CMT107 : Visual Computing - Lab Sheet 1

Introduction to OpenGL programming

1. Installation of Jogl on Intellij

a) Download a short version of Jogl (Jogl3S.7z) from the learningcentral, or the newest full version of jogamp (jogamp-all-platforms.7z; plus jogl-javadoc.7z and gluegen-javadoc.7z if you need javadoc) from http://jogamp.org/deployment/jogamp-current/archive/ , unzip it to your home file folder. We assume you have extracted it into the folder home\Jogl3S).

b) Start Intellij. Open or import a project, or create a new project if no Intellij projects exist in your computer.

c) Define a global library as follows:

- Go to menu: File->Project Structure
- Under the Platform Settings, select Global Libraries
- Click the '+' button in the ***middle*** window, and select Java in the New Global Library, then navigate to the unzipped Jogl3S folder, find the "jar" subdirectory, and add both gluegen-rt.jar and jogl-all.jar
- If you want to add javadoc (for use in context-sensitive help), click the '+' button in the ***right*** window, navigate to the "doc" subdirectory in Jogl3S folder, and add the gluegen/javadoc and jogl/javadoc subdirectory
- Optionally, you can change the name in the Name field as Jogl3S (or any name you like)

2. Use Jogl on Intellij

a) Create a new project on the Intellij IDEA. Name it as Ex01 (or any name you like);

b) Download the file Simple.java from the learningcentral, and copy it to the \src folder of the project Ex01;

c) Add Jogl library into the project as follows:

- Go to menu: File->Project Structure
- Under the Project Settings, select Modules, then click the Dependencies tab in the right window
- Click the '+' button on the right side, and select '2 Library…', then in the 'Choose Libraries' window, select Jogl3S, and click 'Add Selected', and then click OK.

d) Now you can run the project Ex01, and you'll get a red point at the centre of the window[1,2].

e) Add more points in the vertex array to draw more points, lines, or triangles by replacing the following lines:

```
private final int numVertices = 1;
float[] vertexArray = { 0.0f, 0.0f };
```

[1] If you are using Java 9 or later, the red point may be not at the centre of the whole window, but in the bottom left quadrant.
[2] Refer to the notes on the last page of this lab sheet, if version errors occur.

with different `numVertices` and `vertexArray`.

- If you want draw 3D points, add the third coordinates in `vertexArray`, and replace the number 2 by 3 in the following line:
  ```
  gl.glVertexAttribPointer(vPosition, 2, GL_FLOAT, false, 0, 0L);
  ```

  For example, you have defined vertex array as:
  ```
  float[] vertexArray = { 0.0f, 0.0f, 0.0f, 0.5f, 0.3f, -0.7f };
  ```

  If they are 2D points, it represents 3 points with coordinates
  `[0.0, 0.0], [0.0, 0.5], and [0.3, -0.7].`

  If they are 2D points, it represents 2 points with coordinates
  `[0.0, 0.0, 0.0] and [0.5, 0.3, -0.7].`

  Please try to draw the above 2D or 3D points.

- You can draw other different primitives by replacing the first parameter in the following line
  `gl.glDrawArrays(GL_POINTS, 0, numVertices);`
  with `GL_LINES, GL_LINE_STRIP, GL_LINE_LOOP, GL_TRIANGLES, GL_TRIANGLE_STRIP,` and `GL_TRIANGLE_FAN.`

- In drawing triangle type of primitives, you can select a polygon rasterization mode using `void gl.glPolygonMode(int face,int mode);` where face must be GL_FRONT_AND_BACK for front- and back-facing polygons (GL_FRONT and GL_BACK means only rendering front or back face). Mode can be GL_POINT, GL_LINE, and GL_FILL. The initial value is GL_FILL for both front- and back-facing polygons.

  Uncomment the line:
  gl.glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
  Use different `face` and `mode` to see what happens.

- Finally, you should know the obvious meanings of
  `gl.glPointSize(5);` and
  `gl.glLineWidth(5);`

  Change the parameter values to see the effects.

3. Build your own Jogl program on Eclipse

a) Create a new java project on the Intellij IDEA. Name it as Ex02;

b) Add Jogl3S into this project as described above in Section 2.c);

c) Click on Ex02 in the project window; right click the \src folder, then click New-> Java Class to create a new java source file. Input a file name (e.g. CG01) inside the name box, you will get a class file CG01.java, with the following code.

```
public class CG01{
}
```

d) On the first line, add “extends JFrame implements GLEventListener” after “CG01” and before “{”. The Intellij IDEA should have automatically added two import lines:

```
import com.jogamp.opengl.GLEventListener;
import javax.swing.*;
```

Otherwise, it has two errors. Click on JFrame, then press 'Alt + Enter'. The line of "import javax.swing.*;" will be automatically added. Similarly, click on GLEventListener, and then press 'Alt + Enter', the other import line will be added. Further click the line of "public class CG01 extends JFrame implements GLEventListener", press 'Alt + Enter', and then click implement methods. Four methods will be added:

```
public void display(GLAutoDrawable drawable)
public void dispose(GLAutoDrawable drawable)
public void init(GLAutoDrawable drawable)
public void reshape(GLAutoDrawable drawable, int x, int y, int width,
                    int height)
```

The following line will also be automatically added in the front of the file:

```
import com.jogamp.opengl.GLAutoDrawable;
```

e) Add '**import static** com.jogamp.opengl.GL3.*;' at the beginning of the code, so that you can use OpenGL constants as in C language, i.e., you can omit "GL3." prefix when you use any OpenGL constants.

f) You can fill codes inside the display(), init(), and reshape() functions. As an example you can copy some code directly from Simple.java to see whether you can successfully draw a point on the screen. Then draw more different types of primitives.

4. Modularised Jogl program

SimpleM.java is a modularised version of Simple.java, which performs the same function as Simple, but save the shader sources in independent files (Pass.vert and Pass.frag for vertex and fragement shader, separately). Also, the shader program compiling and linking code is separated from the main file, and stored in the file ShaderProc.java

a) You can reuse the Ex01 or Ex02 project, or if you like, create your Ex03 project
b) First, copy the files SimpleM.java and ShaderProc.java into the project .\src folder, and copy files Pass.vert and Pass.frag into the project root folder, i.e., the upper level of .\src folder
c) Run SimpleM.java, you will see a red point at the centre of the window
d) Make changes in the similar way as described above in Section 2 and 3
e) You can also edit the colour in Pass.frag to draw different colours of points, lines, or triangles.

Notes about version errors:

If you see “Error: java: invalid source release: 12”, this may be because your project language level is different from the Project SDK version or the Module SDK version. Go to File -> Settings -> Building, Execution, Deployment ->Compiler ->Java Compiler. Delete the version value in the Project bytecode version field on the right window, and it will show ‘the same as language version’. This can make the versions consistent. You need to rebuild project before the change is in effect.

To set the project Java version, go to Run -> Edit configuration, under the application will be the java file name, click it, and on the right window, select the required JRE version.

You can also check some version information in the following way:
Go to File -> Project Structure. Under the Project Settings, select Project, change the Project language level to the same level as the Project SDK version. Also, select Modules, change the Module SDK version to the same version as the Project language level.