# Calculate Frontal Area

paul dot mcintosh at internetscooter dot com

## Overview

This calculates the frontal area of a mesh based on a camera position provided by the user. There is a lot of code that could be improved but for my purposes it does the trick so I will leave as is.

Here's how it works:

1. We load an STL and work out the dimensions of the mesh to place it nicely in the scene
2. We place a parallel projection camera at some offset from the centre of the mesh. The offset number just needs to be big so that you are looking at the parallel projection from a distance and direction you wish to calculate the frontal area.
3. The VTK camera scale is done in real world sizes and is above+below the camera centre point. Setting a size of 1 (metre) makes a window which is 2 (metres) high. Note: Units are "whatever" i.e. could be 1 mm/m/nm
4. We then take a screenshot at some resolution magnification, the bigger the resolution the more accurate, however the slower the calculation (we just count pixels one by one - anything that is black is a hit).
5. Once we have the pixel count we can calculate area using the camera scale and the number of overall

## Usage

e.g. On Windows to process a mesh looking from the X access, with an image magnified 48 times...

*"CalculateFrontalArea.exe D:\scooter\vespalabs\openfoam\motorBike\constant\triSurface\motorBike.stl" 48 1000 0 0"*

Required parameters:

Filename (Full path to stl file e.g. something.stl): **motorBike.stl**  
Magnification (Image resolution size multipler e.g. 2): **48**  
x (x camera offset position from centre): **1000**  
y (y camera offset position from centre): **0**  
z (z camera offset position from centre): **0**

Also it will generate a screen shot used for the calculation

## Validation and Accuracy Considerations

Associated with this code is some sample data. The follow gives some examples of how accurate this approach is. First the good news... and then the so-so news...

### box\_1square.stl

This data is a 1 x 1 x 1 square. *CalculateFrontalArea* consistently calculates the area as 1, from x1-x48 magnification.

### shere\_64thetaRes\_1radius.stl

This data is a more realistic data set of a sphere with a radius of 1. This should give an area result of pi (3.1415926535897932384626433832795) assuming that the sphere, generated by Paraview, is perfect. There will also be some error at lower resolution and in theory we should get more accurate results at higher resolutions.

Below is what I see running resolutions from 1-43. For curved surfaces it appears that a resolution of 12+ is needed to get accurate results. It also appears that the Paraview circle is not perfect.

At resolutions 12+ there is still varying accuracy even at 30+. We must therefore assume that this method has limitations. Accuracy to 4 decimal places can be achieved at res 32, possibly higher accuracy can achieved by averaging readings for resolutions 32-42.