Articles on the topic of SLICING

**A Method for Slicing CAD Models in Binary STL Format**

TOBB University of Economics and Technology, Ankara, Turkey

<http://web.firat.edu.tr/iats/cd/subjects/Manufacturing/MTE-31.pdf>

* “This study concentrates on the process planning stage of the [advanced manufacturing] and presents a STL slicing algorithm with the ability of G-Code generation…”(1)

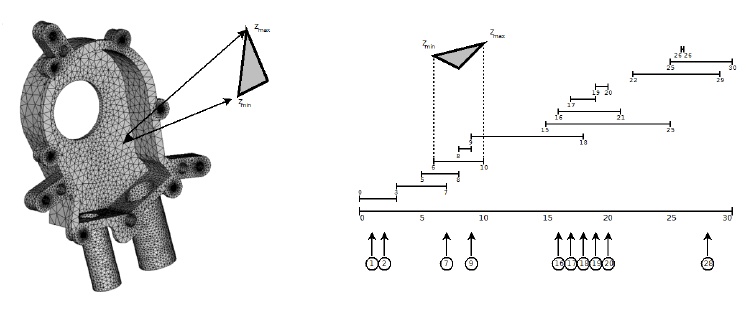
**Slicing Triangle Meshes: An Asymptotically Optimal Algorithm**

Federal University of Technology, Curitiba, Brazil

<http://www.academia.edu/8352847/Slicing_Triangle_Meshes_An_Asymptotically_Optimal_Algorithm>

(also accessible here) <http://www.dainf.ct.utfpr.edu.br/~murilo/public/slicing.pdf>

* “While current algorithms for slicing run on O(n2+k2) or O(n2 + n log nk) for n triangles and k planes, the algorithm proposed in this paper runs on O(nk), where k is the average number of slices cutting each triangle, what is asymptotically the best that can be achieved under certain common assumptions…”



**Marching Cubes: A High Resolution 3D Surface Construction Algorithm**

<http://fab.cba.mit.edu/classes/S62.12/docs/Lorensen_marching_cubes.pdf>

* Not relevant to 3D-printing (is a medical paper), but is relevant to the slicing of 3D surfaces

**Stereolithography (3D Printing) Algorithms and Thoughts**

<https://ravehgonen.wordpress.com/2013/02/19/stereolithography-3d-printing-algorithms-and-thoughts/>

* Aim of blog entry is to briefly consider the process of generating 2D slices of a 3D model
* Presents pseudo code of process
* Shows pictures of code running on the Stanford Bunny model
* Presents some example code that the author created

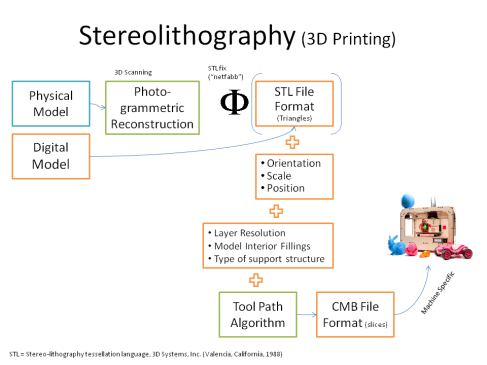
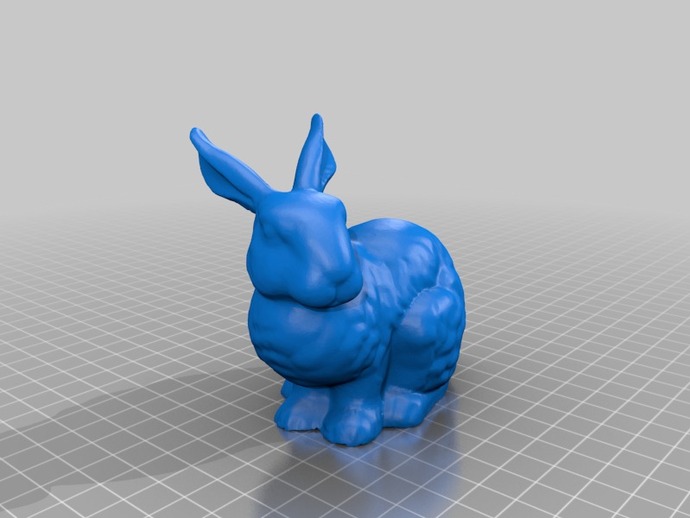
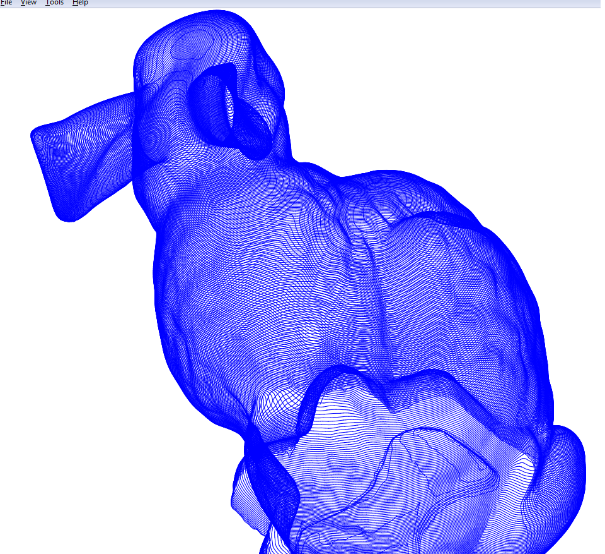
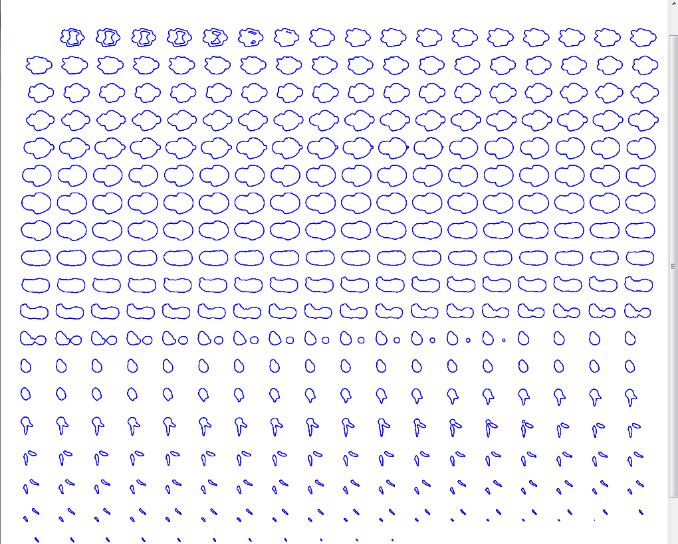


Figure 1: Presents a picture from 3D Systems



* **“A Guide to Creating Good STL Files”**
* <http://www.padtinc.com/blog/the-rp-resource/a-guide-to-creating-good-stl-files>
  + Slicing algorithms only need 3D models made up of polygons
    - STLs are used over NURB (non-uniform rational basis spline)
      * STereoLithography or Standard Tessellation Language
      * Invented by 3D Systems
      * Consists of polygons
        + Defined by outward vector and vertices
        + Supports more than three vertices, but in practice only triangles are used
      * Rules for STL creation
        + No corners can touch an edge
        + Triangles are flat
    - Works by taking the intersection of triangles in 3D and the current “z plane” to create lines that represent the perimeter of a part at a certain height.
  + **“3D Printing: your friend the Slicer”**
  + <https://exploreideasdaily.wordpress.com/2013/04/24/3d-printing-your-friend-the-slicer/>
    - Output of a slicer engine is a g-code file
      * Contains instructions for the printer
    - Examples of slicing engines:
      * Cura
      * Slic3r
      * KISSlicer
    - Slicing is an optimization problem and has multiple solutions
      * Improvements are still needed to deal with:
        + Stringing
        + Seams
    - Cura has a “Joris Mode”
      * Prints single-wall objects in a single path
      * (ex: a cone shell would be printed as a spiral with a decreasing diameter)
    - **How slicing engines work:**
      * Note: Article does the following explanation in 2D to simplify things, but would obviously have to be done in 3D, which requires some additional work
      * For each Z level, engine needs to find some 2D geometry to describe the model at that plane.
        + Not efficient to calculate intersection of *every* triangle. Instead, only the triangles that *do* intersect the plane need to have their geometry referenced.
        1. Identify triangles that need to be referenced, working from the bottom of the model to the top:

Start by creating two lists (since only the minimum and maximum points in a triangle are needed to figure out if the triangle intersects):

Min List: contains the minimum Z-value of the 3 points in a given triangle, alongside the ID for the triangle, in order from lowest Z to highest Z.

Max List: The other lists the maximum Z-value of each triangle’s three points, alongside the ID of the triangle, also in order from lowest to highest.

Create a third “working” list of IDs describing the triangles with minimum Z’s below the Z level being checked

Check the Min List and take the section that ends with the triangle with the highest minimum below the Z level

The triangles should still be in order of how low their minimum Z is

Remove from that set the triangle IDs with maximum Z’s below the Z level being checked

This same triangle that was removed from the working set should be removed from both lists for the next plane since it will be below the Z-plane and is no longer useful

This has the effect of reducing the number of triangles being checked as the z-plane rises, meaning a uniform part will be processed slower at the beginning and faster towards the end.

* + 1. Project the triangles onto the plane
       - Provides some imagery describing ways to project the geometry of a triangle onto the plane
  + Shows the outline resulting from gear composed of triangles
  + Links to example code which the author provides.
  + (Article doesn’t continue).

**HydraRaptor (blog by nophead)**

<http://hydraraptor.blogspot.com/>

Github: <https://github.com/nophead/Mendel90>

Thingiverse: <http://www.thingiverse.com/nophead/designs>

* Extremely comprehensive blog about nophead’s work developing his own 3D-printers and documents his developments in and observations on both hardware and software
* Article about accommodating for hole shrinkage in 3D-printing:
  + <http://hydraraptor.blogspot.com/2011/02/polyholes.html>
* Article about poor dimensioning by slicer engines
  + Explains the math behind determining extrusion geometry and tuning printers further than what most do
  + <http://hydraraptor.blogspot.com/2014/06/why-slicers-get-dimensions-wrong.html>
* Article about embedding nuts inside 3D-printed parts
  + <http://hydraraptor.blogspot.com/2014/03/buried-nuts-and-hanging-holes.html>
* Many others…

SLICING ENGINES

* **Slic3r: G-code generator for 3D printers**
* <http://slic3r.org/blog>
  + Is one of the most popular, free, open source slicing engines available
  + Has many interesting features and supports many printers and usage cases
  + Features:
    - Selective application of settings for local regions of parts
    - Supports multiple extruders
    - Wide compatibility with many printers
    - Print thin layers on the perimeters and thick layers on the interior to save time and increase accuracy
  + Has a blog that explains new features and what’s under development, but doesn’t really focus on the “how” of the engine
* **Skeinforge**
* <http://fabmetheus.crsndoo.com/wiki/index.php/Skeinforge>
  + Very powerful, but can be overwhelming
    - Gives control over dozens of parameters
      * Not just control over tuning and optimizing printing
      * Also provides features like:
        + Dwindle

“Reduces the feed rate and flow rate at the end of the thread, in order to reduce the ooze when traveling.”

* + - * + Fillet

Rounds the corners slightly in a variety of ways in order to reduce corner blobbing and sudden extruder acceleration.

* + - * + Jitter

Change where every layer starts in order to prevent a ridge from being created on the side of the object.