CSE 590 PROJECT 1 CP2 REPORT

IMPLEMENTATION OF COLLISION DETECTION FOR 3D PRINTERS

	Name	UB Person #
1	Libing Wu	50042948
2	Amaan Akhtarali Modak	50206525
3	Shashank Suresh	50208025

Submitted On: 04/24/2017

I. ABSTRACT

LINE INTERSECTION:

We need to design a module that takes in a line segment coordinate pairs (Unsigned 8 bits, [x1,y1,z1,x2,y2,z2]) and whether they intersect (Boolean 1bit). As of this checkpoint, we have implemented the project aimed at collision detection for line segments in 2 Dimensional space. We will try to implement 3D printing collision detection in the next checkpoint.

Preconditions:

- All the coordinates are positive integers
- All the line segments are partitioned by layers
- All the line segments are in ascending order

To perform the implementation for the 2D printing collision detection we needed to complete the following functions, in order to successfully complete the process:

A. orientation Function

In this function, we determine the orientation of the input line segment, using the coordinate pairs. The outputs of this function will be used to implement the last function, which is the doIntersect function. The output of the following function will be given as:

Case 1: Collinear – *Orientation = 0*

Case 2: Clockwise – *Orientation = 1*

Case 3: Counter-clockwise – *Orientation = 2*

B. onSegment Function

In this function, we determine whether a given point lies on a line segment. This function will output 0 if the point does not lie on the line segment (not collinear), and will output 1 if the point does lie on the line segment (collinear).

C. doIntersect Function

In this function, we determine whether there is a collision occurring between the two given line segments. The following function consists of three steps as discussed below:

Step 1: First, we consider the output of the orientation function. If the orientations (o1 & o2, o3 & o4) are not equal, that is if the orientation of the two line segments are different, then we determine that the line segments do collide. If this is the case, we will move on to the next coordinate pairs.

Step 2: From the first step, we have safely determined that the orientations are the same. Now, we check whether the points are collinear, that is we make use of the OnSegment function. If it is found that the point is collinear, which means that if the point lies on the given line segment, then we determine that there is a collision occurring as the lines intersect.

Step 3: Based on the outputs of both the above steps, if we are getting the output as 0, which means that if we determine that the orientations of the given line segments are the same, and the points are not collinear, then we can safely deduce that there is no collision occurring as there is no line intersection.

We also needed to handle the boundary cases, where the lines were duplicated or when the intersection occurred at one of the end-points of the lines. This was done by comparing the coordinates of the two lines in question, and checking whether they are the same or different points.

Also, since we are currently operating in 2D space, the z-coordinates are not considered for calculation, and we only compare the lines with the same z-coordinate. If the z-coordinate changes, then we skip the previous lines with the old z-coordinate and only compare the lines with the new z-coordinate.

II. DESIGN FLOW CHART

The example flow chart consists of sub-modules is shown in Fig. 1.

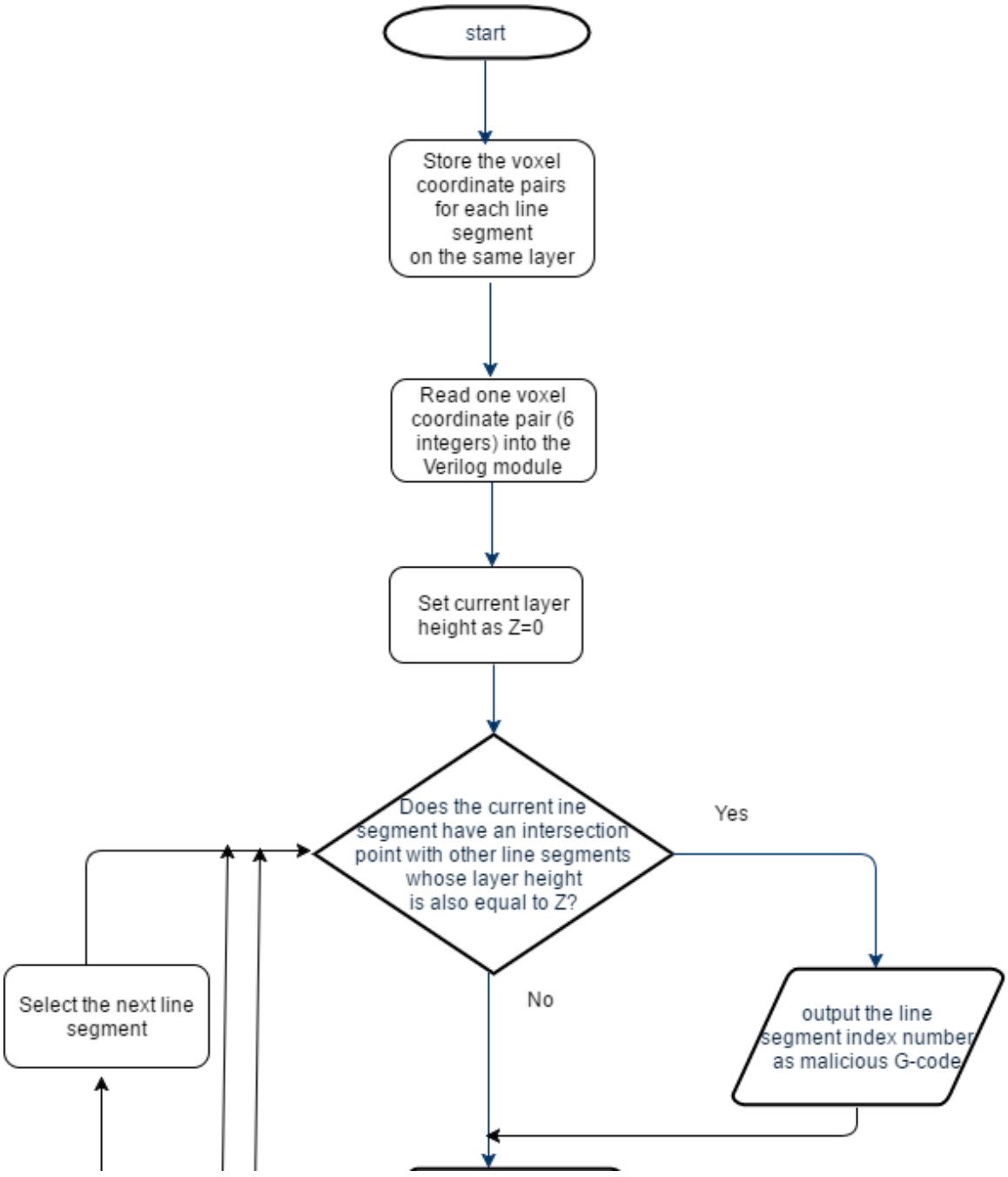
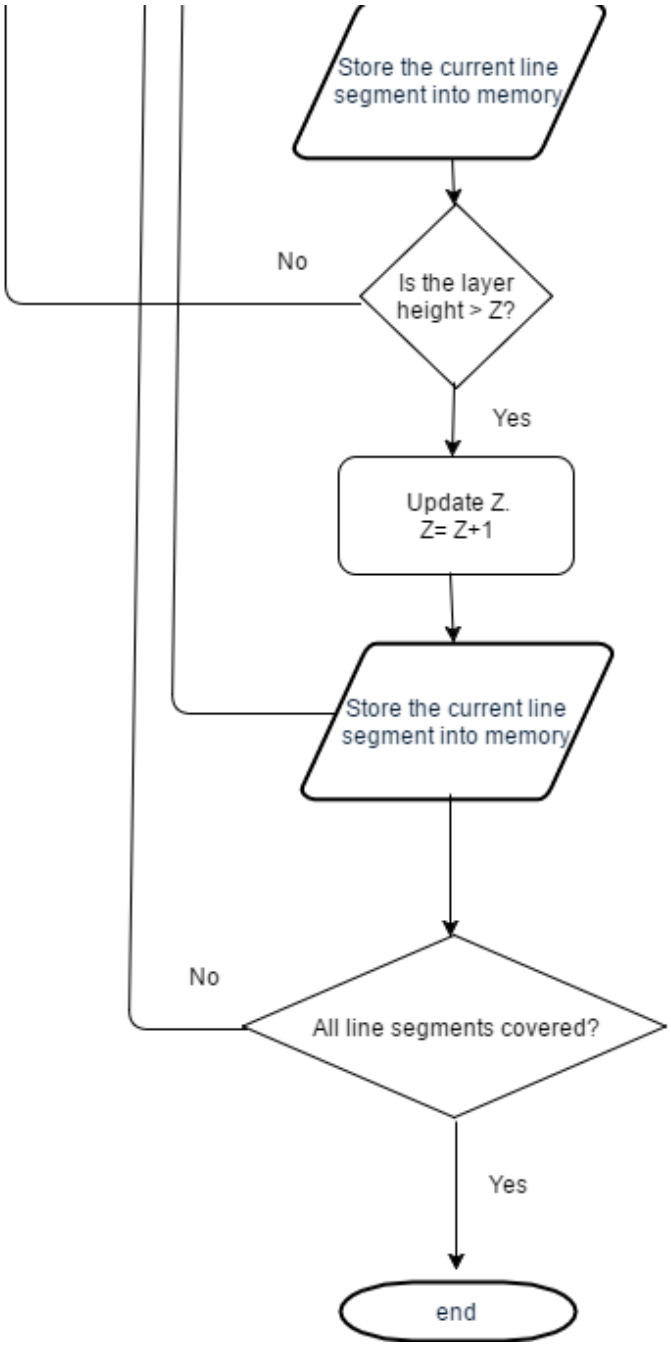


Fig.1a. Flow Chart


Store the current line segment into memory
No
Is the layer height > Z?
Yes
Update Z.
Z= Z+1
Store the current line segment into memory
No
All line segments covered?
Yes
end

III. SIMULATED RESULTS

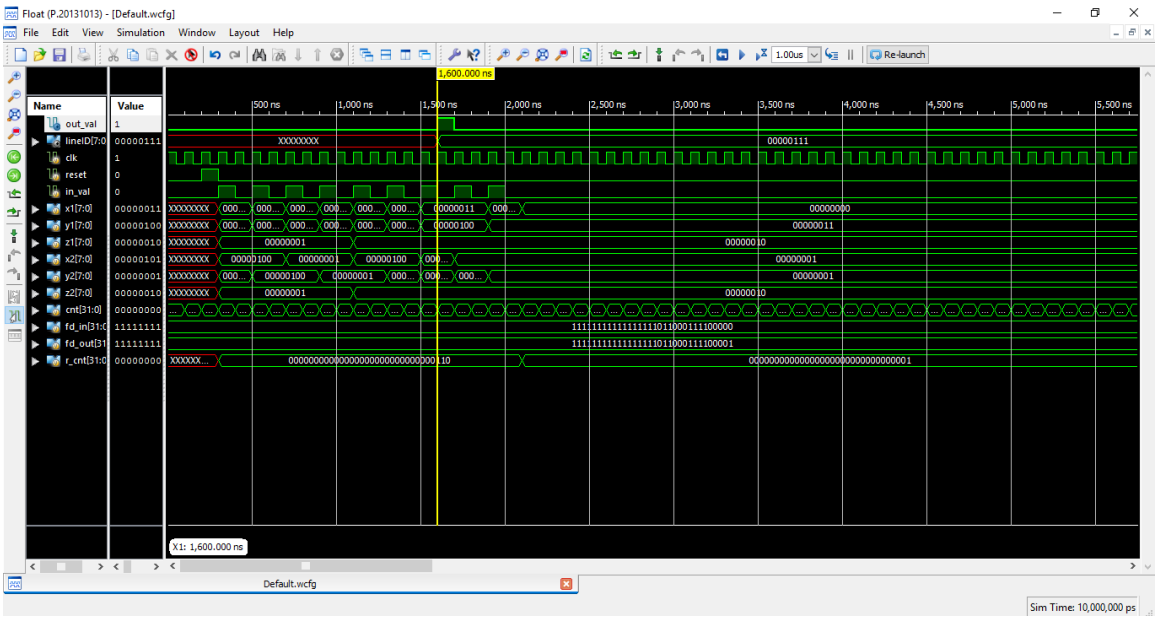
A. Output 1

The below simulation waveform shows the output of our program for the original gcode.txt

In this, we see that whenever there is a collision occurring, there is a spike in the waveform of *out_val,* that is the value of the output valid variable. For instance, according to the given results, we determine that there is a collision occurring at line 7 since the *out_val* value is 1.

Based on the below waveform, we can say that there is a collision occurring only on line 7; since that is the line that is being printed as per the *lineID* row in the image.

The *in_val* value denotes the number of lines that are present in our input file. Based on the waveform, we see that there are 9 input lines.



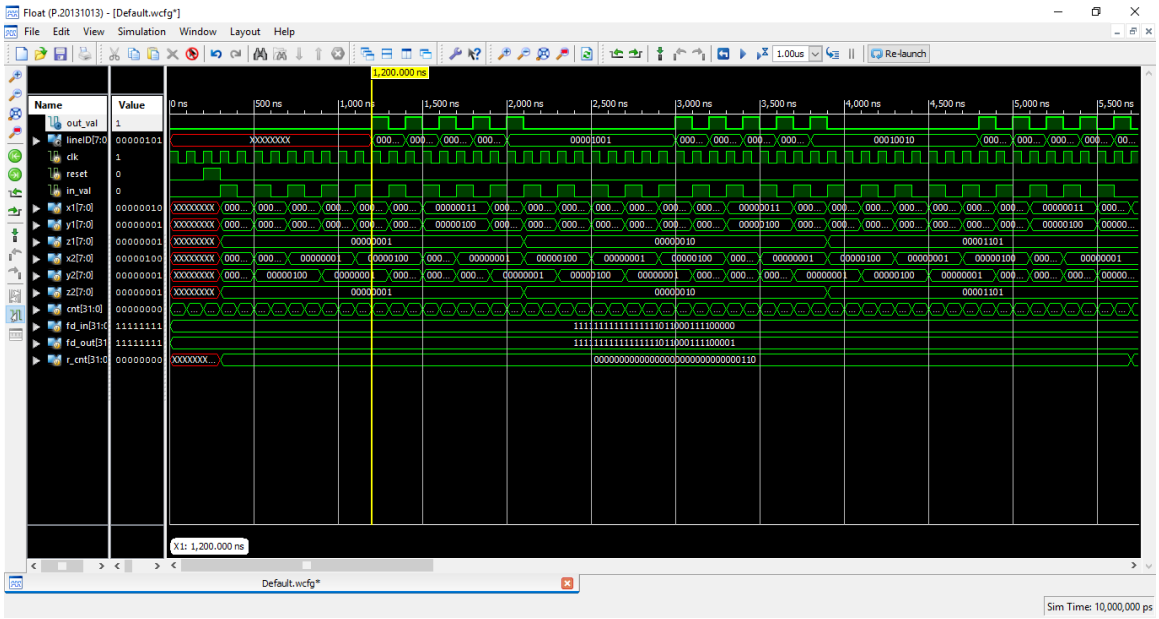
B. Output 2

The below simulation waveform shows the output of our program for the gcode.txt given below.

In this, we see that whenever there is a collision occurring, there is a spike in the waveform of *out_val,* that is the value of the output valid variable. For instance, according to the given results, we determine that there is a collision occurring at line 5 since the *out_val* value is 1.

Based on the below waveform, we can say that there is a collision occurring on lines 5, 6, 7, 8, 9, 14, 15, 16, 17, 18, 23, 24, 25, 26 and 27; since those are the lines that are being printed as per the *lineID* row in the image.

The *in_val* value denotes the number of lines that are present in our input file. Based on the waveform, we see that there are 27 input lines.



Given below is the gcode.txt input which we have tested for our program for output 2:

```
3 3 1 3 0 1
4 2 1 4 4 1
3 4 1 1 4 1
1 3 1 1 1 1
2 1 1 4 1 1
4 2 1 4 4 1
3 4 1 5 1 1
3 4 1 1 4 1
1 3 1 1 1 1
2 1 2 4 1 2
4 2 2 4 4 2
3 4 2 1 4 2
1 3 2 1 1 2
2 1 2 4 1 2
4 2 2 4 4 2
3 4 2 5 1 2
3 4 2 1 4 2
1 3 2 1 1 2
2 1 13 4 1 13
4 2 13 4 4 13
3 4 13 1 4 13
1 3 13 1 1 13
2 1 13 4 1 13
4 2 13 4 4 13
3 4 13 5 1 13
3 4 13 1 4 13
1 3 13 1 1 13
```