PROGRESS REVIEW 7

Astha Prasad

Team F / ADD_IN

Teammates: Daniel Berman, Nikhil Baheti, Ihsane Debbache

ILR #6

January 28, 2016

Introduction

After spending nearly a month apart during the winter break, Team F had a meeting early this semester to draw up the upcoming goals for the next Progress Review. The goals were largely divided into software, microcontroller firmware and nozzle design. Among these, Dan and I were in charge of the Software goals.

Overview

Over the fall semester, the software tasks were mainly aimed at providing a proof of concept for the necessary functions. These functions included insertion layer selection, Path planning and r-axis G-Code generation, 3D part visualization, etc. While these tasks had all been executed separately, there were several drawbacks:

- They were inefficient as they read, processed and wrote G-Code from a text file line by line
- Difficult to use for those not acquainted with the code
- Not easily scalable to incorporate more functionality

As a result, the Dan and I decided to lay an extensive framework for the software that would rely on object oriented concepts. All the different parameters and functions pertaining to a 3D part would be tied together using classes, thus creating a more robust base on which to build and expand the functionality of our subsystem such as introduce thermal mapping and visualization of COTS parts. Although the results of the work over the last 2 weeks are the same as those achieved last semester, the back end is completely object oriented.

Classes

The subsystem currently employs the use of two Classes: ADDIN and PrinterState.

PrinterState

The class PrinterState binds all the state variables related to one G-Code command together. These include the XYZR positions of the motors, temperature of the extruder and bed, speed, extrusion rate, etc. With the help of this class, we are able to read G-Code line by line, translate all the state variables into an object of class PrinterState and hence create a table as shown below. Once all the G-code has been read and converted to an array of states, we are free to process the data without having to deal with the .gcode file.

Table 1 : G-Code represented in tabular format

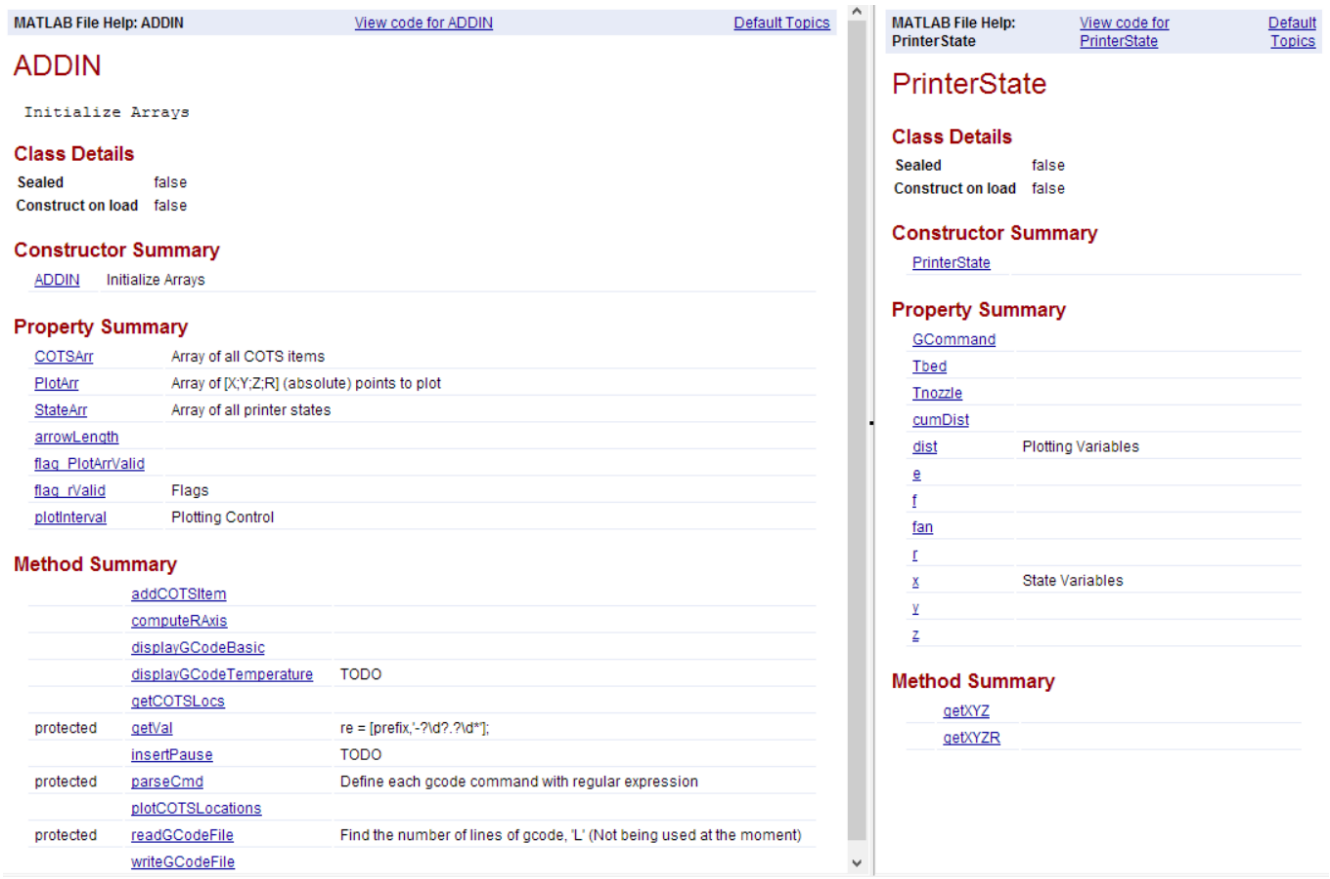
Index	GCommand	X	Y	Z	R	E	F	Tbed	TNozzle
Obj 1	'G01'	2.434	1.876	1.5	30	10	1800	60	215
Obj 2	'G01'	4.234	4.764	1.5	30	15	1800	60	215
Obj 3	'G01'	5.876	8.764	1.5	30	20	1800	60	215
…									

ADDIN

The ADDIN class contains all the data and functions pertaining to one particular 3D part. It can perform the following functions:

- Read G-Code file into array of PrinterStates
- Compute the R axis commands
- Plot the 3D part
- Write PrinterStates back into a .gcode file

An overview of the two classes as created in MATLAB is given below:



MATLAB File Help: ADDIN — View code for ADDIN — Default Topics

ADDIN

Initialize Arrays

Class Details

Sealed	false
Construct on load	false

Constructor Summary

ADDIN	Initialize Arrays

Property Summary

COTSArr	Array of all COTS items
PlotArr	Array of [X;Y;Z;R] (absolute) points to plot
StateArr	Array of all printer states
arrowLength	
flag_PlotArrValid	
flag_rValid	Flags
plotInterval	Plotting Control

Method Summary

	addCOTSItem	
	computeRAxis	
	displayGCodeBasic	
	displayGCodeTemperature	TODO
	getCOTSLocs	
protected	getVal	re = [prefix,'-?\d?.?\d*'];
	insertPause	TODO
protected	parseCmd	Define each gcode command with regular expression
	plotCOTSLocations	
protected	readGCodeFile	Find the number of lines of gcode, 'L' (Not being used at the moment)
	writeGCodeFile	

MATLAB File Help: PrinterState — View code for PrinterState — Default Topics

PrinterState

Class Details

Sealed	false
Construct on load	false

Constructor Summary

PrinterState

Property Summary

GCommand	
Tbed	
Tnozzle	
cumDist	
dist	Plotting Variables
e	
f	
fan	
r	
x	State Variables
y	
z	

Method Summary

getXYZ
getXYZR

Figure 1: Class descriptions by MATLA

Software Work Distribution

After drawing up an initial structure for the Software subsystem, Dan and I divided up the work as equally as possible. Dan was responsible for creating the classes and defining the necessary parameters and functions they would require. He also integrated the stand alone functions written last semester into these classes. These included the 4th asix G-Code generation and path planning. I was in charge of writing the two new functions for reading and writing G-Code which were then integrated into the ADDIN class.

Challenges

Since the software Dan and I wrote was closely tied together, one of the main issues I faced was getting used to collaborating with someone in order to write related code. I was nervous about how well our separate codes would integrate. However, rigorous discussions with Dan about our eventual goal for the subsystem helped me understand exactly what was required.

Teamwork

Remaining goals for Progress Review 7 were distributed as follows:

Ishane Debbache

Ihsane was in charge of designing the next nozzle iteration in SolidWorks which will be machined and tested in the coming weeks. Dan too designed a different iteration of the nozzle which too will be assembled soon.

Nikhil Baheti

Nikhil took up the task of modifying the firmware to enable it to drive 4 motors synchronously. He ran into some hardware limitations which will be tackled in the coming weeks.

Future Goals

Our goals for the next 2 weeks include:

1. Complete firmware modifications
2. Machine and test both nozzle designs
3. Attempt printing around COT items
4. SOFTWARE: Build a GUI capable of
    - Plotting temperature maps
    - Displaying COTS items
    - Integrating with Slic3r
    - Insertion layer selection