Прежде чем начать использовать JOGL, необходимо скачать и инсталлировать 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.

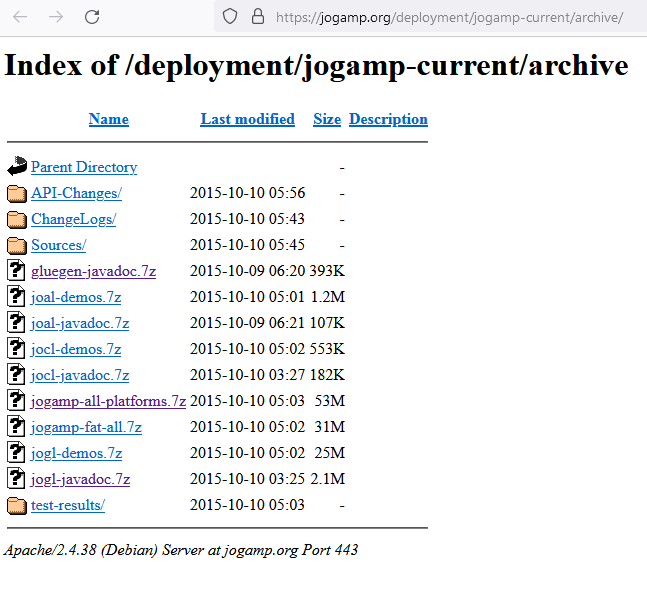


Рисунок 1. Страница загрузки стабильной версии JOGL

Скачайте и распакуйте в каком-либо каталоге файл [jogamp-all-platforms.7z](https://jogamp.org/deployment/jogamp-current/archive/jogamp-all-platforms.7z). Содержимое этого каталога показано на рис. 2.

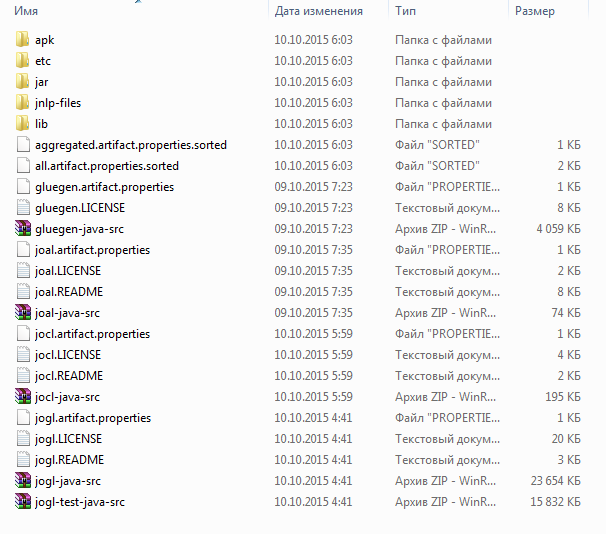


Рисунок 2. Содержимое каталога после распаковки файла jogamp-all-platforms.7z.

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

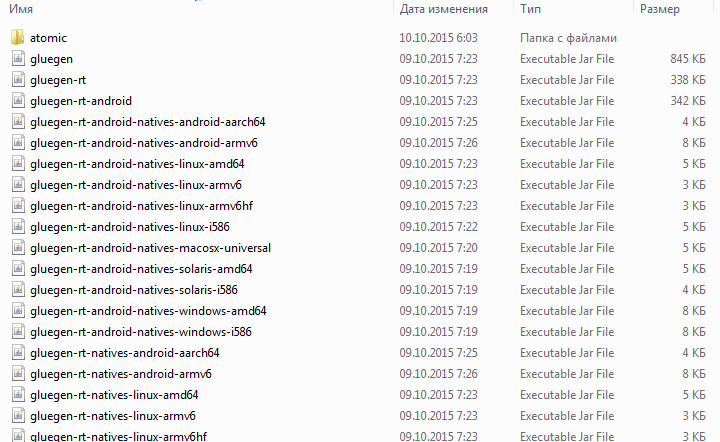


Рисунок 3. Часть содержимого каталога jar

Для каждой платформы необходимы определенные файлы.

|  |  |  |
| --- | --- | --- |
| **64-bit Windows** | **64-bit Linux** | **32/64-bit Mac** |
| gluegen-rt.jar | gluegen-rt.jar | gluegen-rt.jar |
| jogl-all.jar | jogl-all.jar | jogl-all.jar |
| gluegen-java-src.zip | gluegen-java-src.zip | gluegen-java-src.zip |
| jogl-java-src.zip | jogl-java-src.zip | jogl-java-src.zip |
| gluegen-rt-natives-windows-amd64.jar | gluegen-rt-natives-linux-amd64.jar | gluegen-rt-natives-macosx-universal.jar |
| jogl-all-natives-windows-amd64.jar | jogl-all-natives-linux-amd64.jar | jogl-all-natives-macosx-universal.jar |

|  |  |
| --- | --- |
| **32-bit Windows** | **32-bit Linux** |
| gluegen-rt.jar | gluegen-rt.jar |
| jogl-all.jar | jogl-all.jar |  |
| gluegen-java-src.zip | gluegen-java-src.zip |  |
| jogl-java-src.zip | jogl-java-src.zip |  |
| gluegen-rt-natives-windows-i586.jar | gluegen-rt-natives-linux-i586.jar |  |
| jogl-all-natives-windows-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 *(создайте папку)* ***в каталоге вашего рабочего пространства Eclipse***. Если это платформа **Windows 64-bit**, то выберите следующие файлы:

- gluegen-rt.jar,

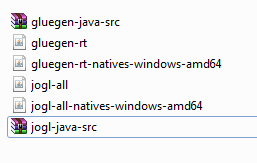
- jogl-all.jar,

- 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). *Создастся проект поверх созданного каталога в прошлом пункте.*

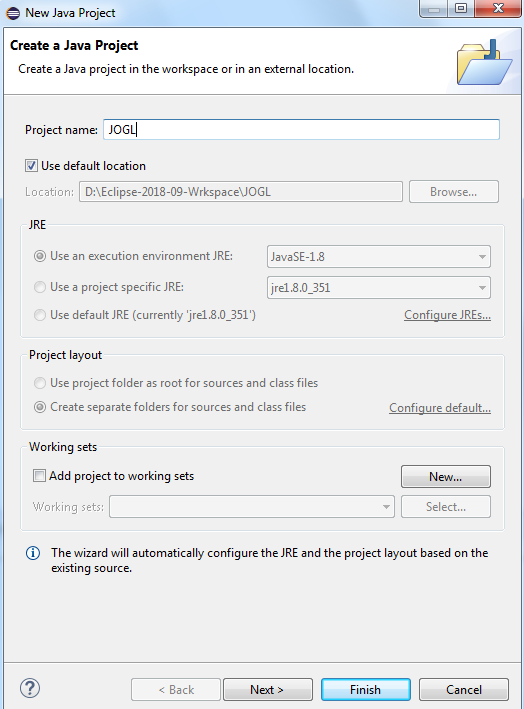


Рисунок 4. Создание проекта с именем «JOGL»

* 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.

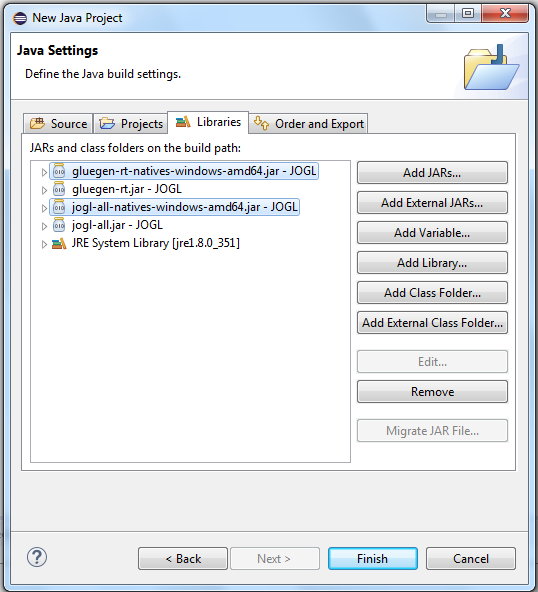


Рисунок 5. Удаление нативных JAR

* 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).

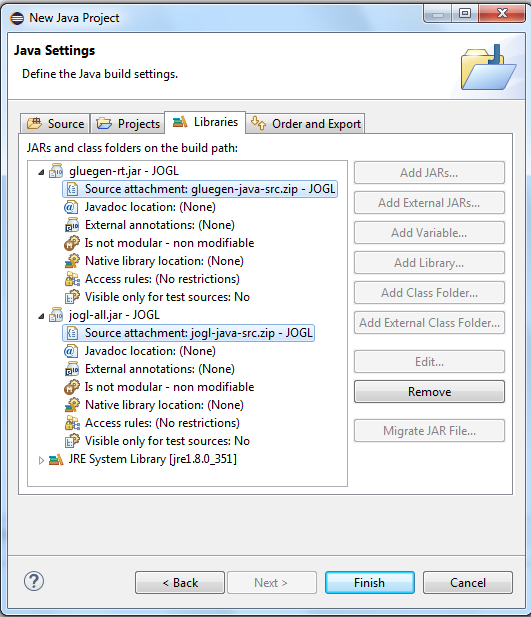


Рисунок 6. Прикрепление файлов с исходным кодом

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

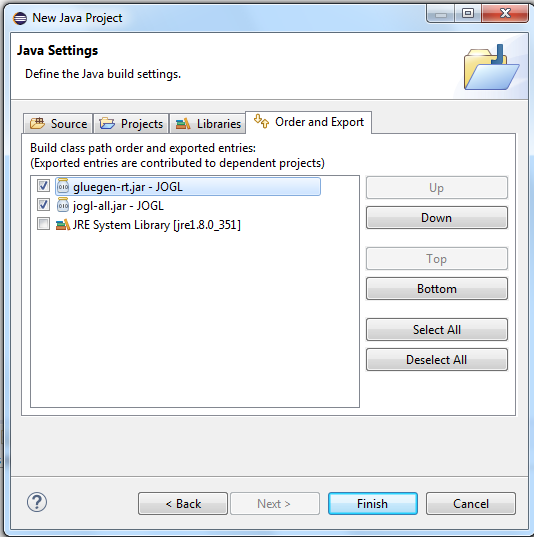


Рисунок 7. Экспорт JAR - файлов JOGL

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

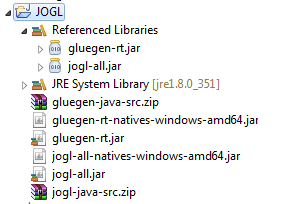


Рисунок 8. Структура проекта с именем «JOGL»

**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).

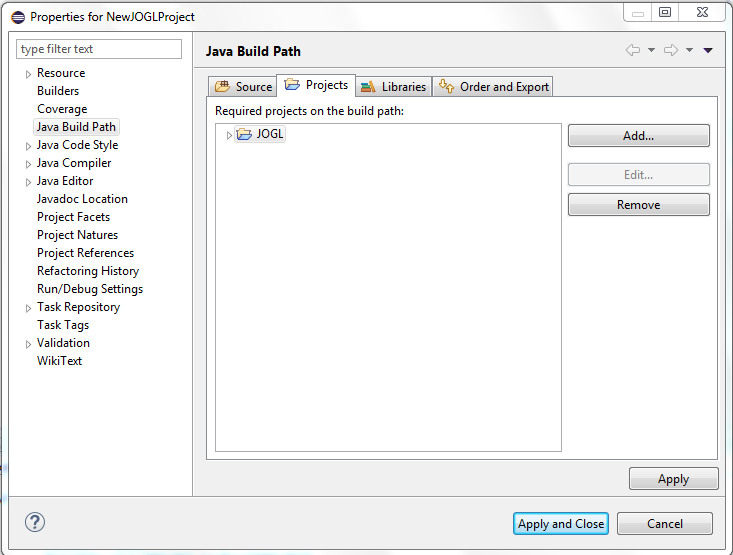


Рисунок 9. Добавление зависимости проекта NewJOGLProject от проекта с именем «JOGL»

Теперь в проекте NewJOGLProject можно использовать библиотеку JOGL. Если это необходимо, то можно создать и другие проекты, зависимые 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>

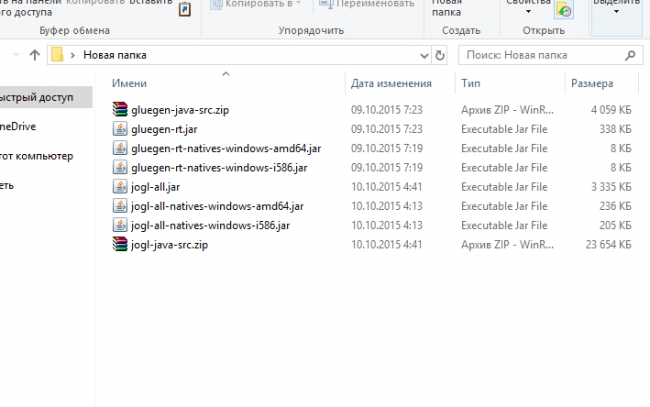
После того как архив получен с сайта, извлекаем в новую папку следующие файлы:

Файлы zip:

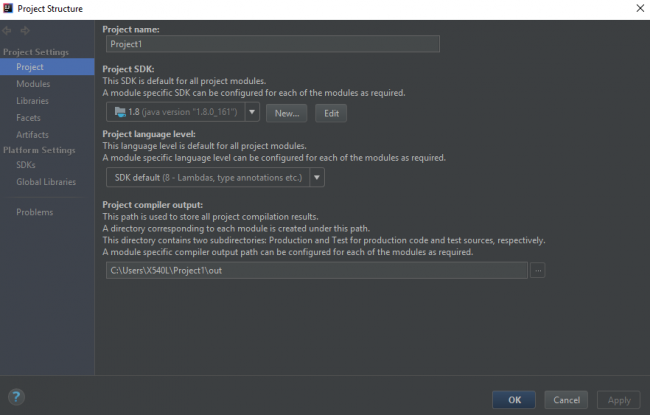
* jogl-java-src.zip
* gluegen-java-src.zip

Из папки jar:

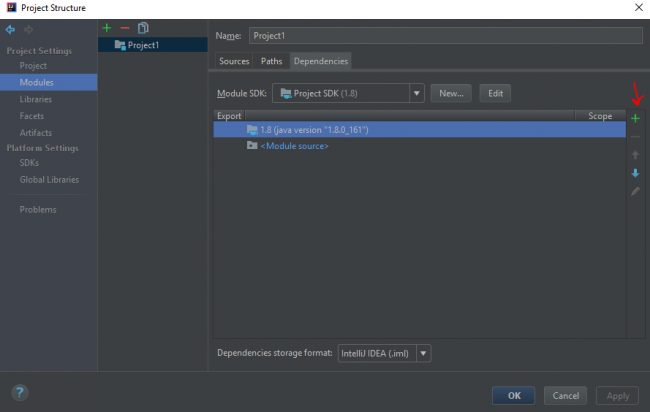
* jogl-all.jar
* jogl-all-natives-windows-amd64.jar
* gluegen-rt.jar
* gluegen-rt-natives-windows-amd64.jar



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

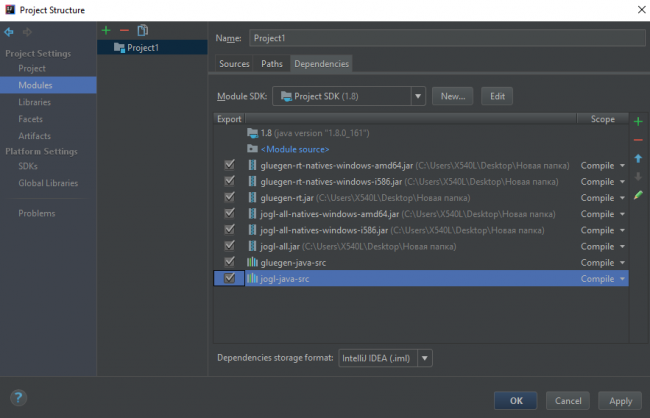


Переходим в **Modules** и нажимаем на зеленый плюс (может быть и не зеленый и находиться в другом месте, это зависит от версии IntelliJ IDEA).



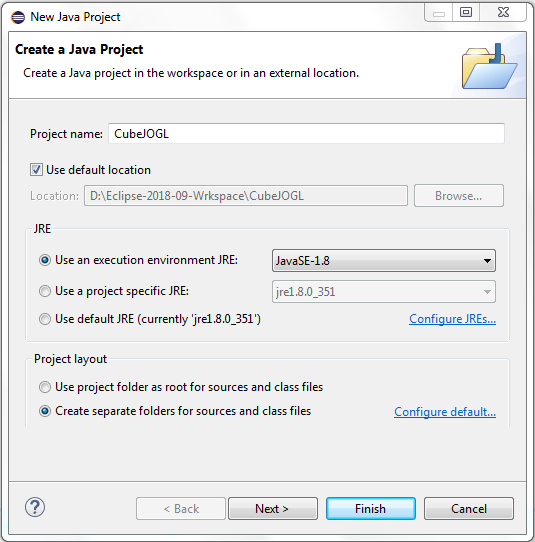
Далее выбираем JAR из папки, в которую извлекли файлы, добавляем файлы с разрешением jar. Затем выбираем Library, а потом Java и добавляем файлы с разрешением zip. Отмечаем все что добавили галочками и нажимаем Apply и затем OK.

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

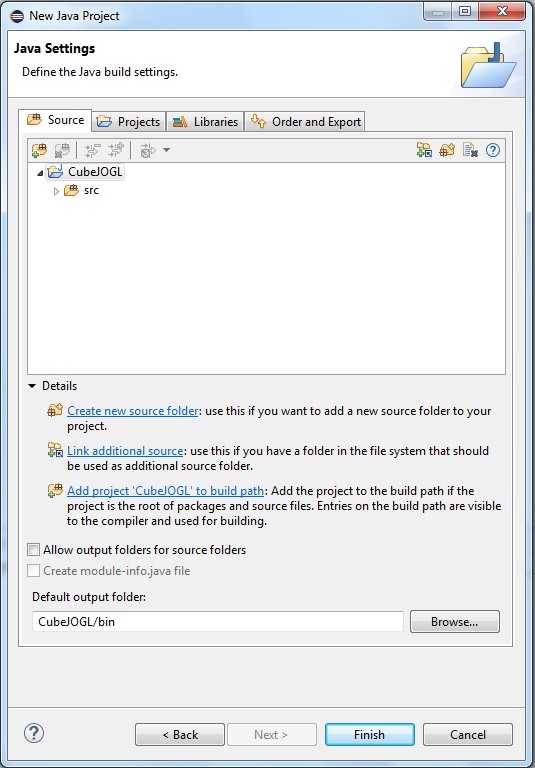


**Пробный проект JOGL в Eclipse**

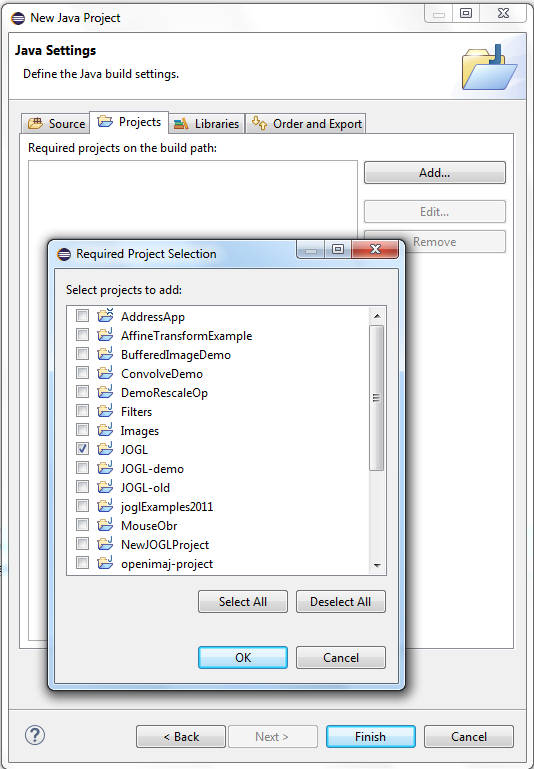
1. Создайте в Eclipse Java – проект CubeJOGL.



1. 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:



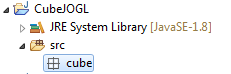
1. Выберите закладку Projects и нажмите кнопку Add (расположена справа в закладке Projects). Появится окно выбора проектов. Выберите проект JOGL.



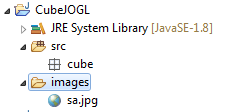
1. Нажмите кнопку Ok. Затем нажмите кнопку Finish в окне создания проекта.
2. Создастся проект со следующей структурой:



1. В проекте CubeJOGL создайте пакет cube.



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



1. 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**);

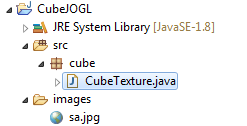
**final** FPSAnimator animator = **new** FPSAnimator(glcanvas, 300, **true**);

animator.start();

}

}

1. Теперь проект CubeJOGL имеет следующую структуру:



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