# Design Section

## 1. Structure

The starting point of the program is the program object, which stores references to the different parts of the program. The program object has three such members: window, diagram, and preferences. diagram and window both have references to the program object, allowing them to communicate.

The diagram object stores the currently loaded diagram including all of the current plots. It also handles serialising diagrams to .arg files.

The window object is a QMainWindow subclass which stores the entire GUI. It is responsible for handling inputs and drawing plots, as well as creating dialogs boxes.

preferences is a simple object which stores a few diagram unspecific preferences.

## 2. Data Dictionary

## 3. Inputs/Outputs

### GUI DesignE:\Documents\Design\file_menu.jpgE:\Documents\Design\help_menu.jpgE:\Documents\Design\view_menu.jpg

## 4. Processes

### Procedures

**Parse an entered locus:**

* **Validate the locus:**
  + Does the locus take the form , where:
    - is one of , , , , , , or a constant .
      * All are a single letter other than .
      * Each is either a real number, a real number succeeded by the letter (an imaginary number), or a sum of the two.
    - is one of , , , , or .
  + If is or , is a constant, and is the locus an equation?
  + If is or , is a real constant or another modulus function?
  + If is or , is a real constant?
  + These cases cover all the diagrams the software is required to plot. The following cases may also be simple to implement, but are not required:
  + If is or , is or (where the are equal)?
  + If is or , is also an argument function?
* **Store the locus:**
  + If the locus represents a point, *i.e.* takes the form , store the point as .
  + If the locus represents the perpendicular bisector of points and , find and store the Cartesian equation of the line.
    - The midpoint of and is .
    - The gradient of the line is given by .
    - Hence, the imaginary intercept of the line is .
    - So the equation is written .
  + If the locus represents a circle, *i.e.* takes the form , store the centre and radius as and respectively.
  + If the locus represents a ray, *i.e.* takes the form , store the start point as , and the gradient as .

**Draw a stored locus:**

* **If the locus is a point:**
  + Draw the point using the graphics library.
  + If enabled in the preferences, label the point with its coordinates.
* **If the locus is a circle:**
  + Use the graphics library to draw a circle with the stored radius and centre.
* **If the locus is a disk:**
  + Draw a circle with a dashed stroke if applicable, and use a solid colour fill.
* **If the locus is a perpendicular bisector:**
  + Draw the line:
    - Find the current lower and upper visible values on screen.
    - Plug these into the line equation to find the start and end-points of the line.
* **If the locus is a half-plane:**
  + Draw the line, with a dashed stroke if applicable.
  + Draw the inequality region with a solid colour fill.

**Set window size:**

* **On window close:**
  + If config.ini does not exist, create it.
  + Update variables window\_width and window\_height.
* **On window open:**
  + If config.ini does exist:
    - Set the window size to the values of window\_width and window\_height.
  + Otherwise create it, and use the default window size.

### Regular Expressions

This regular expression matches a single numerical term. This could be used to find values of .

/([+-]\s\*)?(\d\*\.)?\d+i?(?=\s\*[\)|<=>+-])/g

This regular expression matches a single occurrence of an unknown, with or without a coefficient. This can be used to find instances of during validation, and to check the letter used for the unknown is consistent.

/([+-]\s\*)?((\d\*\.)?\d+)?[a-hj-z](?=\s\*($|[\)|<=>+-]))/g

This regular expression is needed to match a group of numerical or unknown terms separated by whitespace, without matching whitespace on either end.

/([+-]\s\*)?((\d\*\.)?\d+i?|((\d\*\.)?\d+)?[a-hj-z])(\s\*([+-]\s\*)?((\d\*\.)?\d+i?|((\d\*\.)?\d+)?[a-hj-z]))\*(?=\s\*($|[\)|<=>+-]))/g

However, if all whitespace is stripped from the input, the expression can be reduced to this.

/(([+-])?((\d\*\.)?\d+i?|((\d\*\.)?\d+)?[a-hj-z]))+(?=$|[\)|<=>+-])/g

*From this point, all regular expressions require the input contain no whitespace.*

This regular expression matches a modulus function and its contents.

/\|[^|]\*\|(?=$|[<=>+-])(?<=[<=>+-])/g

This similar regular expression matches an argument function and its contents.

/arg\([^\(]\*\)(?=$|[<=>+-])(?<=[<=>+-])/g

Finally, this regular expression matches all valid equality symbols: “<=”, “<”, “=”, “>” and “>=”.

/(=|[<>]=?)(?=[^<=>]+)(?<=[^<=>]+)/g

## 5. Storage

### config.ini definition

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Purpose** |
| label\_points | Boolean | Whether labels should be drawn for points. |
| label\_axes | Boolean | Whether labels should be drawn for axes. |
| font\_size | Integer | Font size to use when drawing labels. |
| stroke | Integer | Thickness in pixels of plotted lines. |
| window\_width | Integer | Stores the last known width of the window. |
| window\_height | Integer | Stores the last known height of the window. |

### .ARG DEFINITION

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Purpose** |
| version | String | Used to check if the file is compatible with the program. |
| plots... | Array | Array of serialized plot objects that can be recreated using pickle. |
| translate | Coordinate | Stores the last point the display was centred around. |
| zoom | Number | Stores the last zoom value used for the diagram. |

## 5. Security

The system will not store any sensitive data, nor will it communicate over any network. There is very little security risk involved with the system, so it does not require any special measures.