function varargout = PuSL\_Use\_GUI\_Windows(varargin)

% PUSL\_USE\_GUI\_WINDOWS MATLAB code for PuSL\_Use\_GUI\_Windows.fig

%

% This code enables the user to splice any .stl file in a way that

% it can be used in Dr. Cheng Sun’s Projection Stereolithography

% 3D printer. The required .txt file is also generated.

% To use this program on the new CLIP printer will require the following

% modifications

% 1. Change image format from .bmp to .png also the .png must be

% 2 bit.

% 2. You could create a ‘movie’ of images, or simply a ton of

% images, creating thousands of 2 bit images will be easier.

%

% Inputs: 1. New File Name and bitmap pre-fix.

% 2. One .stl file

% 3. File Orientation (Y-up, Z-up or X-up)

% 4. Desired layer thickness (20-50um recommended)

% 5. UV Exposure per layer (12-24s recommended)

% 6. Scale Magnification (0.1-1000%)

% 7. # of Designs to print (1-4)

% 8. Gap between designs (60-100 pixels recommended)

% Optional Inputs:

% 1. 2nd .stl file

% 2. Scale Magnification

% Outputs: 1. A new folder with the title of ‘New File Name (input)

% this folder will be located in MATLABs current working

% directory

% 2. Bitmap images representing ‘slices’ of your input .stl file

% where each slice represents one layer of the file.

% 3. A .txt control file. The printer reads this file as an

% input and it explains to the printer in what order to read

% each file, for how long each file should be displayed, and

% how far the motorized z-stage should move between each

% layer.

%

% Tips:

% Last Modified by Evan Baker v1.0 10-Dec-2015

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @PuSL\_Use\_GUI\_Windows\_OpeningFcn, ...

'gui\_OutputFcn', @PuSL\_Use\_GUI\_Windows\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before PuSL\_Use\_GUI\_Windows is made visible.

function PuSL\_Use\_GUI\_Windows\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to PuSL\_Use\_GUI\_Windows (see VARARGIN)

% Choose default command line output for PuSL\_Use\_GUI\_Windows

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes PuSL\_Use\_GUI\_Windows wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = PuSL\_Use\_GUI\_Windows\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

a=imread('CAD Software ImageZUp.png');

axes(handles.axes6);

imshow(a);

% --- Executes on button press in pushbutton1.

%Display the default image (Z-up) as soon as the program is opened.

%This displays the PUSLSplicingImage1

aa=imread('PuSLSplicingImage1.png');

axes(handles.axes4);

imshow(aa);

hold off

%-----------------------------

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

%Pushbutton1 may be unused.

function edit1\_Callback(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text

% str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.

function edit1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit2\_Callback(hObject, eventdata, handles)

function edit2\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit3\_Callback(hObject, eventdata, handles)

function edit3\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit4\_Callback(hObject, eventdata, handles)

function edit4\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit5\_Callback(hObject, eventdata, handles)

function edit5\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit6\_Callback(hObject, eventdata, handles)

function edit6\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit7\_Callback(hObject, eventdata, handles)

function edit7\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit8\_Callback(hObject, eventdata, handles)

function edit8\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit9\_Callback(hObject, eventdata, handles)

function edit9\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit10\_Callback(hObject, eventdata, handles)

function edit10\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit11\_Callback(hObject, eventdata, handles)

function edit11\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit12\_Callback(hObject, eventdata, handles)

function edit12\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on selection change in popupmenu1.

function popupmenu1\_Callback(hObject, eventdata, handles)

% hObject handle to popupmenu1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell array

% contents{get(hObject,'Value')} returns selected item from popupmenu1

%popupmenu1 contains the code for the orientation selection.

contents = cellstr(get(hObject,'String'));

popupmenu1Value = contents{get(hObject,'Value')};

%Setting the values in popupmenu. Also when a value in popupmenu1 is

%selected, a new image is displayed.

if popupmenu1Value == 'X'

a=imread('CAD Software ImageXUp.png');

axes(handles.axes6);

imshow(a);

hold off;

end

if popupmenu1Value == 'Y'

a=imread('CAD Software ImageYUp.png');

axes(handles.axes6);

imshow(a);

hold off;

end

if popupmenu1Value == 'Z'

a=imread('CAD Software ImageZUp.png');

axes(handles.axes6);

imshow(a);

hold off;

end

% --- Executes during object creation, after setting all properties.

function popupmenu1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to popupmenu1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit13\_Callback(hObject, eventdata, handles)

function edit13\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function popupmenu2\_Callback(hObject, eventdata, handles)

% popupmenu2 contains the code for printing multiple structures

popupmenu3Value = getCurrentPopupString(handles.popupmenu3);

popupmenu2Value = getCurrentPopupString(handles.popupmenu2);

if strcmp(popupmenu3Value,'One') == 1;

if strcmp(popupmenu2Value,'One') == 1;

a=imread('PuSL\_Splicing\_Repeat1.png');

end

if strcmp(popupmenu2Value,'Two') == 1;

a=imread('PuSL\_Splicing\_Repeat2.png');

end

if strcmp(popupmenu2Value,'Three') == 1;

a=imread('PuSL\_Splicing\_Repeat3.png');

end

if strcmp(popupmenu2Value,'Four') == 1;

a=imread('PuSL\_Splicing\_Repeat4.png');

end

end

%if popupmenu2Value == 'Two'

if strcmp(popupmenu3Value,'Two') == 1;

if strcmp(popupmenu2Value,'One') == 1;

a=imread('PuSL\_Splicing\_Repeat2.png');

end

if strcmp(popupmenu2Value,'Two') == 1;

a=imread('PuSL\_Splicing\_Repeat4.png');

end

if strcmp(popupmenu2Value,'Three') == 1;

a=imread('NotEnabled.png');

end

if strcmp(popupmenu2Value,'Four') == 1;

a=imread('NotEnabled.png');

end

end

axes(handles.axes4);

imshow(a);

hold off;

function popupmenu2\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function popupmenu3\_Callback(hObject, eventdata, handles)

popupmenu3Value = getCurrentPopupString(handles.popupmenu3);

% popupmenu3 contains the code for the ‘optional’ multiple .stl designs.

popupmenu2Value = getCurrentPopupString(handles.popupmenu2);

if strcmp(popupmenu3Value,'One') == 1;

if strcmp(popupmenu2Value,'One') == 1;

a=imread('PuSL\_Splicing\_Repeat1.png');

end

if strcmp(popupmenu2Value,'Two') == 1;

a=imread('PuSL\_Splicing\_Repeat2.png');

end

if strcmp(popupmenu2Value,'Three') == 1;

a=imread('PuSL\_Splicing\_Repeat3.png');

end

if strcmp(popupmenu2Value,'Four') == 1;

a=imread('PuSL\_Splicing\_Repeat4.png');

end

end

%if popupmenu2Value == 'Two'

if strcmp(popupmenu3Value,'Two') == 1;

if strcmp(popupmenu2Value,'One') == 1;

a=imread('PuSL\_Splicing\_Repeat2.png');

end

if strcmp(popupmenu2Value,'Two') == 1;

a=imread('PuSL\_Splicing\_Repeat4.png');

end

if strcmp(popupmenu2Value,'Three') == 1;

a=imread('NotEnabled.png');

end

if strcmp(popupmenu2Value,'Four') == 1;

a=imread('NotEnabled.png');

end

end

axes(handles.axes4);

imshow(a);

hold off;

function popupmenu3\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit14\_Callback(hObject, eventdata, handles)

function edit14\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit15\_Callback(hObject, eventdata, handles)

function edit15\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit16\_Callback(hObject, eventdata, handles)

function edit16\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit17\_Callback(hObject, eventdata, handles)

function edit17\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function checkbox1\_Callback(hObject, eventdata, handles)

function checkbox2\_Callback(hObject, eventdata, handles)

function pushbutton2\_Callback(hObject, eventdata, handles)

%pushbutton2 is currently unused. (was deleted from the GUI)

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

function pushbutton3\_Callback(hObject, eventdata, handles)

%pushbutton3 is currently unused. (was deleted from the GUI)

function pushbutton4\_Callback(hObject, eventdata, handles)

%pushbutton4 is currently unused. (was deleted from the GUI)

% --- Executes during object creation, after setting all properties.

function axes4\_CreateFcn(hObject, eventdata, handles)

% hObject handle to axes4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% --- Executes on button press in pushbutton5.

function pushbutton5\_Callback(hObject, eventdata, handles)

% When pushbutton 5 is pressed, the user can pick a .stl file.

[Filename\_stl\_1,PathName\_stl\_1,FilterIndex\_stl\_1] = uigetfile;

set(handles.text76,'string', num2str(Filename\_stl\_1))

% --- Executes on button press in pushbutton6.

function pushbutton6\_Callback(hObject, eventdata, handles)

% on pushbutton6 press, the user can pick a .stl file (optional section)

[Filename\_stl\_2,PathName\_stl\_2,FilterIndex\_stl\_2] = uigetfile;

set(handles.text77,'string', num2str(Filename\_stl\_2))

%Later on in this section, reference the 'PathName' for the file location.

function edit18\_Callback(hObject, eventdata, handles)

function edit18\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function axes6\_CreateFcn(hObject, eventdata, handles)

% No Code.

% these commands link MATLAB to access text boxes in the GUI.

function text76\_CreateFcn(hObject, eventdata, handles)

function text77\_CreateFcn(hObject, eventdata, handles)

function str = getCurrentPopupString(hh)

%# getCurrentPopupString returns the currently selected string in the popupmenu with handle hh

%# could test input here

if ~ishandle(hh) || strcmp(get(hh,'Type'),'popupmenu')

error('getCurrentPopupString needs a handle to a popupmenu as input')

end

%# get the string - do it the readable way

list = get(hh,'String');

val = get(hh,'Value');

if iscell(list)

str = list{val};

else

str = list(val,:);

end

%----------------------------------------------------------------------------

% This is the most critical part of the code.

function pushbutton7\_Callback(hObject, eventdata, handles)

%This is the push button that controls 'RUN'.

% STL\_to\_BMP\_for\_PuSL

% Written by: Evan Baker 6/10/2013

% The Code was originally developed to convert .stl files generated on

% SolidWorks into .bmp files required for PuSL printing in Dr. Cheng Sun's

% lab at Northwestern University. The code also generates the .txt file

% the PuSL printer requires.

%

% Future recommended improvements:

% After generating the 1st bitmap, check the previous bitmap and make sure

% it is not exactly the same. If it is exactly the same, do not generate a

% new bitmap and repeat the 1st one a second time.

%

% TO RUN THIS CODE ON MAC: skip to line 160 and fix.

%

% This STL file used in this example was created on solidworks. Other tools may orient the

% file differently. You may need to modify the inputs to VOXELISE function

% below.

%

% When printing 2 different input files, fname will be on the left

% side of the bitmap, fname2 will be on the right side

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%Inputs:

%fname='sw\_2x2\_160umSpring.STL'; %Filename of STL file

%{

clear all

close all

fclose all;

%}

popupmenu1Value = getCurrentPopupString(handles.popupmenu1);

if strcmp(popupmenu1Value,'X') == 1;

YUp = 2;

end

if strcmp(popupmenu1Value,'Y') == 1;

YUp = 1;

end

if strcmp(popupmenu1Value,'Z') == 1;

YUp = 0;

end

%YUp = 0; %If YUp = 1 Y is the up orientation, if YUp = 0 Z is the up orientation.

%If YUp = 2 x is the up orientation.

%Designs from Thingiverse.com have Z-Up.

%Most designs originally for our PuSL printer have YUp.

popupmenu2Value = getCurrentPopupString(handles.popupmenu2);

if strcmp(popupmenu2Value,'One')==1;

numstruct = 1;

end

if strcmp(popupmenu2Value,'Two')==1;

numstruct = 2;

end

if strcmp(popupmenu2Value,'Three')==1;

numstruct = 3;

end

if strcmp(popupmenu2Value,'Four')==1;

numstruct = 4;

end

%numinputfiles = 1; %This is the number of input files

%How many structures?

popupmenu3Value = getCurrentPopupString(handles.popupmenu3);

if strcmp(popupmenu3Value,'One')==1;

numinputfiles = 1;

end

if strcmp(popupmenu3Value,'Two')==1;

numinputfiles = 2;

end

%NOTE: if you have 2 input files, numstruct=1 will make 1 of each.

fname = get(handles.text76,'String');

%fname='Madeleine.STL'; %Filename of Dec 27 file

if numinputfiles == 2;

fname2=get(handles.text77,'String'); %The second input file name

%NOTE: both structures should have same dimensions

end

%Outputs:

dir\_bmap = get(handles.edit20,'String'); %Output folder for saving bitmap files

mkdir(dir\_bmap);

ofname\_prefix=dir\_bmap; %Prefix of output file for bitmaps MUST be 'string'.

% Programatically finding the bounding box.

% Begin the code for Calculating the 'box' dimensions.

fname1 = get(handles.text76,'String');

fname1Code = strcat({'S1 := plot::SurfaceSTL("'}, fname1, '"):');

y = evalin(symengine,fname1Code);

z = evalin(symengine,'S1::boundingBox()');

%---------------

%This code is required to separate out Box X Dimension

q = z(1);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1X=sprintf('%s',str);

end

if i==2;

Term2X=sprintf('%s',str);

end

if i==3;

Term3X=sprintf('%s',str);

end

if i==4;

Term4X=sprintf('%s',str);

end

end

Num1X = strcat(Term1X, {'.'}, Term2X);

Num2X = strcat(Term3X, {'.'}, Term4X);

Num1TrueX = str2num(char(Num1X));

Num2TrueX = str2num(char(Num2X));

BoxXDimension = Num2TrueX-Num1TrueX;

%End Code for Box X Dimension

%-----

%---

%Start Code for Box Y Dimension

q = z(2);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1Y=sprintf('%s',str);

end

if i==2;

Term2Y=sprintf('%s',str);

end

if i==3;

Term3Y=sprintf('%s',str);

end

if i==4;

Term4Y=sprintf('%s',str);

end

end

Num1Y = strcat(Term1Y, {'.'}, Term2Y);

Num2Y = strcat(Term3Y, {'.'}, Term4Y);

Num1TrueY = str2num(char(Num1Y));

Num2TrueY = str2num(char(Num2Y));

BoxYDimension = Num2TrueY-Num1TrueY;

%End Code for Box Y Dimension

%------

%----

%Start Code for Box Z Dimension

q = z(3);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1Z=sprintf('%s',str);

end

if i==2;

Term2Z=sprintf('%s',str);

end

if i==3;

Term3Z=sprintf('%s',str);

end

if i==4;

Term4Z=sprintf('%s',str);

end

end

Num1Z = strcat(Term1Z, {'.'}, Term2Z);

Num2Z = strcat(Term3Z, {'.'}, Term4Z);

Num1TrueZ = str2num(char(Num1Z));

Num2TrueZ = str2num(char(Num2Z));

BoxZDimension = Num2TrueZ-Num1TrueZ;

%End Code for Box Z Dimension

%----

%Calculating the X, Y and Z values to use in the program.

Magnification = str2double(get(handles.edit18,'String'));

MagBoxXDimension = Magnification/100\*BoxXDimension;

MagBoxYDimension = Magnification/100\*BoxYDimension;

MagBoxZDimension = Magnification/100\*BoxZDimension;

set(handles.text89,'string', num2str(MagBoxXDimension))

set(handles.text90,'string', num2str(MagBoxYDimension))

set(handles.text91,'string', num2str(MagBoxZDimension))

if YUp == 0

nnmm = MagBoxZDimension;

widmm = MagBoxXDimension;

lenmm = MagBoxYDimension;

end

if YUp == 1;

nnmm = MagBoxYDimension;

widmm = MagBoxXDimension;

lenmm = MagBoxZDimension;

end

if YUp == 2;

nnmm = MagBoxXDimension;

widmm = MagBoxYDimension; %Changed from Z to Y This fixed the problem!

lenmm = MagBoxZDimension; %Changed from Y to Z

end

%-----------------------------

%Below, the case begins under the condition two input files exist with two

%different sets of X,Y,Z dimensions.

if numinputfiles == 2;

fname2 = get(handles.text77,'String');

fname2Code = strcat({'S1 := plot::SurfaceSTL("'}, fname2, '"):');

y = evalin(symengine,fname2Code);

z = evalin(symengine,'S1::boundingBox()');

%---------------

%This code is required to separate out Box X Dimension

q = z(1);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1X=sprintf('%s',str);

end

if i==2;

Term2X=sprintf('%s',str);

end

if i==3;

Term3X=sprintf('%s',str);

end

if i==4;

Term4X=sprintf('%s',str);

end

end

Num1X = strcat(Term1X, {'.'}, Term2X);

Num2X = strcat(Term3X, {'.'}, Term4X);

Num1TrueX = str2num(char(Num1X));

Num2TrueX = str2num(char(Num2X));

BoxXDimension = Num2TrueX-Num1TrueX;

%End Code for Box X Dimension

%-----

%---

%Start Code for Box Y Dimension

q = z(2);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1Y=sprintf('%s',str);

end

if i==2;

Term2Y=sprintf('%s',str);

end

if i==3;

Term3Y=sprintf('%s',str);

end

if i==4;

Term4Y=sprintf('%s',str);

end

end

Num1Y = strcat(Term1Y, {'.'}, Term2Y);

Num2Y = strcat(Term3Y, {'.'}, Term4Y);

Num1TrueY = str2num(char(Num1Y));

Num2TrueY = str2num(char(Num2Y));

BoxYDimension = Num2TrueY-Num1TrueY;

%End Code for Box Y Dimension

%------

%----

%Start Code for Box Z Dimension

q = z(3);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1Z=sprintf('%s',str);

end

if i==2;

Term2Z=sprintf('%s',str);

end

if i==3;

Term3Z=sprintf('%s',str);

end

if i==4;

Term4Z=sprintf('%s',str);

end

end

Num1Z = strcat(Term1Z, {'.'}, Term2Z);

Num2Z = strcat(Term3Z, {'.'}, Term4Z);

Num1TrueZ = str2num(char(Num1Z));

Num2TrueZ = str2num(char(Num2Z));

BoxZDimension = Num2TrueZ-Num1TrueZ;

%End Code for Box Z Dimension

%----

%Calculating the X, Y and Z values to use in the program.

Magnification = str2double(get(handles.edit23,'String'));

MagBoxXDimension = Magnification/100\*BoxXDimension;

MagBoxYDimension = Magnification/100\*BoxYDimension;

MagBoxZDimension = Magnification/100\*BoxZDimension;

set(handles.text100,'string', num2str(MagBoxXDimension))

set(handles.text101,'string', num2str(MagBoxYDimension))

set(handles.text102,'string', num2str(MagBoxZDimension))

if YUp == 0

nnmm2 = MagBoxZDimension;

widmm2 = MagBoxXDimension;

lenmm2 = MagBoxYDimension;

end

if YUp == 1;

nnmm2 = MagBoxYDimension;

widmm2 = MagBoxXDimension;

lenmm2 = MagBoxZDimension;

end

if YUp == 2;

nnmm2 = MagBoxXDimension;

widmm2 = MagBoxZDimension;

lenmm2 = MagBoxYDimension;

end

end

%This is the end of the calculation for the dimensions of the 2nd input

%file.

LayerThickness = str2double(get(handles.edit19,'String'));

nn=round(nnmm/(LayerThickness/1000)); %Height of the 1st .STL file.

if numinputfiles ==2;

nn2 = round(nnmm2/(LayerThickness/1000)); %Height of the 2nd .STL file

end

%round(3.4/0.020) = 170 layers

%nn=180; %Height of spring structure/20microns for new file

wid=round(widmm/0.0071); %Width of spring structure/7.1microns (this is X, 1050 pixel limit))

if numinputfiles==2;

wid2=round(widmm2/0.0071);

end

len=round(lenmm/0.0071); %Length of spring structure/7.1microns (this is Y, 1400 pixel limit)

if numinputfiles ==2;

len2=round(lenmm2/0.0071); %Length of the 2nd .STL file

end

% Overall bitmap dimensions - we may want to drop several springs onto this

% bitmap. For now we will drop one spring at about the middle of the

% bitmap.

b\_nn=nn; %The total number of bitmap layers for the first file.

b\_wid=1050; %Vertical monitor Width for bitmap: 7.455mm

b\_len=1400; %Horizontal monitor length for bitmap: 9.940mm

%Offset for placing spring into the final bitmap

%Spacer represents the pixel separation between designs. Previously set to

%40 pixels. Even at 50 pixels the single spring's tops clung together.

%Testing 60 now. (I also shrunk the top)

spacer = str2double(get(handles.edit22,'String'));

Exp = str2double(get(handles.edit21,'String')); %UV Exposure Time 12-30 seconds.

%Definition of o\_wid, o\_len: o\_wid and o\_len refer to the exact point on

%the fabrication plane where the structure should be drawn.

if numstruct==1;

o\_wid=b\_wid/2-wid/2;

o\_len=b\_len/2-len/2;

end

if numstruct==2;

o\_wid1=b\_wid/2-wid/2;

o\_len1=b\_len/2-len-spacer;

o\_wid2=b\_wid/2-wid/2;

o\_len2=b\_len/2+spacer;

end

if numstruct==4;

o\_wid1=b\_wid/2-wid-spacer;

o\_len1=b\_len/2-len-spacer;

o\_wid2=b\_wid/2+spacer;

o\_len2=b\_len/2-len-spacer;

o\_wid3=b\_wid/2-wid-spacer;

o\_len3=b\_len/2+spacer;

o\_wid4=b\_wid/2+spacer;

o\_len4=b\_len/2+spacer;

end

if numinputfiles == 2;

o\_wid1=b\_wid/2-wid/2;

o\_len1=b\_len/2-len-spacer;

o\_wid2=b\_wid/2-wid/2;

o\_len2=b\_len/2+spacer;

end

if numinputfiles == 2 && numstruct==2;

o\_wid1=b\_wid/2-wid-spacer;

o\_len1=b\_len/2-len-spacer;

o\_wid2=b\_wid/2+40;

o\_len2=b\_len/2-len-spacer;

o\_wid3=b\_wid/2-wid-spacer;

o\_len3=b\_len/2+spacer;

o\_wid4=b\_wid/2+spacer;

o\_len4=b\_len/2+spacer;

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

%oset=[o\_wid,o\_len];

figure

[stlcoords] = READ\_stl(fname);

xco = squeeze( stlcoords(:,1,:) )';

yco = squeeze( stlcoords(:,2,:) )';

zco = squeeze( stlcoords(:,3,:) )';

[hpat] = patch(xco,yco,zco,'b');

axis equal

%Voxelise the STL:

if YUp ==1;

[OUTPUTgrid] = VOXELISE(wid,nn,len,fname,'xyz');

end

if YUp ==0;

[OUTPUTgrid] = VOXELISE(wid,len,nn,fname,'xyz');

end

if YUp ==2;

[OUTPUTgrid] = VOXELISE(nn,wid,len,fname,'xyz'); %Try swapping this to nn,wid,len from nn, len,wid

end

%The final structure will drop these springs onto a volume of dimension:

% x:1400, y:1050, z=height of structure/20microns = 3.4mm/20microns=170

disp('finished voxelise');

%Initialize the final bitmap array to all zeros (or all ones)

bz=zeros(b\_wid,b\_len);

if numinputfiles==2;

figure

[stlcoords] = READ\_stl(fname2);

xco = squeeze( stlcoords(:,1,:) )';

yco = squeeze( stlcoords(:,2,:) )';

zco = squeeze( stlcoords(:,3,:) )';

[hpat] = patch(xco,yco,zco,'b');

axis equal

%Voxelise the STL:

if YUp ==1;

[OUTPUTgrid2] = VOXELISE(wid2,nn2,len2,fname2,'xyz');

end

if YUp ==0;

[OUTPUTgrid2] = VOXELISE(wid2,len2,nn2,fname2,'xyz');

end

if YUp ==2;

[OUTPUTgrid2] = VOXELISE(nn2,len2,wid2,fname2,'xyz');

end

%The final structure will drop these springs onto a volume of dimension:

% x:1400, y:1050, z=height of structure/20microns = 3.4mm/20microns=170

disp('finished voxelise');

end

% For each of the 20micron slices, we want to create a bitmap file

ofname\_base=[dir\_bmap '\' ofname\_prefix]; %comment this to run on mac

%ofname\_base = ofname\_prefix; % un-comment this to run on mac...

figure

for i=1:nn

%For STL files from Solidworks, the height is imported into matlab

%(using the READ\_stl function as the middle variable. We are assuming

%the width is the first variable and the length is the third variable.

%If we build non-symmetrical shapes in the future, then may need to

%switch first and third variables.

if YUp ==1;

xx=OUTPUTgrid(:,i,:);

end

if YUp ==0;

xx = OUTPUTgrid(:,:,i);

end

if YUp ==2;

xx = OUTPUTgrid(i,:,:);

end

z=squeeze(xx);

if numinputfiles==2;

if YUp==1;

xx2=OUTPUTgrid2(:,i,:);

end

if YUp==0;

xx2=OUTPUTgrid2(:,:,i);

end

if YUp ==2;

xx2 = OUTPUTgrid(i,:,:);

end

z2=squeeze(xx2);

end

if numinputfiles==1;

if numstruct==1;

o\_wid

%o\_wid1

wid

o\_len

%o\_len1

len

o\_len=round(o\_len)

if YUp <2;

bz(o\_wid:o\_wid+wid-1,o\_len:o\_len+len-1)=z;

end

if YUp == 2;

bz(o\_wid:o\_wid+wid-1,o\_len:o\_len+len-1)=z;

end

end

if numstruct==2;

bz(o\_wid1:o\_wid1+wid-1,o\_len1:o\_len1+len-1)=z;

bz(o\_wid2:o\_wid2+wid-1,o\_len2:o\_len2+len-1)=z;

end

if numstruct==4;

bz(o\_wid1:o\_wid1+wid-1,o\_len1:o\_len1+len-1)=z;

bz(o\_wid2:o\_wid2+wid-1,o\_len2:o\_len2+len-1)=z;

bz(o\_wid3:o\_wid3+wid-1,o\_len3:o\_len3+len-1)=z;

bz(o\_wid4:o\_wid4+wid-1,o\_len4:o\_len4+len-1)=z;

end

end

if numinputfiles==2;

if numstruct==1;

bz(o\_wid1:o\_wid1+wid-1,o\_len1:o\_len1+len-1)=z;

bz(o\_wid2:o\_wid2+wid2-1,o\_len2:o\_len2+len2-1)=z2; %This line was modified, len was changed to len2 and wid changed to wid2

end

if numstruct==2;

bz(o\_wid1:o\_wid1+wid-1,o\_len1:o\_len1+len-1)=z;

bz(o\_wid2:o\_wid2+wid-1,o\_len2:o\_len2+len-1)=z;

bz(o\_wid3:o\_wid3+wid2-1,o\_len3:o\_len3+len2-1)=z2; %this line and the one below were modifeid

bz(o\_wid4:o\_wid4+wid2-1,o\_len4:o\_len4+len2-1)=z2; %wid changed to wid2 and len changed to len2

end

end

imagesc(bz);

if i<1000

zer='';

end

if i<100

zer='0';

end

if i<10

zer='00';

end

ofname=[ofname\_base zer sprintf('%i.bmp',i)];

imwrite(bz,ofname,'bmp');

colormap(gray(256));

xlabel('X-direction');

ylabel('Y-direction');

axis equal tight

%pause(.05); %optional pause for viewing small bitmaps

end

Layer = 1; %The starting layer.

Thick=20; %Thickness 20um per layer.

%Exp=12; %Exposure Time 12 seconds (moved to top)

fileID = fopen([ofname\_base '.txt'],'w');

fprintf(fileID,'%s\t','Layer');

%fprintf(fileID,'%5s\t %4s\t %5s\t %3s\n','Layer','File','Thick','Exp');

fprintf(fileID,'%s\t','File');

fprintf(fileID,'%s\t','Thick');

fprintf(fileID,'%s\n','Exp');

for i=1:nn

if i<1000

zer='';

end

if i<100

zer='0';

end

if i<10

zer='00';

end

File=[ofname\_prefix zer sprintf('%i.bmp',i)];

Layer=i;

% fprintf(fileID,'%1.0f\t %-5s\t %2.0f\t %2.0f\n',Layer,File,Thick,Exp);

fprintf(fileID,'%1.0f\t',Layer);

fprintf(fileID,'%s\t',File);

fprintf(fileID,'%2.0f\t',Thick);

fprintf(fileID,'%2.0f\n',Exp);

end

fclose(fileID);

function edit19\_Callback(hObject, eventdata, handles)

function edit19\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function text82\_CreateFcn(hObject, eventdata, handles)

function edit20\_Callback(hObject, eventdata, handles)

function edit20\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit21\_Callback(hObject, eventdata, handles)

function edit21\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function edit22\_Callback(hObject, eventdata, handles)

function edit22\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function pushbutton9\_Callback(hObject, eventdata, handles)

% Programmatically finding the bounding box.

% Begin the code for Calculating the 'box' dimensions.

fname1 = get(handles.text76,'String');

fname1Code = strcat({'S1 := plot::SurfaceSTL("'}, fname1, '"):');

y = evalin(symengine,fname1Code);

%y = evalin(symengine,'S1 := plot::SurfaceSTL("HW350umStentAusteticDesign3.STL"):');

z = evalin(symengine,'S1::boundingBox()');

%---------------

%This code is required to separate out Box X Dimension

q = z(1);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1X=sprintf('%s',str);

end

if i==2;

Term2X=sprintf('%s',str);

end

if i==3;

Term3X=sprintf('%s',str);

end

if i==4;

Term4X=sprintf('%s',str);

end

end

Num1X = strcat(Term1X, {'.'}, Term2X);

Num2X = strcat(Term3X, {'.'}, Term4X);

Num1TrueX = str2num(char(Num1X));

Num2TrueX = str2num(char(Num2X));

BoxXDimension = Num2TrueX-Num1TrueX

%End Code for Box X Dimension

%-----

%---

%Start Code for Box Y Dimension

q = z(2);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1Y=sprintf('%s',str);

end

if i==2;

Term2Y=sprintf('%s',str);

end

if i==3;

Term3Y=sprintf('%s',str);

end

if i==4;

Term4Y=sprintf('%s',str);

end

end

Num1Y = strcat(Term1Y, {'.'}, Term2Y);

Num2Y = strcat(Term3Y, {'.'}, Term4Y);

Num1TrueY = str2num(char(Num1Y));

Num2TrueY = str2num(char(Num2Y));

BoxYDimension = Num2TrueY-Num1TrueY

%End Code for Box Y Dimension

%------

%----

%Start Code for Box Z Dimension

q = z(3);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1Z=sprintf('%s',str);

end

if i==2;

Term2Z=sprintf('%s',str);

end

if i==3;

Term3Z=sprintf('%s',str);

end

if i==4;

Term4Z=sprintf('%s',str);

end

end

Num1Z = strcat(Term1Z, {'.'}, Term2Z);

Num2Z = strcat(Term3Z, {'.'}, Term4Z);

Num1TrueZ = str2num(char(Num1Z));

Num2TrueZ = str2num(char(Num2Z));

BoxZDimension = Num2TrueZ-Num1TrueZ

%End Code for Box Z Dimension

%Displaying all of these values in the GUI

%If under the Z orientation)

Magnification = str2double(get(handles.edit18,'String'));

MagBoxXDimension = Magnification/100\*BoxXDimension;

MagBoxYDimension = Magnification/100\*BoxYDimension;

MagBoxZDimension = Magnification/100\*BoxZDimension;

set(handles.text89,'string', num2str(MagBoxXDimension))

set(handles.text90,'string', num2str(MagBoxYDimension))

set(handles.text91,'string', num2str(MagBoxZDimension))

%----

function text89\_CreateFcn(hObject, eventdata, handles)

function text90\_CreateFcn(hObject, eventdata, handles)

function text91\_CreateFcn(hObject, eventdata, handles)

function edit23\_Callback(hObject, eventdata, handles)

function edit23\_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in pushbutton10.

function pushbutton10\_Callback(hObject, eventdata, handles)

% Begin the code for Calculating the 'box' dimensions.

fname2 = get(handles.text77,'String');

fname2Code = strcat({'S1 := plot::SurfaceSTL("'}, fname2, '"):');

y = evalin(symengine,fname2Code);

z = evalin(symengine,'S1::boundingBox()');

%---------------

%This code is required to separate out Box X Dimension

q = z(1);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1X=sprintf('%s',str);

end

if i==2;

Term2X=sprintf('%s',str);

end

if i==3;

Term3X=sprintf('%s',str);

end

if i==4;

Term4X=sprintf('%s',str);

end

end

Num1X = strcat(Term1X, {'.'}, Term2X);

Num2X = strcat(Term3X, {'.'}, Term4X);

Num1TrueX = str2num(char(Num1X));

Num2TrueX = str2num(char(Num2X));

BoxXDimension = Num2TrueX-Num1TrueX;

%End Code for Box X Dimension

%-----

%---

%Start Code for Box Y Dimension

q = z(2);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1Y=sprintf('%s',str);

end

if i==2;

Term2Y=sprintf('%s',str);

end

if i==3;

Term3Y=sprintf('%s',str);

end

if i==4;

Term4Y=sprintf('%s',str);

end

end

Num1Y = strcat(Term1Y, {'.'}, Term2Y);

Num2Y = strcat(Term3Y, {'.'}, Term4Y);

Num1TrueY = str2num(char(Num1Y));

Num2TrueY = str2num(char(Num2Y));

BoxYDimension = Num2TrueY-Num1TrueY;

%End Code for Box Y Dimension

%------

%----

%Start Code for Box Z Dimension

q = z(3);

remain = char(q);

i=0;

while true

i=i+1;

[str, remain] = strtok(remain, '..');

if isempty(str), break; end

if i==1;

Term1Z=sprintf('%s',str);

end

if i==2;

Term2Z=sprintf('%s',str);

end

if i==3;

Term3Z=sprintf('%s',str);

end

if i==4;

Term4Z=sprintf('%s',str);

end

end

Num1Z = strcat(Term1Z, {'.'}, Term2Z);

Num2Z = strcat(Term3Z, {'.'}, Term4Z);

Num1TrueZ = str2num(char(Num1Z));

Num2TrueZ = str2num(char(Num2Z));

BoxZDimension = Num2TrueZ-Num1TrueZ;

%End Code for Box Z Dimension

%Displaying all of these values in the GUI

%If under the Z orientation)

Magnification = str2double(get(handles.edit23,'String'));

MagBoxXDimension = Magnification/100\*BoxXDimension;

MagBoxYDimension = Magnification/100\*BoxYDimension;

MagBoxZDimension = Magnification/100\*BoxZDimension;

set(handles.text100,'string', num2str(MagBoxXDimension))

set(handles.text101,'string', num2str(MagBoxYDimension))

set(handles.text102,'string', num2str(MagBoxZDimension))

function text100\_CreateFcn(hObject, eventdata, handles)

function text101\_CreateFcn(hObject, eventdata, handles)

function text102\_CreateFcn(hObject, eventdata, handles)