Introduction – To the Max

This assignment option was selected as it appeared to be more realistic than the other five options and as something that could be cloned from in the future.

The focus, as a starting point, was on the input file supplied, snow.slope. This involved interrogating it in terms of format (rows, columns and separators) and the data cell values.

Some of this document demonstrates the understanding of the requirements of the brief and the input data.

Key challenges

- **Key data class**

  Before any programming could commence, the first challenge was to identify the primary data class. Deciding this determined the structure of the program modules, the methods within it and how the programming logic would subsequently flow.

  The principle of the D8 algorithm takes a cell and pairs it with its eight immediate neighbours for calculation purposes.

  This 3 x 3 data / cell grid formed the primary data class.

  Class name: Neighbourhood.

- **Edges**

  To state the obvious, the entire top and bottom rows and the first and last cells of each row are all edge cell. In order to form a full 3 x 3 neighbourhood instance for each cell, five cells adjacent outside of corner edges and 3 cells adjacent out of all other edges had to assume a temporary value. Each outside cell took on the value of the processing / cell of the class instance. This approach was mentioned in various sources including ESRI (2019) and Purves (2016).

- **At this stage of development**

  Construction of the code began here. About 80% of coding was able to be put together to form a working prototype.

  ESRI (2019) illustrates the D8 algorithm in a minuscule form. Here, a 6 x 6 raster data grid is used to demonstrate a resulting grid containing D8 slope direction. To develop and test the prototype, a temporary .txt file was created containing this same data. The purpose was to replicate the accompanying output.

  35 of 36 cells were matched. The mismatched cell was in position 5,4. Here, it was an edge cell and with no downhill neighbours. This led to a further challenge (documented below) that was dealt with towards the end of the main coding process.

  Having successfully developed and tested the prototype on a small-scale basis, subsequent development and testing then reverted to using the snow.slope dataset as input.

  At this stage, each Neighbourhood object had the following information calculated for it:

  - Edge location in terms of compass direction, North, South-east etc,
  - Maximum slope (percentage and degrees,
  - <u>One or more</u> D8 directions (starting from East and working clockwise) that related to the maximum slope calculated.

- **Multiple maximum downhill gradients**

  The challenge here was to reduce the list of downhill directions for each Neighbourhood, where there were more than one, down to one target cell.

  Purves (2016) published university student study notes that referenced the D8 algorithm several times including one for solving flow dispersing in multiple directions. One option was to "Assign 1st direction to cell" (p30). Using this would have kept the program relatively uncomplicated but generating bias towards directional processing, East first, North-east last.

  Another source was encountered. Tarboton (1997) writes about the limitations of the D8 algorithm when it comes to this issue. While most of what was being mentioned would be a time-consuming exercise to implement, diversion into these areas would have been considered out of scope for this project.

  Reading on, an idea then occurred from a phrase "advantage of specifying direction continuously", Tarboton (1997, p310). The interpretation of this, together with an option suggested by Purves (2016), mentioned above, was implemented as code to select the target cell of best fit from the list of identified cells. This took the current Neighbourhood instance and used at least two other neighbouring Neighbourhood instances (one cell away and diagonally opposite) for final decision making.

  For edge cells, as there was not a full set of surrounding Neighbourhoods, bias was coded to accept the first direction encountered.

  This whole process was programmed into the app so that all Neighbourhoods are looped through multiple times until there is one whole pass that occurs without any changes.

- **At this stage of development**

  Each Neighbourhood instance has a single downhill slope direction or none.

- **No downhill gradients (Sinks)**

  The final major challenge here was to fill in the Sinks. The approach adopted here took the neighbour with the lowest elevation within the bounds of the Terrain and set the gradient as zero and the direction towards the neighbour used. It was limited to cells that that were not an edge cell. Edges remained unresolved. This approach is used by ESRI (2019).

- **Remaining challenges (as a collective)**

  - First impressions, the content of the snow.slope dataset looked like ascii files downloaded from Digimap in the DVA2 unit that preceded this unit. Upon a more detailed look, Digimap datasets have georeferencing data included. The remainder of these files were formatted similarly, albeit integer values vs float values. This brought back a recollection of an email conversation for assignment 1 of this unit where it was mentioned the dataset there, in.txt, was not georeferenced as is snow.slope. This became an inspiration to design the application so that any raster dataset could be selected as input to further test the coding and compare it with what was viewed as output in ArcGIS Pro.

    Additional testing was performed against the NN22.asc file. This file, corresponding output files, test cases and testing evidence have been included with the final submission.

  - The remaining challenges encountered from here mostly dealt with programmatical "how to do" issues rather than functional issues, like described above. These included:

    - GUI front-end navigation and input,

    - Matplotlib display maps – getting 4 subplots into one figure,
    - Input file selection from Windows functionality,
    - Catering for input files other than snow.slope,
    - Argument use and defaulting,
    - Argument use gathered from georeferenced datasets,
    - Option to fill or not to fill Sinks,
    - Slope map output option – Percentage, Degrees or Both,
    - Aspect map output option – Yes or No,
    - Geographical map starting references,
    - Hemisphere referencing.

- **Additional Class**

  Towards the end of development, the input dataset became a data object class. This came about to avoid having the same code in two python modules. Having to read the same input dataset in different modules was necessary as the design of the application could be initiated with or with GUI.

Other Key Sources of information

Refer to the Reference (2) section. There, some references are high level links. Much information was obtained from these for various purposes to successfully create a robust application.

Other

If time was unlimited, much more effort could be spent coding to deal with more advanced GIS functionality like what is seen in software like ArcGIS Pro. These include:

- Edge handling,
- Multi directional downhill gradients.

The submitted software is a robust set of program modules that deals with the key functionality of the D8 algorithm. It has been thoroughly tested with various sources of data.

References (1):

ESRI. 2019. *How flow direction Works.* [Online]. Redlands: Environmental Systems Research Institute Inc. [Accessed 20 April 2020]. Available from: https://pro.arcgis.com/

Purves, R. 2016. *GEO372 Vertiefung GIScience Working with terrain models*. [Online]. Zurich: University of Zurich. [Accessed 22 April 2020]. Available from: http://www.geo.uzh.ch/

Tarboton, D. 1997. *A new method for the determination of flow directions and upslope areas in grid digital elevation models*. Water Resources Research. Volume 33, Issue 2, pp.309-319.

References (2):

ESRI. 2020. *How Slope Works.* [Online]. Redlands: Environmental Systems Research Institute Inc. [Accessed 20 April 2020]. Available from: https://pro.arcgis.com/

Geeks for Geeks. 2020. *Python Programming Language.* [Online]. [Accessed April and May 2020]. Available from: https://www.geeksforgeeks.org/

Kite. 2020. *The Python Language.* [Online]. San Francisco. [Accessed April and May 2020]. Available from: https://www.kite.com/

Hunter, J. 2020. *Matplotlib: Visualization with Python.* [Online]. The Matplotlib Development Team. [Accessed April and May 2020]. Available from: https://matplotlib.org/

Tutorials Point. 2020. *Python Tutorial.* [Online]. Kavuri Hills: Incor9 Building. [Accessed April and May 2020]. Available from: https://www.tutorialspoint.com/