AG2411

GIS Architecture and Algorithms

User Manual

Alexandre Barbusse barbusse@kth.se

Anna Nordlöv anordlov@kth.se

Petteri Pesonen tppe@kth.se

Oliver Stromann stromann@kth.se

Kungliga Tekniska Högskolan

Table of Contents

List of Figures

1 Abstract of the application

PAOA is a raster-based geoinformation system that was developed in 2016 by the students Alexandre Barbusse, Anna Nordlöv, Petteri Pesonen and Oliver Stromann under the course *AG2411: GIS architecture and algorithms* at the Royal Institute of Technology in Stockholm.

1.1 Focus group

PAOA has been designed to be accessible to both people who are beginners in GIS utilisation and need to work with raster data, and people that want to implement some specialized raster-based layer operations.

PAOA is easy to learn, and provides a user-friendly interface conceived to be less intimidating as most common GIS. Indeed, all commands are accessible using the same toolbar, which has been thought to be straightforward, avoiding confusions caused by mysterious symbols. Figure X allow you to check yourself that PAOA's GUI is clear and user-friendly.

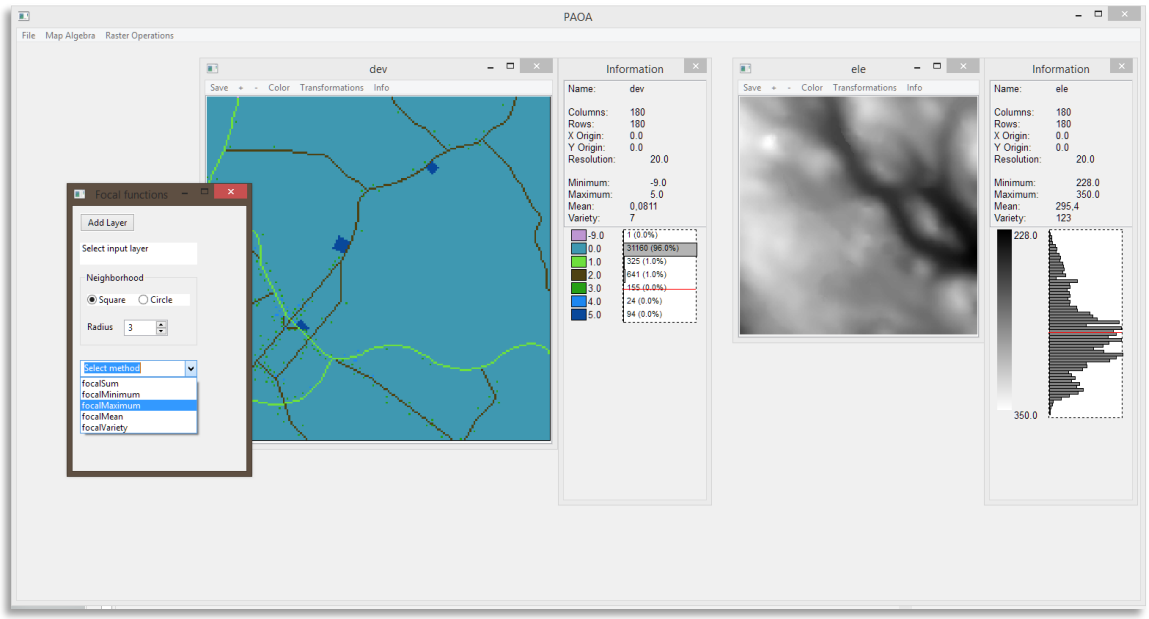


Figure 1: PAOA user interface presentation

To more advanced users, please note that even though PAOA initially focus on users needing to process terrain surface operations for digital elevation models (DEM), the architecture presented in section 1 allows you to create all kinds of extensions by developing new Java methods.

1.2 Features

PAOA can handle raster data layer files in ASCII format and provides a toolbox with different manipulation options. The raster operations you can access using PAOA are divided in two main types: map algebra operations and general (raster) operations.

Map algebra operations provide local operations tools, focal operations tools and zonal operations tools. They all apply mathematical operations on input the raster values. A more detailed description of these different map algebra operations tools, please refer to the toolbox manual in section 4.

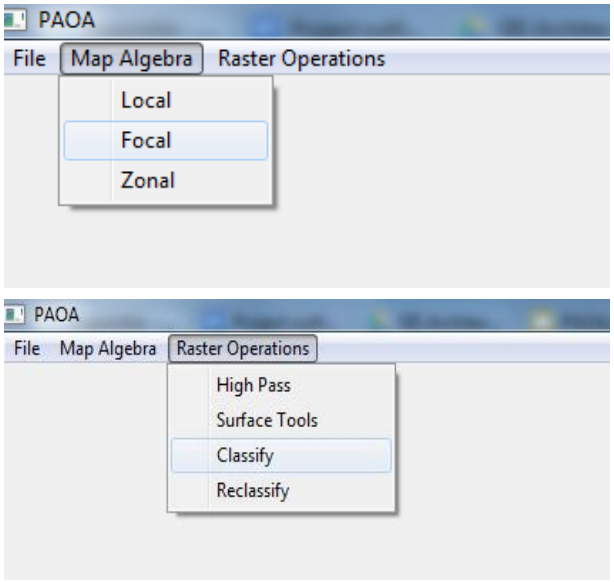


Figure 2: Accessing available raster operations

Among the general operations in the PAOA software, high pass filter, classify, reclassify, and surface tools for digital elevation models, such as slope, aspect and hillshade are available. Find more information about these tools in the toolbox manual.

PAOA also allows you to work directly on a layer you have previously opened by assigning colors on it, transforming the layer while conserving values using geometrical operations such as rotation, mirroring or stretching, or simply visualizing metadata and elementary statistics in the layer using the info tool.

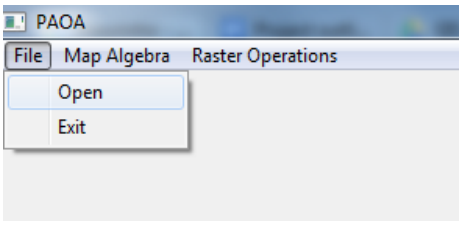


Figure 3: Opening a new layer

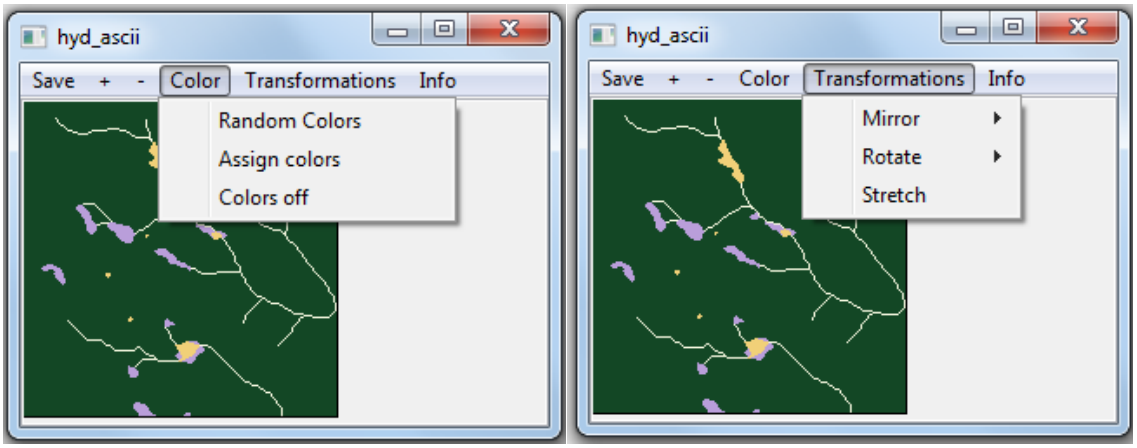


Figure 4: Working with operations directly on the layer

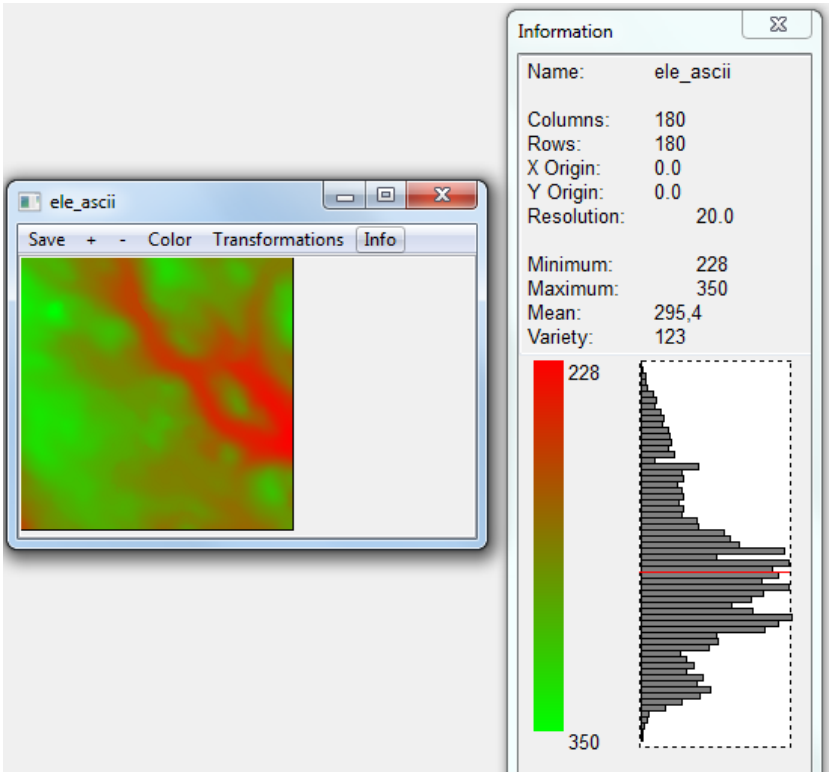
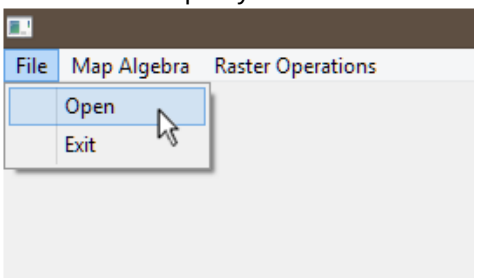


Figure 5: Opening the info toolbox

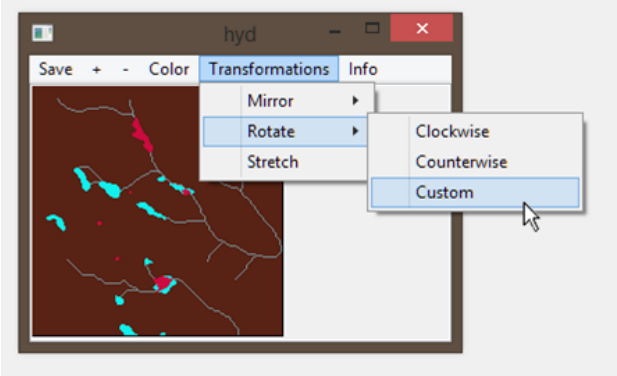
1.3 Workflow

RASTER LAYER BASIC WORKFLOW

1. Open your raster file



2. Do your manipulations



3. Save your output raster

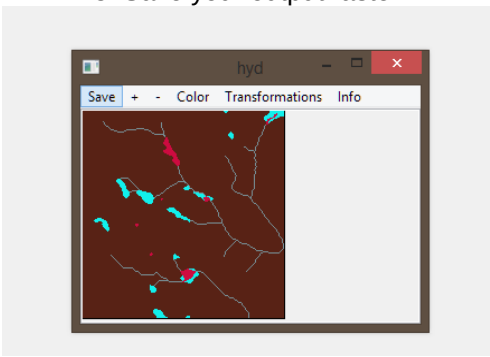
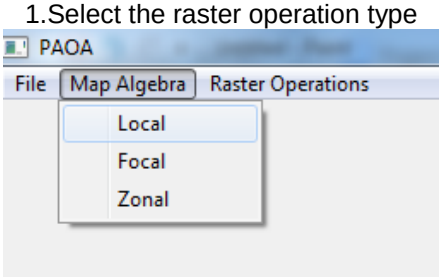


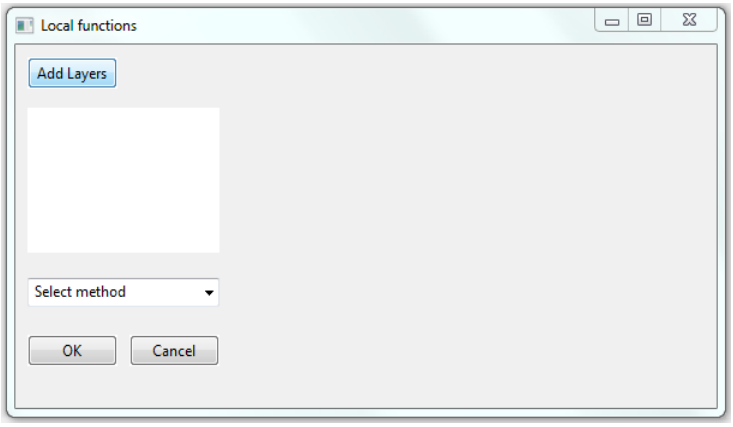
Figure 6.1 to 6.3: raster layer basic workflow

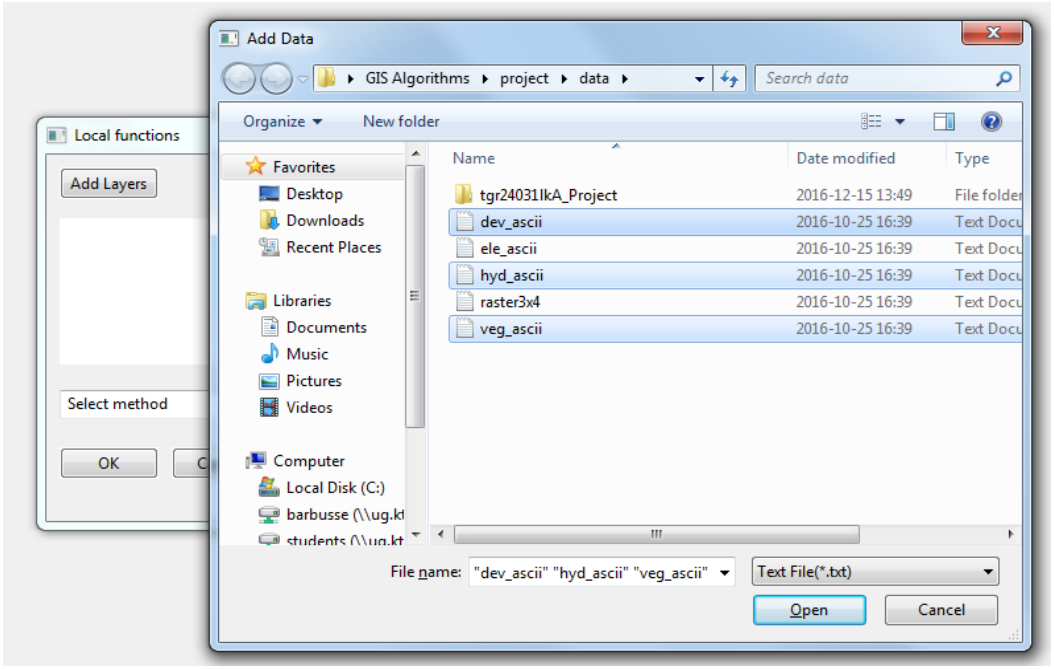
RASTER OPERATION WORKFLOW

1.Select the raster operation type

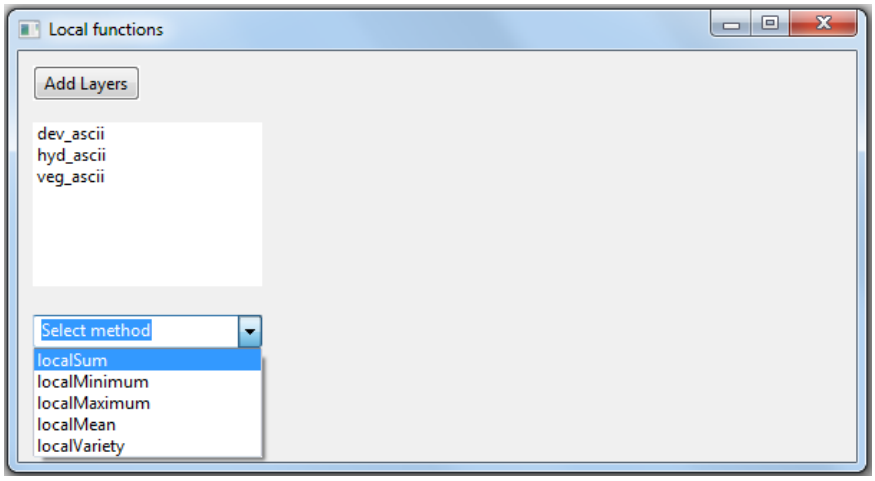


2. Select your input raster files

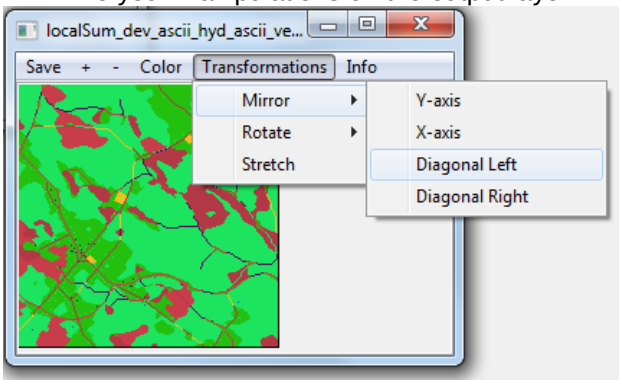




3. Select the function



4.Do your manipulations on the output layer



5. Save the final output raster

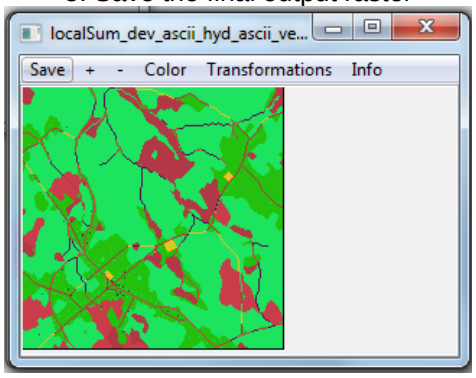


Figure 7.1 to 7.5: raster operation basic workflow

2 Data formats

2.1 Formats of data which can be read or saved

PAOA is designed to work exclusively with ESRI ASCII format raster file as input, and creates rasters in the same format. Consequently, any output saved with PAOA will also be an ESRI ASCII format raster file. You can find the basic source information about this format at:

http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=ESRI%20ASCII%20Raster%20format.

Please note that the following subsection is extracted from the above mentioned webpage.

An ESRI ASCII format raster file is divided in two main data blocks: a header, with present information of the raster structure, and a value block with all the cell value data.

```
ncols         XXX
nrows         XXX
xllcorner     XXX
yllcorner     XXX
cellsize      XXX
NODATA_value  XXX
row 1
row 2
...
...
row n
```

Figure 8: Header structure required for the ESRI ASCII raster file format[1]

2.1.1 Header format

As explained by ESRI, the syntax of the header information is a set of keywords associated by a set of corresponding values. Here are the definitions and requirements for these keywords and values.

[1] Adapted from:
http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=ESRI%20ASCII%20Raster%20format

Table 1: Header information format requirements[2]

Parameter	Description	Requirements
ncols	Number of cell columns	Integer greater than 0
nrows	Number of cell rows	Integer greater than 0
xllcorner	X coordinate of the lower left corner of the cell	double
yllcorner	Y coordinate of the lower left corner of the cell	double
cellsize	Cell size	double greater than 0
NODATA_value	The input values to be NoData in the output raster	Optional. Default is 20

2.1.2 Data format

As explained by ESRI, the data of the ESRI ASCII raster must follow the header information.

Cell values have to be delimited by spaces.

No line break are required at the end of each row in the raster. The number of columns (nCols) of the header information determines when a new row begins

Row 1 of the data is the first one at the top of the raster, followed by row 2 under row 1, and so on until the corresponding number of rows declared with nRows is reached

2.1.3 Example ASCII raster

Here is an example of the ASCII raster file format:

```
ncols 180
nrows 180
xllcorner 0.0
yllcorner 0.0
cellsize 20.0
NODATA_value 20.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0
3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 3.0 4.0 4.0 4.0
```

Figure 9: Example of an ASCII raster file

[2] Adapted from: http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=ESRI%20ASCII%20Raster%20format

2.2 NoData-value limitation

PAOA software DOES NOT handle NoData-values. Please make sure that your input raster file does not contain any when opening an input file; many operations may still work but the output will be incorrect.

2.3 Coordinate-system limitation

Please note that PAOA software is preserving the X and Y coordinates of the lower left corner of the raster, which is playing the role of a coordinate system, when a geometrical transformation of the raster - such as a rotation - is performed.

Consequently, the output raster file can be attributed a coordinate system which is not updated to match the transformed layer. Please be aware of the consequences involved when using a file saved after applying a geometrical operation on your input files.

3 Application architecture

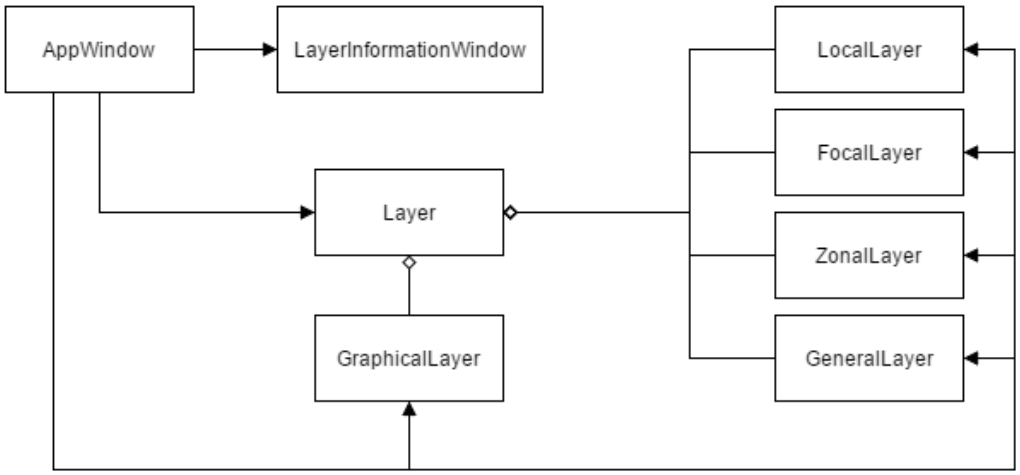


Figure 10: Abstract application class diagram

Abstract application architecture is shown in Figure 10, where the abstract layout of the classes and their relationships can be seen. Current architecture was made to be as extensible as possible. Extensibility means that each component in the architecture can be enhanced with more methods or classes example being the Layer superclass. For example every subclass of the Layer class is extending the Layer class, by using the same constructor principle to create new Layer objects. With this same fashion adding more classes is easy to implement to the current architecture. However, it is good to understand that some changes may cause additional changes other classes. For example, if tools are added as new classes they need to be made accessible from the AppWindow. Therefore, adding new toolboxes there may be needed. The following chapters explain class diagram components in more detail.

AppWindow class is the constructor for the GUI and has the executable main method, which runs the whole application. Another GUI component besides the AppWindow class is the LayerInformationWindow that creates information windows for layer specifically when user desires so. Initially when launching the application from the AppWindow class, it creates a main shell frame that is used for containing every layer, tool or information window that the user can act with inside the application. Also, the main frame has functionalities that the user can access such as opening new layers and toolboxes. Every time a new window is created by opening a layer, through a toolbox or an information window, it is considered as a child shell and they have different content based on the method in hand. Therefore, these child shells have different ways of viewing and disposing them. For example, if a color picker is closed when a color dialog is

open, it will be disposed. This kind of handling is necessary to prevent bugs in the application.

As mentioned previously the child shells will have different content based on what they present. These child shells are classified into three: layer, information and toolbox windows. The creation of these are different for each of them. For the layers a frame is first created that has the menu bar content that the user can use to do different functionalities to this layer shell. After that the content is created, which consists of a scrolled composite that will act as a frame for an SWT image (raster image) that is created in the Layer class and returned. Information shell only has the content, which is layer specific and it is created in the LayerInformationWindow class. Toolboxes have tool specific content, which have options for the user to pick from that are based on the method inputs. These methods are called through the Layer superclass that they extend. However, the toolbox content is created in the AppWindow class.

Figure 11: AppWindow class variables and methods

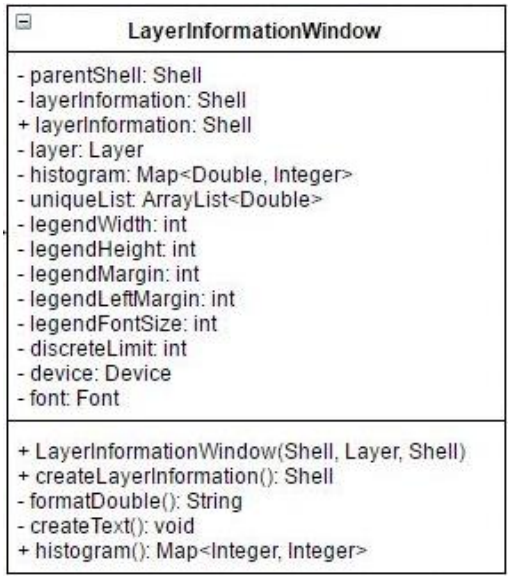


Figure 12: LayerInformationWindow class variables and methods

Layer class is one of the most centric classes in the application, since it is a superclass of many other extending classes. The purpose of the Layer class is to create the layer objects that can be viewed and manipulated inside the application. In addition, the Layer class has many other useful methods such as getting the amount of unique values of specific layer. These methods can then be used in classes such as AppWindow and LayerInformationWindows.

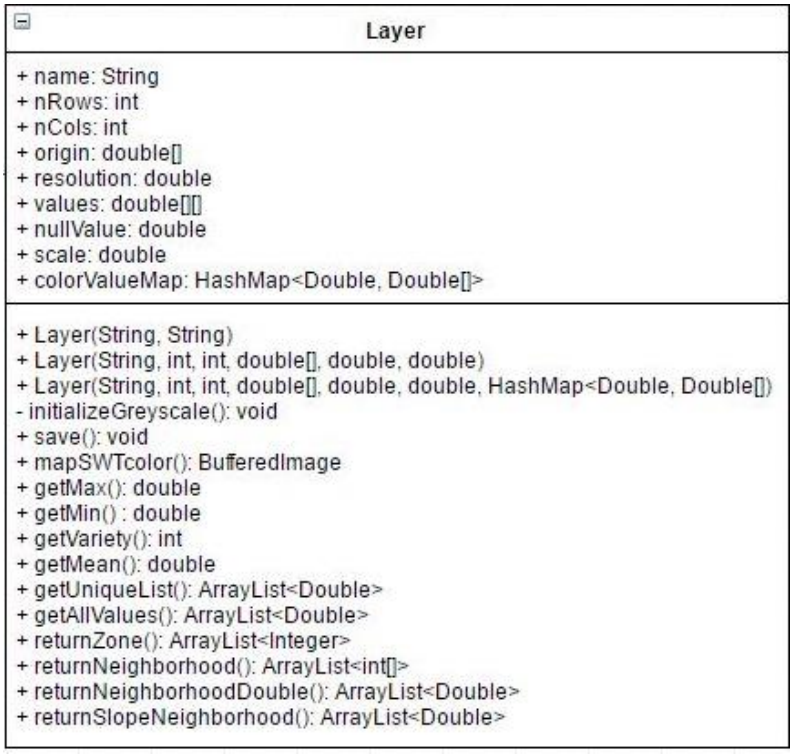


Figure 13: Layer class variables and methods

The rest of the mentioned classes are subclasses of the Layer class. These subclasses create new superclass Layer objects using different methods that are not present in the Layer class. Depending on whether the methods create completely new layer objects or inherit some values from an existing layer object, these subclasses can be divided into two: toolbox and layer specific methods. Toolbox specific classes LocalLayer, FocalLayer, ZonalLayer and GeneralLayer create new layer objects without inheriting the colorValues. Instead in the layer specific class GraphicalLayer also the layer specific colors (if there exists any) are inherited.

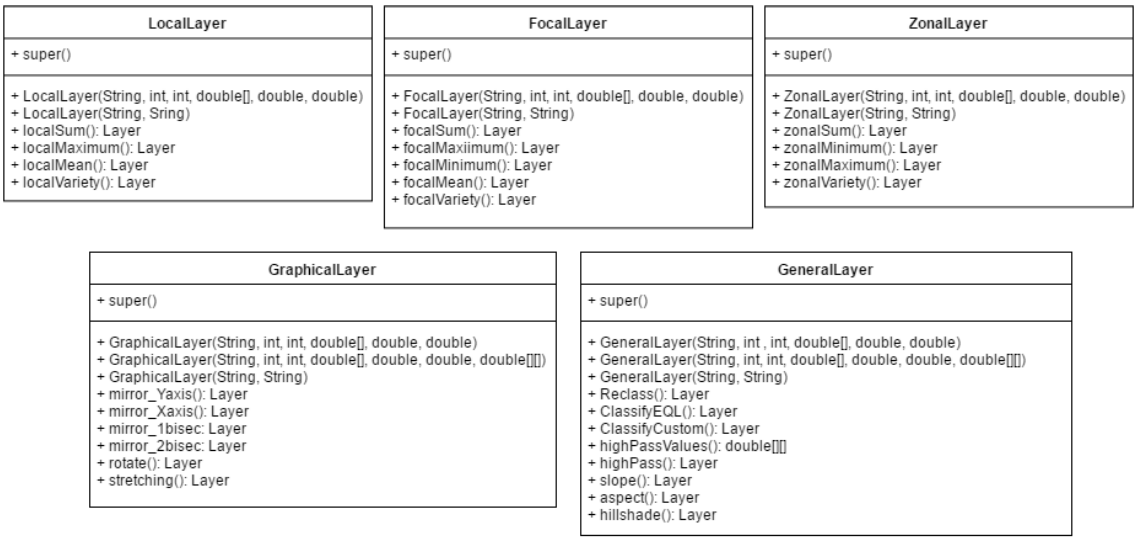


Figure 14: Extended Layer subclasses

4 Tools

The tools in PAOA are divided into six categories; coloring, zonal-, focal-, local-, general- and graphical (transformations) operations. Also for every layer there exist an info-tool that produces an information box containing metadata for the chosen layer. All tools regarding zonal-, focal-, local- and general operations are located in the main window in the upper left corner, where you also open new raster files. Zonal-, focal- and local operations are found under *Map Algebra*, the general operations are found under *Raster Operations*. The graphical operations are found under *Transformations* at the top of every layer-window where also the info-box is accessed. All the different tools are further explained below including their different limitations and restrictions.

NOTE: Do not forget to save your raster after you have done any kind of manipulation, it will not be saved automatically!

4.1 Coloring

The default setting is that all files are opened in grey scale where white represent the highest value and black the lowest value. This is possible to change by clicking *Color* in the top bar of the raster window, however the user must consider the type of data in the file.

Discrete data

- Random colors: All values in the raster will randomly be assigned a color. If you do not like the color this tool can be used multiple times.
- Assign colors: This tool lets the user assign an optional color to one or several values in the raster.
- Color off: Regardless of previous coloring, this tool will restore the raster to its original grey scale values.

Continuous data

- Random colors: Works the same as for discrete data
- Assign colors: For rasters containing continuous data, the software will ask you if you want the data to be discrete. If the user chooses *Yes* the above description is followed. However if the user chooses *No,* another coloring option will appear where a color ramp may be chosen.

- Color off: Works the same as for continuous data. Note: because of a bug, the values might be slightly changed from the original grey scale values if the *Assign colors* option has been used previously.

4.2 Local operations (Map algebra)

Local operations take two input layers and then the specific function is applied to the same cell in both input layers, for every cell. Access the tool at the top bar in the main window by clicking *Map Algebra*. See an example below with the specific function localSum; the adding of every cell in the first input layer to the same cell in the second input layer.

The local-operation toolbox include the following functions:

- Sum
- Min
- Max
- Variety
- Mean

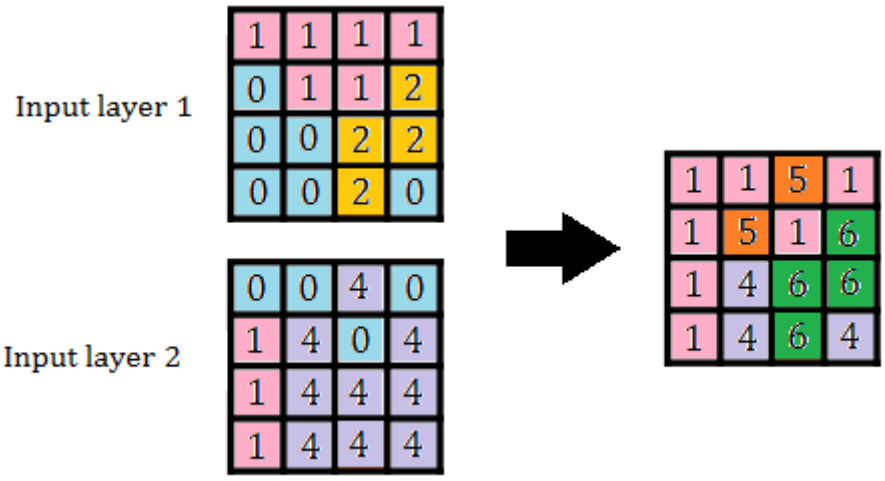


Figure 15: Local sum operation example

4.3 Focal operations (Map algebra)

Focal operations takes one input layer and a specified neighborhood; the specific function is then applied to the whole of the neighborhood of every individual cell and the result of the function is the output value of the cell whose neighborhood has been used.

Access the tool at the top bar in the main window by clicking *Map Algebra*. See an example below of the operation focalVariety shown with different types of neighborhoods. PAOA offers two different types of neighborhoods, square and circular; both of which expands outward to specified radius with the current cell in the middle. The radius can be specified by the user for both types of neighborhoods. For the circular option, all cells whose middle point falls inside the radius is included in the neighborhood.

The focal-operation toolbox include the following functions:

- Sum
- Min
- Max
- Variety
- Mean

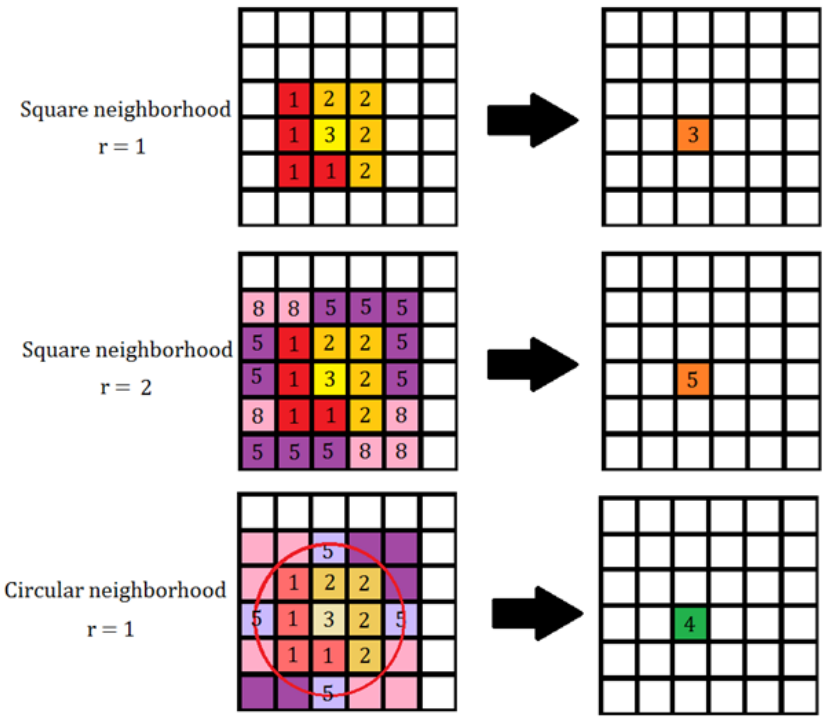


Figure 16: Focal variety example

4.4 Zonal operations (Map algebra)

Zonal operations take two input layers; a *value* layer and a *zone* layer. The zonal layer decides in what areas the specific function should be applied in the value layer. Access

the tool at the top bar in the main window by clicking *Map Algebra*. See an example of the operation zonalVariety in the image below.

The zonal-operations toolbox include the following operations:

- Sum
- Min
- Max
- Variety

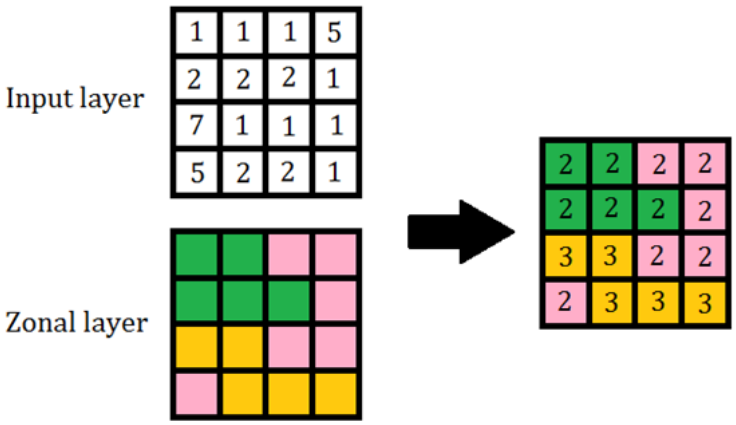


Figure 17: Zonal Variety operation examples

4.5 General operations (Raster operations)

This toolbox include a couple of operations used for different types of manipulation of data. They can be accessed by clicking *Raster operations* at the top bar in the main window. Read more about each of the tools further below, tools included are:

- Surface tools
  - Slope
  - Aspect
  - Hillshade
- High pass filter
- Reclass
- Classify, equal interval
- Classify, custom

4.5.1 Surface tools

4.5.1.1 Slope

Takes an input layer of elevation data and calculates the difference of values between the current cell and its neighbors, this is done for very cell. The neighborhood is a 3x3 pixel area with the current pixel in the middle; in the case of some neighborhood-cells falling outside the raster, the current pixel's value is applied to these cells.

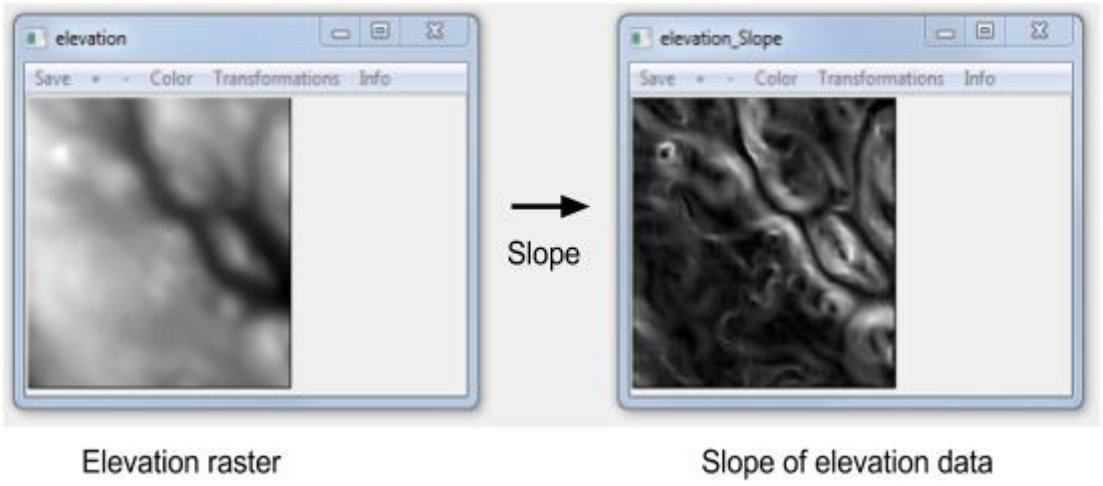


Figure 18: Slope tools applied to elevation data

a	b	c
d	e	f
g	h	i

Figure 19: Slope algorithm neighborhood - Source: Help, ArcGIS for Desktop[3]

[3] Adapted from:
http://desktop.arcgis.com/en/arcmap/10.3/tools/spatial-analyst-toolbox/how-slope-works.htm

```
delta_x := 0
delta_y := 0
rise_run :=0
slope_deg := 0

for each k in K
    out(k):=0

    with (a,b,c,d,e,f,g,h,i) in N(k)
    delta_x := ((c+2f+i) - (a+2d+g) /  (8*x_cellsize)
    delta_y := ((g+2*h+i) - (a+2b+c) / (8*y_cellsize))
    rise_run := sqrt(delta_x² + delta_y²)
    slope_deg := atan(rise_run) * 180/Π
    out(k) := slope_deg

end
```

Where:

- N(k) is the set of all cells that are in cell k's neighbourhood
- Out(k) is the output on cell k

Figure 20: Pseudo-code for the slope algorithm

4.5.1.2 Aspect

Takes an input layer of elevation data and calculates which direction a downslope faces. For example an analyst can discover whether a mountain side faces north or south. The output layer's color/greyscale values defines the direction of the slope according to the following map below.

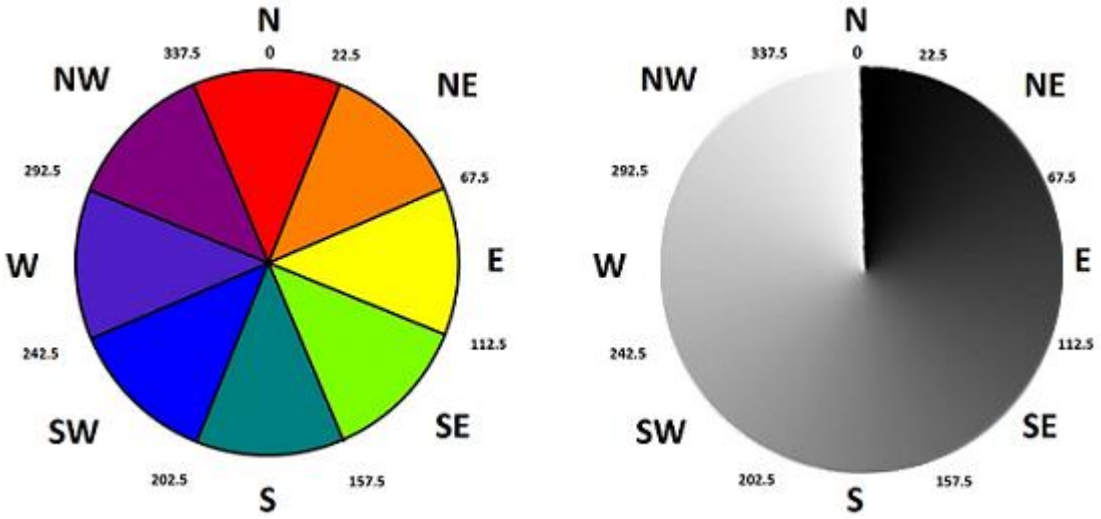


Figure 21: Orientation map of the slope

```
delta_x := 0
delta_y := 0
aspect_rad := 0
aspect_deg := 0

for each k in K
    out(k):=0

    with (a,b,c,d,e,f,g,h,i) in N(k)
    delta_x := ((c+2f+i) - (a+2d+g) /  8)
    delta_y := ((g+2*h+i) - (a+2b+c) / 8)
    aspect_rad := aspect_rad := atan2(delta_y, -delta_x)

    if delta_x ≠ 0 then
        aspect_rad := atan2(delta_y, -delta_x)

        if aspect_rad < 0 then
            aspect_rad := 2*Π + aspect_rad
        end

    end

    else
        if delta_y > 0 then
            aspect_rad := Π
        end

        else
            if delta_y < 0 then
                aspect_rad := (3/2)*Π
            end
        end
    end

    aspect_deg := aspect_rad * (180/Π)
    out(k) := aspect_deg
end
```

Figure 22: Pseudo-code for the aspect algorithm

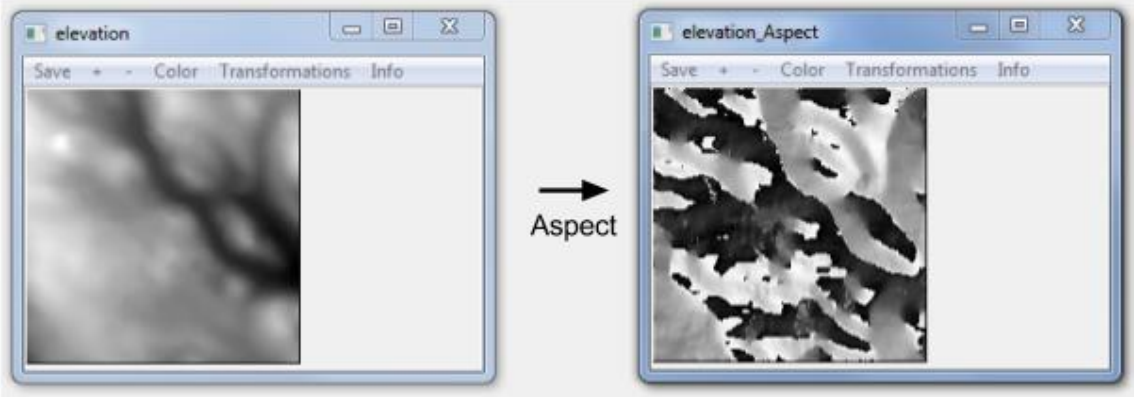


Figure 23: Aspect tool applied to elevation data

4.5.1.3 Hillshade

Based on the light source's location (e.g. the sun), this operation will generate an output layer showing if areas are shaded by other nearby features; where white represent more light and black represent shade. It takes an input layer of elevation data, an azimuth and angle. The azimuth decides in what direction the light source comes from and the altitude is how high above the horizon the source is.

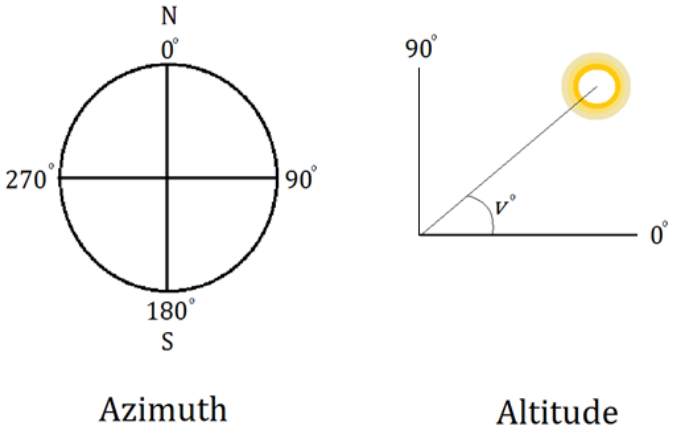


Figure 24: Azimuth and altitude of the Hillshade tool

Figure 25: Hillshade tool applied to elevation data

```
delta_x := 0
delta_y := 0
aspect_rad := 0
rise_run := 0
slope_rad := 0
zenith_deg := 0
zenith_rad := 0
azimuth_deg := 0
azimuth_rad := 0
hillshade := 0

zenith_deg := 90 - in1
zenith_rad := zenith_deg * Π/180

azimuth_deg := 360 - in2 + 90
if azimuth_deg >= 360 then
    azimuth_deg := azimuth_deg - 360
end
azimuth_rad := azimuth_deg * Π/180

for each k in K
    out(k):=0

    with (a,b,c,d,e,f,g,h,i) in N(k)
    delta_x := ((c+2f+i) - (a+2d+g) /  8)
    delta_y := ((g+2*h+i) - (a+2b+c) / 8)
    aspect_rad := aspect_rad := atan2(delta_y, -delta_x)

    if delta_x ≠ 0 then
        aspect_rad := atan2(delta_y, -delta_x)

        if aspect_rad < 0 then
            aspect_rad := 2*Π + aspect_rad
        end

    end

    else
        if delta_y > 0 then
            aspect_rad := Π
        end

        else
            if delta_y < 0 then
                aspect_rad := (3/2)*Π
            end
        end
    end

    rise_run := sqrt(delta_x² + delta_y²)
    slope_rad := atan(rise_run)

    hillshade := 255 * ( (cos(zenith_rad)*cos(slope_rad)) +
    (sin(zenith_rad)*sin(slope_rad)*cos(azimuth_rad-aspect_rad)) )

    out(k) := hillshade

end
```

Figure 26: Pseudo-code fort he hillshade algorithm

4.5.2 High pass filter

Takes an input layer and highlights the edges between different values in the input raster (e.g. makes the boundary of land and water stand out); simply put, an edge-enhancement filter. It works like a focalSum operation with a square neighborhood filter of 3x3 pixels. This filter in turn contains specific values, seen below, that are multiplied with the corresponding cells in the input layer. See image below for an example.

-0.7	-1.0	-0.7
-1.0	6.8	-1.0
-0.7	-1.0	-0.7

Figure 27: Neighborhood filter values. Source: Esri, ArcMap.

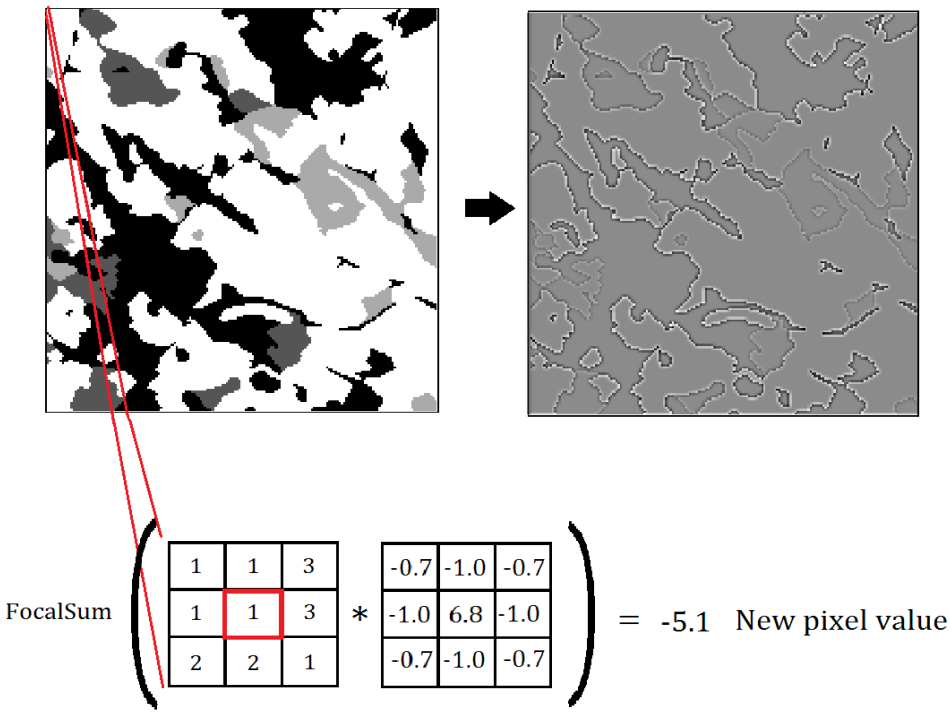


Figure 28: High pass filter procedure

```
High pass filter method
Input parameters: neighborhood shape and radius
    Create new empty raster layer with same parameters as input file

    for every pixel in input layer:
        get neighborhood of current pixel
        get the filter values

        for every neighboring pixel in the neighborhood:
            get the old value
            get the filter value
            set new value as (old value * filter value)
            add to previous new values in the neighborhood
        end

        insert the summarized new value in the new raster
    end

    return new raster
end
```

Figure 29: Pseudo-code of the high pass filter

4.5.3 Classify

This function lets the user classify values in a raster. The tool is accessed from the main window at the top bar under *Raster Operations,* where there are two different classification options; Equal classes or Custom classes.

4.5.3.1 Equal interval classes

This part of the classification tool will classify the data in the raster into an equal sized, user specified, number of classes. For example the values

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

will, if the specified number of classes is 3, result in the following classes:

Class	Values
class 1	0, 1, 2, 3
class 2	4, 5, 6
class 3	7, 8, 9

4.5.3.2 Custom classes

This part of the classification tool lets the user specify which ranges of values that should be classified; the user also need to decide what class number the ranges are classified into. As seen in the figure below the user specifies the number of ranges wanted, then the user click on the first minimum range value, presses *Enter*, enter the min value and then presses *Enter* again. The same procedure is made for maximum range value and the class number (which is entered in the *Value*-field). The ranges are designed so that

the min-value is included in the class range but not the max value; hence if a user wants all values from 1 to 2, including 2, then the max-value of that range must be higher than 2, in the figure below the example is 2.1.

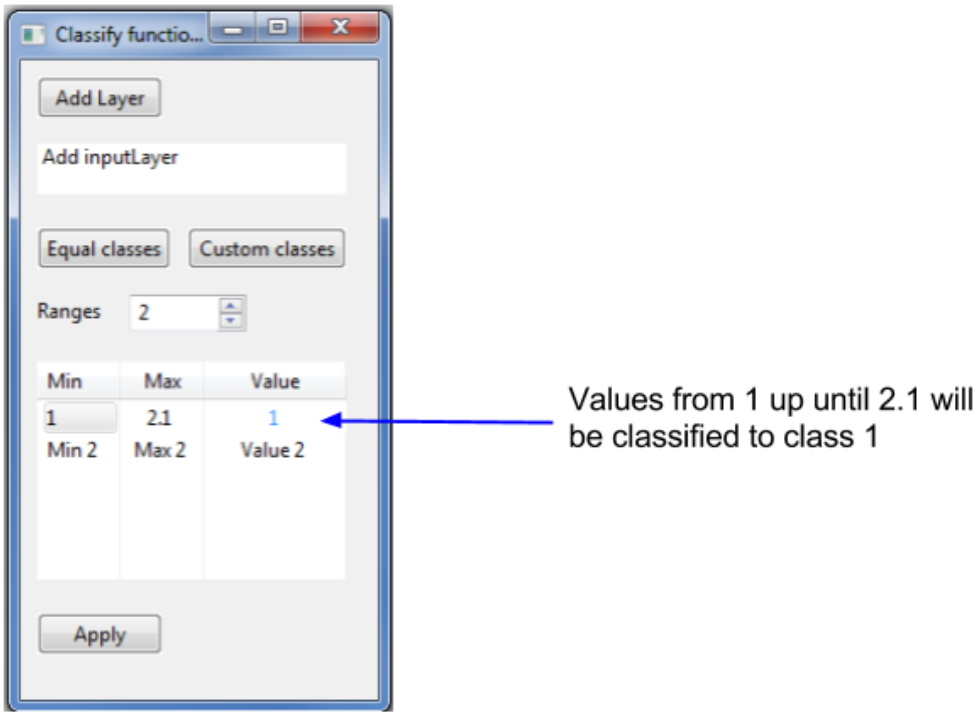


Figure 30: Classify custom window

4.5.4 Reclass

This tool lets the user change all or individual values of a raster. The tool is accessed by clicking *Raster Operations* in the main window. When the tool opens, after choosing an input layer, the existing values of the layer is added to the list “Old Values”. To change a value, simply click it and type in the new value, the chosen value will appear in the list “New Values”. Note that all values need not be changed and that several old values can be reclassified to the same new value.

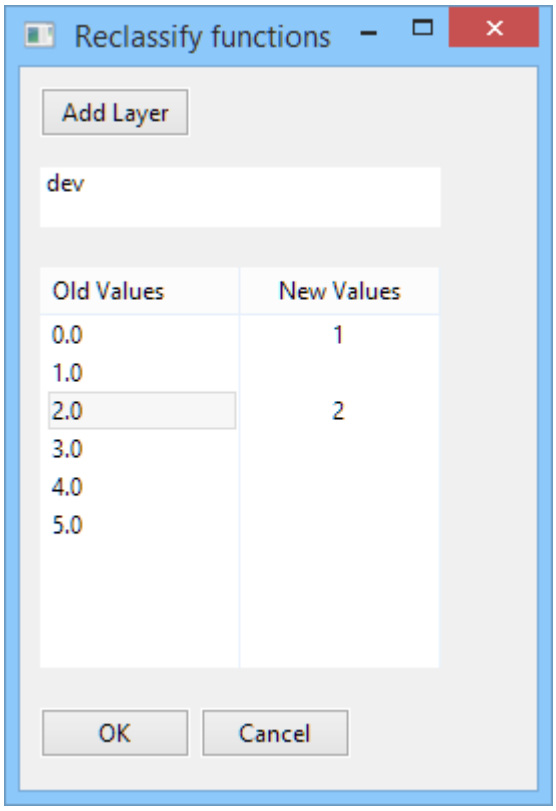


Figure 31: Reclassify window

4.6 Graphical operations (Transformations)

The graphical operations include some tools that lets the user correct or adjust files by means of rotation, mirroring and stretching. These can be accessed at the top bar of each opened raster file by clicking *Transformations*. The following tools are included:

4.6.1 Mirror

Different mirroring options are available for the user to choose from, these are as follows:

- Mirror along the X axis (horizontally)
- Mirror along the Y axis (vertically)
- Mirror along left diagonal (from upper left corner)
- Mirror along right diagonal (from upper right corner)

4.6.2 Rotation

- Clockwise or counter clockwise: The raster will rotate 90 degrees in the chosen direction.
- Custom: The user specifies the amount of degrees the image should be rotated by value, the raster will then rotate *counter clockwise.* Important note, the raster will always be displayed in the matter of a straight box, thus if the user specifies an angle that is not strictly horizontal or vertical this will cause some pixels being left blank, see example image below. These will be assigned a value of NoData taken from the input file. Since the software does not manipulate files including NoData values correctly, the user should take care of using a file that has not been rotated vertically or horizontally.

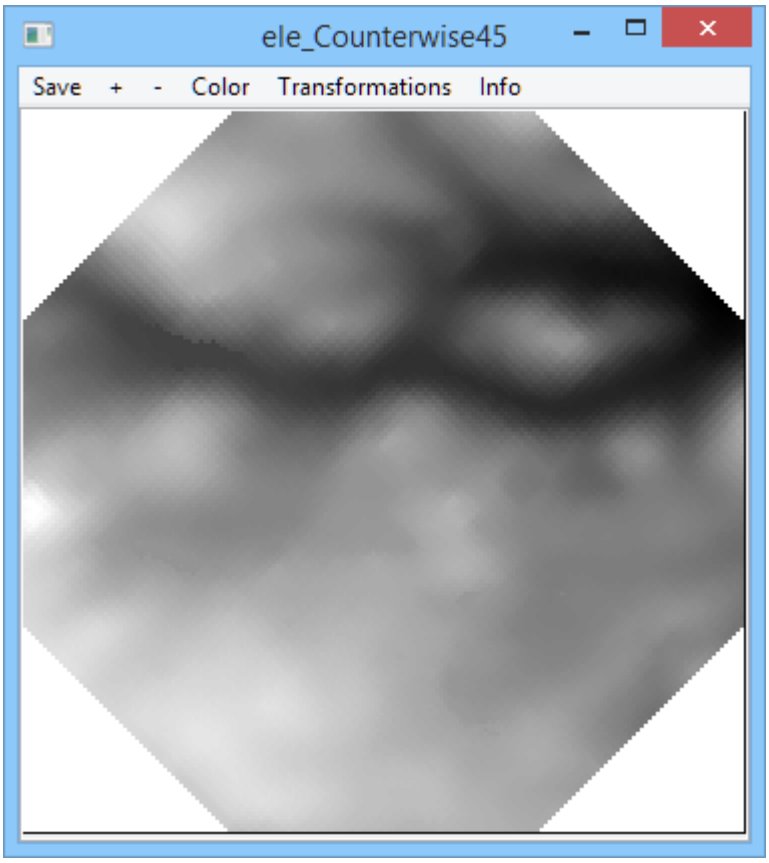


Figure 32: Raster rotate counter clockwise by 45 degrees

4.6.3 Stretch

The user specifies how many times in both the X and the Y direction the raster should be stretched, e.g. if the user specifies 2 in the X direction the map will be doubled in length but kept in height.

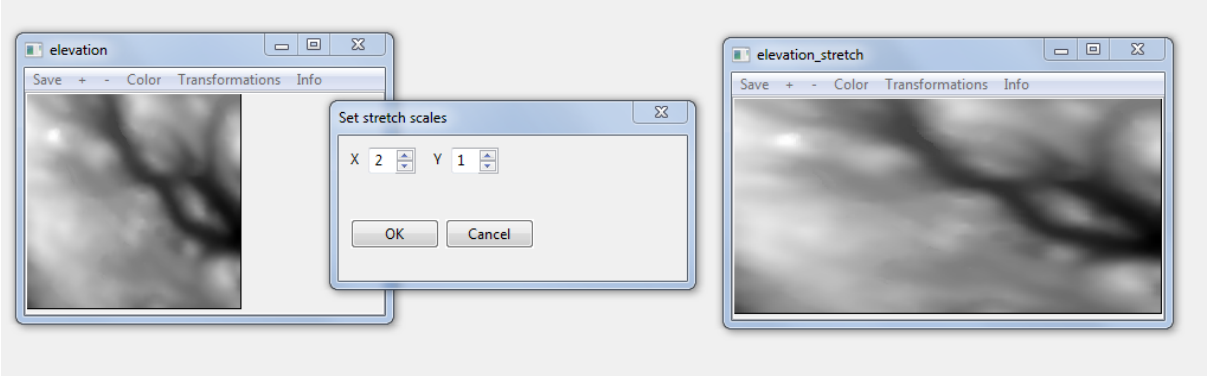


Figure 33: Stretch tool applied to a layer with X-stretch 2 and Y-stretch 1

4.7 Layer Information Window

This tool provides the user with metadata about an opened raster, access it by clicking *Info* at the top bar of the raster window. The information provided are:

- Name of raster file
- Number of column
- Number of rows
- X origin
- Y origin
- Resolution (cell size)
- Maximum value in the raster
- Minimum value in the raster
- Mean values of the raster values
- Variety - the number of different values in the raster
- Histogram, showing the distribution and assigned colors of the values in the raster. The red line indicates the location of the mean.

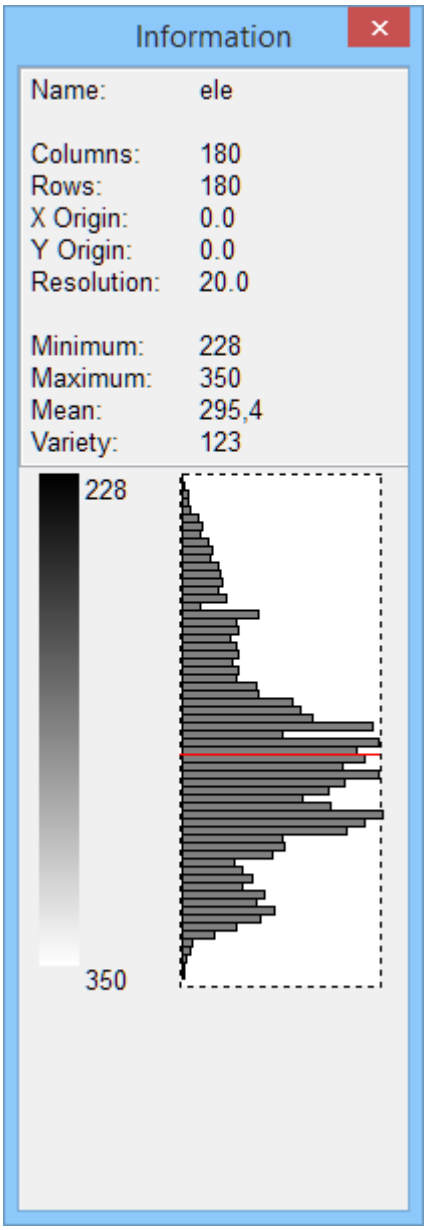


Figure 34: Layer information window

Appendix 1: Overview of the application class diagram