**Master 2 3D Printing project**

The project is individual. It deals with slicing and toolpath generation in the context of 3D printing. The whole project is on **20 points** and has two parts.

It needs to be developed in Java using the Eclipse (Neon or later version) JDT IDE. You are given:

* startup code to start developing with. This startup code contains:
  + a library to read 3D model files in OBJ file format (project called ‘com.owens.oobjloader.builder’)
  + an example 3D model, ‘CuteOcto.obj’
  + an application project entitled ‘fc.PrintingApplication.Students’
  + a library to perform vector offsetting/morphology called Clipper (project named ‘de.lighti.clipper’)

To get you started with Eclipse development, please refer to the ‘GettingStarted’ PDF document, which is also provided.

**The project**

The following things need to be implemented:

1. Vector slicing of a 3D model along the Z axis using a step of 0.2 millimeter. Slicing takes place Z-wise along the bounding box of the model. The output of the slicing process should be a set of raw vector contours for each slice. A filled raster bitmap should be created for each slice. This part is on **11 points**.
2. The generation of circumscribed vector toolpaths to completely fill each slice. We consider the nozzle diameter is 0.4 millimeter. The Clipper library should be used to generate circumscribed contours, taking as input the raw vector contours, until these contours completely fill the slice. This part is on **9 points**.

To generate the slices, you will need to first compute the bounding box of the input model and create slices within that box, starting from the lowest Z plane, upwards (Z has a top-down orientation). You will expand the box sideways by 10% along the X and Y axis to create a margin around that box, making the creation of bitmaps clearer as otherwise drawn contours could be too close to the edge of the bitmaps. Please refer to the provided course’s PDF (near the end) for details.

**Slice bitmaps you need to generate**

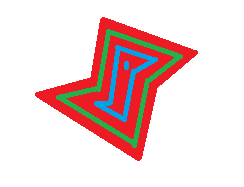
Your application must generate one PNG image file per slice. Files are to be created in the local application directory (do not use absolute paths) and should be named and numbered with respect to the slice number (ex : slice001.png, slice002.png and so on.). The size of one image pixel should be 0.05 mm. Toolpaths will be drawn using the standard Java AWT (BufferedImage class, drawLine etc.).

**What slice bitmaps should look like**

Your bitmaps should have a black background. You will draw the filled slice in red using the raw, outer slice contours (eg. the raw contours obtained when intersecting slice planes and the 3D model).

The perimeter and all other circumscribed contours that will be created using the Clipper library should be drawn in green and blue. Remember the perimeter is not the outer contour.

The following figure shows the filled slice in red (the outer edges match the raw contour of the slice). The first circumscribed contour is the perimeter and is offset inwards from the raw contour using half the nozzle diameter, that is, 0.4 / 2 = 0.2 mm (as discussed during the course) and is draw in green. All subsequent contours (called ‘shells’), still going inwards, each move more and more away from the perimeter using a step of 0.4 mm at every iteration. They show up in blue and we have two shells here, one of which is tiny.



**Using Clipper to generate offset contours**

Remember that Clipper demands vector contours that are:

* Watertight. ‘CuteOcto’ is a watertight 3D model, eg. there is no ‘hole’, no degeneracy, no gap between the triangles. The model is well formed. All slices are guaranteed to also be well formed and watertight.
* Correctly oriented. Please refer to the course PDF for details. All outer slice contours should have a clockwise orientation. All holes should have a counterclockwise orientation. Islands in holes are clockwise, too. So you need:
  + To figure out if a contour represents an outer border, an island, or a hole (see course PDF)
  + Once you know that, change contour orientations accordingly. This is also explained in the PDF.

Offsetting can be achieved by:

* Creating a ClipperOffset object
* For each contour of a slice, create a Clipper ‘Path’.
* Add LongPoint points to the Path for each point of the contour. Note that LongPoint uses integer coordinates, so you have to convert your 3D, float-point coordinates to 2D integer coordinates while working with Clipper (Clipper needs to work internally with integer coordinates and hence provides an integer coordinate API).
* Add the Path to the ClipperOffset object.

Example pseudo code:

//

// See

// http://www.angusj.com/delphi/clipper.php

// for an example in C#

//

ClipperOffset offset = **new** ClipperOffset();

**for** (Contour contour : myContours) // a slice may have multiple contours

{

// Warning: you might have the first and last coord of ‘contour' be the same

Coord2D[] positions = contour.getPositions();

Path path = **new** Path();

**for** (**int** i=0; i < (positions.length - 1); i++)

{

path.add(**new** LongPoint(positions[i].x, positions[i].y));

}

offset.addPath(path, JoinType.***MITER***, EndType.***CLOSED\_POLYGON***);

}

**int** n = 1;

**while** (**true**)

{

Paths solution = **new** Paths();

offset.execute(solution, <amount>); // use amount matching your desired offset

**if** (solution.size() == 0) // Nothing generated? This is your exit criterion.

**break**;

**for** (Path path : solution)

{

LongPoint p = path.get(i % path.size());

[...]

**Sending your project**

Please send your project by email to [frederic.claux@unilim.fr](mailto:frederic.claux@unilim.fr) in a ZIP archive. Do not use any other archive format (RAR etc.). The whole project folder (below the Workspace folder) should be zipped and sent (sending the ‘src’ folder is not sufficient, the whole project needs to be sent). It should **not** contain PNG files. Your program will be run and the generated PNG files examined with an image viewer. It is important your code compiles and run successfully.

Remember that the ‘fc.PrintingApplication.Students’ project should be renamed, with your name showing up in place of ‘Students’, as described in the ‘GettingStarted’ document.

It is recommended to use Windows for development, as your project will be run and tested under Windows, but not mandatory.

The project deadline is **July 31**. Feel free to ask any project-related question to [frederic.claux@unilim.fr](mailto:frederic.claux@unilim.fr).