PROGRESS REVIEW 8

Astha Prasad

Team F / ADD_IN

Teammates: Daniel Berman, Nikhil Baheti, Ihsane Debbache

ILR #7

February 11, 2016

Introduction

After developing a basic object oriented framework for the software subsystem, the next step was to write the remaining functions and build a GUI that calls the necessary functions in order to generate G-Code using the user's input. Over the last two weeks, Dan and I have developed a working GUI capable of:

- Selecting STL files & configuration files and generating regular G-Code
- COTS item addition
- Computation of R axis commands
- STL file display with tool path/temperature/speed/rotation maps

GUI

The fundamental structure of the GUI is given by Figure 1.

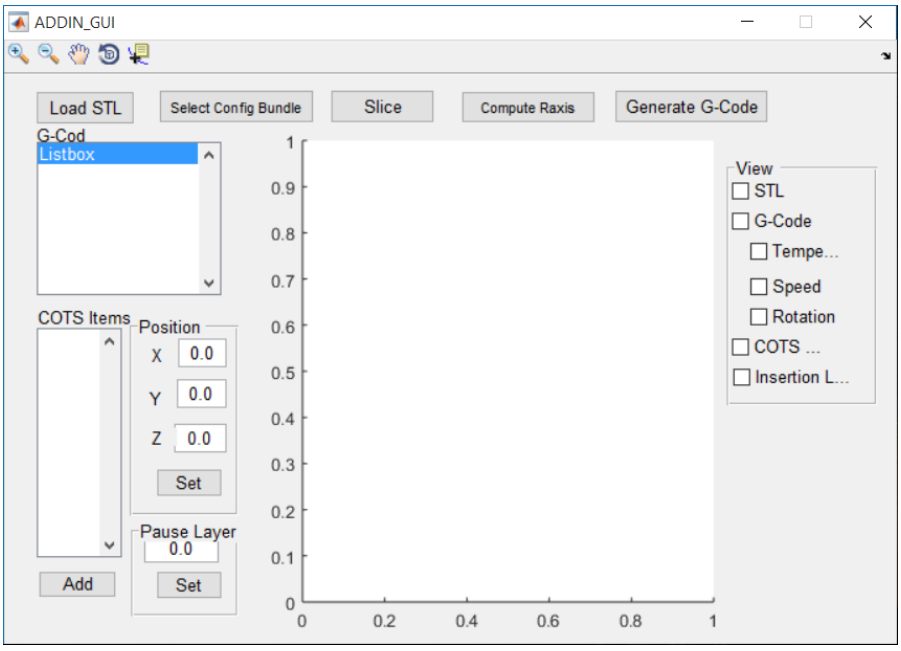

Figure 1: Basic GUI

The basic functionality of the various buttons is given below:

1. Load STL file: This allows the user to select the STL file of the part to be printed
2. Select Config Bundle: This button allows the user to select the .ini file containing all the specifications necessary for slicer to slice the 3D part.
3. Slice: This command invokes Slic3r from command line using the provided STL file and configuration bundle, saving the generated G-Code in the same location as the STL file.

Using the above buttons, we are able to obtain the standard output of that Slic3r provides. The remaining post processing is undertaken by the remaining options.

COTS items

We can select the STL file of the COTS items to be inserted into the main print. Selecting 'Add' under the COTS Items list leads the user to select the STL file and then opens up the user interface shown by Figure 2a.

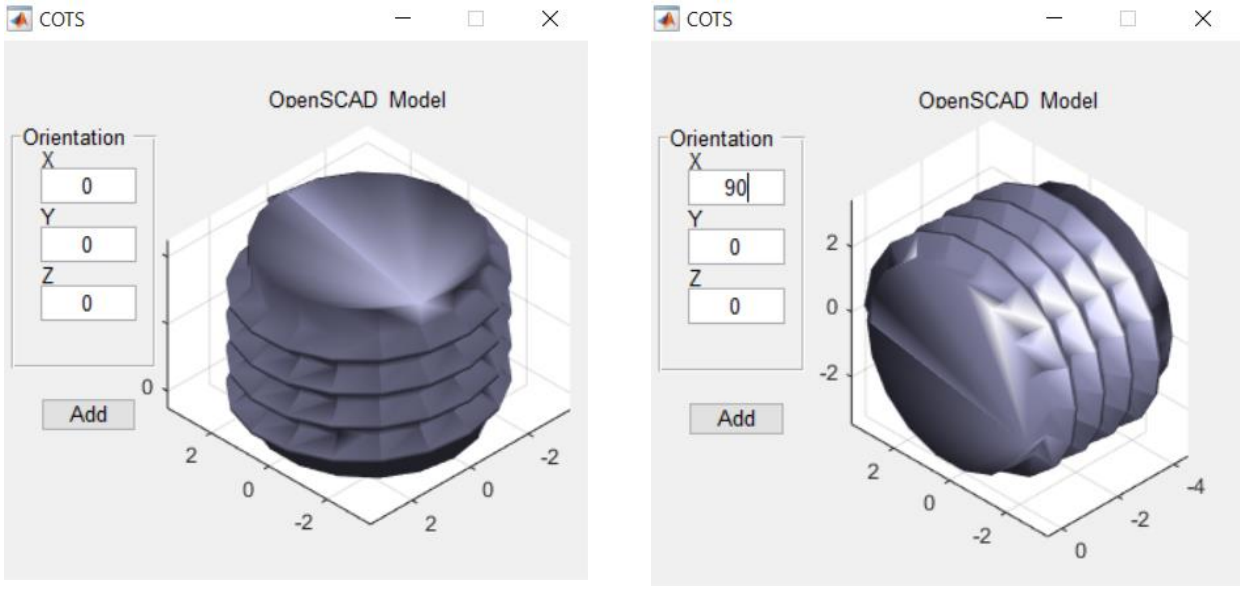

Figure 2 : **a. COTS GUI with screw** **b. Screw rotated by 90 degrees**

This COTS GUI was developed by Dan. It allows the user to rotate the object about the XYZ axes and then add it to the system by hitting 'Add'. The user can similarly introduce several COTS items to the system, which get populated in the 'COTS Items' list. Figure 2b shows a screw rotated by 90 degrees about the X axis.

Position

Once we have added the STL files for the 3D part and COTS item, we can select their corresponding check boxes to view them in the same coordinate frame. Using the 'Position' options from the left of the plot, we can move the COTS item along the XYZ axes in order to position it within the 3D part. The positioning of the COTS item determines the angle at the R commands. Figures 4 shows the both STL parts in the same coordinate frame. Figure 5 shows the COTS part translated along XYZ by [10, 10, 5].

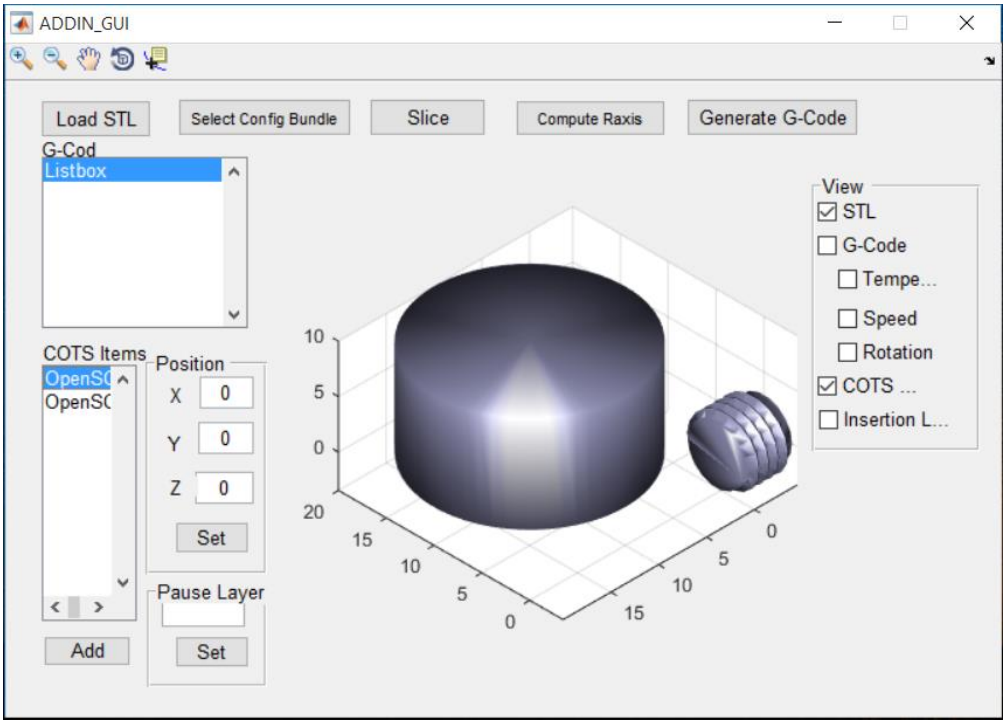

Figure 4: 3D part and COTS item in one frame

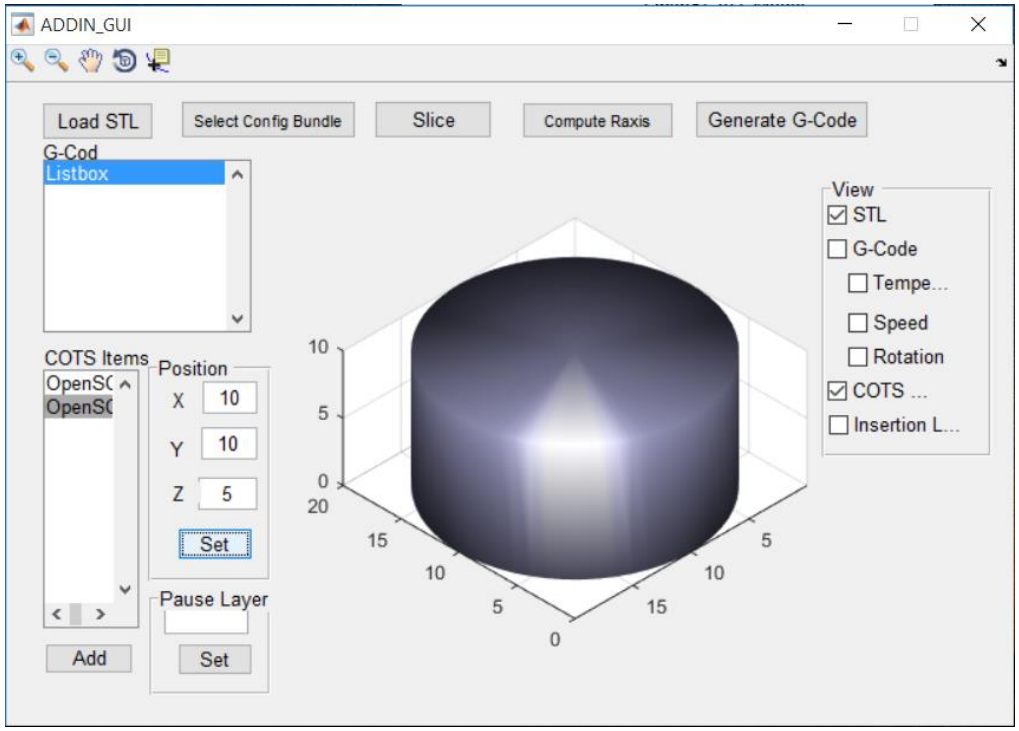

Figure 3 : COTS item completely enclosed in 3D part

Once the item has been placed in its position, we can hit the 'Compute R Axis' button in order to generate the R commands for each printer state/G-Code command.

Plots

The ‘Compute R Axis’ button calls a function that reads G-Code and saves it into an array of Printer States. Once we have this information readily accessible, we are able to plot several variables pertaining to the print, such as the speed of the nozzle (Figure 5), the temperature of the nozzle (Figure 6), tool path (Figure 7), rotation angle (Figure 8), etc. The functions to plot these variables were written by me and their efficiencies increased by Dan.

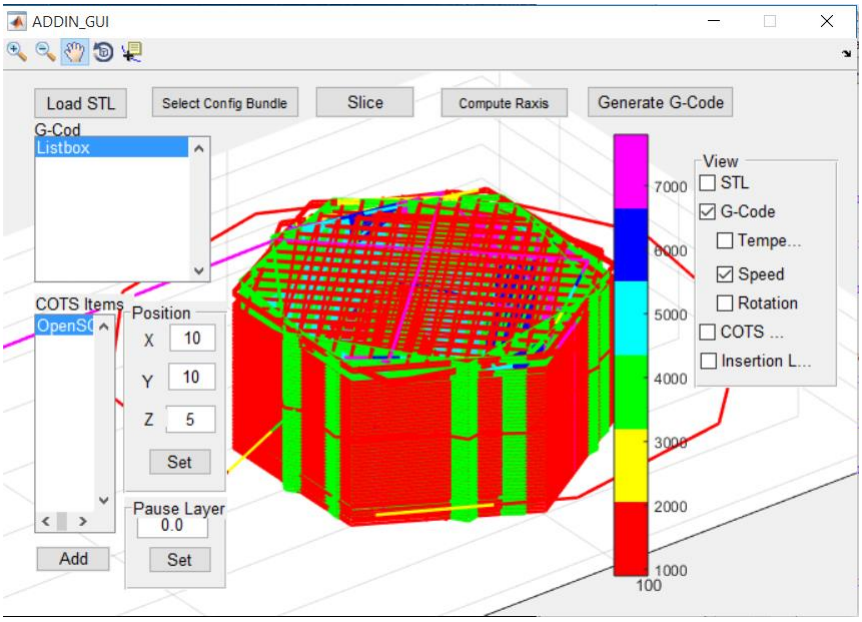

Figure 5: Speed Map

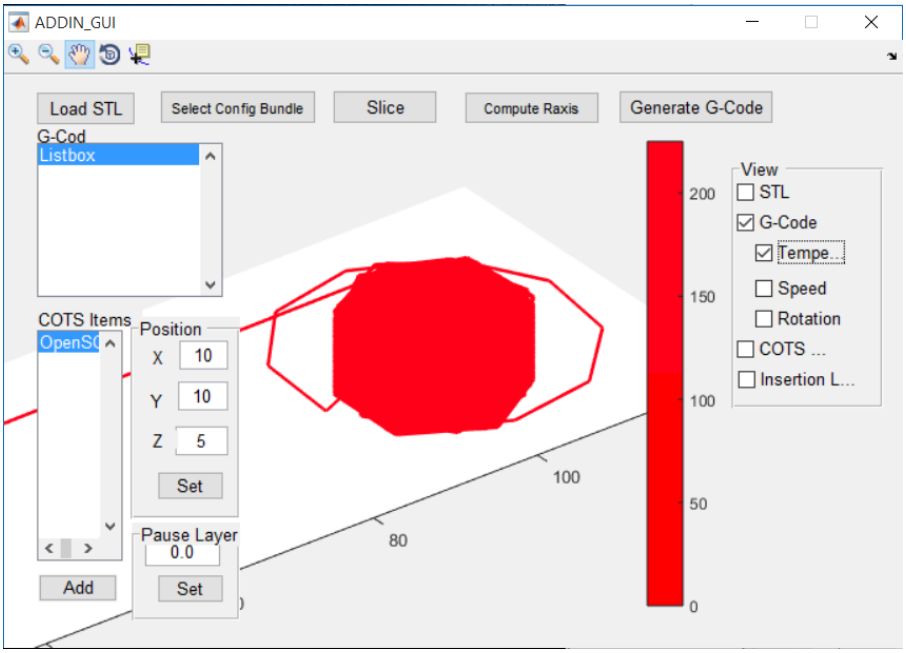

Figure 6 : Temperature map (This file maintains one temperature throughout the print)

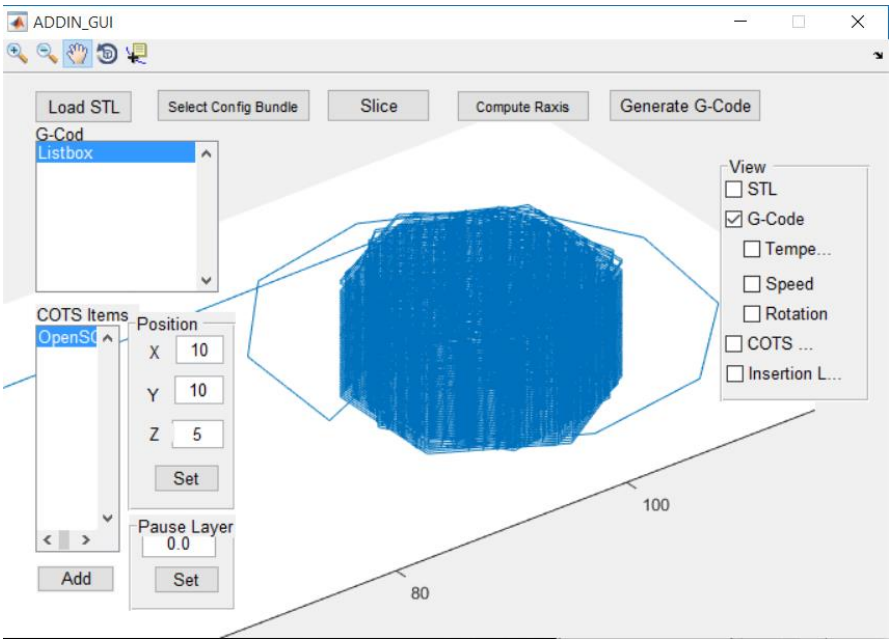

Figure 7 : XYZ positions of the nozzle

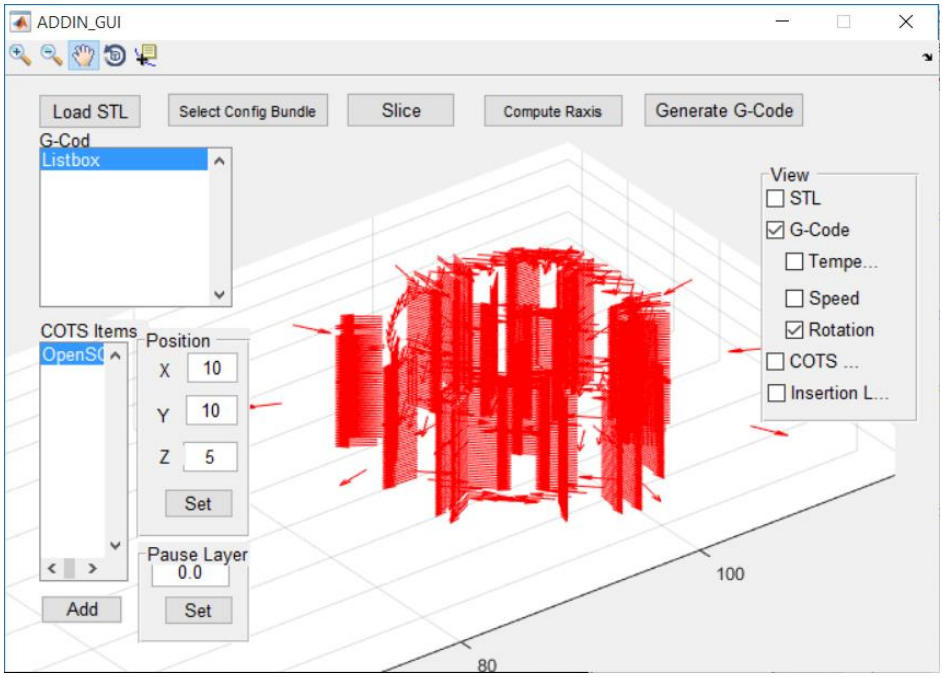

Figure 8 : Rotation angles

In order to print efficiently around COTS items, we may require to increase the temperature of the nozzle when it is close to the part. Hence, having this information readily viewable will be helpful during testing and assist in increasing the performance of our printer.

Since plotting these maps for all possible Printer States is a very slow process, we wrote a function called 'computePlotArr' which saves all the Printer states that are a certain Euclidean distance away from each other. This greatly increased the speed of producing plots.

Generate G-Code

Finally, once the post processing is done, we are able to export the final G-Code into a .gcode file using the 'writeGCodeFile' which is called when the user presses the 'Generate G-Code' button. Hence, a user is able to generate 4 axis GCode using this GUI without any knowledge about the code behind it. This GUI will greatly help us in the testing process, allowing the entire team to generate the necessary code for various 3D parts.

Challenges

There were no particular challenges faced by the software team over the last 2 weeks. Team F in general faces quite a few challenges with firmware and testing. Due to this, we are running slightly behind schedule with regards to testing the printer moving and printing in all 4 axes. However, we expect to tackle these issues before our coming PR.

Teamwork

Remaining goals for Progress Review 8 were distributed as follows:

Ishane Debbache

Ihsane was in charge of assembling the nozzle with the slip ring and mounting it onto the printer. However, the filament got jammed during testing, due to which he was held up in his tasks. The entire team was curious to find out how the new nozzle will perform so we all intend to help with the subsequent testing processes.

Nikhil Baheti

Nikhil was working towards modifying the firmware on the RAMBO board in order to be able to drive all the motors simultaneously. Due to some unidentifiable bug, he is currently unable to integrate all the functionality into one code. I plan to assist him in debugging his code as a fresh set of eyes might be helpful.

Dan Berman

Dan and I worked together on developing the GUI. We took care of the COTS item addition and improving the return time on the plotting functions while I tackled the overall pipeline for generating G-Code. He also machined his own nozzle design a well as the one Ihsane designed. His nozzle was able to extrude filament smoothly. He plans to mount it onto the printer this week.

Future Goals

The GUI presented by the software team this week was a preliminary/rough working model. There are still corner cases where certain functions fail to work. For example, in the computeRAxis function, we noticed that some of the angles were returned as 'NaN's which we plan to fix in the coming weeks. Also, we plan to implement a Collision Check function which will check for collisions in every Printer State. Remaining team goals include: Firmware

1. Control port E1 along with all other ports
2. Have one working firmware

Nozzle
1. Complete mounting of the nozzle
2. Test both nozzles