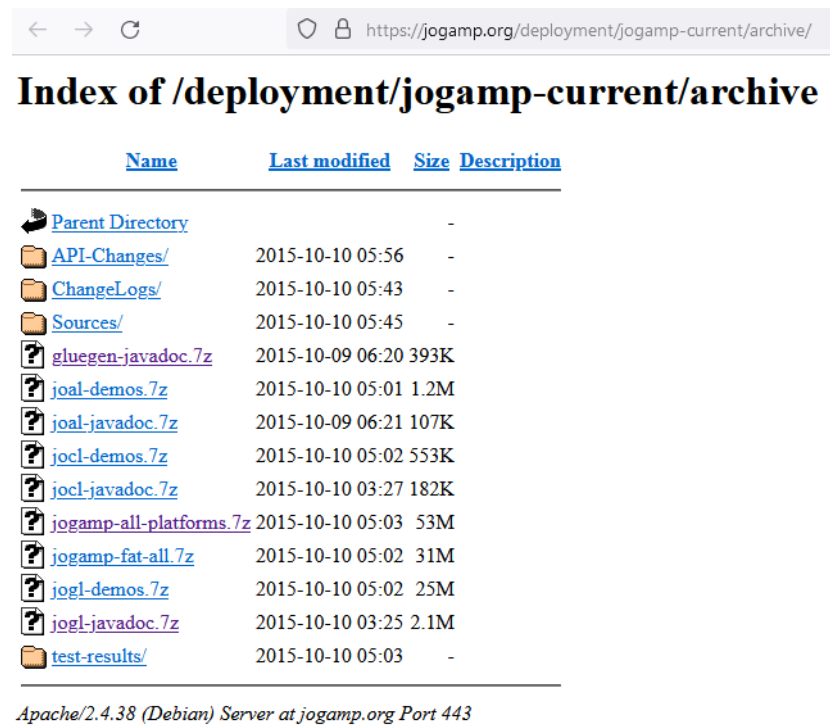


Прежде чем начать использовать JOGL, необходимо скачать и установить файлы JOGL JAR и нативные JARs или нативные библиотечные файлы. Последняя стабильная версия JOGL может быть получена по адресу <https://jogamp.org/deployment/jogamp-current/archive/>.

При переходе появится страница, показанная на рис. 1.



Name	Last modified	Size	Description
Parent Directory		-	
API-Changes/	2015-10-10 05:56	-	
ChangeLogs/	2015-10-10 05:43	-	
Sources/	2015-10-10 05:45	-	
gluegen-javadoc.7z	2015-10-09 06:20	393K	
joal-demos.7z	2015-10-10 05:01	1.2M	
joal-javadoc.7z	2015-10-09 06:21	107K	
jocl-demos.7z	2015-10-10 05:02	553K	
jocl-javadoc.7z	2015-10-10 03:27	182K	
jogamp-all-platforms.7z	2015-10-10 05:03	53M	
jogamp-fat-all.7z	2015-10-10 05:02	31M	
jogl-demos.7z	2015-10-10 05:02	25M	
jogl-javadoc.7z	2015-10-10 03:25	2.1M	
test-results/	2015-10-10 05:03	-	

Apache/2.4.38 (Debian) Server at jogamp.org Port 443

Рисунок 1. Страница загрузки стабильной версии JOGL

Скачайте и распакуйте в каком-либо каталоге файл [jogamp-all-platforms.7z](#). Содержимое этого каталога показано на рис. 2.

Имя	Дата изменения	Тип	Размер
apk	10.10.2015 6:03	Папка с файлами	
etc	10.10.2015 6:03	Папка с файлами	
jar	10.10.2015 6:03	Папка с файлами	
jnlp-files	10.10.2015 6:03	Папка с файлами	
lib	10.10.2015 6:03	Папка с файлами	
aggregated.artifact.properties.sorted	10.10.2015 6:03	Файл "SORTED"	1 КБ
all.artifact.properties.sorted	10.10.2015 6:03	Файл "SORTED"	2 КБ
gluegen.artifact.properties	09.10.2015 7:23	Файл "PROPERTIE..."	1 КБ
gluegen.LICENSE	09.10.2015 7:23	Текстовый докум...	8 КБ
gluegen-java-src	09.10.2015 7:23	Архив ZIP - WinR...	4 059 КБ
joal.artifact.properties	09.10.2015 7:35	Файл "PROPERTIE..."	1 КБ
joal.LICENSE	09.10.2015 7:35	Текстовый докум...	2 КБ
joal.README	09.10.2015 7:35	Текстовый докум...	8 КБ
joal-java-src	09.10.2015 7:35	Архив ZIP - WinR...	74 КБ
jocl.artifact.properties	10.10.2015 5:59	Файл "PROPERTIE..."	1 КБ
jocl.LICENSE	10.10.2015 5:59	Текстовый докум...	4 КБ
jocl.README	10.10.2015 5:59	Текстовый докум...	2 КБ
jocl-java-src	10.10.2015 5:59	Архив ZIP - WinR...	195 КБ
jogl.artifact.properties	10.10.2015 4:41	Файл "PROPERTIE..."	1 КБ
jogl.LICENSE	10.10.2015 4:41	Текстовый докум...	20 КБ
jogl.README	10.10.2015 4:41	Текстовый докум...	3 КБ
jogl-java-src	10.10.2015 4:41	Архив ZIP - WinR...	23 654 КБ
jogl-test-java-src	10.10.2015 4:41	Архив ZIP - WinR...	15 832 КБ

Рисунок 2. Содержимое каталога после распаковки файла jogamp-all-platforms.7z.

Перейдите в каталог jar. Часть его содержимого показана на рис. 3.

Имя	Дата изменения	Тип	Размер
atomic	10.10.2015 6:03	Папка с файлами	
gluegen	09.10.2015 7:23	Executable Jar File	845 КБ
gluegen-rt	09.10.2015 7:23	Executable Jar File	338 КБ
gluegen-rt-android	09.10.2015 7:23	Executable Jar File	342 КБ
gluegen-rt-android-natives-android-aarch64	09.10.2015 7:25	Executable Jar File	4 КБ
gluegen-rt-android-natives-android-armv6	09.10.2015 7:26	Executable Jar File	8 КБ
gluegen-rt-android-natives-linux-amd64	09.10.2015 7:23	Executable Jar File	5 КБ
gluegen-rt-android-natives-linux-armv6	09.10.2015 7:23	Executable Jar File	3 КБ
gluegen-rt-android-natives-linux-armv6hf	09.10.2015 7:23	Executable Jar File	3 КБ
gluegen-rt-android-natives-linux-i586	09.10.2015 7:22	Executable Jar File	5 КБ
gluegen-rt-android-natives-macosx-universal	09.10.2015 7:20	Executable Jar File	5 КБ
gluegen-rt-android-natives-solaris-amd64	09.10.2015 7:19	Executable Jar File	5 КБ
gluegen-rt-android-natives-solaris-i586	09.10.2015 7:19	Executable Jar File	4 КБ
gluegen-rt-android-natives-windows-amd64	09.10.2015 7:19	Executable Jar File	8 КБ
gluegen-rt-android-natives-windows-i586	09.10.2015 7:19	Executable Jar File	8 КБ
gluegen-rt-natives-android-aarch64	09.10.2015 7:25	Executable Jar File	4 КБ
gluegen-rt-natives-android-armv6	09.10.2015 7:26	Executable Jar File	8 КБ
gluegen-rt-natives-linux-amd64	09.10.2015 7:23	Executable Jar File	5 КБ
gluegen-rt-natives-linux-armv6	09.10.2015 7:23	Executable Jar File	3 КБ
gluegen-rt-natives-linux-armv6hf	09.10.2015 7:23	Executable Jar File	3 КБ

Рисунок 3. Часть содержимого каталога jar

Для каждой платформы необходимы определенные файлы.

64-bit Windows

gluegen-rt.jar
jogl-all.jar
gluegen-java-src.zip
jogl-java-src.zip
gluegen-rt-natives-
windows-amd64.jar
jogl-all-natives-
windows-amd64.jar

64-bit Linux

gluegen-rt.jar
jogl-all.jar
gluegen-java-src.zip
jogl-java-src.zip
gluegen-rt-natives-
linux-amd64.jar
jogl-all-natives-linux-
amd64.jar

32/64-bit Mac

gluegen-rt.jar
jogl-all.jar
gluegen-java-src.zip
jogl-java-src.zip
gluegen-rt-natives-macosx-
universal.jar
jogl-all-natives-macosx-
universal.jar

32-bit Windows

gluegen-rt.jar
jogl-all.jar
gluegen-java-src.zip
jogl-java-src.zip
gluegen-rt-natives-windows-i586.jar
jogl-all-natives-windows-i586.jar

32-bit Linux

gluegen-rt.jar
jogl-all.jar
gluegen-java-src.zip
jogl-java-src.zip
gluegen-rt-natives-linux-i586.jar
jogl-all-natives-linux-i586.jar

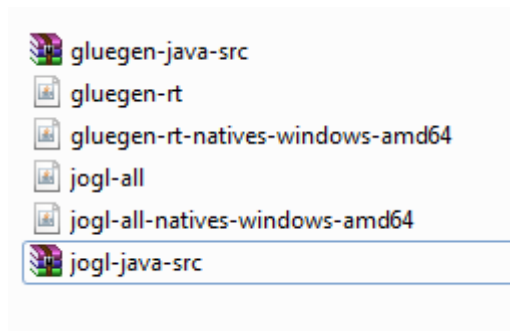
Создание проекта JOGL в Eclipse

Самый простой путь создания проекта с использованием JOGL состоит в подключении необходимых jar-файлов к проекту с использованием в дальнейшем зависимости от этого проекта.

Создание проекта JOGL

- Поместите все JAR с кодом, нативные JARs для всех платформ, которые должно поддерживать ваше приложение, и ZIP – файлы с исходными кодами в каталог JOGL *в каталоге вашего рабочего пространства Eclipse*. Если это платформа **Windows 64-bit**, то выберите следующие файлы:
 - gluegen-rt.jar,
 - jogl-all.jar,
 - gluegen-java-src.zip,
 - jogl-java-src.zip,
 - gluegen-rt-natives-windows-amd64.jar,

- jogl-all-natives-windows-amd64.jar.



- В Eclipse выберите File > New > Java Project. Укажите имя проекта «JOGL» и нажмите кнопку «Next» (рис. 4).

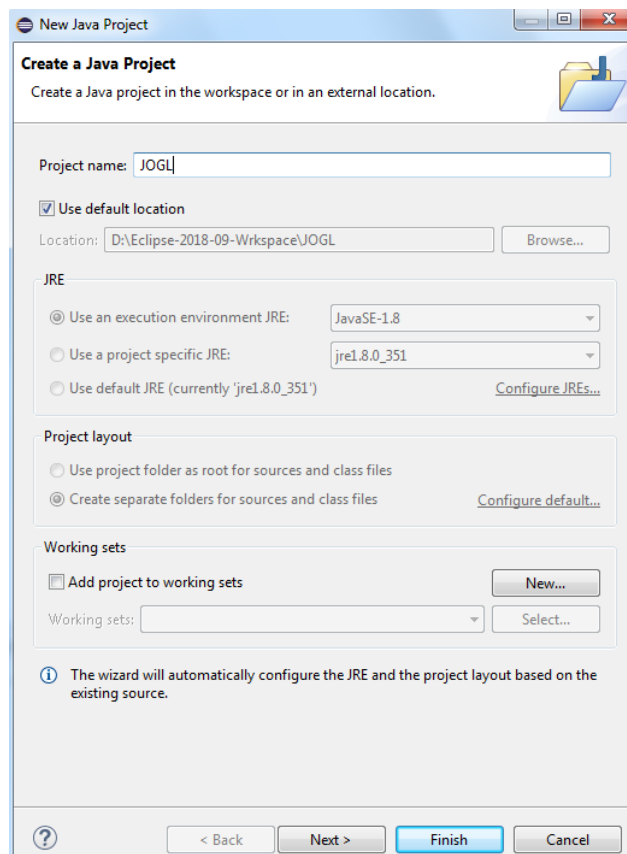


Рисунок 4. Создание проекта с именем «JOGL»

- Удалите все нативные jar из проекта, оставив только gluegen-rt.jar и jogl-all.jar (рис. 5).

Файлы gluegen-rt-natives-*-*.jar и jogl-all-natives-*-*.jar не требуются в пути к классам JOGL, но требуются Eclipse для экспорта проекта в виде файла JAR, который можно запускать.

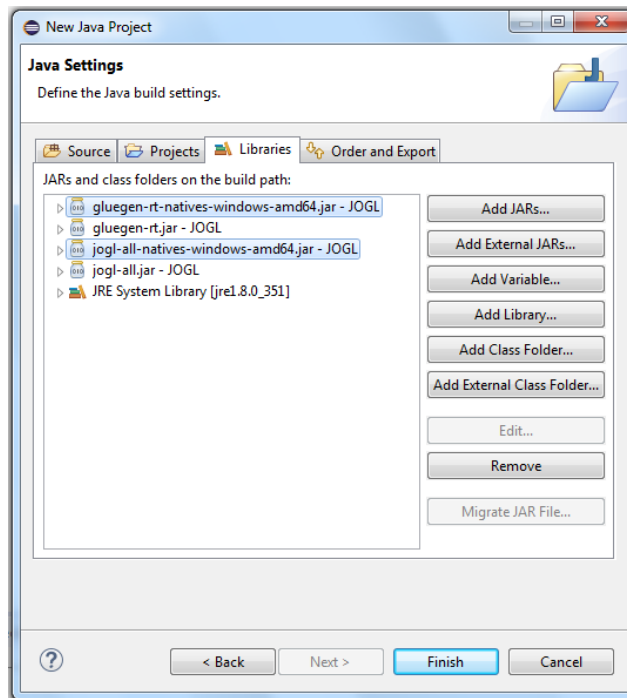


Рисунок 5. Удаление нативных JAR

- Разверните JAR – файлы jogl-all и gluegen-rt. Выберите "Source attachment" (прикрепить источник), укажите соответствующий файл с именем *java-src.zip в проекте (рис. 6).

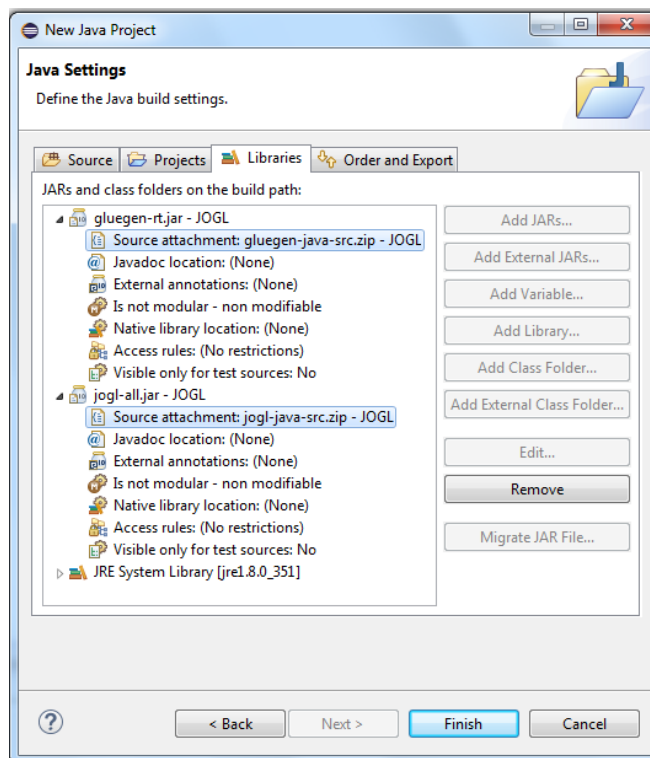


Рисунок 6. Прикрепление файлов с исходным кодом

Выберите закладку "Order and Export". Выберите оба JAR и нажмите кнопку "Finish" (рис. 7).

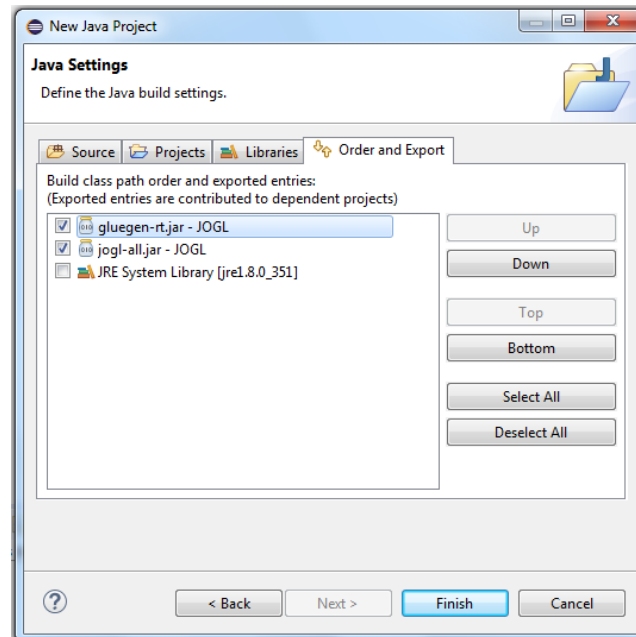


Рисунок 7. Экспорт JAR - файлов JOGL

В Eclipse получим следующую структуру проекта с именем «JOGL» (рис. 8).

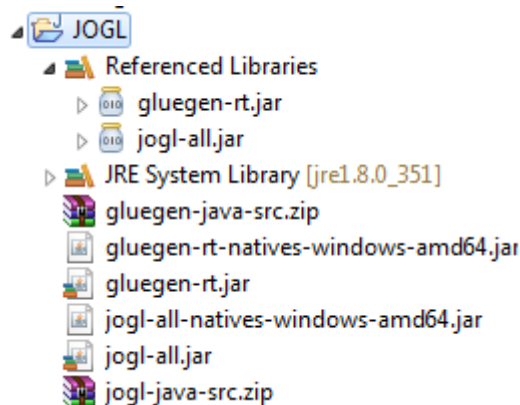


Рисунок 8. Структура проекта с именем «JOGL»

Добавление зависимости от проекта JOGL

- Создайте в Eclipse новый проект, например, NewJOGLProject.
- Нажмите правую кнопку мыши на новом проекте NewJOGLProject и выберите "Properties" (в самом низу выпадающего списка).
- Выберите в левом столбце появившегося окна пункт "Java Build Path".

- Нажмите на кнопку "Add...", выберите проект "JOGL" и нажмите "OK" (рис. 9).

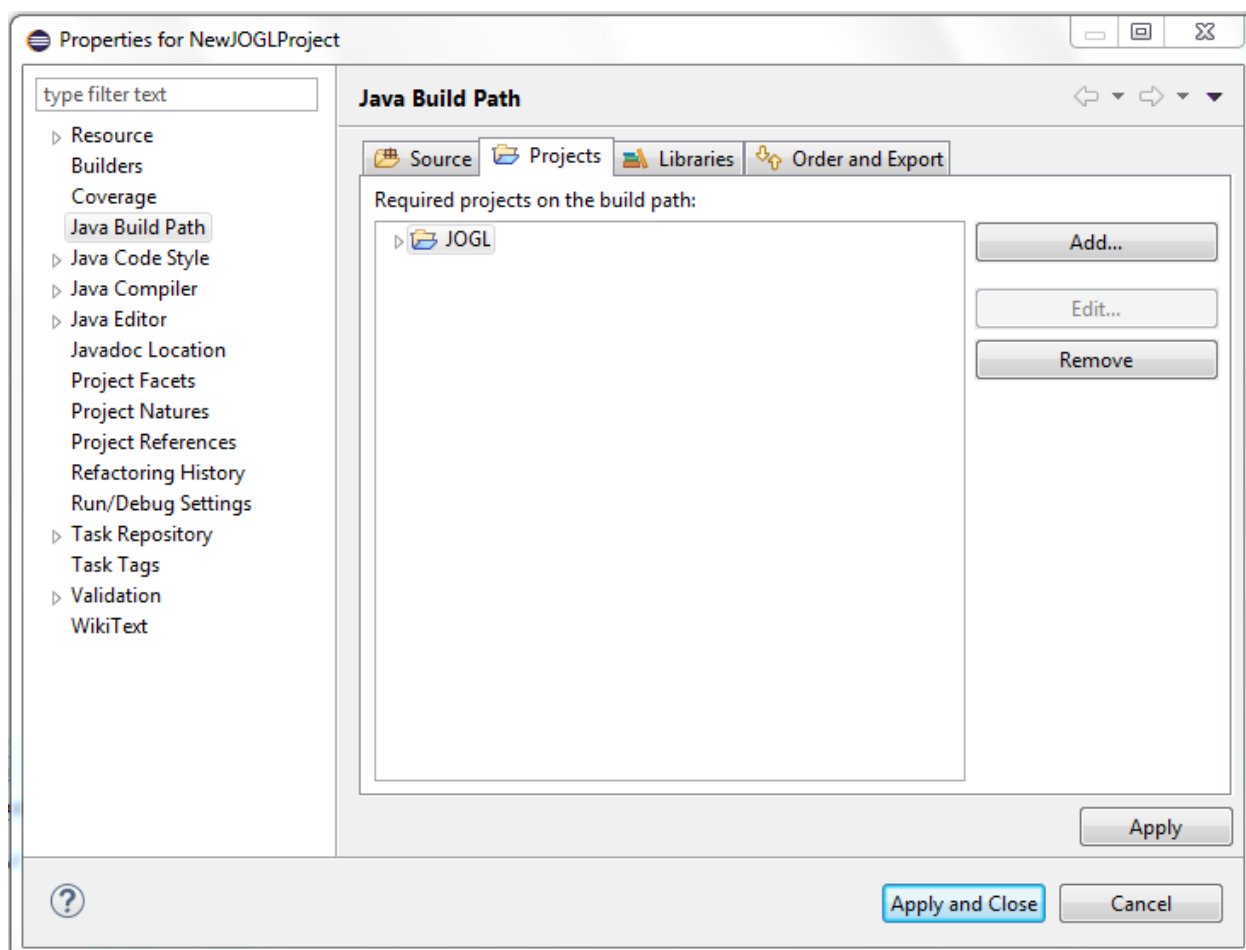


Рисунок 9. Добавление зависимости проекта NewJOGLProject от проекта с именем «JOGL»

Теперь в проекте NewJOGLProject можно использовать библиотеку JOGL. Если это необходимо, то можно создать и другие проекты, зависящие от проекта с именем «JOGL».

IntelliJ IDEA

Чтобы начать работать с OpenGL нужно сначала скачать архив jogamp-all-platforms.7z с сайта:

https://jogamp.org/wiki/index.php/Downloading_and_installing_JOGL

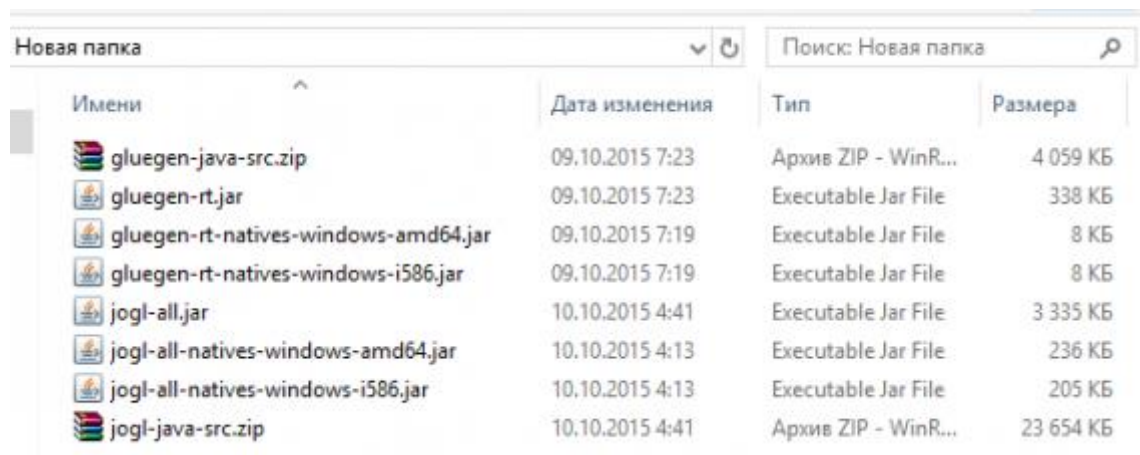
После того как архив получен с сайта, извлекаем в новую папку следующие файлы:









Файлы zip:

- jogl-java-src.zip
- gluegen-java-src.zip

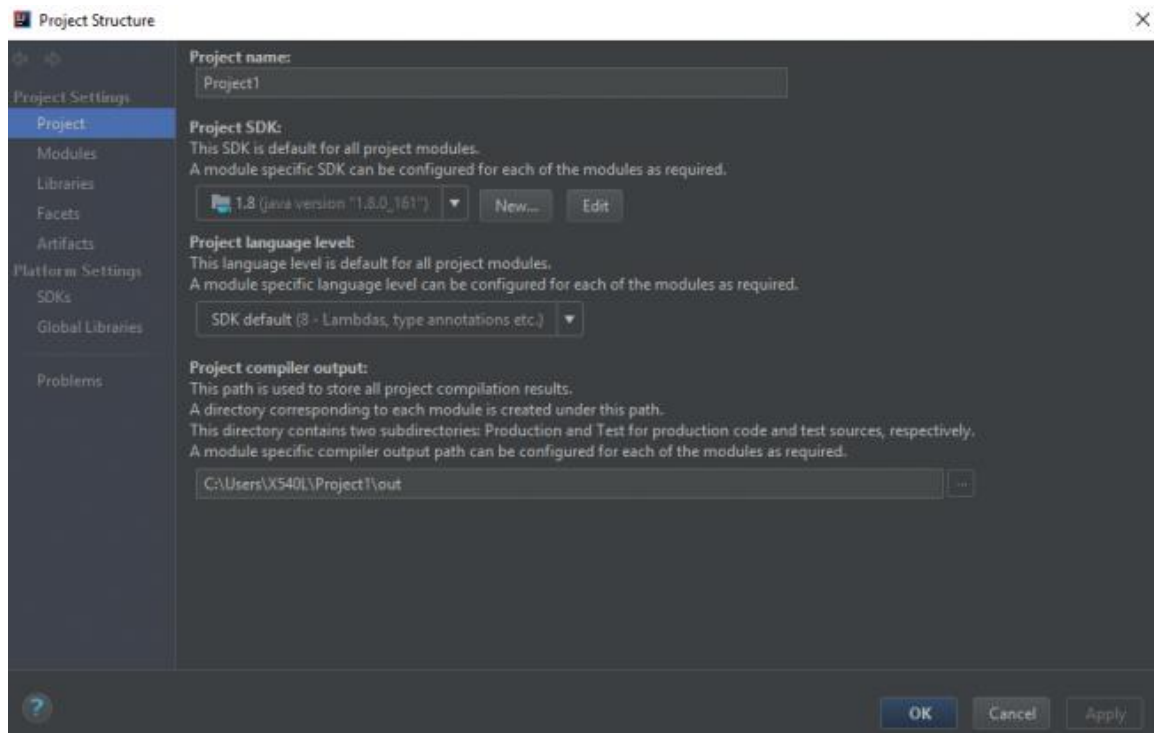
Из папки jar:

- jogl-all.jar
- jogl-all-natives-windows-amd64.jar
- gluegen-rt.jar
- gluegen-rt-natives-windows-amd64.jar

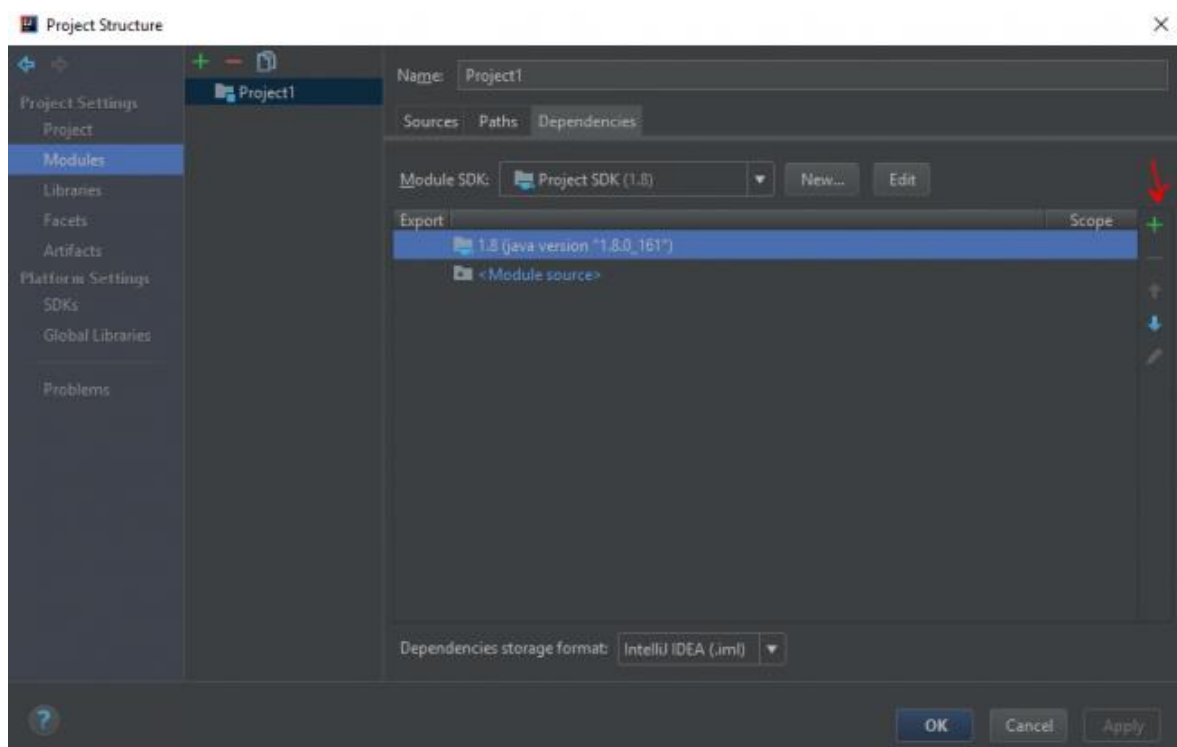


Новая папка				Поиск: Новая папка	
Имени	Дата изменения	Тип	Размера		
 gluegen-java-src.zip	09.10.2015 7:23	Архив ZIP - WinR...	4 059 КБ		
 gluegen-rt.jar	09.10.2015 7:23	Executable Jar File	338 КБ		
 gluegen-rt-natives-windows-amd64.jar	09.10.2015 7:19	Executable Jar File	8 КБ		
 gluegen-rt-natives-windows-i586.jar	09.10.2015 7:19	Executable Jar File	8 КБ		
 jogl-all.jar	10.10.2015 4:41	Executable Jar File	3 335 КБ		
 jogl-all-natives-windows-amd64.jar	10.10.2015 4:13	Executable Jar File	236 КБ		
 jogl-all-natives-windows-i586.jar	10.10.2015 4:13	Executable Jar File	205 КБ		
 jogl-java-src.zip	10.10.2015 4:41	Архив ZIP - WinR...	23 654 КБ		

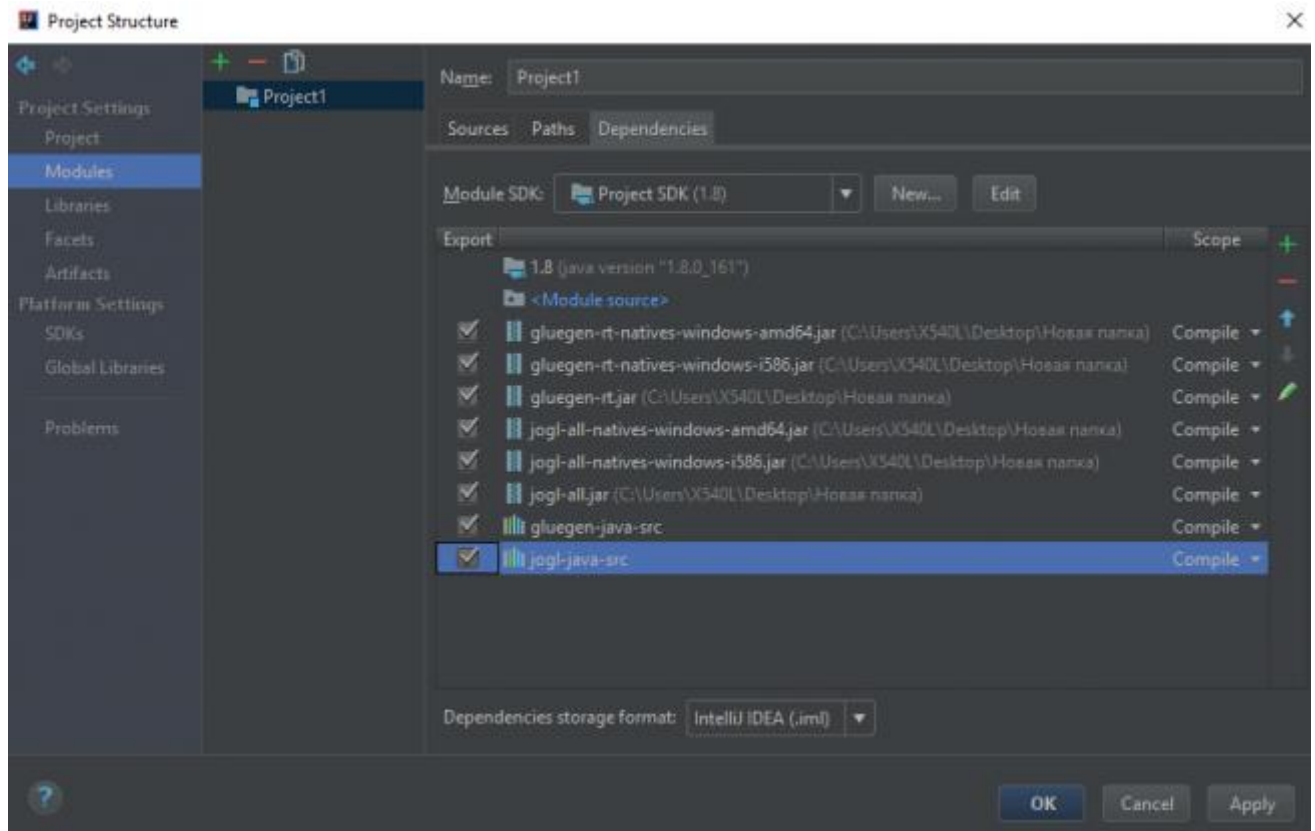
Запускаем IntelliJ IDEA и создаем проект или открываем ранее созданный. Заходим в меню File и выбираем пункт Project Structure.



Переходим в **Modules** и нажимаем на зеленый плюс (может быть и не зеленый и находиться в другом месте, это зависит от версии IntelliJ IDEA).



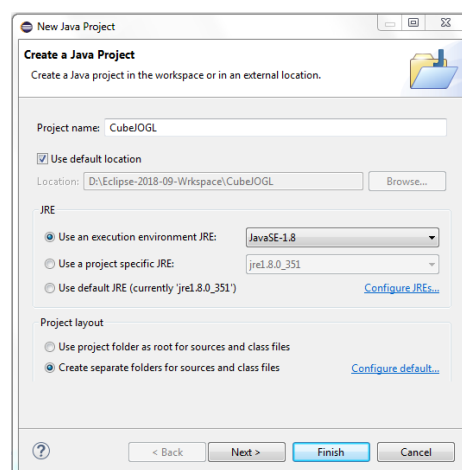
Далее выбираем JAR из папки, в которую извлекли файлы, добавляем файлы с разрешением jar. Затем выбираем Library, а потом Java и добавляем файлы с разрешением zip. Отмечаем все что добавили галочками и нажимаем Apply и затем OK.



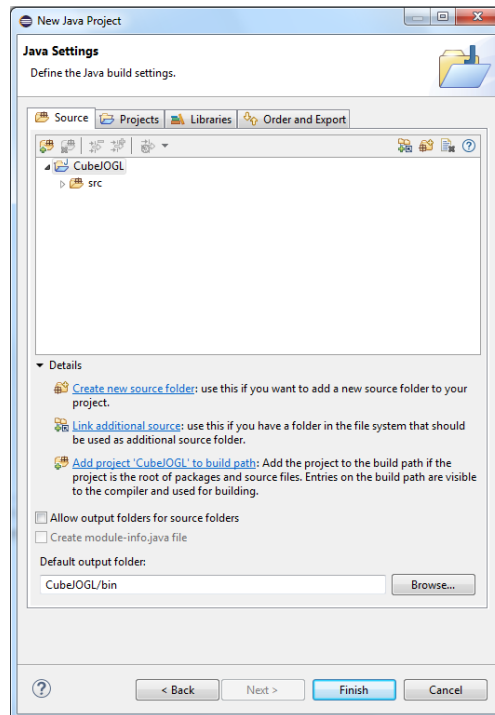
Теперь JOGL подключен. Если нажать в проекте на **External Libraries**, то можно увидеть все что мы подключили.

Пробный проект JOGL в Eclipse

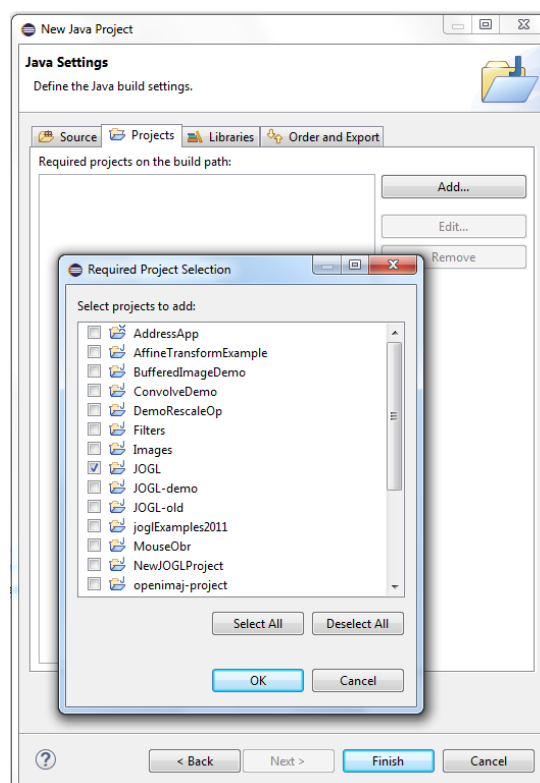
1. Создайте в Eclipse Java – проект CubeJOGL.



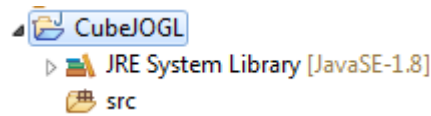
2. Нажмите кнопку Next. *Нужно нажать именно Next, а не Finish.* Это нужно для того, чтобы определить зависимость от проекта JOGL, который мы создали ранее. Появится следующее окно:



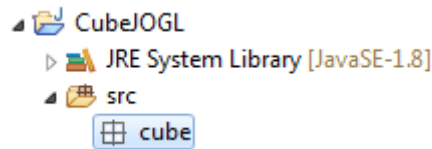
3. Выберите закладку Projects и нажмите кнопку Add (расположена справа в закладке Projects). Появится окно выбора проектов. Выберите проект JOGL.



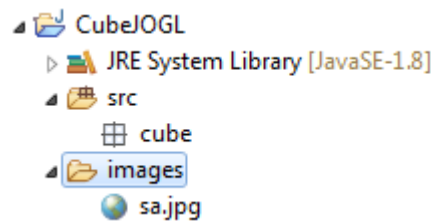
4. Нажмите кнопку Ok. Затем нажмите кнопку Finish в окне создания проекта.
5. Создастся проект со следующей структурой:



6. В проекте CubeJOGL создайте пакет cube.



7. В проекте CubeJOGL создайте каталог images и скопируйте в него файл sa.jpg (находится на сетевом диске).



8. Создайте в пакете cube класс CubeTexture. Код, который необходимо поместить в класс CubeTexture, приводится ниже.

```
package cube;

import java.awt.DisplayMode;
import java.io.File;
import java.io.IOException;
import javax.swing.JFrame;
import com.jogamp.opengl.GL2;
import com.jogamp.opengl.GLAutoDrawable;
import com.jogamp.opengl.GLCapabilities;
import com.jogamp.opengl.GLEventListener;
import com.jogamp.opengl.GLProfile;
import com.jogamp.opengl.awt.GLCanvas;
import com.jogamp.opengl.glu.GLU;
import com.jogamp.opengl.util.FPSAnimator;
import com.jogamp.opengl.util.texture.Texture;
import com.jogamp.opengl.util.texture.TextureIO;

public class CubeTexture implements GLEventListener {
    public static DisplayMode dm, dm_old;
    private GLU glu = new GLU();
    private float xrot,yrot,zrot;
    private int texture;

    @Override
    public void display(GLAutoDrawable drawable) {
        final GL2 gl = drawable.getGL().getGL2();
        gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);
        gl.glLoadIdentity();
    }
}
```

```

gl.glTranslatef(0f, 0f, -5.0f);

gl.glRotatef(xrot, 1.0f, 0.0f, 0.0f);
gl.glRotatef(yrot, 0.0f, 1.0f, 0.0f);
gl.glRotatef(zrot, 0.0f, 0.0f, 1.0f);

gl.glBindTexture(GL2.GL_TEXTURE_2D, texture);

gl.glBegin(GL2.GL_QUADS);

gl.glTexCoord2f(0.0f, 0.0f); gl.glVertex3f(-1.0f, -1.0f, 1.0f);
gl.glTexCoord2f(1.0f, 0.0f); gl.glVertex3f( 1.0f, -1.0f, 1.0f);
gl.glTexCoord2f(1.0f, 1.0f); gl.glVertex3f( 1.0f, 1.0f, 1.0f);
gl.glTexCoord2f(0.0f, 1.0f); gl.glVertex3f(-1.0f, 1.0f, 1.0f);

gl.glTexCoord2f(1.0f, 0.0f); gl.glVertex3f(-1.0f, -1.0f, -1.0f);
gl.glTexCoord2f(1.0f, 1.0f); gl.glVertex3f(-1.0f, 1.0f, -1.0f);
gl.glTexCoord2f(0.0f, 1.0f); gl.glVertex3f( 1.0f, 1.0f, -1.0f);
gl.glTexCoord2f(0.0f, 0.0f); gl.glVertex3f( 1.0f, -1.0f, -1.0f);

gl.glTexCoord2f(0.0f, 1.0f); gl.glVertex3f(-1.0f, 1.0f, -1.0f);
gl.glTexCoord2f(0.0f, 0.0f); gl.glVertex3f(-1.0f, 1.0f, 1.0f);
gl.glTexCoord2f(1.0f, 0.0f); gl.glVertex3f( 1.0f, 1.0f, 1.0f);
gl.glTexCoord2f(1.0f, 1.0f); gl.glVertex3f( 1.0f, 1.0f, -1.0f);

gl.glTexCoord2f(1.0f, 1.0f); gl.glVertex3f(-1.0f, -1.0f, -1.0f);
gl.glTexCoord2f(0.0f, 1.0f); gl.glVertex3f( 1.0f, -1.0f, -1.0f);
gl.glTexCoord2f(0.0f, 0.0f); gl.glVertex3f( 1.0f, -1.0f, 1.0f);
gl.glTexCoord2f(1.0f, 0.0f); gl.glVertex3f(-1.0f, -1.0f, 1.0f);

gl.glTexCoord2f(1.0f, 0.0f); gl.glVertex3f( 1.0f, -1.0f, -1.0f);
gl.glTexCoord2f(1.0f, 1.0f); gl.glVertex3f( 1.0f, 1.0f, -1.0f);
gl.glTexCoord2f(0.0f, 1.0f); gl.glVertex3f( 1.0f, 1.0f, 1.0f);
gl.glTexCoord2f(0.0f, 0.0f); gl.glVertex3f( 1.0f, -1.0f, 1.0f);

gl.glTexCoord2f(0.0f, 0.0f); gl.glVertex3f(-1.0f, -1.0f, -1.0f);
gl.glTexCoord2f(1.0f, 0.0f); gl.glVertex3f(-1.0f, -1.0f, 1.0f);
gl.glTexCoord2f(1.0f, 1.0f); gl.glVertex3f(-1.0f, 1.0f, 1.0f);
gl.glTexCoord2f(0.0f, 1.0f); gl.glVertex3f(-1.0f, 1.0f, -1.0f);

gl.glEnd();

gl.glFlush();

xrot += .1f;
yrot += .1f;
zrot += .1f;
}

@Override
public void dispose(GLAutoDrawable drawable) {
}

@Override
public void init(GLAutoDrawable drawable) {
    final GL2 gl = drawable.getGL().getGL2();

    gl.glShadeModel(GL2.GL_SMOOTH);
    gl.glClearColor(0f, 0f, 0f, 0f);
    gl.glClearDepth(1.0f);
    gl.glEnable(GL2.GL_DEPTH_TEST);
    gl.glDepthFunc(GL2.GL_LEQUAL);

```

```

gl.glHint(GL2.GL_PERSPECTIVE_CORRECTION_HINT, GL2.GL_NICEST);

gl.glEnable(GL2.GL_TEXTURE_2D);
try{
    File im = new File("images\\sa.jpg");
    Texture t = TextureIO.newTexture(im, true);
    texture= t.getTextureObject(gl);
}catch(IOException e){
    e.printStackTrace();
}
}

@Override
public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
    final GL2 gl = drawable.getGL().getGL2();
    if(height <= 1)
        height = 1;

    final float h = (float) width / (float) height;
    gl.glViewport(0, 0, width, height);
    gl.glMatrixMode(GL2.GL_PROJECTION);
    gl.glLoadIdentity();

    glu.gluPerspective(45.0f, h, 1.0, 20.0);
    gl.glMatrixMode(GL2.GL_MODELVIEW);
    gl.glLoadIdentity();
}

public static void main(String[] args) {
    final GLProfile profile = GLProfile.get(GLProfile.GL2);
    GLCapabilities capabilities = new GLCapabilities(profile);

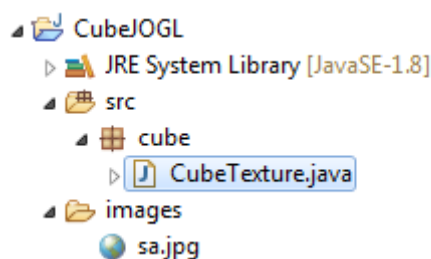
    final GLCanvas glcanvas = new GLCanvas(capabilities);
    CubeTexture r = new CubeTexture();

    glcanvas.addGLEventListener(r);
    glcanvas.setSize(400, 400);

    final JFrame frame = new JFrame (" Куб с текстурой ");
    frame.getContentPane().add(glcanvas);
    frame.setSize(frame.getContentPane().getPreferredSize());
    frame.setVisible(true);
    final FPSAnimator animator = new FPSAnimator(glcanvas, 300, true);
    animator.start();
}
}

```

9. Теперь проект CubeJOGL имеет следующую структуру:



10. Запустите класс CubeJOGL на выполнение. Если вы увидели вращающийся куб, значит все сделано правильно и у вас заработала программа с использованием JOGL.

Базовый фрейм JOGL

Чтобы программа могла использовать API JOGL, необходимо реализовать интерфейс `GLEventListener`.

Интерфейс `GLEventListener`

Все методы интерфейса `GLEventListener` требуют в качестве параметра объект класса, реализующего интерфейс `GLAutoDrawable`. Тип возвращаемого значения всех методов - `void`.

Метод	Описание
<u><code>display</code></u> (<u><code>GLAutoDrawable</code></u> drawable)	Вызывается drawable для инициализации рендеринга OpenGL клиентом. Метод содержит логику, используемую для рисования графических элементов с использованием OpenGL API.
<u><code>dispose</code></u> (<u><code>GLAutoDrawable</code></u> drawable)	Уведомляет слушателя о выполнении освобождения всех ресурсов OpenGL для каждого GLContext, таких как буферы памяти и программы GLSL.
<u><code>init</code></u> (<u><code>GLAutoDrawable</code></u> drawable)	Вызывается drawable сразу после инициализации контекста OpenGL.
<u><code>reshape</code></u> (<u><code>GLAutoDrawable</code></u> drawable, int x, int y, int width, int height)	Вызывается объектом рисования во время первого перерисовывания после изменения размера компонента. Параметры: drawable - запуск <code>GLAutoDrawable</code> x - x-координата окна просмотра в пикселях y - y-координата окна просмотра в пикселях width - ширина области просмотра в пикселях height - высота окна просмотра в пикселях

Интерфейс GLAutoDrawable

GLAutoDrawable создает основной контекст рендеринга JOGL.

Интерфейс GLAutoDrawable расширяет интерфейсы GLDrawable и NativeSurfaceHolder.

Этот интерфейс реализуют классы *GLAutoDrawableBase*, *GLAutoDrawableDelegate*, *GLCanvas*, *GLJCanvas*, *GLJPanel*, *GLWindow*.

Методы интерфейса GLAutoDrawable

Модификатор и тип	Метод	Описание
void	<u><a>addGLEventListener</u> (int index, <u><a>GLEventListener</u> listener)	Добавляет заданный <code>listener</code> по заданному индексу этой очереди рисования.
void	<u><a>addGLEventListener</u> (<u><a>GLEventListener</u> listener)	Добавляет заданное значение <code>listener</code> в конец очереди рисования.
boolean	<u><a>areAllGLEventListenerInitialized</u> ()	Возвращает true, если все <code>GLEventListener</code> инициализированы, в противном случае false .
<u><a>GLContext</u>	<u><a>createContext</u> (<u><a>GLContext</u> shareWith)	Создает новый контекст для рисования для этого объекта рисования, который при необходимости будет совместно использовать объекты буфера, текстуры и другие объекты OpenGL на стороне сервера с указанным <code>GLContext</code> .
void	<u><a>destroy</u> ()	Уничтожает все ресурсы, связанные с этим GLAutoDrawable, включая <code>GLContext</code> .
void	<u><a>display</u> ()	Вызывает выполнение рендеринга OpenGL для этого GLAutoDrawable в следующем порядке: вызов <code>display(..)</code> всех зарегистрированных <code>GLEventListener</code> .
<u><a>GLEventListener</u>	<u><a>disposeGLEventListener</u> (<u><a>GLEventListener</u> listener, boolean remove)	Удаляет указанный <code>listener</code> посредством <code>dispose(..)</code> , если он был инициализирован и добавлен в очередь.
void	<u><a>flushGLRunnables</u> ()	Сбрасывает все <code>enqueued GLRunnable</code> этого GLAutoDrawable, включая уведомление ожидающего исполнителя.

<u>GLAnimatorControl</u>	<u>getAnimator()</u>	
boolean	<u>getAutoSwapBufferMode()</u>	Указывает, включена ли автоматическая замена буфера для этого объекта рисования.
<u>GLContext</u>	<u>getContext()</u>	<p>Возвращает контекст, связанный с этим объектом рисования.</p> <p>Возвращенный контекст будет синхронизирован. Не полагайтесь на его идентификатор, контекст может измениться.</p>
int	<u>getContextCreationFlags()</u>	
<u>GLDrawable</u>	<u>getDelegatedDrawable()</u>	Если реализация использует делегирование, возвращает делегированный экземпляр <code>GLDrawable</code> , в противном случае возвращает экземпляр <code>this</code> .
<u>Thread</u>	<u>getExclusiveContextThread()</u>	
<u>GL</u>	<u>getGL()</u>	Возвращает объект конвейера <code>GL</code> , который использует этот <code>GLAutoDrawable</code> .
<u>GLEventListener</u>	<u>getGLEventListener(int index)</u>	Возвращает значение <code>GLEventListener</code> по заданному индексу очереди для рисования.
int	<u>getGLEventListenerCount()</u>	Возвращает номер <code>GLEventListener</code> этой очереди для рисования.
boolean	<u>getGLEventListenerInitState(GLEventListener listener)</u>	Возвращает, инициализирован ли данный <code>listener</code> или нет.
<u>RecursiveLock</u>	<u>getUpstreamLock()</u>	Возвращает рекурсивный объект блокировки <code>upstream widget</code> для синхронизации многопоточного доступа поверх <code>NativeSurface.lockSurface()</code> .
<u>Object</u>	<u>getUpstreamWidget()</u>	Метод <i>может</i> возвращать вышестоящий объект инструментария пользовательского интерфейса, содержащий экземпляр <code>GLAutoDrawable</code> , если он существует.

boolean	<u>invoke</u> (boolean wait, <u>GLRunnable</u> glRunnable)	Ставит в очередь одноразовый запрос <u>GLRunnable</u> , который будет выполнен в рамках следующего вызова <u>display()</u> после вызова всех зарегистрированных методов <u>GLEventListenerdisplay</u> (<u>GLAutoDrawable</u>).
boolean	<u>invoke</u> (boolean wait, <u>List</u> < <u>GLRunnable</u> > glRunnables)	Расширяет <u>invoke</u> (boolean, <u>GLRunnable</u>) функциональность, позволяя вводить список <u>GLRunnable</u> файлов.
boolean	<u>isThreadGLCapable</u> ()	Указывает, способен ли текущий поток выполнять работу, связанную с OpenGL.
<u>GLEventListener</u>	<u>removeGLEventListener</u> (<u>GLEventListener</u> listener)	Удаляет данные <u>listener</u> из очереди рисования.
void	<u>setAnimator</u> (<u>GLAnimatorControl</u> animatorControl)	Регистрирует использование аниматора, реализующего <u>GLAnimatorControl</u> .
void	<u>setAutoSwapBufferMode</u> (boolean enable)	Включает или отключает автоматическую замену буфера для этого объекта рисования.
<u>GLContext</u>	<u>setContext</u> (<u>GLContext</u> newCtx, boolean destroyPrevCtx)	Связывает новый контекст, <u>newCtx</u> , с этим автоматически отрисовываемым.
void	<u>setContextCreationFlags</u> (int flags)	
<u>Thread</u>	<u>setExclusiveContextThread</u> (<u>Thread</u> t)	Устанавливает этот экземпляр <u>GLContext</u> для данного потока. Поток будет запрашивать исключительно <u>GLContext</u> посредством вызова <u>display()</u> и не освобождать его до тех пор, пока не будет вызван <u>destroy()</u> или <u>setExclusiveContextThread</u> (null).
<u>GL</u>	<u>setGL</u> (<u>GL</u> gl)	Задаёт объект <u>GL</u> конвейера, который использует этот <u>GLAutoDrawable</u> .
void	<u>setGLEventListenerInitState</u> (<u>GLEventListener</u> listener, boolean initialized)	Устанавливает инициализированное состояние <u>listener</u> .

Методы, унаследованные от GLDrawable

Модификатор и тип	Метод	Описание
<u>GLContext</u>	<u>createContext</u> (<u>GLContext</u> shareWith)	Создает новый контекст для рисования в этом drawable, который при необходимости будет совместно использовать объекты буфера, текстуры и другие объекты OpenGL на стороне сервера с указанным GLContext.
<u>GLCapabilitiesImmutable</u>	<u>getChosenGLCapabilities</u> ()	Извлекает <u>GLCapabilitiesImmutable</u> соответствующие выбранные возможности OpenGL (формат пикселей / визуальный / GLProfile) для этого объекта рисования.
<u>GLDrawableFactory</u>	<u>getFactory</u> ()	Возвращает значение <u>GLDrawableFactory</u> , используемой для создания этого экземпляра.
<u>GLProfile</u>	<u>getGLProfile</u> ()	Возвращает <u>GLProfile</u> .
long	<u>getHandle</u> ()	Возвращает дескриптор GL drawable, который гарантированно будет действительным после <i>realization</i> и во время его <i>surfaceLocked</i> создания.
<u>NativeSurface</u>	<u>getNativeSurface</u> ()	Возвращает связанный <u>NativeSurface</u> с этим <u>NativeSurfaceHolder</u> .
<u>GLCapabilitiesImmutable</u>	<u>getRequestedGLCapabilities</u> ()	Извлекает <u>GLCapabilitiesImmutable</u> соответствующие запрошенным пользователем возможностям OpenGL (формат пикселей / визуальный / GLProfile) для этого объекта рисования.
int	<u>getSurfaceHeight</u> ()	Возвращает высоту <u>GLDrawableSurface</u> клиентской области в пикселях.
int	<u>getSurfaceWidth</u> ()	Возвращает ширину <u>GLDrawableSurface</u> клиентской области в пикселях.
boolean	<u>isGLOriented</u> ()	Возвращает <i>true</i> , если объект рисования отображается в системе координат OpenGL, начало <i>координат внизу слева</i> .
boolean	<u>isRealized</u> ()	Возвращает <i>true</i> , если возможность рисования реализована, в противном случае <i>false</i> .
void	<u>setRealized</u> (boolean realized)	Указывает реализациям <u>surface</u> <u>GLDrawable</u> , был ли создан базовый компонент и может ли он быть использован.

void

[swapBuffers](#) ()

Меняет местами передний и задний буферы.

Инициализация GLAutoDrawable

Реализация должна инициализироваться как можно скорее, что возможно только *после* того, как прикрепленное `NativeSurface` станет видимым и будет реализовано.

Должна быть реализована следующая последовательность инициализации:

- Создайте `GLDrawable` с запрошенным `GLCapabilities`
- Уведомлять `GLDrawable` для проверки `GLCapabilities` путем вызова `setRealized(true)`.
- Создайте новый `GLContext`.
- Инициализируйте все ресурсы OpenGL, вызывая `init(..)` все зарегистрированные ресурсы `GLEventListener`. Это можно сделать немедленно или с помощью последующего `display(..)` вызова.
- Отправьте событие изменения формы, вызвав `reshape(..)` всем зарегистрированным `GLEventListener`. Это должно быть сделано после вызова `init(..)`.

Реконфигурация GLAutoDrawable

Другой деталью реализации является реконфигурация `GLDrawable`. Одним из вариантов использования является перетаскивание окна на другой экран с другой конфигурацией пикселей, `GLCapabilities`, т. е. Реализация должна быть способна обнаруживать такие случаи в сочетании с соответствующими `NativeSurface`.

Например, AWT `Canvas` `getGraphicsConfiguration()` способен определять изменение устройства отображения. Это продемонстрировано в рамках `GLCanvas` специализации и NEWT AWT `Canvas` `getGraphicsConfiguration()`. Другой демонстрацией является `NativeWindow` реализация NEWT на платформе Windows, которая использует функцию `MonitorFromWindow (HWND)` *собственной платформы*.

Все ресурсы OpenGL должны быть восстановлены, в то время как `drawable` `GLCapabilities` должен быть выбран снова. Должен выполняться следующий протокол.

- Контролируемая утилизация:
 - Утилизируйте все ресурсы OpenGL, вызывая `dispose(..)` всех зарегистрированных `GLEventListener`.
 - Уничтожьте `GLContext`.
 - Уведомите `GLDrawable` о недопустимом состоянии путем вызова `setRealized(false)`.
- Контролируемая регенерация:
 - Создайте новый `GLDrawable` с запрошенным `GLCapabilities`
 - Уведомите `GLDrawable` о повторной проверке `GLCapabilities` путем вызова `setRealized(true)`.
 - Создайте новый `GLContext`.
 - Инициализируйте все ресурсы OpenGL, вызывая `init(..)` всех зарегистрированных `GLEventListener`. Это можно сделать немедленно или с помощью последующего вызова `display(..)`.
 - Отправьте событие изменения формы, вызвав `reshape(..)` всем зарегистрированным `GLEventListener`. Это должно быть сделано после вызовов `init(..)`.

Блокировка GLAutoDrawable

Реализации `GLAutoDrawable` выполняют блокировку в следующем порядке:

```
1. getUpstreamLock().lock()
2. GLDrawable.getNativeSurface().lockSurface()
```

и соответственно снимает блокировки:

```
1. GLDrawable.getNativeSurface().unlockSurface()
2. getUpstreamLock().unlock()
```

Вышеуказанный *порядок блокировки* является обязательным, чтобы гарантировать атомарность работы и избежать условий гонки. Пользовательская реализация или пользовательские приложения, требующие эксклюзивного доступа, должны следовать *порядку блокировки*. См.:

- `getUpstreamLock()`
- `invoke(boolean, GLRunnable)`
- `invoke(boolean, List)`

Классы GLCanvas и GLJPanel

GLCanvas и GLJPanel - это два основных класса графического интерфейса JOGL, которые реализуют интерфейс GLAutoDrawable, и которые можно использовать в качестве поверхностей рисования для команд OpenGL.

GLCanvas - это тяжеловесный AWT-компонент, который обеспечивает поддержку рендеринга OpenGL. Это основная реализация интерфейса AWTAutoGLDrawable. Он является потомком класса `java.awt.Canvas`.

Поскольку это тяжеловесный компонент, в некоторых случаях GLJCanvas может неправильно интегрироваться с компонентами Swing. Если возникают проблемы с использованием GLCanvas, необходимо использовать класс GLJPanel.

Класс GLCanvas

Конструкторы GLCanvas

Конструктор	Описание
<code>GLCanvas()</code>	Создает новый компонент GLCanvas с набором возможностей OpenGL по умолчанию, используя механизм выбора возможностей OpenGL по умолчанию, на экранном устройстве по умолчанию.
<code>GLCanvas(GLCapabilitiesImmutable capsReqUser)</code>	Создает новый компонент GLCanvas с запрошенным набором возможностей OpenGL, используя механизм выбора возможностей OpenGL по умолчанию, на экранном устройстве по умолчанию.

GLCanvas
(GLCapabilitiesImmutable
capsReqUser, GLCapabilities
Chooser chooser, GraphicsDevice device)

Создает новый компонент GLCanvas.

Методы GLCanvas

Модификатор и тип	Метод	Описание
void	<u>addGLEventListener</u> (int index, <u>GLEventListener</u> listener)	Добавляет данные <code>listener</code> по заданному индексу в эту очередь для рисования.
void	<u>addGLEventListener</u> (<u>GLEventListener</u> listener)	Добавляет заданное значение <code>listener</code> в конец этой очереди рисования.
void	<u>addNotify</u> ()	Переопределен для отслеживания добавления этого компонента в контейнер.
boolean	<u>areAllGLEventListenerInitialized</u> ()	Возвращает true, если все добавленные <u>GLEventListener</u> инициализированы, в противном случае false .
<u>GLContext</u>	<u>createContext</u> (<u>GLContext</u> shareWith)	Создает новый контекст для рисования для этого объекта рисования, который при необходимости будет совместно использовать объекты буфера, текстуры и другие объекты OpenGL на стороне сервера с указанным GLContext.
void	<u>destroy</u> ()	Уничтожает все ресурсы, связанные с этим GLAutoDrawable, включая GLContext.
void	<u>display</u> ()	Вызывает выполнение рендеринга OpenGL для этого GLAutoDrawable в следующем порядке: вызов <code>display(..)</code> всех зарегистрированных <u>GLEventListener</u> файлов.
<u>GLEventListener</u>	<u>disposeGLEventListener</u> (<u>GLEventListener</u> listener, boolean remove)	Удаляет указанный <code>listener</code> с помощью <code>dispose(..)</code> , если он был инициализирован и добавлен в эту очередь.

void	<u>flushGLRunnables()</u>	Сбрасывает все <code>enqueued GLRunnable</code> этого <code>GLAutoDrawable</code> , включая уведомление ожидающего исполнителя.
<u>GLAnimatorControl</u>	<u>getAnimator()</u>	
boolean	<u>getAutoSwapBufferMode()</u>	Указывает, включена ли автоматическая замена буфера для этого объекта рисования.
<u>GLCapabilitiesImmutable</u>	<u>getChosenGLCapabilities()</u>	Извлекает <code>GLCapabilitiesImmutable</code> соответствующие возможности OpenGL
<u>GLContext</u>	<u>getContext()</u>	Возвращает контекст, связанный с этим объектом рисования.
int	<u>getContextCreationFlags()</u>	
float[]	<u>getCurrentSurfaceScale(float[] result)</u>	Возвращает текущий масштаб пикселя связанного с <code>NativeSurface</code> .
<u>WindowClosingProtocol.WindowClosingMode</u>	<u>getDefaultCloseOperation()</u>	
<u>GLDrawable</u>	<u>getDelegatedDrawable()</u>	Если реализация использует делегирование, возвращает делегированный экземпляр <code>GLDrawable</code> , в противном случае возвращает <code>this</code> .
<u>Thread</u>	<u>getExclusiveContextThread()</u>	
<u>GLDrawableFactory</u>	<u>getFactory()</u>	Возвращает значение <code>GLDrawableFactory</code> , использованное для создания этого экземпляра.
<u>GL</u>	<u>getGL()</u>	Возвращает объект <code>GL</code> конвейера, который использует этот <code>GLAutoDrawable</code> .
<u>GLEventListener</u>	<u>getGLEventListener(int index)</u>	Возвращает значение <code>GLEventListener</code> по заданному индексу этой очереди для рисования.
int	<u>getGLEventListenerCount()</u>	Возвращает номер <code>GLEventListener</code> этой очереди для рисования.

boolean	<u>getGLEventListenerInitState</u> (<u>GLEventListener</u> listener)	Извлекает, инициализирован ли данный <code>listener</code> или нет.
<u>GLProfile</u>	<u>getGLProfile</u> ()	Возвращает <code>GLProfile</code>
<u>GraphicsConfiguration</u>	<u>getGraphicsConfiguration</u> ()	
long	<u>getHandle</u> ()	Возвращает дескриптор GL, который гарантированно будет действительным после <code>realization</code> и во время его <code>surfaceLocked</code> создания.
float[]	<u>getMaximumSurfaceScale</u> (float[] result)	Возвращает максимальный масштаб пикселя связанного <code>NativeSurface</code> .
float[]	<u>getMinimumSurfaceScale</u> (float[] result)	Возвращает минимальный масштаб пикселя связанного <code>NativeSurface</code> .
<u>NativeSurface</u>	<u>getNativeSurface</u> ()	Возвращает связанный <code>NativeSurface</code> этим <code>NativeSurfaceHolder</code> .
<u>GLCapabilitiesImmutable</u>	<u>getRequestedGLCapabilities</u> ()	Извлекает <code>GLCapabilitiesImmutable</code> соответствующие запрошенным пользователем возможностям OpenGL
float[]	<u>getRequestedSurfaceScale</u> (float[] result)	Возвращает масштаб <code>requested</code> пикселя связанного <code>NativeSurface</code> .
boolean	<u>getShallUseOffscreenLayer</u> ()	Возвращает свойство, установленное с помощью <code>OffscreenLayerOption.setShallUseOffscreenLayer(boolean)</code> .
int	<u>getSurfaceHeight</u> ()	Возвращает высоту <code>GLDrawablesurface</code> клиентской области в пикселях.
int	<u>getSurfaceWidth</u> ()	Возвращает ширину <code>GLDrawablesurface</code> клиентской области в пикселях.
<u>RecursiveLock</u>	<u>getUpstreamLock</u> ()	Возвращает рекурсивный объект блокировки <code>upstream widget</code> для синхронизации многопоточного доступа поверх <code>NativeSurface.lockSurface()</code> .

<u>Object</u>	<u>getUpstreamWidget()</u>	Метод <i>может</i> возвращать вышестоящий объект инструментария пользовательского интерфейса, содержащий этот экземпляр <code>GLAutoDrawable</code> , если он существует.
boolean	<u>invoke</u> (boolean wait, <u>GLRunnable</u> glRunnable)	Ставит в очередь одноразовый <code>GLRunnable</code> запрос, который будет выполнен в рамках следующего <code>GLAutoDrawable.display()</code> вызова после вызова всех зарегистрированных методов <code>GLEventListener.display(GLAutoDrawable)</code> .
boolean	<u>invoke</u> (boolean wait, <u>List<GLRunnable></u> glRunnable s)	Расширяет <code>GLAutoDrawable.invoke(boolean, GLRunnable)</code> функциональность, позволяя вводить список <code>GLRunnable</code> файлов.
boolean	<u>isGLOriented()</u>	Возвращает <code>true</code> , если отрисовываемый объект отображается в системе координат OpenGL, начало <i>координат внизу слева</i> .
boolean	<u>isOffscreenLayerSurfaceEnabled()</u>	Возвращает <code>true</code> , если этот экземпляр использует закадровый слой, в противном случае <code>false</code> .
boolean	<u>isRealized()</u>	Возвращает <code>true</code> , если эта возможность рисования реализована, в противном случае <code>false</code> .
boolean	<u>isThreadGLCapable()</u>	Указывает, способен ли текущий поток выполнять работу, связанную с OpenGL.
static void	<u>main</u> (<u>String</u> [] args)	Самая простая тестовая запись JOGL AWT
void	<u>paint</u> (<u>Graphics</u> g)	Переопределен, чтобы вызвать выполнение рендеринга OpenGL во время циклов перерисовки.
void	<u>print</u> (<u>Graphics</u> graphics)	
void	<u>releasePrint()</u>	Должны быть вызваны после <code>PrinterJob.print()</code> .

<u>GLEventListener</u>	<u>removeGLEventListener</u> (<u>GLEventListener</u> listener)	Удаляет данные listener из этой очереди рисования.
void	<u>removeNotify</u> ()	Переопределяется для отслеживания, когда этот компонент удаляется из контейнера.
void	<u>reshape</u> (int x, int y, int width, int height)	Переопределен, чтобы вызвать GLDrawableHelper.reshape(com.jogamp.opengl.GLAutoDrawable, int, int, int, int) вызов для всех зарегистрированных GLEventListener файлов.
void	<u>setAnimator</u> (<u>GLAnimatorControl</u> animatorControl)	Регистрирует использование аниматора, реализацию GLAnimatorControl.
void	<u>setAutoSwapBufferMode</u> (boolean onOrOff)	Включает или отключает автоматическую замену буфера для этого объекта рисования.
<u>GLContext</u>	<u>setContext</u> (<u>GLContext</u> newCtx, boolean destroyPrevCtx)	Связывает новый контекст, newCtx, с этим автоматически извлекаемым.
void	<u>setContextCreationFlags</u> (int flags)	
<u>WindowClosingProtocol.WindowClosingMode</u>	<u>setDefaultCloseOperation</u> (<u>WindowClosingProtocol.WindowClosingMode</u> op)	
<u>Thread</u>	<u>setExclusiveContextThread</u> (<u>Thread</u> t)	Выделяет этот экземпляр GLContext для данного потока. Поток будет запрашивать исключительно GLContext via GLAutoDrawable.display() и не освобождать его до GLAutoDrawable.destroy() тех пор, пока не будет вызван или setExclusiveContextThread(null) не будет вызван.
<u>GL</u>	<u>setGL</u> (<u>GL</u> gl)	Задаёт объект GL конвейера, который использует этот GLAutoDrawable.

void	<u>setGLEventListenerInitState</u> (<u>GLEventListener</u> listener, boolean initialized)	Устанавливает заданное инициализированное состояние <code>listener</code> .
void	<u>setRealized</u> (boolean realized)	Указывает для реализаций <code>surface GLDrawable</code> , был ли создан базовый компонент и может ли он быть использован.
void	<u>setShallUseOffscreenLayer</u> (boolean v)	Возвращает закадровый слой, если он поддерживается.
void	<u>setSharedAutoDrawable</u> (<u>GLAutoDrawable</u> sharedAutoDrawable)	Указывает <code>GLAutoDrawable</code> , который <code>OpenGL context</code> должен быть общим <code>GLAutoDrawable</code> для этого <code>GLContext</code> .
void	<u>setSharedContext</u> (<u>GLContext</u> sharedContext)	Указывает <code>OpenGL context</code> , который должен быть общим <code>GLAutoDrawable</code> для этого <code>GLContext</code> .
boolean	<u>setSurfaceScale</u> (float[] pixelScale)	Запрашивать масштаб пикселей в направлениях x и y для связанного <code>NativeSurface</code> , где <code>size_in_pixel_units = pixel_scale * size_in_window_units</code> .
void	<u>setupPrint</u> (double scaleMatX, double scaleMatY, int numSamples, int tileWidth, int tileHeight)	Должен быть вызван раньше <code>PrinterJob.print()</code> .
void	<u>swapBuffers</u> ()	Меняет местами передний и задний буферы.
<u>String</u>	<u>toString</u> ()	
void	<u>update</u> (<u>Graphics</u> g)	Переопределен из <code>Canvas</code> , чтобы предотвратить очистку <code>canvas AWT</code> от вмешательства в рендеринг <code>OpenGL</code> .

Класс GLJPanel

Конструкторы GLJPanel

Конструктор	Описание
<u>GLJPanel()</u>	Создает новый компонент GLJPanel с набором возможностей OpenGL по умолчанию и с использованием механизма выбора возможностей OpenGL по умолчанию.
<u>GLJPanel</u> (<u>GLCapabilitiesImmutable</u> userCapabilitiesRequest)	Создает новый компонент GLJPanel с запрошенным набором возможностей OpenGL, используя механизм выбора возможностей OpenGL по умолчанию.
<u>GLJPanel</u> (<u>GLCapabilitiesImmutable</u> userCapabilitiesRequest, <u>GLCapabilitiesChooser</u> chooser)	Создает новый компонент GLJPanel.

Методы GLJPanel

Модификатор и тип	Метод	Описание
void	<u>addGLEventListener</u> (int index, <u>GLEventListener</u> listener)	Добавляет данные <code>listener</code> по заданному индексу в эту очередь для рисования.
void	<u>addGLEventListener</u> (<u>GLEventListener</u> listener)	Добавляет заданное значение <code>listener</code> в конец этой очереди рисования.
void	<u>addNotify()</u>	Переопределяется для отслеживания добавления этого компонента в контейнер.
boolean	<u>areAllGLEventListenerInitialized()</u>	Возвращает true, если все добавленные <code>GLEventListener</code> инициализированы, в противном случае false .
<u>GLContext</u>	<u>createContext</u> (<u>GLContext</u> shareWith)	Создает новый контекст для рисования для этого объекта рисования, который при необходимости будут совместно использовать объекты буфера, текстуры и другие объекты OpenGL на стороне сервера с указанным GLContext.

void	<u>destroy()</u>	Просто псевдоним для removeNotify
void	<u>display()</u>	Вызывает выполнение рендеринга OpenGL для этого GLAutoDrawable в следующем порядке: вызов <code>display(..)</code> всех зарегистрированных <code>GLEventListener</code> файлов.
<u>GLEventListener</u>	<u>disposeGLEventListener</u> (<u>GLEventListener</u> listener, boolean remove)	Удаляет указанный <code>listener</code> через <code>dispose(..)</code> , если он был инициализирован и добавлен в эту очередь.
void	<u>flushGLRunnables()</u>	Сбрасывает все <code>enqueued GLRunnable</code> это <code>GLAutoDrawable</code> , включая уведомление ожидающего исполнителя.
<u>GLAnimatorControl</u>	<u>getAnimator()</u>	
boolean	<u>getAutoSwapBufferMode()</u>	Указывает, включена ли автоматическая замена буфера для этого объекта рисования.
<u>GLCapabilitiesImmutable</u>	<u>getChosenGLCapabilities()</u>	Извлекает <code>GLCapabilitiesImmutable</code> соответствующие выбранным возможностям OpenGL
<u>GLContext</u>	<u>getContext()</u>	Возвращает контекст, связанный с этим объектом рисования.
int	<u>getContextCreationFlags()</u>	
float[]	<u>getCurrentSurfaceScale</u> (float[] result)	Возвращает текущий масштаб пикселя связанного <code>NativeSurface</code> .
<u>AWTGLPixelBuffer.AWTGLPixelBufferProvider</u>	<u>getCustomPixelBufferProvider()</u>	
<u>WindowClosingProtocol.WindowClosingMode</u>	<u>getDefaultCloseOperation()</u>	
<u>GLDrawable</u>	<u>getDelegatedDrawable()</u>	Если реализация использует делегирование, возвращает делегированный экземпляр <code>GLDrawable</code> , в противном случае <code>this</code> .

<u>Thread</u>	<u>getExclusiveContextThread()</u>	
<u>GLDrawableFactory</u>	<u>getFactory()</u>	Возвращает значение <code>GLDrawableFactory</code> , используемое для создания этого экземпляра.
<u>GL</u>	<u>getGL()</u>	Возвращает объект <code>GL</code> конвейера, используемый этим <code>GLAutoDrawable</code> .
<u>GLEventListener</u>	<u>getGLEventListener(int index)</u>	Возвращает значение <code>GLEventListener</code> по заданному индексу этой очереди рисования.
int	<u>getGLEventListenerCount()</u>	Возвращает номер <code>GLEventListener</code> этой очереди для рисования.
boolean	<u>getGLEventListenerInitState(GLEventListener listener)</u>	Извлекает, инициализирован ли данный <code>listener</code> или нет.
<u>GLProfile</u>	<u>getGLProfile()</u>	Извлекает <code>GLProfile</code>
long	<u>getHandle()</u>	Возвращает дескриптор <code>GL</code> , который гарантированно будет действительным после <code>realization</code> и во время его создания <code>surfaceLocked</code> .
float[]	<u>getMaximumSurfaceScale(float[] result)</u>	Возвращает максимальный масштаб пикселя связанного <code>NativeSurface</code> .
float[]	<u>getMinimumSurfaceScale(float[] result)</u>	Возвращает минимальный масштаб пикселя связанного <code>NativeSurface</code> .
<u>NativeSurface</u>	<u>getNativeSurface()</u>	Возвращает значение, связанное <code>NativeSurfaceC</code> этим <code>NativeSurfaceHolder</code> .
<u>GLCapabilitiesImmutable</u>	<u>getRequestedGLCapabilities()</u>	Извлекает <code>GLCapabilitiesImmutable</code> соответствующие запрошенным пользователем возможностям OpenGL
float[]	<u>getRequestedSurfaceScale(float[] result)</u>	Возвращает <code>requested</code> пиксельный масштаб связанного <code>NativeSurface</code> .

boolean	<u>getSkipGLOrientationVerticalFlip()</u>	Смотри <code>setSkipGLOrientationVerticalFlip(boolean)</code> .
int	<u>getSurfaceHeight()</u>	Возвращает высоту клиентской области <code>GLDrawablesurface</code> в пикселях.
int	<u>getSurfaceWidth()</u>	Возвращает ширину клиентской области <code>GLDrawablesurface</code> в пикселях.
int	<u>getTextureUnit()</u>	Возвращает используемую текстурную единицу.
<u>RecursiveLock</u>	<u>getUpstreamLock()</u>	Возвращает рекурсивный объект блокировки <code>upstream widget</code> для синхронизации многопоточного доступа поверх <code>NativeSurface.lockSurface()</code> .
<u>Object</u>	<u>getUpstreamWidget()</u>	Метод <i>может</i> возвращать вышестоящий объект инструментария пользовательского интерфейса, содержащий этот экземпляр <code>GLAutoDrawable</code> , если он существует.
boolean	<u>initializeBackend</u> (boolean offthread)	Пытается инициализировать серверную часть, если она еще не инициализирована.
boolean	<u>invoke</u> (boolean wait, <u>GLRunnable</u> glRunnable)	Ставит в очередь одноразовый запрос <code>GLRunnable</code> , который будет выполнен в рамках следующего <code>GLAutoDrawable.display()</code> вызова после вызова всех зарегистрированных методов <code>GLEventListenerdisplay(GLAutoDrawable)</code> .
boolean	<u>invoke</u> (boolean wait, <u>List<GLRunnable></u> glRunnables)	Расширяет функциональность <code>GLAutoDrawable.invoke(boolean, GLRunnable)</code> , позволяя вводить список файлов <code>GLRunnable</code> .
boolean	<u>isGLOriented()</u>	Возвращает <code>true</code> , если отрисовываемый объект отображается в системе координат OpenGL, начало <i>координат внизу слева</i> .

boolean	<u>isRealized()</u>	Возвращает true, если эта возможность рисования реализована, в противном случае false.
boolean	<u>isThreadGLCapable()</u>	Указывает, способен ли текущий поток выполнять работу, связанную с OpenGL.
void	<u>print</u> (<u>Graphics</u> graphics)	
void	<u>releasePrint()</u>	Должен быть вызван после <code>PrinterJob.print()</code> .
<u>GLEventListener</u>	<u>removeGLEventListener</u> (<u>GLEventListener</u> listener)	Удаляет данные listener из этой очереди рисования.
void	<u>removeNotify()</u>	Переопределяется для отслеживания, когда этот компонент удаляется из контейнера.
void	<u>reshape</u> (int x, int y, int width, int height)	Переопределено, чтобы вызвать <code>GLDrawableHelper.reshape(com.jogamp.opengl.GLAutoDrawable, int, int, int, int)</code> ВЫЗОВ для всех зарегистрированных <code>GLEventListener</code> файлов.
void	<u>setAnimator</u> (<u>GLAnimatorControl</u> animatorControl)	Регистрирует использование аниматора, реализацию <code>GLAnimatorControl</code> .
void	<u>setAutoSwapBufferMode</u> (boolean enable)	Включает или отключает автоматическую замену буфера для этого чертежа.
<u>GLContext</u>	<u>setContext</u> (<u>GLContext</u> newCtx, boolean destroyPrevCtx)	Связывает новый контекст, newCtx, с этим автоматически извлекаемым.
void	<u>setContextCreationFlags</u> (int flags)	
<u>WindowClosingProtocol.WindowClosingMode</u>	<u>setDefaultCloseOperation</u> (<u>WindowClosingProtocol.WindowClosingMode</u> op)	

<u>Thread</u>	<u>setExclusiveContextThread</u> (<u>Thread</u> t)	Выделяет этот экземпляр <code>GLContext</code> для данного потока. Поток будет запрашивать исключительно <code>GLContext</code> через <code>GLAutoDrawable.display()</code> и не освобождать его до тех пор, пока <code>GLAutoDrawable.destroy()</code> не будет вызван или не будет вызван <code>setExclusiveContextThread(null)</code> .
<u>GL</u>	<u>setGL</u> (<u>GL</u> gl)	Задаёт объект <code>GL</code> конвейера, который использует этот <code>GLAutoDrawable</code> .
void	<u>setGLEventListenerInitState</u> (<u>GLEventListener</u> listener, boolean initialized)	Устанавливает заданное <code>listener</code> инициализированное состояние.
void	<u>setOpaque</u> (boolean opaque)	
void	<u>setPixelBufferProvider</u> (<u>AWTGLPixelBuffer</u> . <u>AWTGLPixelBufferProvider</u> custom)	
void	<u>setRealized</u> (boolean realized)	Указывает для реализаций <code>surfaceGLDrawable</code> , был ли создан базовый компонент и может ли он быть использован.
void	<u>setRequestedGLCapabilities</u> (<u>GLCapabilitiesImmutable</u> caps)	Задайте новый запрос <code>GLCapabilitiesImmutable</code> для этой <code>GLJPanel</code> , разрешающий реконфигурацию.
void	<u>setSharedAutoDrawable</u> (<u>GLAutoDrawable</u> sharedAutoDrawable)	Указывает <code>GLAutoDrawable</code> , который <code>OpenGL context</code> должен быть общим <code>GLAutoDrawable</code> для этого <code>GLContext</code> .
void	<u>setSharedContext</u> (<u>GLContext</u> sharedContext)	Указывает an <code>OpenGL context</code> , который должен быть общим <code>GLAutoDrawable</code> для этого <code>GLContext</code> .
void	<u>setSkipGLOrientationVerticalFlip</u> (boolean v)	<code>isGLOriented()</code> Вертикальный флип на основе пропуска, который обычно требуется серверной частью вне экрана,.

boolean	<u>setSurfaceScale</u> (float[] pixelScale)	Запрашивайте масштаб пикселей в направлениях x и y для связанного <code>NativeSurface</code> , где <code>size_in_pixel_units = pixel_scale * size_in_window_units</code> .
void	<u>setTextureUnit</u> (int v)	Позволяет пользователю запрашивать текстурный модуль для использования, который должен быть вызван перед первой инициализацией
void	<u>setupPrint</u> (double scaleMatX, double scaleMatY, int numSamples, int tileWidth, int tileHeight)	Должен быть вызван раньше <code>PrinterJob.print()</code> .
boolean	<u>shouldPreserveColorBufferIfTranslucent</u> ()	Для полупрозрачной <code>GLJPanel</code> (для которой <code>setOpaque</code> было вызвано (<code>false</code>)) указывает, должно ли приложение сохранять буфер цвета OpenGL (<code>GL_COLOR_BUFFER_BIT</code>) для корректного отображения <code>GLJPanel</code> и базовых виджетов, которые могут отображаться через части <code>GLJPanel</code> со значениями альфа меньше 1.
void	<u>swapBuffers</u> ()	Меняет местами передний и задний буферы
<u>String</u>	<u>toString</u> ()	

Класс `GLCapabilities`

Чтобы создать экземпляр классов `GLCanvas` или `GLJPanel`, необходим объект класса, реализующего интерфейс `GLCapabilitiesImmutable`, который определяет неизменяемый (`immutable`) набор возможностей OpenGL.

Один из способов получить объект интерфейса `CapabilitiesImmutable` - создать экземпляр класса `GLCapabilities`, который реализует этот интерфейс.

Класс `GLCapabilities` описывает желаемые возможности, которые должен поддерживать контекст рендеринга, например, профиль OpenGL.

Конструктор класса GLCapabilities

Конструктор	Описание
<u>GLCapabilities</u> (<u>GLProfile</u> glp)	Создает объект GLCapabilities.

Методы класса GLCapabilities

Модификатор и тип	Метод	Описание
<u>Object</u>	<u>clone</u> ()	
<u>Object</u>	<u>cloneMutable</u> ()	
int	<u>compareTo</u> (<u>CapabilitiesImmutable</u> o)	только сравнение hw / sw, стерео, мультисэмплов, трафаретов, RGBA и глубины
<u>GLCapabilities</u>	<u>copyFrom</u> (<u>CapabilitiesImmutable</u> source)	Копирует все <u>CapabilitiesImmutable</u> значения из source этого экземпляра.
<u>GLCapabilities</u>	<u>copyFrom</u> (<u>GLCapabilitiesImmutable</u> source)	Копирует все <u>GLCapabilitiesImmutable</u> значения из source этого экземпляра.
boolean	<u>equals</u> (<u>Object</u> obj)	Равенство по неизменяемым атрибутам обоих объектов
int	<u>getAccumAlphaBits</u> ()	Возвращает количество бит для альфа-компонента буфера накопления.
int	<u>getAccumBlueBits</u> ()	Возвращает количество бит для синего компонента накопительного буфера.
int	<u>getAccumGreenBits</u> ()	Возвращает количество бит для зеленого компонента накопительного буфера.
int	<u>getAccumRedBits</u> ()	Возвращает количество битов для красного компонента буфера накопления.
int	<u>getDepthBits</u> ()	Возвращает количество битов буфера глубины.
boolean	<u>getDoubleBuffered</u> ()	Возвращает, запрошена ли двойная буферизация, доступна или выбрана.

<u>GLProfile</u>	<u>getGLProfile</u> ()	Возвращает профиль GL, который вы хотите или который используется для рисования.
boolean	<u>getHardwareAccelerated</u> ()	Возвращает, запрошено ли аппаратное ускорение, доступно или выбрано.
int	<u>getNumSamples</u> ()	Возвращает количество буферов выборки, которые должны быть выделены, если буферы выборки включены, в противном случае возвращает 0.
boolean	<u>getSampleBuffers</u> ()	Возвращает, следует ли выделять буферы выборки для сглаживания всей сцены (FSAA) для этого чертежа.
<u>String</u>	<u>getSampleExtension</u> ()	Возвращает расширение для сглаживания всей сцены (FSAA).
int	<u>getStencilBits</u> ()	Возвращает количество битов буфера трафарета.
boolean	<u>getStereo</u> ()	Возвращает, запрошен ли stereo, доступен или выбран.
int	<u>hashCode</u> ()	хэш-код над неизменяемыми атрибутами обоих объектов
boolean	<u>isFBO</u> ()	Возвращает, запрошен ли, доступен или выбран ли закадровый режим FBO.
boolean	<u>isPBuffer</u> ()	Возвращает, запрошен ли, доступен или выбран закадровый режим pbuffer.
void	<u>setAccumAlphaBits</u> (int accumAlphaBits)	Задаёт количество битов, запрашиваемых для альфа-компонента накопительного буфера.
void	<u>setAccumBlueBits</u> (int accumBlueBits)	Задаёт количество битов, запрашиваемых для синего компонента накопительного буфера.
void	<u>setAccumGreenBits</u> (int accumGreenBits)	Задаёт количество битов, запрашиваемых для зеленого компонента накопительного буфера.
void	<u>setAccumRedBits</u> (int accumRedBits)	Задаёт количество битов, запрашиваемых для красного компонента накопительного буфера.
void	<u>setDepthBits</u> (int depthBits)	Задаёт количество битов, запрашиваемых для буфера глубины.
void	<u>setDoubleBuffered</u> (boolean enable)	Включает или отключает двойную буферизацию.

void	<u>setFBO</u> (boolean enable)	Запрашивает закадровый режим FBO.
void	<u>setGLProfile</u> (<u>GLProfile</u> profile)	Устанавливает желаемый профиль GL
void	<u>setHardwareAccelerated</u> (boolean enable)	Включает или отключает аппаратное ускорение.
void	<u>setNumSamples</u> (int numSamples)	Если буферы выборки включены, указывает количество буферов, которые должны быть выделены.
void	<u>setPBuffer</u> (boolean enable)	Запрашивает закадровый режим pbuffer.
void	<u>setSampleBuffers</u> (boolean enable)	По умолчанию используется значение false. Указывает, следует ли выделять буферы выборки для сглаживания всей сцены (FSAA) для этого чертежа. Имейте в виду, что для этого требуется альфа-компонент. Если этот метод включен, он также вызывает <code>setAlphaBits(1)</code> если <code>Capabilities.getAlphaBits() == 0</code> .
void	<u>setSampleExtension</u> (<u>String</u> se)	Задаёт желаемое расширение для сглаживания всей сцены (FSAA), по умолчанию <code>GLCapabilitiesImmutable.DEFAULT_SAMPLE_EXTENSION</code> .
void	<u>setStencilBits</u> (int stencilBits)	Задаёт количество битов, запрашиваемых для буфера трафарета.
void	<u>setStereo</u> (boolean enable)	Включает или отключает просмотр стерео.
<u>String</u>	<u>toString</u> ()	Возвращает текстовое представление этого объекта GLCapabilities.
<u>StringBuilder</u>	<u>toString</u> (<u>StringBuilder</u> sink)	Возвращает текстовое представление этого объекта.

Класс GLProfile

Поскольку было выпущено несколько версий OpenGL API, то необходимо указать версию OpenGL API, используемую в вашей программе. Это делается с помощью класса GLProfile.

Класс GLProfile задаёт профиль OpenGL. Статическая одноэлементная инициализация этого класса запрашивает доступность всех профилей OpenGL и создает

экземпляры одноэлементных объектов `GLProfile` для каждого доступного профиля. Профиль платформы по умолчанию может указываться, используя вызов `GLProfile.GetProfileDefault()`, или более специализированные версии, использующие другие статические методы `getProfile`.

Поля класса `GLProfile`

Модификатор и тип	Поле	Описание
<code>static boolean</code>	<code>DEBUG</code>	
<code>static boolean</code>	<code>disabledEGL</code>	В случае, если реализация EGL не доступна, как на платформе <code>Platform.OSType.IOS</code> устанавливается значение <code>true</code> .
<code>static boolean</code>	<code>disableOpenGLARBContext</code>	В случае, если реализация расширения для создания контекста <code>ARB_create_context</code> дает сбой на платформе, установленное свойство <code>jogl.disable.openglarbcontext</code> отключает его использование.
<code>static boolean</code>	<code>disableOpenGLCore</code>	В случае, если собственные профили ядра OpenGL не требуются и если одна платформа может иметь ошибочную реализацию, установка свойства <code>jogl.disable.openglcore</code> отключает запрос возможных существующих собственных профилей ядра OpenGL.
<code>static boolean</code>	<code>disableOpenGLDesktop</code>	В случае, если профили рабочего стола OpenGL не требуются и если платформа может иметь ошибочную реализацию, установка свойства <code>jogl.disable.opengldesktop</code> отключает запрос возможных существующих профилей рабочего стола OpenGL.
<code>static boolean</code>	<code>disableOpenGLES</code>	В случае, если профили OpenGL ES не требуются и если платформа может иметь ошибочную реализацию, установка свойства <code>jogl.disable.opengles</code> отключает запрос возможных существующих профилей OpenGL ES.
<code>static boolean</code>	<code>disableSurfacelessContext</code>	Отключите возможность бесповерхностного контекста OpenGL и его зондирования, установив свойство <code>jogl.disable.surfacelesscontext</code> .

static boolean	<u>enableANGLE</u>	Необходимо отключить поддержку ANGLE, эмуляции D3D ES2 в Windows, предоставляемой с Firefox и Chrome.
static <u>String</u> []	<u>GL_PROFILE_LIST_ALL</u>	Все профили GL в порядке определения по умолчанию.
static <u>String</u> []	<u>GL_PROFILE_LIST_MAX</u>	Порядок максимального количества профилей.
static <u>String</u> []	<u>GL_PROFILE_LIST_MAX_FIXEDFUNC</u>	Порядок максимальных фиксированных функциональных профилей GL4bc GL3bc GL2 GLES1
static <u>String</u> []	<u>GL_PROFILE_LIST_MAX_MOBILE</u>	Заказ максимального количества оригинальных мобильных профилей.
static <u>String</u> []	<u>GL_PROFILE_LIST_MAX_PROGSHADER</u>	Порядок максимального количества программируемых шейдерных профилей GL4bc GL4 GL3bc GL3 GLES3 GL2 GLES2
static <u>String</u> []	<u>GL_PROFILE_LIST_MAX_PROGSHADER_CORE</u>	Порядок максимального количества программируемых шейдерных <i>ядер</i> <i>профилирует только</i> GL4 GL3 GLES3 GLES2
static <u>String</u> []	<u>GL_PROFILE_LIST_MIN</u>	Порядок минимальных профилей.
static <u>String</u> []	<u>GL_PROFILE_LIST_MIN_DESKTOP</u>	Заказ минимального количества оригинальных профилей рабочего стола.
static <u>String</u>	<u>GL2</u>	Рабочий стол OpenGL profile 1.x до версии 3.0
static <u>String</u>	<u>GL2ES1</u>	Пересечение рабочего GL2 и встроенного ES1 профиля
static <u>String</u>	<u>GL2ES2</u>	Пересечение рабочего GL3, GL2 и встроенного профиля ES2
static <u>String</u>	<u>GL2GL3</u>	Пересечение профилей GL3 и GL2 для рабочего стола
static <u>String</u>	<u>GL3</u>	Рабочий стол OpenGL core profile 3.x, с x>= 1
static <u>String</u>	<u>GL3bc</u>	Профиль совместимости рабочего стола с OpenGL 3.x, с x>= 1, т.Е. GL2 плюс GL3. bc расшифровывается как обратная совместимость.

static <u>String</u>	<u>GL4</u>	Рабочий стол OpenGL core profile 4.x, с x>= 0
static <u>String</u>	<u>GL4bc</u>	Профиль совместимости рабочего стола с OpenGL 4.x, с x> = 0, т.Е. GL2 плюс GL4. bc расшифровывается как обратная совместимость.
static <u>String</u>	<u>GL4ES3</u>	Пересечение профиля рабочего стола GL4 и ES3, доступное только при наличии ES3 или GL4 w/GL_ARB_ES3_compatibility.
static <u>String</u>	<u>GLES1</u>	Встроенный профиль OpenGL составляет 1.x, при этом x >= 0
static <u>String</u>	<u>GLES2</u>	Встроенный профиль OpenGL ES 2.x, с x>= 0
static <u>String</u>	<u>GLES3</u>	Встроенный профиль OpenGL - это 3.x, с x>= 0

Методы класса GLProfile

Модификатор и тип	Метод	Описание
static <u>GLProfile</u>	<u>createCustomGLProfile</u> (<u>String</u> profile, <u>GLProfile</u> profileImpl)	
boolean	<u>equals</u> (<u>Object</u> o)	
static <u>GLProfile</u>	<u>get</u> (<u>AbstractGraphicsDevice</u> device, <u>String</u> profile)	Возвращает объект GLProfile.
static <u>GLProfile</u>	<u>get</u> (<u>AbstractGraphicsDevice</u> device, <u>String</u> [] profiles, boolean favorHardwareRasterizer)	Возвращает первый профиль из заданного списка, в котором доступна реализация.
static <u>GLProfile</u>	<u>get</u> (<u>String</u> profile)	Использует устройство по умолчанию
static <u>GLProfile</u>	<u>get</u> (<u>String</u> [] profiles, boolean favorHardwareRasterizer)	Использует устройство по умолчанию
static <u>GLProfile</u>	<u>getDefault</u> ()	Возвращает объект GLProfile по умолчанию, отражающий наилучший вариант для текущей платформы.

static <u>GLProfile</u>	<u>getDefault</u> (<u>AbstractGraphicsDevice</u> device)	Возвращает объект GLProfile по умолчанию, отражающий наилучший вариант для текущей платформы.
static <u>AbstractGraphicsDevice</u>	<u>getDefaultDevice</u> ()	
static <u>GLProfile</u>	<u>getGL2ES1</u> ()	Вызовы <u>getGL2ES1</u> (<u>AbstractGraphicsDevice</u>) с использованием устройства по умолчанию.
static <u>GLProfile</u>	<u>getGL2ES1</u> (<u>AbstractGraphicsDevice</u> device)	Возвращает реализацию профиля GL2ES1, следовательно, совместимую с GL2ES1. Он возвращается:
static <u>GLProfile</u>	<u>getGL2ES2</u> ()	Вызовы <u>getGL2ES2</u> (<u>AbstractGraphicsDevice</u>) с использованием устройства по умолчанию.
static <u>GLProfile</u>	<u>getGL2ES2</u> (<u>AbstractGraphicsDevice</u> device)	Возвращает реализацию профиля GL2ES2, следовательно, совместимую с GL2ES2. Он возвращается:
static <u>GLProfile</u>	<u>getGL2GL3</u> ()	Вызовы <u>getGL2GL3</u> (<u>AbstractGraphicsDevice</u>) с использованием устройства по умолчанию.
static <u>GLProfile</u>	<u>getGL2GL3</u> (<u>AbstractGraphicsDevice</u> device)	Возвращает реализацию профиля GL2GL3, следовательно, совместимую с GL2GL3. Он возвращается:
static <u>GLProfile</u>	<u>getGL4ES3</u> ()	Вызовы <u>getGL4ES3</u> (<u>AbstractGraphicsDevice</u>) с использованием устройства по умолчанию.
static <u>GLProfile</u>	<u>getGL4ES3</u> (<u>AbstractGraphicsDevice</u> device)	Возвращает реализацию профиля GL4ES3, следовательно, совместимую с GL4ES3. Он возвращается:

static <u>String</u>	<u>getGLArrayName</u> (int array)	
<u>Constructor</u> <?>	<u>getGLCtor</u> (boolean glObject)	
<u>String</u>	<u>getGLImplBaseClassName</u> ()	
static <u>String</u>	<u>getGLTypeName</u> (int type)	
<u>GLProfile</u>	<u>getImpl</u> ()	верните эту реализацию профилей, например.
<u>String</u>	<u>getImplName</u> ()	возвращает имя реализации этого профиля, например.
static <u>GLProfile</u>	<u>getMaxFixedFunc</u> (boolean favorHardwareRasterizer)	Использует устройство по умолчанию
static <u>GLProfile</u>	<u>getMaxFixedFunc</u> (<u>AbstractGraphicsDevice</u> device, boolean favorHardwareRasterizer)	Возвращает самый высокий профиль, реализуя конвейер фиксированных функций.
static <u>GLProfile</u>	<u>getMaximum</u> (boolean favorHardwareRasterizer)	Использует устройство по умолчанию
static <u>GLProfile</u>	<u>getMaximum</u> (<u>AbstractGraphicsDevice</u> device, boolean favorHardwareRasterizer)	Возвращает самый высокий профиль.
static <u>GLProfile</u>	<u>getMaxProgrammable</u> (boolean favorHardwareRasterizer)	Использует устройство по умолчанию
static <u>GLProfile</u>	<u>getMaxProgrammable</u> (<u>AbstractGraphicsDevice</u> device, boolean favorHardwareRasterizer)	Возвращает самый высокий профиль, реализуя программируемый конвейер шейдеров.
static <u>GLProfile</u>	<u>getMaxProgrammableCore</u> (boolean favorHardwareRasterizer)	Использует устройство по умолчанию
static <u>GLProfile</u>	<u>getMaxProgrammableCore</u> (<u>AbstractGraphicsDevice</u> device, boolean favorHardwareRasterizer)	Возвращает самый высокий профиль, реализуя <i>только</i> программируемый конвейер шейдерного ядра.
static <u>GLProfile</u>	<u>getMinimum</u> (boolean favorHardwareRasterizer)	Использует устройство по умолчанию

static <u>GLProfile</u>	<u>getMinimum</u> (<u>AbstractGraphicsDevice</u> device, boolean favorHardwareRasterizer)	Возвращает самый низкий профиль.
<u>String</u>	<u>getName</u> ()	верните это имя профиля
static <u>String</u>	<u>glAvailabilityToString</u> ()	Использует устройство по умолчанию
static <u>String</u>	<u>glAvailabilityToString</u> (<u>AbstractGraphicsDevice</u> device)	
static <u>StringBuilder</u>	<u>glAvailabilityToString</u> (<u>AbstractGraphicsDevice</u> device, <u>S</u> <u>tringBuilder</u> sb)	
static <u>StringBuilder</u>	<u>glAvailabilityToString</u> (<u>AbstractGraphicsDevice</u> device, <u>S</u> <u>tringBuilder</u> sb, <u>String</u> indent, int indentCount)	
boolean	<u>hasGLSL</u> ()	Указывает, поддерживает ли данный профиль GLSL, т.е.
int	<u>hashCode</u> ()	
static void	<u>initProfiles</u> (<u>AbstractGraphicsDevice</u> device)	Инициализируйте активную инициализацию GLProfiles для данного устройства, если это еще не сделано.
static void	<u>initSingleton</u> ()	Статическая инициализация JOGL.
static boolean	<u>isAnyAvailable</u> ()	Возвращает доступность любого профиля на устройстве по умолчанию.
static boolean	<u>isAvailable</u> (<u>AbstractGraphicsDevice</u> device, <u>S</u> <u>tring</u> profile)	Возвращает доступность профиля на устройстве.
static boolean	<u>isAvailable</u> (<u>String</u> profile)	Возвращает доступность профиля на устройстве по умолчанию.
static boolean	<u>isAWTAvailable</u> ()	
boolean	<u>isGL2</u> ()	Указывает , поддерживает ли этот профиль GL2 .

boolean	<u>isGL2ES1()</u>	Указывает, поддерживает ли этот профиль GL2ES1.
boolean	<u>isGL2ES2()</u>	Указывает, поддерживает ли этот профиль GL2ES2.
boolean	<u>isGL2ES3()</u>	Указывает, поддерживает ли этот профиль GL2ES3.
boolean	<u>isGL2GL3()</u>	Указывает, поддерживает ли этот профиль GL2GL3.
boolean	<u>isGL3()</u>	Указывает, поддерживает ли этот профиль GL3.
boolean	<u>isGL3bc()</u>	Указывает, поддерживает ли этот профиль GL3bc.
boolean	<u>isGL3ES3()</u>	Указывает, поддерживает ли этот профиль GL3ES3.
boolean	<u>isGL4()</u>	Указывает, поддерживает ли этот профиль GL4.
boolean	<u>isGL4bc()</u>	Указывает, поддерживает ли этот профиль GL4bc.
boolean	<u>isGL4ES3()</u>	Указывает, поддерживает ли этот профиль GL4ES3.
boolean	<u>isGLES()</u>	Указывает, поддерживает ли этот профиль GLES.
boolean	<u>isGLES1()</u>	Указывает, поддерживает ли этот профиль GLES1.
boolean	<u>isGLES2()</u>	Указывает, поддерживает ли этот профиль GLES2.
boolean	<u>isGLES3()</u>	Указывает, поддерживает ли этот профиль GLES3.
boolean	<u>isHardwareRasterizer()</u>	возвращает true, если impl.
static boolean	<u>isInitialized()</u>	
boolean	<u>isValidArrayType</u> (int index, int comps, int type, boolean isVertexAttribPointer, boolean throwException)	

boolean	<u>isValidDataType</u> (int type, boolean throwException)	Общая проверка, если тип является допустимым типом данных GL для текущего профиля
static void	<u>shutdown</u> ()	Метод ручного завершения работы может быть вызван после последнего использования JOGL в запущенной JVM. Он освобождает все временно созданные ресурсы, то есть проблемы <code>GLDrawableFactory.shutdown()</code> . Реализация завершения работы вызывается через перехват завершения работы JVM, если не вызывается вручную.
<u>String</u>	<u>toString</u> ()	
boolean	<u>usesNativeGLES</u> ()	Указывает, использует ли этот профиль любую из нативных реализаций OpenGL ES.
static boolean	<u>usesNativeGLES</u> (<u>String</u> profileImpl)	Указывает, используется ли какой-либо из нативных профилей OpenGL ES.
boolean	<u>usesNativeGLES1</u> ()	Указывает, использует ли этот профиль нативных реализации OpenGL ES1.
static boolean	<u>usesNativeGLES1</u> (<u>String</u> profileImpl)	Указывает, используется ли собственный профиль OpenGL ES1.
boolean	<u>usesNativeGLES2</u> ()	Указывает, использует ли этот профиль собственные реализации OpenGL ES2.
static boolean	<u>usesNativeGLES2</u> (<u>String</u> profileImpl)	Указывает, используется ли собственный профиль OpenGL ES3 или ES2.
boolean	<u>usesNativeGLES3</u> ()	Указывает, использует ли этот профиль собственные реализации OpenGL ES3.

static boolean	<u>usesNativeGLS3</u> (<u>String</u> profileImpl)	Указывает, используется ли собственный профиль OpenGL ES2.
void	<u>verifyEquality</u> (<u>GLProfile</u> glp)	

Пример. Пытаемся понять профили, возможности и т.п.

```
import javax.swing.*;
import static com.jogamp.opengl.GL2.*;
import com.jogamp.opengl.*;
import com.jogamp.opengl.awt.GLCanvas;

public class Code extends JFrame implements GLEventListener{
    private static final long serialVersionUID = 1L;
    private GLCanvas myCanvas;

    public Code(){
        setTitle(" ");
        setSize(600, 400);
        setLocation(200, 200);

        myCanvas = new GLCanvas();

        //Свойства только что созданного GLCanvas
        System.out.println("Свойства только что созданного GLCanvas с конструктором по
        умолчанию");
        System.out.println(myCanvas);

        myCanvas.addGLEventListener(this);

        this.add(myCanvas);
        setVisible(true);

        System.out.println("Свойства GLCanvas, после того как к нему подключили слушателя");
        System.out.println("и добавили его в окно JFrame");
        System.out.println(myCanvas);
        //System.out.println();
    }

    public void display(GLAutoDrawable drawable){
        GL2 gl = (GL2) GLContext.getCurrentGL();

        gl.glClearColor(0.1f, 0.5f, 0.5f, 0.5f);
        gl.glClear(GL_COLOR_BUFFER_BIT);
        gl.glColor3f(1.0f, 0.0f, 0.0f);
        gl.glRecti(0, 0, 100, 100);
        gl.glFlush();
    }

    public static void main(String[] args){
        new Code();
    }

    public void init(GLAutoDrawable drawable) {

        // Профиль
        GLProfile glp = GLProfile.getDefault();
```

```

System.out.println("Возможность использования профилей");
System.out.println(GLProfile.glAvailabilityToString());

System.out.println("Максимальный профиль");
System.out.println(GLProfile.getMaximum(true));

System.out.println("Профиль по умолчанию");
System.out.println(glp);

System.out.println("Аппаратная растеризация?");
System.out.println(glp.isHardwareRasterizer());

System.out.println("Имя профиля");
System.out.println(glp.getImplName());

// ВОЗМОЖНОСТИ
GLCapabilities cap = new GLCapabilities(glp);
System.out.println("Возможности для профиля");
System.out.println(cap);

System.out.println("Возможности для профиля - установлена альфа 16 бит");
cap.setAlphaBits(16);
System.out.println(cap);
}

public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) { }

public void dispose(GLAutoDrawable drawable) { }
}

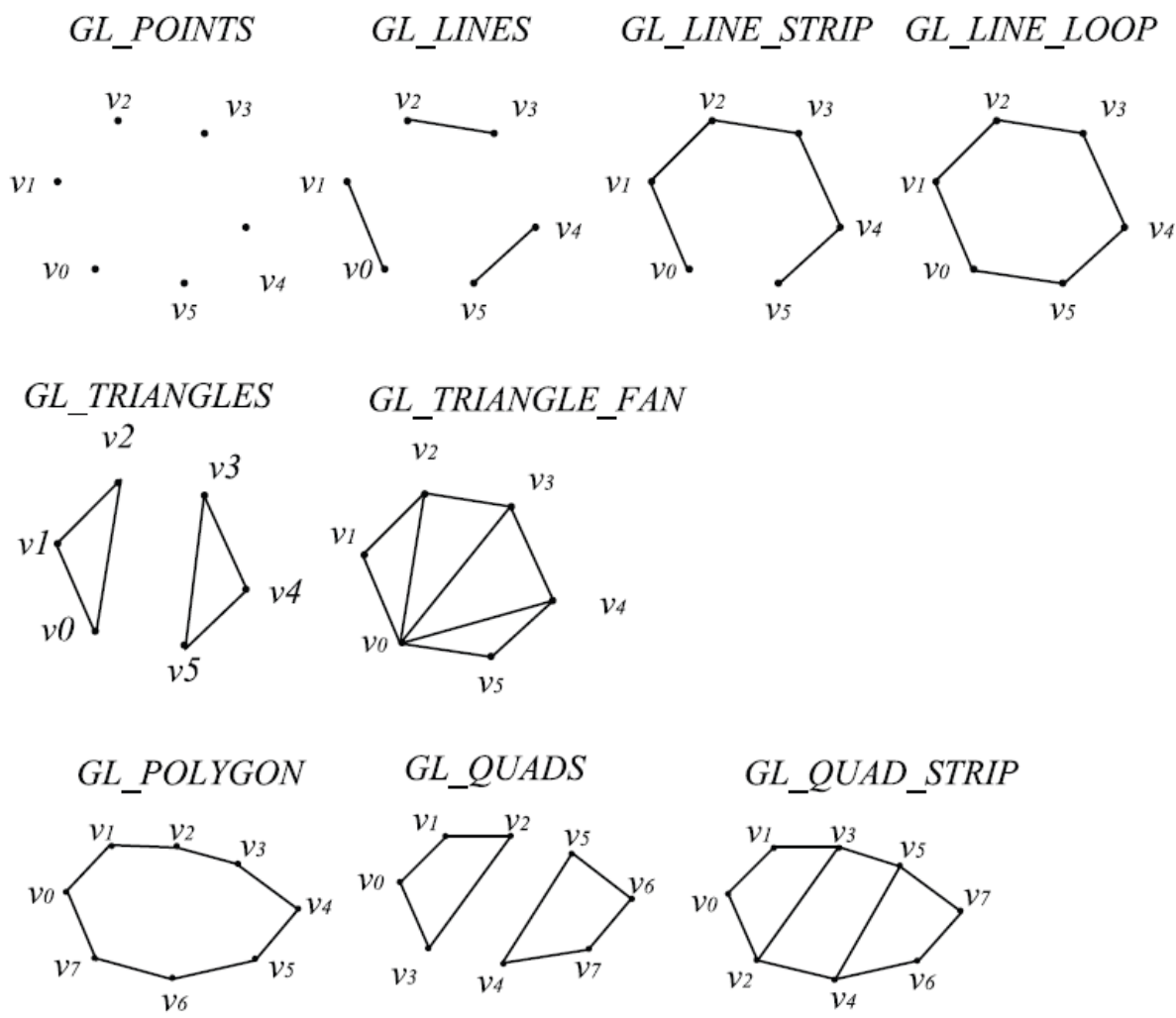
```

Двумерная графика на JOGL

Ниже приведены различные встроенные параметры OpenGL, поддерживаемые JOGL:

Примитивный	Описание
GL_POINT	Каждая вершина рассматривается как отдельная точка
GL_LINES	Он обрабатывает каждую пару вершин как отрезок прямой.
GL_LINES_STRIP	Соединяет группу отрезков друг с другом.
GL_LINES_LOOP	Соединяет группу отрезков в цикле, т.е. от первого к последнему, а затем обратно к первому.
GL_TRIANGLE	Каждая вершина триплета ведет себя как независимый треугольник

GL_TRIANGLE_FAN	Создает связную группу треугольников, где каждый треугольник определен для определенной вершины, представленной после первых двух вершин.
GL_TRIANGLE_STRIP	Соединяет группу треугольников друг с другом.
GL_QUADS	Каждая группа из четырех вершин ведет себя как независимый четырехугольник.
GL_QUAD_STRIP	Соединяет группу четырехугольников друг с другом.
GL_POLYGON	Рисует один выпуклый многоугольник.



Исследуем возможности

```
import javax.swing.*;
import static com.jogamp.opengl.GL2.*;
import com.jogamp.opengl.*;
import com.jogamp.opengl.awt.GLCanvas;

public class Code extends JFrame implements GLEventListener{
    private static final long serialVersionUID = 1L;
    private GLCanvas myCanvas;

    public Code(){
        setTitle(" ");
        setSize(600, 400);
        setLocation(200, 200);

        myCanvas = new GLCanvas();

        //Свойства только что созданного GLCanvas
        System.out.println("Свойства только что созданного GLCanvas с конструктором по
        умолчанию");
        System.out.println(myCanvas);

        myCanvas.addGLEventListener(this);

        this.add(myCanvas);
        setVisible(true);

        System.out.println("Свойства GLCanvas, после того как к нему подключили слушателя");
        System.out.println("и добавили его в окно JFrame");
        System.out.println(myCanvas);
        //System.out.println();
    }

    public void display(GLAutoDrawable drawable){
        GL2 gl = (GL2) GLContext.getCurrentGL();

        gl.glClearColor(0.1f, 0.5f, 0.5f, 0.5f);
        gl.glClear(GL_COLOR_BUFFER_BIT);
        gl.glColor3f(1.0f, 0.0f, 0.0f);
        gl.glRecti(0, 0, 100, 100);
        gl.glFlush();
    }

    public static void main(String[ ] args){
        new Code();
    }

    public void init(GLAutoDrawable drawable) {

        // Профиль
        GLProfile glp = GLProfile.getDefault();

        System.out.println("Возможность использования профилей");
        System.out.println(GLProfile.glAvailabilityToString());

        System.out.println("Максимальный профиль");
        System.out.println(GLProfile.getMaximum(true));

        System.out.println("Профиль по умолчанию");
        System.out.println(glp);
    }
}
```

```

        System.out.println("Аппаратная растеризация?");
        System.out.println(glp.isHardwareRasterizer());

        System.out.println("Имя профиля");
        System.out.println(glp.getImplName());

        // Возможности
        GLCapabilities cap = new GLCapabilities(glp);
        System.out.println("Возможности для профиля");
        System.out.println(cap);

        System.out.println("Возможности для профиля - установлена альфа 16 бит");
        cap.setAlphaBits(16);
        System.out.println(cap);
    }

    public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) { }

    public void dispose(GLAutoDrawable drawable) { }
    }

```

Пример. Рисуем двумерную линию

```

import javax.swing.JFrame;

import com.jogamp.opengl.GL2;
import com.jogamp.opengl.GLAutoDrawable;
import com.jogamp.opengl.GLCapabilities;
import com.jogamp.opengl.GLEventListener;
import com.jogamp.opengl.GLProfile;
import com.jogamp.opengl.awt.GLCanvas;

public class BasicLine implements GLEventListener {

    @Override
    public void init(GLAutoDrawable arg0)
    {

    }

    @Override
    public void display(GLAutoDrawable drawable) {
        final GL2 gl = drawable.getGL().getGL2();

        //Каждую пару точек трактуем как отдельную линию
        gl.glBegin (GL2.GL_LINES);
        gl.glVertex2d(-0.60, 0.10);
        gl.glVertex2d(0.60, 0.10);
        gl.glVertex2d(0.10, -0.60);
        gl.glVertex2d(0.10, 0.60);
        gl.glEnd();

    }

    @Override
    public void reshape(GLAutoDrawable arg0, int arg1, int arg2, int arg3, int arg4)
    {

    }

    @Override

```

```

public void dispose(GLAutoDrawable arg0)
{

}

public static void main(String[] args) {

    final GLProfile gp = GLProfile.get(GLProfile.GL2);
    GLCapabilities cap = new GLCapabilities(gp);
    final GLCanvas gc = new GLCanvas(cap);

    BasicLine bl = new BasicLine();
    gc.addGLEventListener(bl);

    final JFrame frame = new JFrame ("Линия");
    frame.add(gc);
    frame.setSize(500,400);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
}
}

```

Рисуем простые трехмерные фигуры

```

import java.awt.Dimension;
import java.awt.Frame;
import java.awt.Toolkit;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;

import javax.swing.JFrame;

import com.jogamp.opengl.GL2;
import com.jogamp.opengl.GLAutoDrawable;
import com.jogamp.opengl.GLCapabilities;
import com.jogamp.opengl.GLEventListener;
import com.jogamp.opengl.GLProfile;
import com.jogamp.opengl.awt.GLCanvas;
import com.jogamp.opengl.glu.GLU;
import com.jogamp.opengl.util.gl2.GLUT;

public class JOGL3dDemo implements GLEventListener {
    //GL2 gl;
    private GLUT glut = new GLUT();
    private GLU glu = new GLU();

    public static void main(String[] args) {
        // Получаем совместимость объекта с профилем GL2
        final GLProfile profile = GLProfile.get(GLProfile.GL2);
        GLCapabilities capabilities = new GLCapabilities(profile);
        // Холст
        final GLCanvas glcanvas = new GLCanvas(capabilities);

        JOGL3dDemo b = new JOGL3dDemo();
        glcanvas.addGLEventListener(b);

        //Создание фрейма
        final JFrame frame = new JFrame ("Трехмерные фигуры");
    }
}

```

```

//System.out.println(frame);
//Добавление холста к фрейму
frame.add(glcanvas);

//Определяем размер фрейма так
//или так - размер фрейма равен размеру экрана
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
frame.setSize(screenSize.width, screenSize.height);
glcanvas.setSize(screenSize.width, screenSize.height);

frame.addWindowListener(new WindowListener() {

    @Override
    public void windowActivated(WindowEvent arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void windowClosed(WindowEvent arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void windowClosing(WindowEvent arg0) {
        System.exit(0);

    }

    @Override
    public void windowDeactivated(WindowEvent arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void windowDeiconified(WindowEvent arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void windowIconified(WindowEvent arg0) {
        // TODO Auto-generated method stub

    }

    @Override
    public void windowOpened(WindowEvent arg0) {
        // TODO Auto-generated method stub

    }

});
frame.setVisible(true);

}

@Override
public void display(GLAutoDrawable drawable) {
    GL2 gl = drawable.getGL().getGL2();

```

```

gl.glMatrixMode(GL2.GL_PROJECTION);
gl.glLoadIdentity();
gl.glOrtho(-2, 2, -2, 2, 0.1, 100);

gl.glMatrixMode(GL2.GL_MODELVIEW);
gl.glLoadIdentity();
glu.gluLookAt(1.0, 1.0, 2.0, 0, 0, 0, 0, 1, 0);

gl.glClear(GL2.GL_COLOR_BUFFER_BIT);
gl.glColor3ub((byte)255, (byte)0, (byte)0);
// ось z
this.axis(drawable, 0.5);
// ось y
gl.glPushMatrix();
gl.glRotated(90, 0, 1, 0);
this.axis(drawable, 0.5);
gl.glPopMatrix();

// ось x
//gl.glPushMatrix();
gl.glRotated(-90, 1, 0, 0);
this.axis(drawable, 0.5);
gl.glPopMatrix();

// Куб
gl.glPushMatrix();
gl.glTranslated(0.7, 0.7, 0.7);
glut.glutWireCube((float)1.0);
gl.glPopMatrix();

// Сфера
gl.glPushMatrix();
gl.glTranslated(1.0, 1.0, 0);
glut.glutWireSphere(0.2, 20, 18);
gl.glPopMatrix();

// Конус
gl.glPushMatrix();
gl.glTranslated(1.0, 0, 1.0);
glut.glutWireCone(0.2, 0.5, 10, 8);
gl.glPopMatrix();

// Чайник
gl.glPushMatrix();
gl.glRotated(90, 90, 90, 0);
gl.glTranslated(-0.5, -0.5, -0.5);
glut.glutWireTeapot(0.2);
gl.glPopMatrix();
}

@Override
public void dispose(GLAutoDrawable arg0) {
    // TODO Auto-generated method stub
}

@Override
public void init(GLAutoDrawable arg0) {
    // TODO Auto-generated method stub
}

```

```

@Override
public void reshape(GLAutoDrawable arg0, int arg1, int arg2, int arg3, int
arg4) {
    // TODO Auto-generated method stub

}
// Рисуем ось
void axis (GLAutoDrawable drawable, double lenght) {
    final GL2 gl = drawable.getGL().getGL2();
    gl.glPushMatrix();

    gl.glBegin(GL2.GL_LINES);
        gl.glVertex3d(0, 0, 0);
        gl.glVertex3d(0, 0, lenght);
    gl.glEnd();

    gl.glTranslated(0, 0, lenght - 0.2);
    glut.glutWireCone(0.04, 0.2, 12, 9);
    gl.glPopMatrix();
}
}

```

Варианты заданий

