Ruben Heyse 20254261

Computer Engineering and Mechatronics Project: Developing software for a drawing robot

Software Description

The task of this program is to do as follows:

1. Based on const defined parameters, generate a lookup table (via a stuct array) for the (X, Y) start position for each shape on the grid
    1. Define
        1. Square size and x, y count
        2. Store local (0, 0) for where the shape is to be drawn in each grid square.
2. Define the shapes that can be drawn
    1. Read ShapeStrokeData.txt
    2. Dynamically create struct for each shape, storing the name and drawing instructions
3. Read the drawing instructions from an ASCII text file, specified by the user.
4. Relate the instructions given, to the stored shape stroke data from ShapeStrokeData.txt.
    1. Read shape name from instruction file and find correct shape struct
5. Generate the G-Code required to draw the shapes defined in the instructions file.
    1. Given the position of the shape defined in the instruction file, find the required x, y coordinate in the lookup table
    2. Generate the required G-Code for the shape given the position found in the lookup table
6. Send the required G-Code to the robot to initialise its state.
7. Send the G-Code of the required shapes to the robot.

Upon completion of the drawing, the pen should finish in the pen-up position. Furthermore, when the multiple drawig instruction files are given by the user, these should all be drawn in the same grid. As such, the pen should reset to (0,0), so as to re-index itself.

Key Data Items

Name	Data type	Rationale
grid	Struct	Pseudo OOP allows for separation of concerns
grid.cell_width_mm	Int	It would be an irrational over-complication to expect millimetre dimensions to be floating point
grid.cell_height_mm	Int	See above
grid.cell_x_count	Int	A non-integer number of cells would be illogical
grid.cell_y_count	Int	See above
grid.total_width_mm	Int	Allows the value to only need to be calculated once and then can be looked up. Since the cell dimensions and count are count are integers, it may be assumed that the total dimension should be an integer.
grid.total_height_mm	Int	See above
cell	Struct	Will be initialised as an array of structures, so that each element will describe a cell. An example of this may be found here: https://www.javatpoint.com/array-of-structures-in-c. This allows the array to be treated as a runtime JSON store.
Cell.global_origin	Array	The origin coordinates for the cell. Since all dimensions are in integer millimetres, it may be assumed that the origin is also an integer value.
Cell.shape_local_origin	Array	Based on the desired scaled shape (defined below), the origin of a shape drawn within the cell may be stored here. This stops repeatedly calculating the same coordinate.
Shape	Struct	Handled identical to the cell struct, this will be initialised as an array of structs, so that each element can hold the descriptive information of a shape.
Shape.name	Char (String)	Strings in C are handled as a 1D array of chars.
Shape.drawing_information	Struct	Again, handled like as a struct array, each element will contain a line from the drawing instruction file. While this can also be done as a 2D array, using a struct allows the code to be semantic and readable.
scaled_shape_width_mm	Const	Since the shape aspect ratio is constant, only the width needs to be defined for the scaling factor to be calculated. As this is pre-defined, the type can be a constant.
shape_point_width	Const	This needs to be defined since a shape may not take up the entirety of the 16 point grid.

		However, since this value does not change at runtime, a const may be used.
shape_scale_factor	Float	Since the calculation is a division, this value will need to be a decimal. Using the example data to scaling the 16 point shape to 20mm, the scale factor in 1.25.
shape_file_name	Char (String)	The name of the shape file that is to be read.
shape_file_data	Char (String)	The current line of the shape file.
drawing_instructions_filename	Char (String)	The name of the drawing file that is to be read.
drawing_instructions	Char (String)	The current instruction being executed.
g_code_instruction	Char (String)	The generated GCode instruction that is to be streamed to the robot.

Function Declarations

initialise_robot

- Params
  - void
- Return
  - void

initialise_grid

- Params
  - *grid
- Return
  - success → int

initialise_cells

- Params
  - *grid
  - *cell
  - scaled_shape_width_mm
- Return
  - success → int

define_shapes

- Params
  - *shape_file_data
  - *shape
- Return
  - success → int

generate_g_code

- Params
  - *shape
  - *cell
  - *grid
- Return
  - g_code_instruction → char

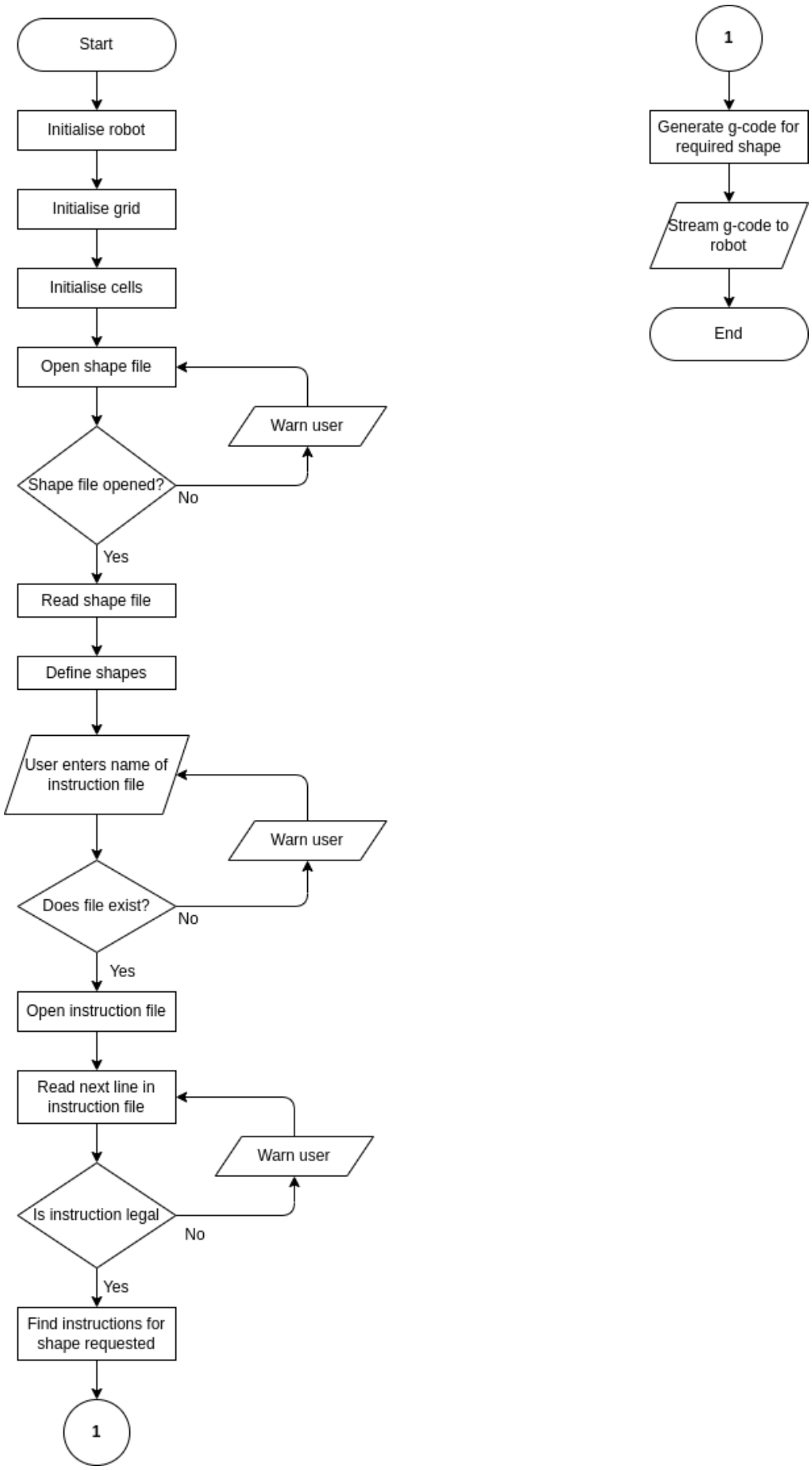
Testing Information

Function	Test Case	Test Data	Expected Output

initialise_grid	Test for illegal data	Grid{
		cell_width_mm = 1000
		cell_height_mm = 1000
		cell_x_count = 30
		cell_y_count = 30
		}
			0

initialise_cells	Test for correct positioning	Grid{ cell_width_mm = 30 cell_height_mm = 30 cell_x_count = 3 cell_y_count = 3 }, scaled_shape_width_mm = 20	1

Flowchart(s)


Start
Initialise robot
Initialise grid
Initialise cells
Open shape file
Warn user
Shape file opened?
No
Yes
Read shape file
Define shapes
User enters name of instruction file
Warn user
Does file exist?
No
Yes
Open instruction file
Read next line in instruction file
Warn user
Is instruction legal
No
Yes
Find instructions for shape requested
1
1
Generate g-code for required shape
Stream g-code to robot
End