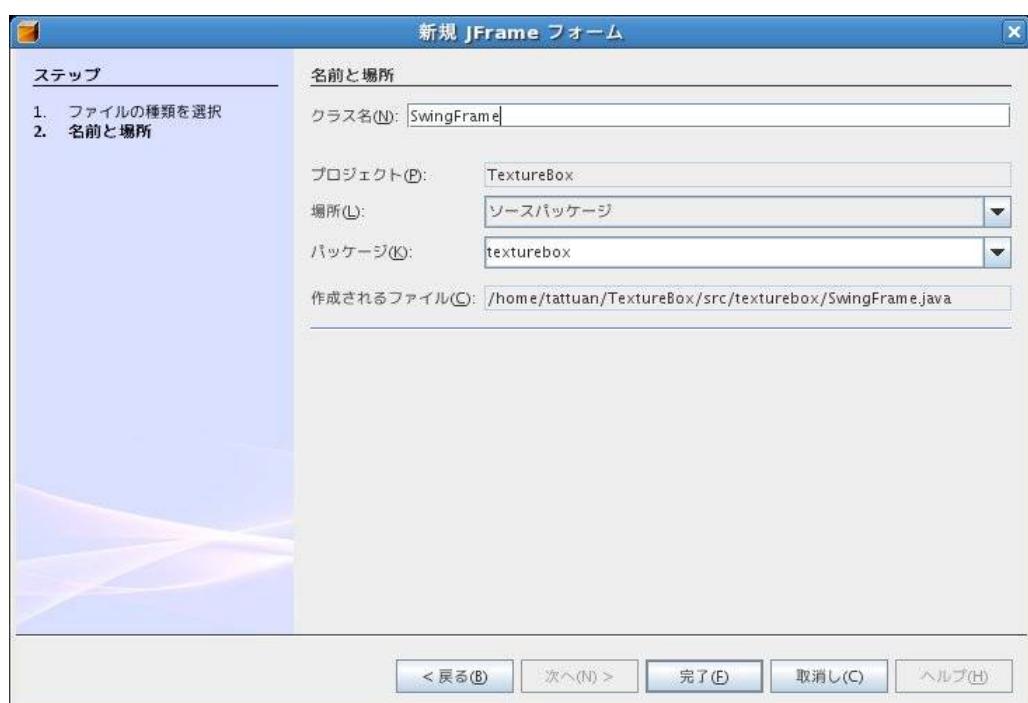
左側のパネルが「**プロジェクト**」となっていることを確認し、TextureBox プロジェクト内の「ソースパッケージ」→「texturebox」を選択します。選択後、右クリックメニューが表示されるので「**新規**」→「**JFrame フォーム**」を選択します。



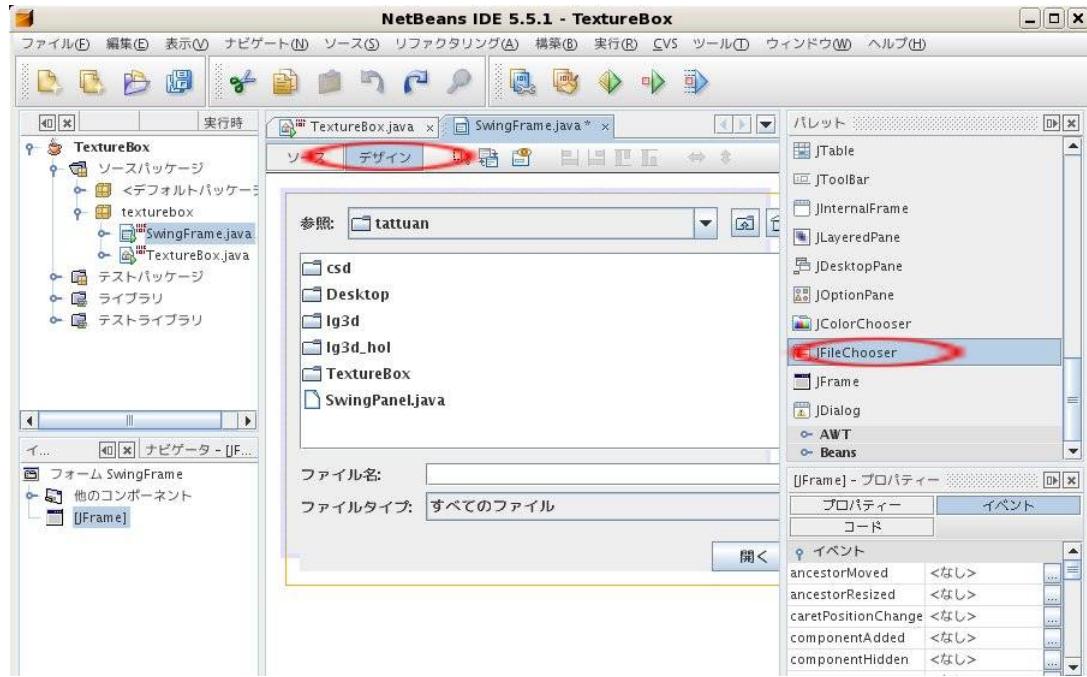
2. クラス情報を入力するためのダイアログが表示されます。

- クラス名 : **SwingFrame**

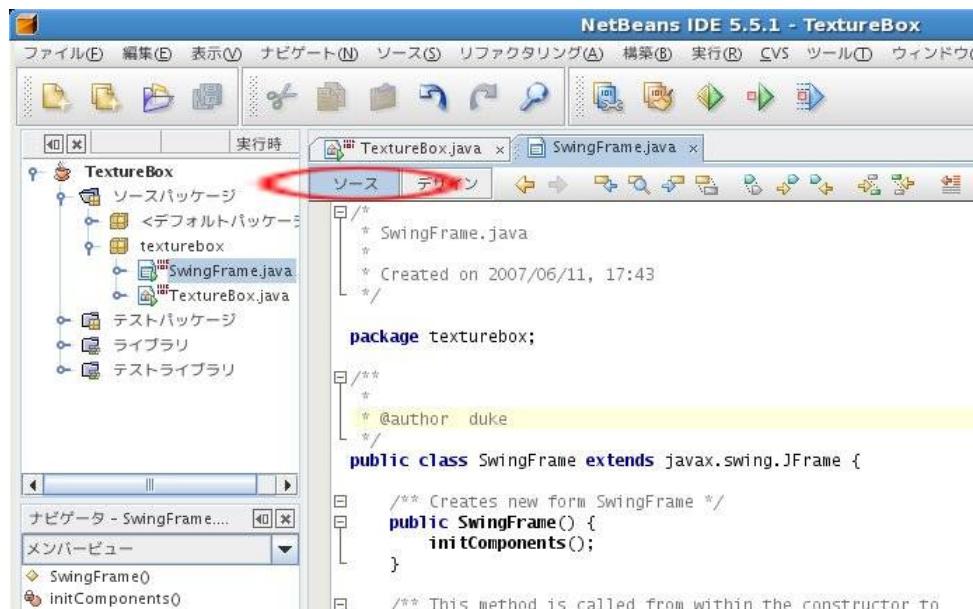
と入力し、「完了」を選択します。



3. 左側のパネルに「SwingFrame」が追加されます。  
 真ん中のエディタに「SwingFrame」が表示されます。  
 また、エディタには通常のソースコードではなく、GUIエディタが表示されます。  
 右側のパネルにパレットが表示され、Swingのコンポーネントが選択できます。  
 パレットのJFileChooserを選択し、エディタに貼り付けます。  
 コンポーネントのサイズも自由に変更できるので好みのサイズに変更してください。



真ん中のエディタの「ソース」タグを選択すると、以下のようなソースが自動で作成されているのがわかります。



```
package texturebox;

public class SwingFrame extends javax.swing.JFrame {

    /** Creates new form SwingFrame */
    public SwingFrame() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {
        jFileChooser1 = new javax.swing.JFileChooser();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        javax.swing.GroupLayout layout =
            new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(jFileChooser1,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(containerGap, javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(0, 0, 0)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addComponent(jFileChooser1,
                        javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 100, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(containerGap, javax.swing.GroupLayout.PREFERRED_SIZE,
                        javax.swing.GroupLayout.DEFAULT_SIZE,
                        javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(0, 0, 0)
        );
        pack();
    }

    // Variables declaration - do not modify
    private javax.swing.JFileChooser jFileChooser1;
    // End of variables declaration
}
```

4. TextureBox.java を以下のように変更します。

```
/*
 * TextureBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package texturebox;

/**
 *
 * @author duke
 */

import java.io.IOException;
import java.net.URL;
import java.lang.ClassLoader;
import javax.vecmath.Vector3f;
import org.jdesktop.j3d_sg.Appearance;
import org.jdesktop.j3d_sg.Shape3D;
import org.jdesktop.j3d_utils.shape.Box;
import org.jdesktop.j3d_utils.shape.SimpleAppearance;
import org.jdesktop.j3d_wg.Component3D;
import org.jdesktop.j3d_wg.Frame3D;

public class TextureBox {

    /** Creates a new instance of TextureBox */
    public TextureBox() {

        // フレームの生成
        Frame3D frame = new Frame3D();

        // シェイプの生成および設定
        // ボックス 6 面の画像を指定した Appearance の設定
        Appearance appearanceFront = null;
        Appearance appearanceRight = null;
        Appearance appearanceLeft = null;
        Appearance appearanceTop = null;
        Appearance appearanceBottom = null;
        Appearance appearanceBack = null;
        URL url = null;
        ClassLoader loader = this.getClass().getClassLoader();
    }
}
```

```

// SimpleAppearance オブジェクトを作成する際に、
// テクスチャファイルの読み込みに失敗した場合 IOException
// を発生させますので、それに対応するために Exception 処理を追加
try {
    url = loader.getResource("photo01.jpg");
    appearanceFront = new SimpleAppearance( url );
    // 前面のアピアランスを切替可能にする
    appearanceFront.setCapability(Appearance.ALLOW_TEXTURE_WRITE);
    url = loader.getResource("photo02.jpg");
    appearanceRight = new SimpleAppearance( url );
    url = loader.getResource("photo03.jpg");
    appearanceLeft = new SimpleAppearance( url );
    url = loader.getResource("photo04.jpg");
    appearanceTop = new SimpleAppearance( url );
    url = loader.getResource("photo05.jpg");
    appearanceBottom = new SimpleAppearance( url );
    url = loader.getResource("photo06.jpg");
    appearanceBack = new SimpleAppearance( url );
} catch (IOException ex) {
    System.err.println("画像ファイルの読み込みに失敗しました。");
    ex.printStackTrace();
}
// Box の生成
// 引数は、幅(x)、高さ(y)、奥行き(z)、フラグ、アピアランス
// Box.GENERATE_TEXTURE_COORDS は
// Box オブジェクトで テクスチャを利用するためには必要なフラグです
// 6 面の Appearance を貼り付けるため、
// とりあえず Box の Appearance を null にします。
Box box = new Box(0.05f, 0.04f, 0.03f,
                  Box.GENERATE_TEXTURE_COORDS, null);

// 正面
Shape3D shape = box.getShape(Box.FRONT);
shape.setAppearance(appearanceFront);

// 右側
shape = box.getShape(Box.RIGHT);
shape.setAppearance(appearanceRight);

// 左側
shape = box.getShape(Box.LEFT);
shape.setAppearance(appearanceLeft);

// 上
shape = box.getShape(Box.TOP);

```

```
shape.setAppearance(appearanceTop);

// 下(底面)
shape = box.getShape(Box.BOTTOM);
shape.setAppearance(appearanceBottom);

// 後
shape = box.getShape(Box.BACK);
shape.setAppearance(appearanceBack);

// コンポーネントの生成
Component3D component = new Component3D();

// シエイプをコンポーネントに追加
component.addChild(box);

// コンポーネントをコンテナに追加
// ここではコンテナを使用していないので、
// 直接フレームに追加
frame.addChild(component);

// フレームの表示
// フレームの大きさを設定
frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));

// フレームの表示
frame.changeEnabled(true);
frame.changeVisible(true);

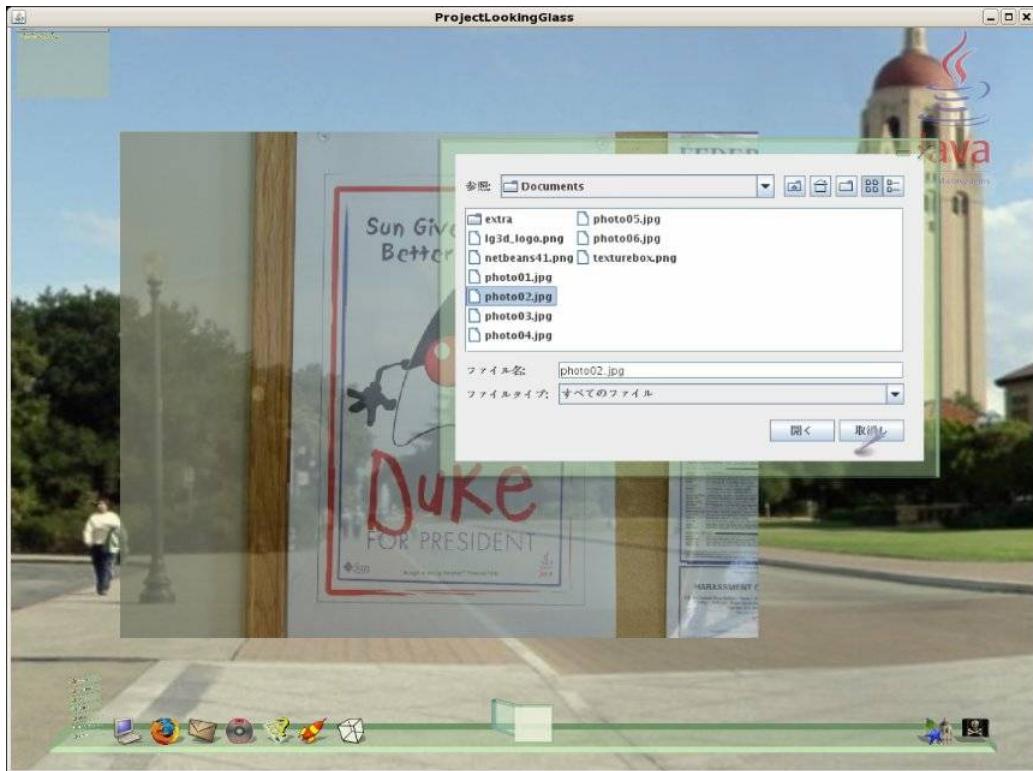
//JFrame呼び出し
new SwingFrame().setVisible(true);
}

<*/>
* @param args the command line arguments
*/
public static void main(String[] args) {
    // TODO code application logic here
    new TextureBox();

}

}
```

5. 「4.3 Jarベースのアプリケーションの作成と実行」にある [「JARベースのアプリケーションの作成」](#) の [「ステップ8」](#) を参照し、プロジェクトを構築します。  
同ページにある [「JARベースのアプリケーションの実行」](#) を参照し、アプリケーションの実行を行います。  
実行画面は下図のようになります。



## 4.8 Swing を利用する (SwingNode 編)

### この章で行うこと :

SwingNode を利用して、swing コンポーネントである JFileChooser を利用し、ボックスの前面の画像を変更できるようにします。また、SwingNode を利用した JFileChooser に 4.7 で行った JFrame を利用した場合に表示されるフレームと似たものを表示させます。

### SwingNode に JPanel を追加して利用する

SwingNode は Component3D を継承したクラスです。setPanel() メソッドを使って、SwingNode に JPanel を追加することができます。

1. JPanel を継承したクラス SwingPanel.java を以下のように作成します。

左側のパネルが「プロジェクト」となっていることを確認し、TextureBox プロジェクト内の「ソースパッケージ」→「texturebox」を選択します。

選択後、右クリックでメニューが表示されるので「新規」→「JPanel フォーム」を選択します。



後は「[4.7 Swing を利用する \(JFrame 編\)](#)」の 2、3 の同様の手順で作成すると以下のようないソースコードが自動で作成されます。

```
/*
 * SwingPanel.java
 *
 * Created on 2007/06/11, 15:49
 */

package texturebox;
/**
 *
 * @author duke
 */
public class SwingPanel extends javax.swing.JPanel {

    /**
     * Creates new form SwingPanel
     */
}
```

```

public SwingPanel() {
    initComponents();
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
//
private void initComponents() {
    jFileChooser1 = new javax.swing.JFileChooser();

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
    this.setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jFileChooser1, javax.swing.GroupLayout.PREFERRED_SIZE, 396,
                               javax.swing.GroupLayout.PREFERRED_SIZE)
                );
            )
        );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jFileChooser1, javax.swing.GroupLayout.PREFERRED_SIZE, 214,
                           javax.swing.GroupLayout.PREFERRED_SIZE)
        );
}
// Variables declaration - do not modify
private javax.swing.JFileChooser jFileChooser1;
// End of variables declaration
}

```

## 2. TextureBox.java を以下のように変更します。

SwingNode オブジェクトを作成します。作成したオブジェクトに SwingNode.setPanel() メソッドを使用して JPanel を追加します。

```

/*
 * TextureBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */

```

```
package texturebox;

/**
 *
 * @author duke
 */

import java.io.IOException;
import java.net.URL;
import java.lang.ClassLoader;
import javax.vecmath.Vector3f;
import org.jdesktop.lg3d.sg.Appearance;
import org.jdesktop.lg3d.sg.Shape3D;
import org.jdesktop.lg3d.utils.shape.Box;
import org.jdesktop.lg3d.utils.shape.SimpleAppearance;
import org.jdesktop.lg3d.wg.Component3D;
import org.jdesktop.lg3d.wg.Frame3D;
import org.jdesktop.lg3d.wg.SwingNode;

public class TextureBox {

    /** Creates a new instance of TextureBox */
    public TextureBox() {

        // フレームの生成
        Frame3D frame = new Frame3D();

        // シェイプの生成および設定
        // ボックス 6 面の画像を指定した Appearance の設定
        Appearance appearanceFront = null;
        Appearance appearanceRight = null;
        Appearance appearanceLeft = null;
        Appearance appearanceTop = null;
        Appearance appearanceBottom = null;
        Appearance appearanceBack = null;
        URL url = null;
        ClassLoader loader = this.getClass().getClassLoader();

        // SimpleAppearance オブジェクトを作成する際に、
        // テクスチャファイルの読み込みに失敗した場合 IOException
        // を発生させますので、それに対応するために Exception 処理を追加
        try {
            url = loader.getResource("photo01.jpg");
            appearanceFront = new SimpleAppearance( url );
            url = loader.getResource("photo02.jpg");
            appearanceRight = new SimpleAppearance( url );
            url = loader.getResource("photo03.jpg");
            appearanceLeft = new SimpleAppearance( url );
        }
    }
}
```

```
url = loader.getResource("photo04.jpg");
appearanceTop = new SimpleAppearance( url );
url = loader.getResource("photo05.jpg");
appearanceBottom = new SimpleAppearance( url );
url = loader.getResource("photo06.jpg");
appearanceBack = new SimpleAppearance( url );
} catch (IOException ex) {
    System.err.println("画像ファイルの読み込みに失敗しました。");
    ex.printStackTrace();
}
// Box の生成
// 引数は、幅(x)、高さ(y)、奥行き(z)、フラグ、アピアランス
// Box.GENERATE_TEXTURE_COORDS は
// Box オブジェクトで テクスチャを利用するため必要なフラグです
// 6 面の Appearance を貼り付けるため、とりあえず Box の
// Appearance を null にします。
Box box = new Box(0.05f, 0.04f, 0.03f,
    Box.GENERATE_TEXTURE_COORDS, null);

// 正面
Shape3D shape = box.getShape(Box.FRONT);
shape.setAppearance(appearanceFront);

// 右側
shape = box.getShape(Box.RIGHT);
shape.setAppearance(appearanceRight);

// 左側
shape = box.getShape(Box.LEFT);
shape.setAppearance(appearanceLeft);

// 上
shape = box.getShape(Box.TOP);
shape.setAppearance(appearanceTop);

// 下(底面)
shape = box.getShape(Box.BOTTOM);
shape.setAppearance(appearanceBottom);

// 後
shape = box.getShape(Box.BACK);
shape.setAppearance(appearanceBack);

// コンポーネントの生成
Component3D component = new Component3D();
```

```
// シェイプをコンポーネントに追加
component.addChild(box);

// SwingNode の作成
SwingNode node = new SwingNode();

// SwingNode に JPanel を追加
node.setPanel(new SwingPanel());

// コンポーネントの表示 位置の移動
node.setTranslation(0.1f,0.08f,-0.04f);

// コンポーネントをコンテナに追加
// ここではコンテナを使用していないので、
// 直接フレームに追加
frame.addChild(component);
frame.addChild(node);

// フレームの表示
// フレームの大きさを設定
frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));

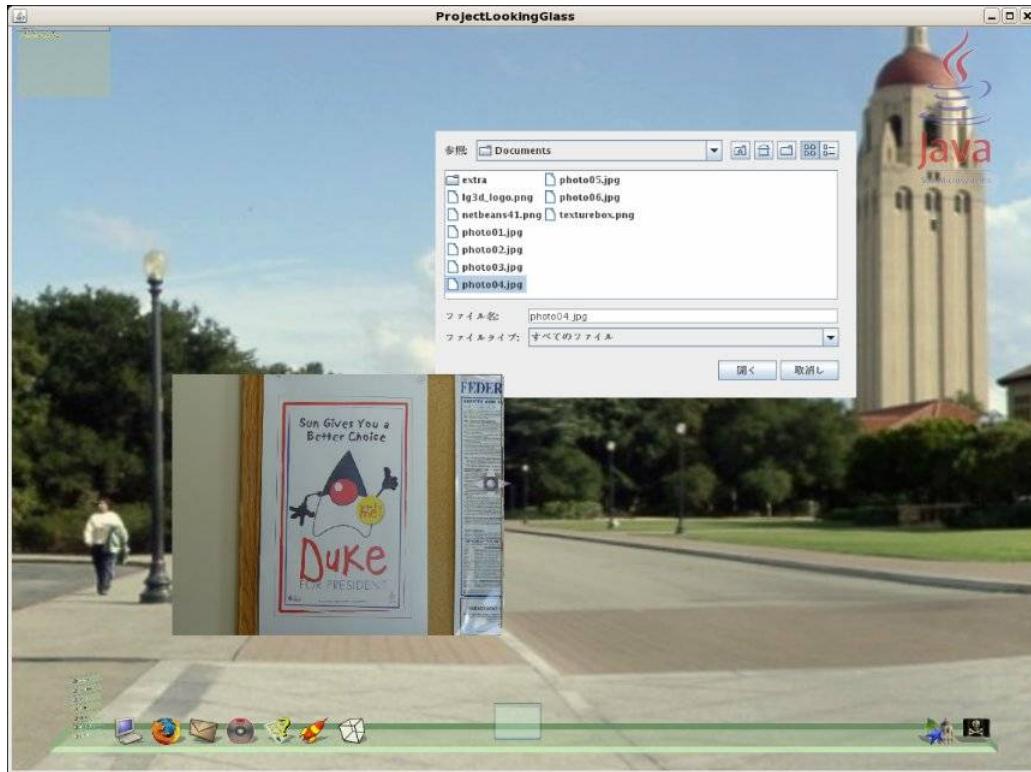
// フレームの表示
frame.changeEnabled(true);
frame.changeVisible(true);

}

/*
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    new TextureBox();
}

}
```

3. 「4.3 Jar ベースのアプリ ケーションの 作成と実行」 にある [「JARベースのアプリケーションの作成」 の ステップ8](#) を参照し、プロジェクトを構築します。  
同ページにある [「JARベースのアプリケーションの実行」](#) を参照し、アプリケーションの実行を行います。  
実行画面は下図のようになります。



## SwingNode にフレームをつける

JFrame を利用して Swing コンポーネントを作成した場合、自動的にフレームが表示されますが、 SwingNode を利用した場合、フレームは表示されません。  
SwingNode にフレームと同じようなものを表示させたい場合、Shape を利用して別途作成する必要があります。  
SwingNode の表示サイズは LG3D を実行する環境、設定により異なります。  
そのため、 SwingNode の LG3D 上の表示サイズを取得し、それを元に Shape の表示サイズを決定します。 SwingNode の表示サイズの取得には SwingNode のメソッド `getLocalWidth()`、`getLocalHeight()` を利用します。

## GlassyPanel の作成

フレームとして表示させるシェイプとして、 `org.jdesktop.lg3d.utils.shape.GlassyPanel` クラスを利用します。 GlassyPanel は Box 同様に x,y,z 座標（すべて float）とアピアランスを必要とします。 座標の値を設定する時に、 `getLocalWidth()`,`getLocalHeight()` を利用します。

1. **TextureBox.java** TextureBox.java を以下のように書き換えます。  
赤(太字)の部分が元のプログラムから変更・追加したところです。

```
/*
 * TextureBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */

package texturebox;

/**
 *
 * @author duke
 */

import java.io.IOException;
import java.net.URL;
import java.lang.ClassLoader;
import javax.vecmath.Vector3f;
import org.jdesktop.j3d.lg3d.sg.Appearance;
import org.jdesktop.j3d.lg3d.sg.Shape3D;
import org.jdesktop.j3d.lg3d.utils.shape.Box;
import org.jdesktop.j3d.lg3d.utils.shape.GlassyPanel;
import org.jdesktop.j3d.utils.shape.SimpleAppearance;
import org.jdesktop.j3d.wg.Component3D;
import org.jdesktop.j3d.wg.Frame3D;
import org.jdesktop.j3d.wg.SwingNode;

public class TextureBox {

    /** Creates a new instance of TextureBox */
    public TextureBox() {

        // フレームの生成
        Frame3D frame = new Frame3D();

        // シェイプの生成および設定
        // ボックス 6 面の画像を指定した Appearance の設定
        Appearance appearanceFront = null;
        Appearance appearanceRight = null;
        Appearance appearanceLeft = null;
        Appearance appearanceTop = null;
        Appearance appearanceBottom = null;
        Appearance appearanceBack = null;
        URL url = null;
    }
}
```

```
ClassLoader loader = this.getClass().getClassLoader();

// SimpleAppearance オブジェクトを作成する際に、
// テクスチャファイルの読み込みに失敗した場合 IOException
// を発生させますので、それに対応するために Exception 処理を追加
try {
    url = loader.getResource("photo01.jpg");
    appearanceFront = new SimpleAppearance( url );
    url = loader.getResource("photo02.jpg");
    appearanceRight = new SimpleAppearance( url );
    url = loader.getResource("photo03.jpg");
    appearanceLeft = new SimpleAppearance( url );
    url = loader.getResource("photo04.jpg");
    appearanceTop = new SimpleAppearance( url );
    url = loader.getResource("photo05.jpg");
    appearanceBottom = new SimpleAppearance( url );
    url = loader.getResource("photo06.jpg");
    appearanceBack = new SimpleAppearance( url );
} catch (IOException ex) {
    System.err.println("画像ファイルの読み込みに失敗しました。");
    ex.printStackTrace();
}
// Box の生成
// 引数は、幅(x)、高さ(y)、奥行き(z)、フラグ、アピアランス
// Box.GENERATE_TEXTURE_COORDS は
// Box オブジェクトで テクスチャを利用するため必要なフラグです
// 6 面の Appearance を貼り付けるため、とりあえず Box の
// Appearance を null にします。
Box box = new Box(0.05f, 0.04f, 0.03f,
                  Box.GENERATE_TEXTURE_COORDS, null);

// 正面
Shape3D shape = box.getShape(Box.FRONT);
shape.setAppearance(appearanceFront);

// 右側
shape = box.getShape(Box.RIGHT);
shape.setAppearance(appearanceRight);

// 左側
shape = box.getShape(Box.LEFT);
shape.setAppearance(appearanceLeft);

// 上
shape = box.getShape(Box.TOP);
shape.setAppearance(appearanceTop);
```

```

// 下(底面)
shape = box.getShape(Box.BOTTOM);
shape.setAppearance(appearanceBottom);

// 後
shape = box.getShape(Box.BACK);
shape.setAppearance(appearanceBack);

// コンポーネントの生成
Component3D component = new Component3D();

// シェイプをコンポーネントに追加
component.addChild(box);

// SwingNode の作成
SwingNode node = new SwingNode();

// SwingNode に JPanel を追加
node.setPanel(new SwingPanel());

// GlassPanel のアピア ランスの作成
Appearance glassyAppearance =
    new SimpleAppearance(0.6f,1.0f,0.6f,1.0f);

// GlassPanel の作成
// x,y 座標は SwingNode の LG3D 上のサイズ + 0 .05f を指定
GlassyPanel glassyPanel = new GlassyPanel(node.getLocalWidth()
    +0.005f,node.getLocalHeight() +0.005f,0.005f,glassyAppearance);

// GlassPanel を SwingNode に追加
node.addChild(glassyPanel);

// コンポーネントの表示位置の移動
node.setTranslation(0.1f,0.08f,0.04f);

// コンポーネントをコンテナに追加
// ここではコンテナを使用していないので、
// 直接フレームに追加
frame.addChild(component);
frame.addChild(node);

// フレームの表示
// フレームの大きさを設定
frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));

// フレームの表示
frame.changeEnabled(true);

```

```

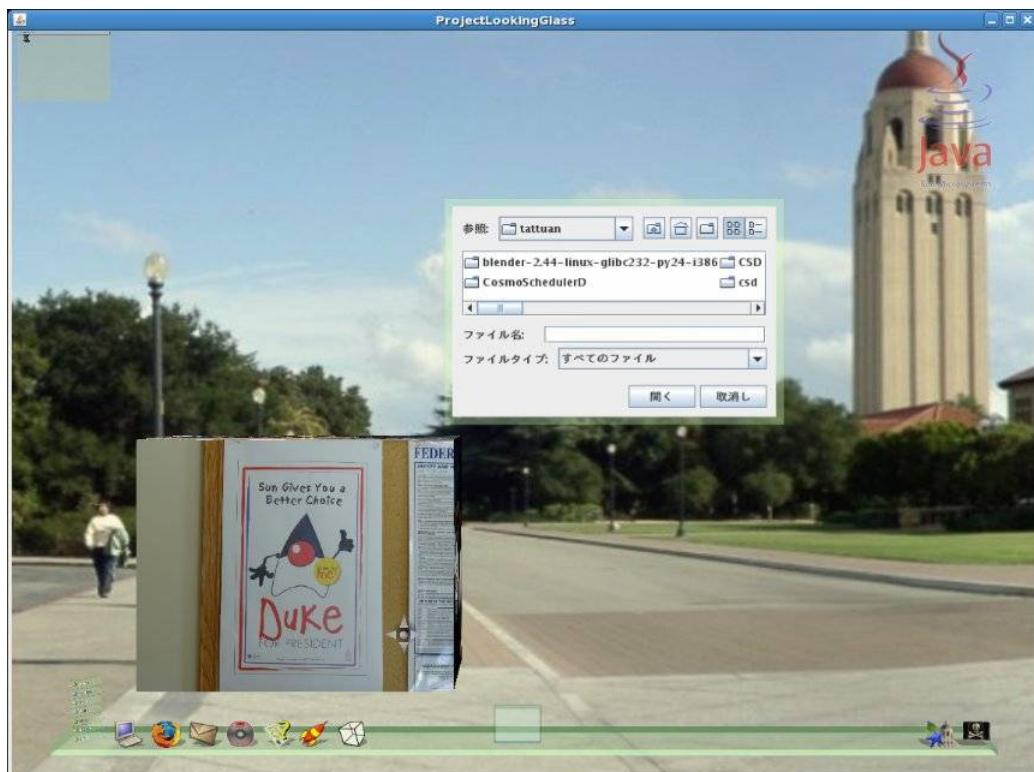
        frame.setVisible(true);

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        new TextureBox();
    }
}

```

2. 「4.3 Jar ベースのアプリ ケーションの 作成と実行」 にある [「JARベースのアプリケーションの作成」 の ステップ8](#) を参照し、プロジェクトを構築します。同ページにある [「JARベースのアプリケーションの実行」](#) を参照し、アプリケーションの実行を行います。  
実行画面は下図のようになります。JFileChooser の回りに半透明の緑のシェイプが表示されています。



## ボックスの画像を変更するアクションを実装する

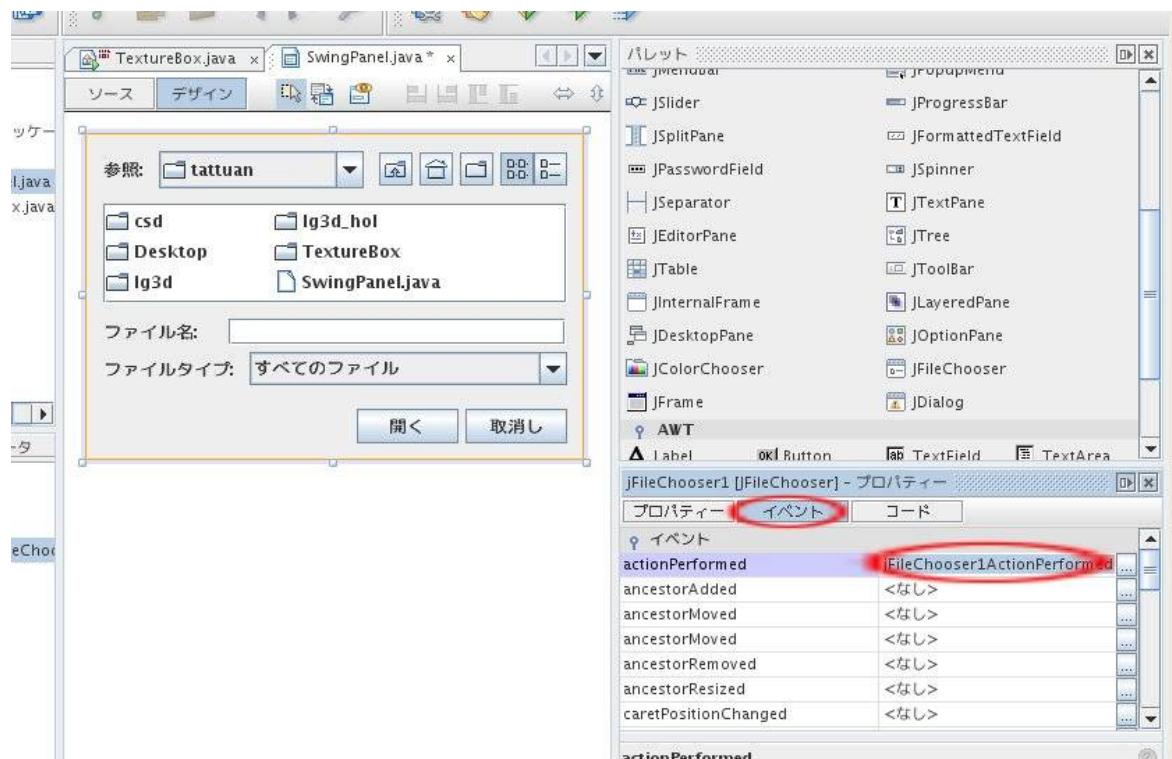
前項まででファイルチューザの表示部分を作成しました。この項ではファイルチューザのアクションを実装します。TextureBox の前面をファイルチューザで選択した画像に変更する処理を作成します。

- ・ テクスチャの切り替えを可能にする
- ・ テクスチャの作成
- ・ 作成したテクスチャをアピアランスに設定する

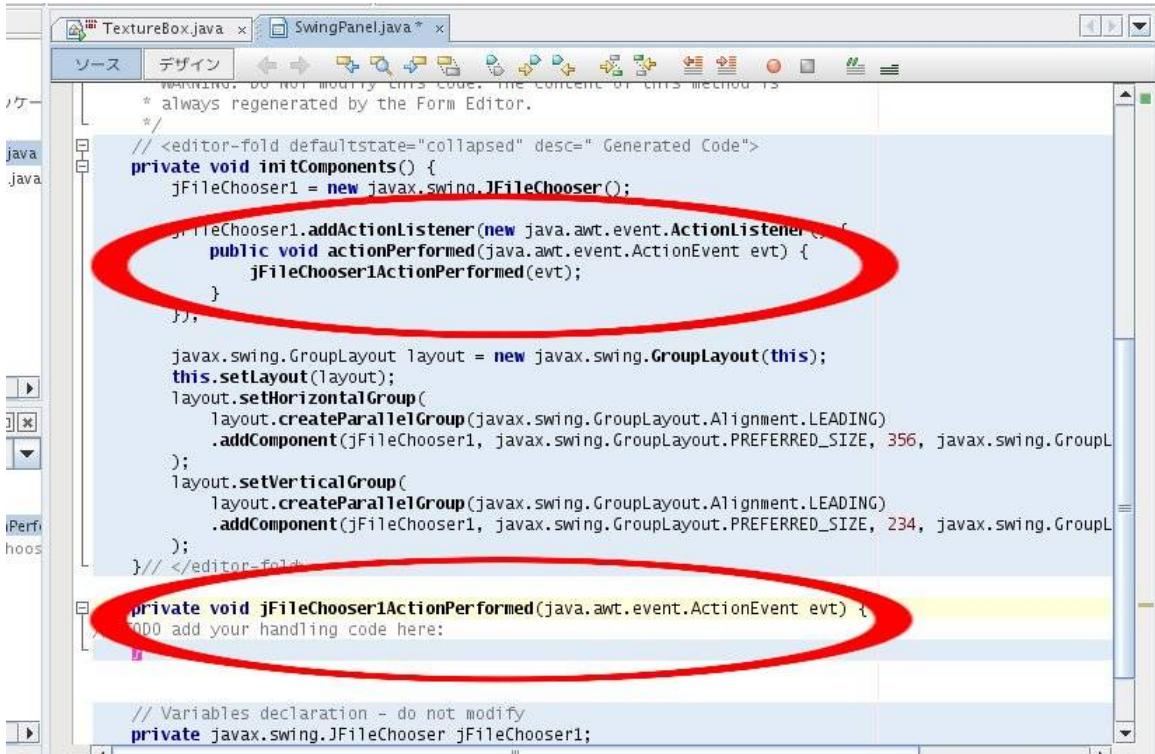
ファイルチューザで指定したファイルからテクスチャを作成する必要があります。テクスチャの保持には `org.jdesktop.lg3d.Texture` クラスを利用します。イメージファイルのロードには `javax.imageio.ImageIO` クラスの `read()` メソッドを、ロードしたイメージからテクスチャを作成するには `org.jdesktop.lg3d.sg.utils.image.TextureLoader` クラスを利用します。`TextureLoader.getTexture()` メソッドでテクスチャを取得することができます。作成したテクスチャをアピアランスに設定するには、`Appearance.setTexture()` メソッドを使用します。

1. Swing のアクションイベントを作成します。

NetBeans の GUI エディタを利用すると簡単に作成できます。SwingPanel.java の編集画面にすると、右側のパネルに **JFileChooser のプロパティ** がでているのがわかると思います。プロパティの「**イベント**」を選択し、「**actionPerformed**」の右側をクリックしアクションの名前を入力してください。



名前を入力すると以下のように自動でソースコードが作成されます。



```
/* WARNING: Do not modify this code. The content of this method is always regenerated by the Form Editor.
 */
// <editor-fold defaultstate="collapsed" desc=" Generated Code">
private void initComponents() {
    jFileChooser1 = new javax.swing.JFileChooser();

    jFileChooser1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jFileChooser1ActionPerformed(evt);
        }
    });
    ...

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
    this.setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jFileChooser1, javax.swing.GroupLayout.PREFERRED_SIZE, 356, javax.swing.GroupLayout.PREFERRED_SIZE)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jFileChooser1, javax.swing.GroupLayout.PREFERRED_SIZE, 234, javax.swing.GroupLayout.PREFERRED_SIZE)
    );
} // </editor-fold>

private void jFileChooser1ActionPerformed(java.awt.event.ActionEvent evt) {
    TODO add your handling code here:
}

// Variables declaration - do not modify
private javax.swing.JFileChooser jFileChooser1;
```

2. SwingPanel.java を以下のように変更します。変更箇所は赤字（太字）で示します。  
テクスチャを設定するアピアランスを引数として与えます。開くボタンを押した際のアクションを作成します。アクションコマンドとして ApproveSelection が送られた場合、選択されたファイルからイメージをロードしテクスチャを作成します。作成したテクスチャをアピアランスに設定します。

```
/*
 * SwingPanel.java
 *
 * Created on 2007/06/11, 15:49
 */

package texturebox;

import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
import org.jdesktop.lg3d.sg.Appearance;
import org.jdesktop.lg3d.sg.Texture;
import org.jdesktop.lg3d.sg.utils.image.TextureLoader;

/**
 *
 * @author duke
 */
public class SwingPanel extends javax.swing.JPanel {
```

```

/** Creates new form SwingPanel */
private Appearance target;

public SwingPanel(Appearance target) {

    if(target==null) {
        throw new IllegalArgumentException
            ("target オブジェクトが指定されていません ");
    }
    this.target = target;
    initComponents();
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
//
private void initComponents() {
    jFileChooser1 = new javax.swing.JFileChooser();

    jFileChooser1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jFileChooser1ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
    this.setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jFileChooser1,
                    javax.swing.GroupLayout.PREFERRED_SIZE, 396,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
            )
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jFileChooser1,
                javax.swing.GroupLayout.PREFERRED_SIZE, 214,
                javax.swing.GroupLayout.PREFERRED_SIZE)
    );
}

private void jFileChooser1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    // 開くボタンが押された場合の処理
}

```

```

if(evt.getActionCommand().trim().equals("ApproveSelection")){
    try {
        // 選択したファイルの取得
        File file = this.jFileChooser1.getSelectedFile();
        // ファイルからイメージをロード
        BufferedImage image = null;
        image = ImageIO.read(file);
        // イメージからテクスチャを作成
        Texture texture = new TextureLoader(image).getTexture();
        // テクスチャをアピアランスに設定
        target.setTexture(texture);
    } catch(java.io.IOException e){
    }
}

// Variables declaration - do not modify
private javax.swing.JFileChooser jFileChooser1;
// End of variables declaration

}

```

3. TextureBox.java を以下のように変更します。

変更箇所は赤字（太字）で示します。

Appearance オブジェクトは、デフォルトではテクスチャの切り替えができません。

テクスチャを切り替えられるようにするには **setCapability()** メソッドを引数

**Appearance.ALLOW\_TEXTURE\_WRITE** を指定してコールします。

今回は、ボックスの前面の画像を指定するので、**appearanceFront** を切り替えられますようにします。

```

/*
 * TextureBox.java
 *
 * Created on 2005/09/29, 15:42
 *
 * To change this template, choose Tools | Options and locate the template under
 * the Source Creation and Management node. Right-click the template and choose
 * Open. You can then make changes to the template in the Source Editor.
 */
package texturebox;

/**
 *
 * @author duke
 */

```

```
import java.io.IOException;
import java.net.URL;
import java.lang.ClassLoader;
import javax.vecmath.Vector3f;
import org.jdesktop.desktop.lg3d.sg.Appearance;
import org.jdesktop.desktop.lg3d.sg.Shape3D;
import org.jdesktop.desktop.lg3d.utils.shape.Box;
import org.jdesktop.desktop.lg3d.utils.shape.SimpleAppearance;
import org.jdesktop.desktop.lg3d.wg.Component3D;
import org.jdesktop.desktop.lg3d.wg.Frame3D;
import org.jdesktop.desktop.lg3d.wg.SwingNode;

public class TextureBox {

    /** Creates a new instance of TextureBox */
    public TextureBox() {

        // フレームの生成
        Frame3D frame = new Frame3D();

        // シエイプの生成および設定
        // ボックス 6 面の画像を指定した Appearance の設定
        Appearance appearanceFront = null;
        Appearance appearanceRight = null;
        Appearance appearanceLeft = null;
        Appearance appearanceTop = null;
        Appearance appearanceBottom = null;
        Appearance appearanceBack = null;
        URL url = null;
        ClassLoader loader = this.getClass().getClassLoader();

        // SimpleAppearance オブジェクトを作成する際に、
        // テクスチャファイルの読み込みに失敗した場合 IOException
        // を発生させますので、それに対応するために Exception 処理を追加
        try {
            url = loader.getResource("photo01.jpg");
            appearanceFront = new SimpleAppearance( url );
            // 前面のアピアランスを 切替可能にする
            appearanceFront.setCapability(Appearance.ALLOW_TEXTURE_WRITE);
            url = loader.getResource("photo02.jpg");
            appearanceRight = new SimpleAppearance( url );
            url = loader.getResource("photo03.jpg");
            appearanceLeft = new SimpleAppearance( url );
            url = loader.getResource("photo04.jpg");
            appearanceTop = new SimpleAppearance( url );
            url = loader.getResource("photo05.jpg");
            appearanceBottom = new SimpleAppearance( url );
        }
    }
}
```

```
url = loader.getResource("photo06.jpg");
appearanceBack = new SimpleAppearance( url );
} catch (IOException ex) {
    System.err.println("画像ファイルの読み込みに失敗しました。");
    ex.printStackTrace();
}
// Box の生成
// 引数は、幅(x)、高さ(y)、奥行き(z)、フラグ、アピアランス
// Box.GENERATE_TEXTURE_COORDS は
// Box オブジェクトで テクスチャを利用するため必要なフラグです
// 6 面の Appearance を貼り付けるため、とりあえず Box の
// Appearance を null にします。
Box box = new Box(0.05f, 0.04f, 0.03f,
                  Box.GENERATE_TEXTURE_COORDS, null);

// 正面
Shape3D shape = box.getShape(Box.FRONT);
shape.setAppearance(appearanceFront);

// 右側
shape = box.getShape(Box.RIGHT);
shape.setAppearance(appearanceRight);

// 左側
shape = box.getShape(Box.LEFT);
shape.setAppearance(appearanceLeft);

// 上
shape = box.getShape(Box.TOP);
shape.setAppearance(appearanceTop);

// 下(底面)
shape = box.getShape(Box.BOTTOM);
shape.setAppearance(appearanceBottom);

// 後
shape = box.getShape(Box.BACK);
shape.setAppearance(appearanceBack);

// コンポーネントの生成
Component3D component = new Component3D();

// シェイプをコンポーネントに追加
component.addChild(box);

// SwingNode の作成
SwingNode node = new SwingNode();
```

```
// SwingNode に JPanel を追加
node.setPanel(new SwingPanel(appearanceFront));

// コンポーネントの表示位置の移動
node.setTranslation(0.1f,0.08f,-0.04f);

// コンポーネントをコンテナに追加
// ここではコンテナを使用していないので、
// 直接フレームに追加
frame.addChild(component);
frame.addChild(node);

// フレームの表示
// フレームの大きさを設定
frame.setPreferredSize(new Vector3f(0.1f, 0.08f, 0.06f));

// フレームの表示
frame.changeEnabled(true);
frame.changeVisible(true);
}

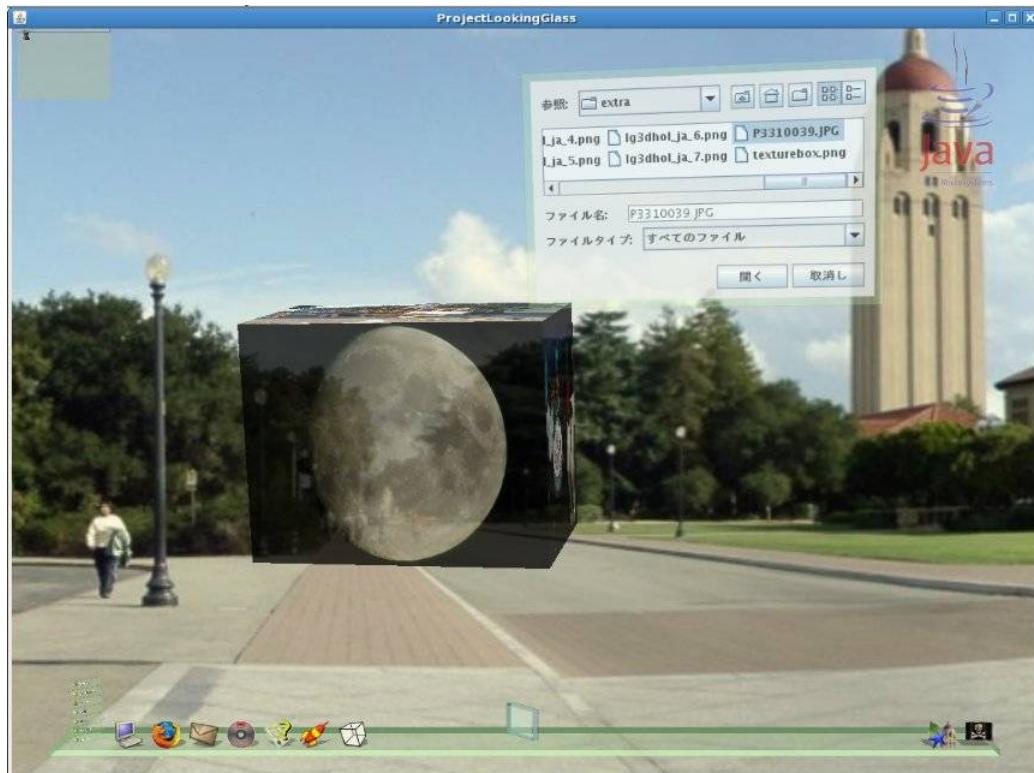
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    // TODO code application logic here
    new TextureBox();

}

}
```

4. 「4.3 Jar ベースのアプリ ケーションの 作成と実行」 にある [「JARベースのアプリケーションの作成」](#) の [「ステップ8」](#) を参照し、プロジェクトを構築します。同ページにある [「JARベースのアプリケーションの実行」](#) を参照し、アプリケーションの実行を行います。

実行画面は下図のようになります。画像ファイルを選択して「開く」ボタンを押すと、ボックスの前面の画像が切り替わります。



## 応用問題

/home/duke/Documents/extra 以下には今回のスライド用アプリケーションで使用した画像があります。 それらを利用して、以下の改良に挑戦してください。

- サムネイル上で中クリックで最小化

Thumbnail オブジェクトにイベント処理を追加します。

Frame3D.changeVisible(boolean) メソッドを利用することにより 最小化を行なうことができます。

- 最小化、終了ボタンの作成

Frame3D.changeEnabled(boolean) メソッドを利用することにより アプリケーションを終了することができます。

また、lg3d-core.jar には

- resources/images/button/window-close.png
- resources/images/button/window-minimize.png

が含まれていますので、最小化、終了ボタン作成にこれらを利用するのも良いでしょう。

- 4枚以上の画像に対応する(表示画像を切り替えていく)

「4.3 テクスチャを使う」ではボックスの全ての面に違う画像を表示させましたが、ここではボックスの上下の面は画像の表示対象からはずします。

一番お手軽なのは、画像ごとに SimpleAppearance オブジェクトを生成し、Shape3D.setAppearance(Appearance) メソッドを利用する方法です。

- ボックス以外のシェイプを使う

Box 以外のシェイプとして

- org.jdesktop.lg3d.utils.shape.ImagePanel
- org.jdesktop.lg3d.utils.shape.FuzzyEdgePanel

などがあります。 ImagePanel と FuzzyEdgePanel はテクスチャを貼るために用意されている シェイプなので、Box のように GENERATE\_TEXTURE\_COORDS を指定をする 必要はありません。

- ページ番号の表示を行う

org.jdesktop.lg3d.utils.shape.Text2D クラスを使う方法が簡単です。

Text2D は簡単な文字列を扱うためのシェイプです。

Text2D.setString(String) で文字列の変更が行えます。 文字が大きすぎる場合は Component3D オブジェクトに格納後 Component3D.setScale(float) で拡大・縮小が行えます。

# 参考文献、URL等

---

## 公式サイト

- Project Looking Glass の総本山  
<https://lg3d.dev.java.net/>
  - Project Looking Glass 日本語サイト  
<https://lg3d.dev.java.net/ja/>
- 

## ML/掲示板

- 日本語 ML : interest\_ja@lg3d.dev.java.net  
登録ページ : 登録には java.net のアカウントが必要です(無料)  
<https://lg3d.dev.java.net/servlets/ProjectMailingListList>
  - 英語 ML : interest@lg3d.dev.java.net  
登録ページ : 登録には java.net のアカウントが必要です(無料)  
<https://lg3d.dev.java.net/servlets/ProjectMailingListList>
  - 日本語掲示板 (上記の日本語 ML と連動しています)  
閲覧は誰でも可能ですが、投稿には javadesktop.org のアカウントが必要です (無料)  
<http://forums.java.net/jive/forum.jspa?forumID=81>
  - 英語掲示板 (上記の英語 ML と連動しています)  
閲覧は誰でも可能ですが、投稿には javadesktop.org のアカウントが必要です (無料)  
<http://forums.java.net/jive/forum.jspa?forumID=80>
- 

## Web サイト

- Java in the Box 内 Project Looking Glass のページ (櫻庭祐一さん)  
インストール、プログラミング、リンク、Eclipse プラグインの情報など  
<http://www.javainthebox.net/lg3d/index.html>
- 

## 雑誌

- Software Design 10月号  
Project Looking Glass 徹底攻略 川原 英哉、藤楓 泰宏、かみや たま、櫻庭 祐一 他  
<http://www.gihyo.co.jp/magazines/SD/archive/200510/>