Decorative Boxes – Step4

1. **Overall description**

The goal of this project is to create decorative boxes within the limitations of an 80 Watt laser etching machine.  The limitations of the boxes that are created, is that they must have faces that intersect at 90 degree angles and that the thickness of the plywood or plexiglass they are made of is no greater than 0.25 inch.  As demonstrated in this proposal, even within these limitations a wide range of design complexity can be created.

For this project, the decorative box that will be created will consist of a series of  intersecting rectilinear prisms unioned together at right angles, and extended to fill larger, convex bounding polyhedra.

Design assumptions:

(1). Edges of each prism will be parallel with either the X, Y or Z axis.

(2). There is no coplanarity between the faces of  adjacent prisms.

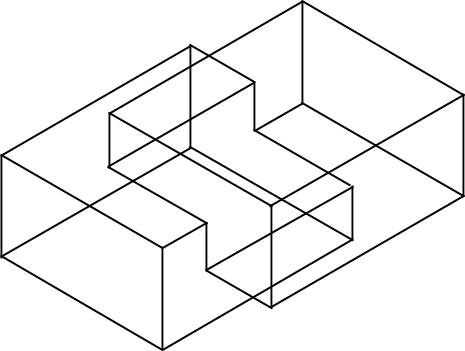
The project is divided into seven steps:

(1). Bounding volume.  (2). Area fill.  (3). Extruding prisms.  (4). Faces of unioned prisms calculated.  (5). Finger joints.  (6). Cut line.  (7). Output device controller.

1. **Step 4 description**

In Steps 1 through 3, the bounding volume is read into the program, a two dimensional pattern of rectangles are read into the program, and this pattern of rectangles is extruded to fill the bounding volume.

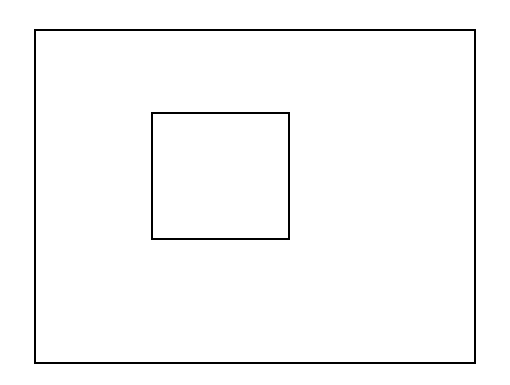
In step 4 the overlapping prisms are unioned together into one contiguous volume.



**Figure 1. Example image of two unioned prisms**

Each Step of this program is interactively completed using AutoCAD computer aided drafting software.  Each of the seven steps of this program communicate with each other using XML.  The output of Step 3 is an XML file that provides ordered sets of line segments completely specifying each of the prisms to be unioned.  To begin Step 4, these sets of line segments are opened and reassembled as prisms in AutoCAD.  In AutoCAD a union operation is performed and exported in Drawing Exchange Format, (DXF).

The output of AutoCAD is a series of unordered lines.  These unordered lines must be sorted into the prisms that they belong to, and then assembled to complete unioned solid.  The specific geometry of the designs may make sorting these elements complex.  One cause of this complexity is the case where one prism entirely pierces another.  This union operation creates the problems of one prism having a hole in its surface, and of another prism becoming divided into two prisms.



**Figure 2. Example of a hole on a surface of a prism**

For Step 4 algorithms were developed to:

1. Efficiently read the unordered lines into prisms.
2. Detect holes on the surfaces of prisms.
3. Read the unordered lines into prisms.

Taking advantage of several facts about the input data simplifies, this process.  One simplifying fact is that every line that defines the edges of each prism are parallel to the X, Y or Z axis.  A second fact is that there exists no coplanarity between faces of adjacent prisms.  Finally, that  each line defines an edge between at most two right angle surfaces.

I classify the surfaces of prisms into three collections: zSurfaces, xSurfaces and ySurfaces.  All zSurfaces are top and bottom faces because they are perpendicular to the vertical to z axis and the value of all points on each zSurface are the same.

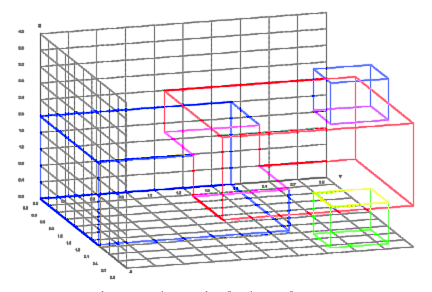
Because each line is the edge of at most two perpendicular faces, each line need only be searched twice in two collections of surfaces.  For example, if a line is parallel to z axis, the algorithm will search it in xSurfaces and ySurfaces to find which two surfaces it belongs to, because it is vertical to zSurfaces so that it cannot exist in zSurfaces.

Using this classification, the Z value of corresponding coordinates can be used to sort only the collection of zSurfaces.  The values of Z coordinate in zSurfaces can then be used as keys to implement binary-search to sort and search zSurfaces.  Because this is a binary search, its complexity will be at most only O(logN).

1. Detect holes on the surfaces of prisms.

Within each surface, the code will first mark connected lines as a group and then check if a group of connected lines are within another group of connected lines.  To achieve this, the Apache commons math library is used.  This library has a function to detect if a point is within the region formed by collected lines in 2D plane.

1. Demonstration of result.



**Figure 3. The result of unit test for Step 4**

The input are the original points of three prisms and all the lines. Each color represents an independent prism. The color of the lines between two prisms is the combination of the colors of the two prisms. Note that in this figure there are four independent prisms, because one input prism is split into two.