HexASCII: a file format for cartographical hexagonal rasters

L. M. de Sousa[†⋆] and J. P. Leitão[†]

[†] Swiss Federal Institute of Aquatic Science and Technology (EAWAG) ,
Urban Water Management Department (SWW) ,
Überlandstrasse 133 CH-8600 Dübendorf, Switzerland
[⋆] `luis.de.sousa@protonmail.ch`

Abstract

Hexagonal segmentations of space have long been known for their advantages *vis à vis* squared partitions in discretising spatial variables, be it natural phenomena or human related features. However, readily available and easy to use tools to manipulate and interact with hexagonal *rasters* remain widely unavailable today. This article presents a first step to enable the use of hexagonal *rasters* in the GIS field. A format to encode cartographical hexagonal meshes as simple ASCII files is specified through a context free grammar. Named HexASCII, this file format provides a simple mean of storing and sharing such *rasters*. A set of simple tools based on the HexASCII format are presented, allowing their creation and basic interaction with traditional GIS software.

Keywords: HexASCII, Hexagons, Hexagonal mesh, ASCII, BNF, *raster*

1 Introduction

Honeycombs are perhaps the most recognisable hexagonal structure in Nature, but are far from being an isolated case. The flora is particularly akin to this pattern, found in the arrangement of plant cells, flower petals, branching of young trees and their distribution in space. While less prominent in the fauna, hexagonal patterns are also found in the distribution of eye cells or fur. Like these, many other structures in the natural world that may appear random are in fact semi-regular hexagonal patterns.

Different authors have explained this predominance of hexagonal patterns in different ways. Perhaps the most significant is the study of Voronoi-Poisson distributions. The majority of Voronoi polygons created by a set of points randomly distributed across a finite space have six sides and six neighbours. Adding an additional constraint of minimum distance between each of the randomly distributed points leads to an even larger number of six-sided Voronoi polygons. Increasing this distance eventually results in a regular hexagonal mesh (Zhu et al. 2001).

Lucarini (2008) approached this subject from the opposite angle, starting with points regularly distributed according to triangular, squared and hexagonal patterns and then adding increasing Gaussian noise. The Voronoi polygons in the squared and triangular patterns break down with the addition of the lightest noise, immediately becoming six-sided figures. In contrast, the Voronoi polygons resulting from the hexagonal point pattern remain six sided figures. Lucarini (2008) also noted that up to a certain level the introduction of noise actually improves the perimeter-to-area ratio of the Voronoi polygons resulting from triangular and squared patterns. This author concludes that hexagons provide the only regular and topologically stable pattern of spatial partitioning.

But in the technical world the approach has been different. Form the earliest days of Computer Science, square grids have dominated discrete data structures in the fields of digital image processing and Geographic Information Systems (GIS). However, the shortcomings of square grids became evident early in the history of Computer Science, motivating suggestions for a shift to hexagonal meshes already in the 1960s. Nevertheless, in spite of further theoretical developments and other advantages identified, this transition never materialised. Likewise in the GIS field, neither hexagonal cartographical *raster* file formats, nor supporting tools were ever fully developed.

Geo-spatial analysts may work with hexagonal tessellations using vector topologies, created with tools such as MMQGIS[1] or with Voronoi tessellations. There are although three important penalties to this method: (i) the topology of each cell must also be stored, increasing the number of associated values from one (the discretised value) to as much as thirteen (adding the coordinates of each cell vertex); (ii) the completeness and regularity of the mesh is not guaranteed; and (iii) since the relationship between an hexagon side and its area is dictated by a rational number, a degree of uncertainty is introduced. In contrast, a *raster* file format provides the distinctive feature of an implicit topology, defined on simple constructs like cell geometry and mesh positioning.

Drawing inspiration from the ASCII *raster* format specified by ESRI [2], this article presents an open format to store hexagonal meshes as text files. Named HexASCII (or HASC for short), it sets the minimum information required to capture mesh geometry and location, plus the cell data themselves. A prototype application is also presented, composed of an Application Programming Interface (API) and a set of command line tools supporting the basic use of HexASCII *rasters* and their integration with traditional GIS software.

These developments intend to be a step towards the adoption of hexagonal *rasters* in GIS. HexASCII is the first stage in a path towards full support to the geo-spatial data work-flow, right from the acquisition phase by field sampling or remote sensing, through the entire spatial analysis process and to the final presentation of results.

This article is structured as follows: Section 2 reviews previous findings regarding hexagonal meshes justifying its adoption in GIS. Section 3 discusses different hexagonal discrete coordinate systems, identifying the minimum information required to describe an hexagonal *raster*. Based on these requirements the HexASCII grammar is presented in Section 4, specifying the encoding of these *rasters* into text files. A set of simple tools supporting the HexASCII format is presented in Section 5. Section 6 summarises the developments proposed herein and discusses possible future developments.

[1]http://michaelminn.com/linux/mmqgis/

[2]http://resources.esri.com/help/9.3/arcgisdesktop/com/gp_toolref/spatial_analyst_tools/esri_ascii_raster_format.htm

2 Advantages of hexagonal *rasters*

This section reviews the most relevant advantages of hexagonal meshes as a *raster* structures in GIS, particularly when compared with traditional squared grids. Rooting in different fields of application, these advantages are nonetheless universal for geo-spatial data.

2.1 Raster data

In fields like Engineering or Earth Sciences there is often the need to model the real world in ways that simplify its complexity. In GIS one of the ways used to describe real world phenomena is through the discretisation of spatial variables into regular meshes of samples. The value at each sampling point is assigned to a pre-defined area around it − a cell − whose dimension and position in the Euclidean space is unequivocal. This kind of data structure is commonly known as *raster* (Tomlin 1990).

There are various ways to divide a plane in small areas of equal shape, but only three forms are possible with regular geometric shapes, i.e. having all sides of equal length and all internal angles of equal amplitude. These shapes are the equilateral triangle, the square and the hexagon. Among these three shapes, hexagons are the most compact; for instance, when storing cylindrical drums of equal diametre within a confined space, an hexagonal pattern accommodates roughly 13% more elements than a squared pattern (i.e. less vacant space is left around each drum).

Rasters in GIS have been predominantly organised in rectangular patterns, in which the space between each row and each column of samples is constant. The use of gridded data this way in GIS is also a consequence of some of the methods employed to acquire spatial data. Particularly in the field of Remote Sensing, the processes for translating meshes of samples sensed in the geographic space into the Euclidean space have been developed exclusively for square grids. This in spite of the fact that these sampled geographic meshes are necessarily irregular, due to the curvature of the Earth's surface and the mechanics of acquisition instruments themselves. The discrete representation with rectangular grids has also the advantage of a direct correspondence between Euclidean coordinates of each cell and its coordinates in the grid itself (column and row numbers). However, this rectangular organisation raises various issues, of which the irregular neighbourhood is possibly the best known.

2.2 Neighbourhoods

The anisotropic neighbourhood of squared grids raised problems early on in Computer Science. In the domain of pattern recognition the concept of connectivity is capital. As exemplified in Figure 1, square grids can easily produce patterns for which connectivity can not be established in a straightforward manner. In this example, a Moore type of neighbourhood (eight cells sharing a vertex or edge) determines that the border is closed, but its interior is still connected to the exterior. With a von Neumann neighbourhood (four cells sharing an edge) the border is open but the interior cell is disconnected from the exterior.

In an hexagonal grid the nearest neighbour logic is straightforward and connectivity unequivocal. This led Golay (1969) to endorse hexagonal meshes for morphological processing, developing various algorithms and proposing schemata for digital systems (i.e. computers) implementing them.

2.3 Frequency domain

Mersereau (1979) explored the advantages of hexagons in the frequency domain, originally intending to devise the minimum regular sampling system required to rebuild a signal within

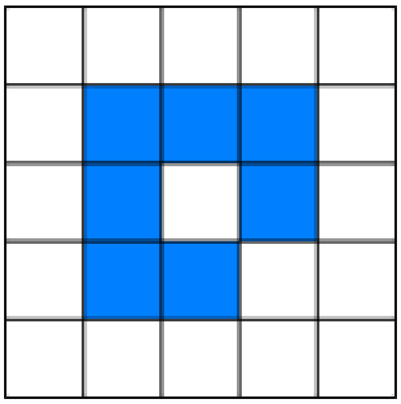


Figure 1: The border paradox.

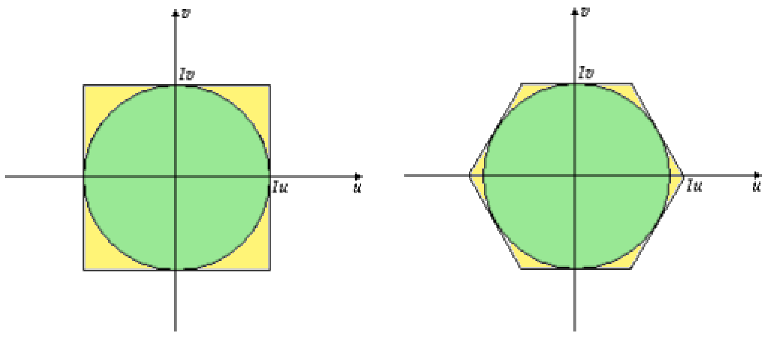


Figure 2: Sampling areas in the Fourier Spectrum for a squared (left) and an hexagonal (right) sampling schemes.

a circular area of the Fourier spectrum. Since average sampling density is proportional to the sampling area in the Fourier spectrum, Mersereau was able to prove that an hexagonal sampling system is able to rebuild the same signal with 13.4% fewer samples than a squared counterpart square. Figure 2 presents this result in a schematic way: the green circular area represents the signal band to sample, the yellow areas the extra frequencies covered by the sampling system. Mersereau (1979) followed on to define the Hexagonal Discrete Fourier Transform (HDFT), the hexagonal counterpart to the Fast Fourier Transform (FFT) employed with squared sampling systems. Doing so, he demonstrated the requirements for 25% fewer computational cycles and 25% less memory storage for the Fourier transform with an hexagonal sampling system.

In a way vindicating Mersereau's findings, Yellot (1981) revealed that light receiving cells in mammals eyes are disposed in hexagonal patterns. Although a property already known in insects, its finding in the complex structure of the spherical chamber of an eye was notable. These findings would later inspire pattern recognition techniques (Thiem and Hartmann 2000).

2.4 Fluid Dynamics

During the 1980s efforts were made to develop two-dimensional discrete microscopic models that could evolve in the continuum to reproduce the laws of fluid dynamics. Initial work was conducted on square grids with four velocity vectors linking each cell to its neighbours; while these models can reach thermodynamic equilibrium, they are anisotropic. Frisch et al. (1986) conceived such a model on an hexagonal grid, with six vectors linking each cell to its neighbourhood. From this discrete system the model progressively approaches the Navier-Stockes equations into the continuum. The six velocity vectors showed necessary and sufficient to simulate fluid flow.

The work on hydrographic modelling by de Sousa et al. (2006) indicates that hexagonal meshes are superior to square grids storing flow directions, preserving more accurately river and watershed geometries. Recent work on flood modelling at the urban scale point to considerably more accurate flood hazard results with hexagonal *rasters* (de Sousa et al. 2016).

2.5 Image processing and morphological accuracy

Work on hexagonal sampling systems for artificial vision date back to the 1980s, with demonstrations of their application to automatic error detection in industrial parts production (Stauton and Storey 1989; Stauton 1989). This subject was again revisited by Snyder et al. (1999), showing how to derive the spatial filters traditionally used in digital image processing for hexagonal meshes.

Brimkov and Barneva (2001) compared squared and hexagonal sampling systems by investigated their efficiency in discretising straight lines. With squared cells one unit long in side, the maximum deviation produced discretising a line is $\sqrt{2}$ $(1.41421\ldots)$; however with an hexagonal scheme this maximum is reduced to $1.11844\ldots$.

2.6 Global Grids

A topic not related to the cartographic domain, but worth mentioning, is the role hexagons have played in the development of global grids, data structures that discretise the entire geographic domain, i.e. the complete surface of the Earth.

Late in the 1980s the Environmental Protection Agency (EPA) of the United States, launched the Environmental Monitoring and Assessment Program (EMAP), an effort to monitor the fauna and flora, scaling up from the national to the global scope. This programme entailed the definition of a uniform sampling network under the following requirements (White et al. 1992; Carr et al. 1997):

- Space partitioned in equal areas.
- Compact cells of similar shape.
- Minimal linear deformation.
- Hierarchical structure for refinement and generalisation.

A sampling system respecting all these requirements is not possible at the global scale; however, it was from them that the concept of Geodesic Discrete Global Grid System (GDGGS) emerged (Sahr and White 1998). White et al. (1992) noted that the spherical projections of the Platonic solids yield the only possible divisions of the spherical surface into equal spherical polygons. Moreover, linear deformations decrease with the higher the number of faces in these solids.

Snyder (1992) presented an equal area projection from the spheroid to the icosahedron, that would be the basis for the Icosahedral Snyder Equal Area Grid (ISEAG). This structure is built by inscribing an hexagon on each of the twenty triangular faces of the icosahedron (*vide* Figure 3). These hexagons are then projected back to the geographic domain, using the inverse of Snyder's projection. The result is a mesh defined on the geographic datum composed

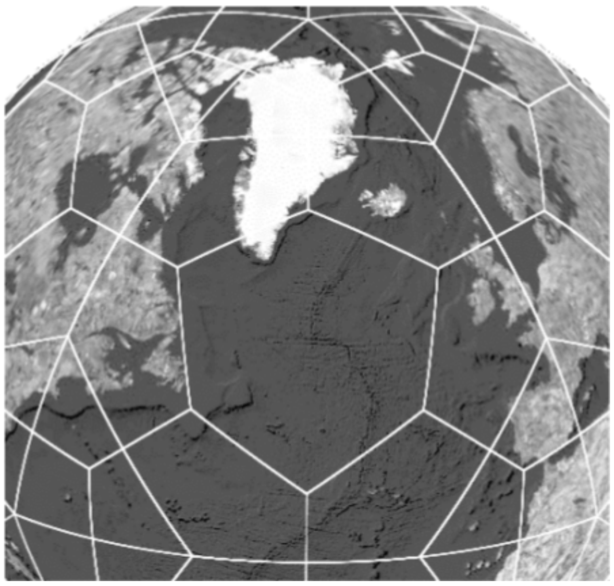


Figure 3: Projected face of the icosahedron and the polygons of the ISEAGs of resolutions 1 and 2 (de Sousa et al. 2007).

by twenty hexagons and twelve pentagons (Sahr and White 1998). Grids of higher resolution are obtained by inscribing more hexagons on each icosahedron face, producing a quasi-regular partition of the globe surface.

The ISEAG would be the basis for the definition of a GDGGS addressing EPA's requirements (Sahr et al. 2003); the European Space Agency (ESA) also uses an ISEAG to publish its Soil Moisture and Ocean Salinity (SMOS) data (Suess et al. 2004). The ISEAG presents a clear path towards a shift from geo-processing on the Euclidean space to the geographic domain, greatly reducing projection deformations.

3 Discrete coordinate systems

An important feature of the *raster* data structure is the indexation of individual cells by their row and column numbers, in what is effectively a discrete coordinate system. While with traditional square grids cell indexation is straightforward, with hexagonal meshes this matter is somewhat more complex. In the scope of the HexASCII file format, cell indexation impacts disk storage space, necessary meta-data and cell sequencing. This section reviews various discrete coordinate systems previously used with hexagons, discussing their advantages and weaknesses and ways to combine them.

3.1 Gridded coordinate systems

A squared grid can be regarded as a matrix of numerical values. To address each of its cells a simple discrete coordinate system suffices. Such a system only allows integer values, expressed in the form (m, n), with the first coordinate identifying the column in which the cell is found and the second identifying the line. The origin cell (usually one of the left corners) is identified with

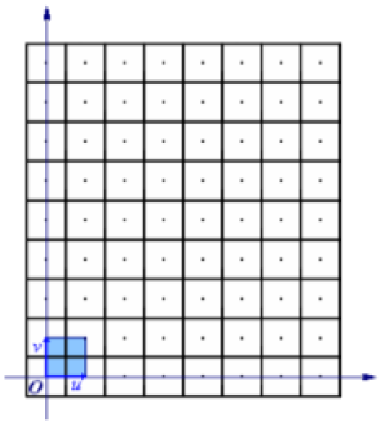


Figure 4: Discrete coordinate system on a square grid.

the coordinates $(0, 0)$, with the cell to its immediate right addressed with the coordinates $(1, 0)$ and so forth. In a grid with M columns by N lines the admissible values are in the intervals $[0, M - 1]$ and $[0, N - 1]$. The versors defined by the axes of this coordinate system define a basis for $\mathbb{R}^2$ (as exemplified by Figure 4). In Computer Science this cell coordinate system is particularly useful, providing for straightforward processing routines, as exemplified in Listing 1.

Listing 1: Example routine processing a numerical matrix

```
float grid[numRows, numColumns];
for (int x = 0; x < numRows; x++)
    for (int y = 0; y < numColumns; y++)
    {
        process(grid[x,y]);
    }
```

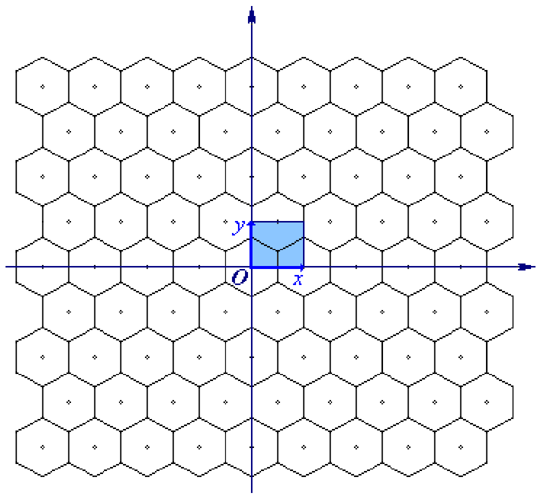
3.2 The natural hexagonal coordinate system

In an hexagonal mesh it is not possible to unequivocally identify all cells using an orthogonal coordinate system like the one described above. However, using two axes on a $\pi/3$ or $2\pi/3$ angle a discrete indexation becomes possible, as shown in Figure 5; the versors of these axes form too a basis for $\mathbb{R}^2$. This system was originally formalised by Snyder et al. (1999) and named *h2*; other authors refer to it simply as natural system (Sahr 2008).

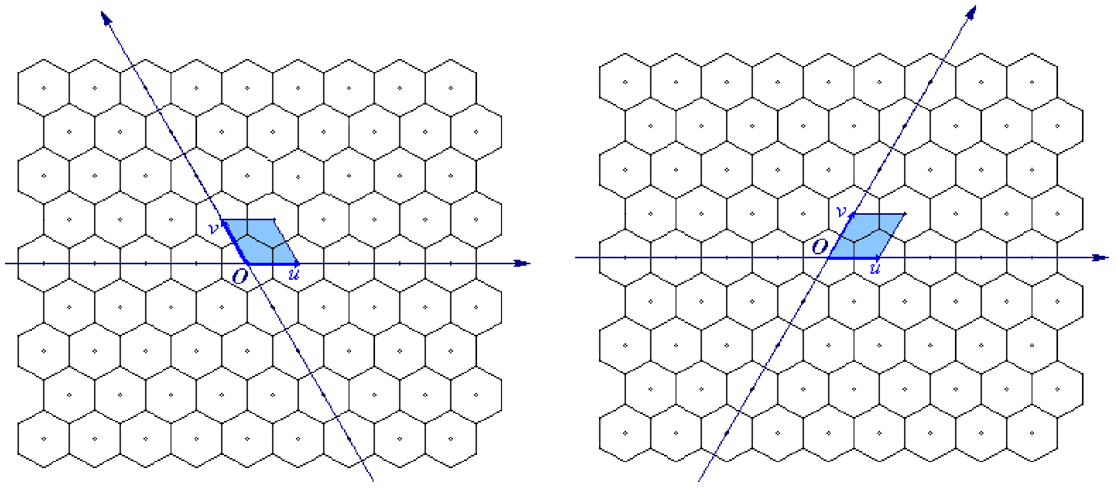
Hexagonal meshes can be oriented in two different ways to have cell sides parallel to the Euclidean system axes. Hexagons may be set with two of its sides are parallel to the Euclidean easting axis, or with two sides parallel to the northing axis. The former of these is called a North-South orientation, while the latter is known as East-West orientation.

In spite of its mathematical soundness, the natural hexagonal coordinate system presents some relevant disadvantages. In order to store a five by five segment of an hexagonal mesh directly employing the traditional matrix structure an expression like the following would be used:

$$\begin{cases} 0 <= x <= 4 \\ 0 <= y <= 4 \end{cases} \tag{1}$$



(a) With orthogonal axes cell indexation is ambigous.



(b) Axes at angles of $\pi/3$ or $2\pi/3$ provide full cell indexation.

Figure 5: Discrete coordinate systems on hexagonal meshes.

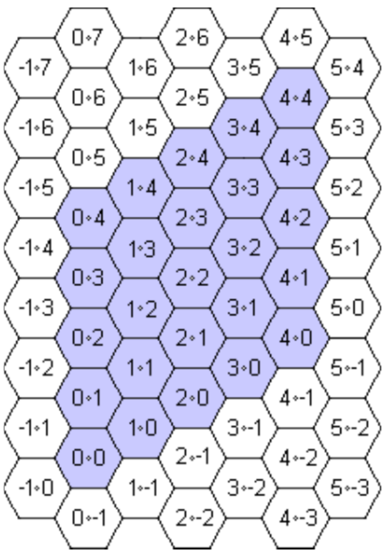


Figure 6: Cell coordinates determined by the natural hexagonal coordinate system.

Such expression actually produces a parallelogram, as Figure 6 shows, not a square or a rectangle, as it would produce in a squared grid. Since an hexagonal *raster* is a digital realisation of a chart, defined relative to the cartographic axes, this parallelogram shape may come across as awkward to users.

One approach to this problem is to store the entire parallelogram and only present the largest rectangle contained within. This, however, raises another problem: waste of storage space. As an example, storing an hexagonal mesh encompassing a rectangular area of interest ten-by-ten cells wide requires forty additional cells, as depicted in Figure 7. For a rectangular area of 100-by-100 hexagonal cells, the number of extra cells required increases to 2 450. This number of additional cells in the minimum bounding parallelogram for a rectangular area in a North-South oriented hexagonal mesh is given by the following expression:

$$\sum_{m=0}^{M_{cols}} 2[(m-1)\backslash 2] \tag{2}$$

Where m is the number of columns and \ symbolises the integer division operation. This expression makes evident the waste in memory and hard drive space imposed.

In essence the parallelogram issue is a consequence of the misalignment between the *h2* system axes and cartographic axes. Alternative methods of storage with this system may be possible, but require additional meta-data and computation.

3.3 The Generalised Balanced Ternary

The lack of a direct hierarchical aggregation mechanism in hexagonal meshes is often pointed as a disadvantage. An hexagon can not be decomposed into smaller hexagons, and reciprocally, can neither various hexagons aggregate into a larger hexagon. This prevents the direct application of hierarchical spatial address structures, such as the *quadtree* in squared grids.

It was in search of such a hierarchical structure that Gibson and Lucas (1982) proposed the Generalised Balanced Ternary (GBT). This system starts by applying a balanced notational cell

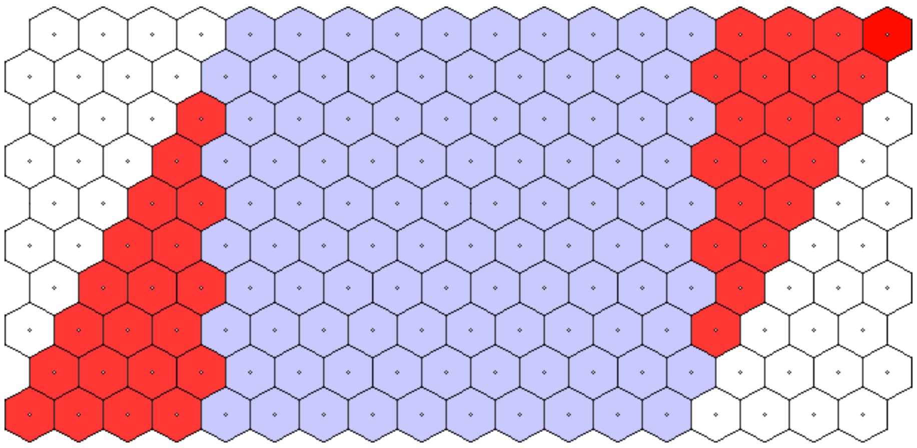


Figure 7: Additional cells (red) in a minimum bounding parallelogram enclosing a rectangular area of interest (blue).

address system, a septenary symmetrical at the origin. In Figure 8 (a) a single digit balanced septenary is presented; cell 6 is placed symmetrically to cell 1, cell 5 to cell 2 and cell 4 to 3.

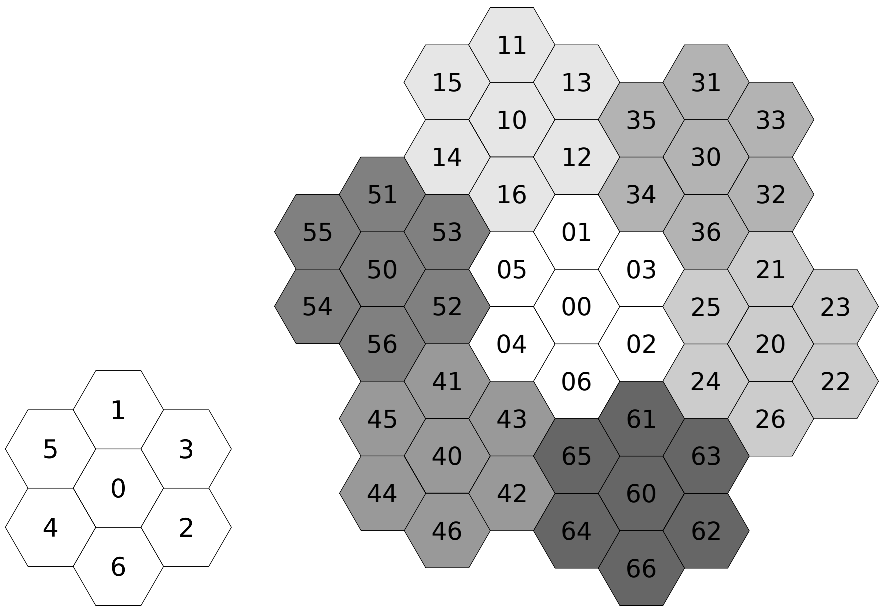
The GBT hierarchy is created by aggregating this set of seven hexagons into a larger element referred as permutahedron. This seven hexagon aggregation can itself be used to partition the Euclidean space into a mesh, while conserving the anisotropic neighbourhood property. To each of these permutahedrons a new notation digit is assigned, again observing the balanced system. Figure 8 (b) presents the second GBT level formed by seven of these permutahedrons. The following GBT level is formed in the same way, by assembling seven elements like those in Figure 8 (b) and assigning a third balanced notation digit.

The GBT hierarchy forms an address tree that can be employed in similar fashion to a *quadtree*. Like the *h2*, the GBT system should not be regarded as a mere alternate addressing system for the Euclidean system, but rather as a complete replacement.

Gibson and Lucas (1982) went on to develop a GBT algebra, that allows basic vector computations. This requires a seven by seven lookup table defining the addition results of each balanced digit with one another. With this table it is possible to compute the sum of two GBT addresses with multiple digits. Importantly, the difference between two GBT addresses is the GBT address of the vector between them.

The GBT has been praised for its elegance, fostering and inspiring further research into the topic of hexagon based hierarchies. However, its use in GIS never expanded, remaining restricted to disciplines where hierarchy is relevant, such as image compression. The increased complexity in basic operations is a likely culprit. The need for a lookup table in the GBT algebra is demanding both in programming and computational terms. What is more, common map algebra operations become rather challenging in a GBT. Figure 9 presents a two level GBT tree and the locations of two different neighbourhoods, those for cell 34 and cell 43. The lack of a common address pattern is apparent; iterating through a neighbourhood in a GBT requires computations of addresses and searches in the tree. Finally, it is important to note that as with the *h2* system, the GBT does not lend itself easily to the representation of rectangular areas.

Another potential issue is the conversion between GBT addresses to or from cartographic coordinates. van Roessel (1988) explored the subject, presenting algorithms for both operations. The transformation from GBT to the Euclidean system is particularly costly, with computation



(a) Level one.

(b) Level two.

Figure 8: Generalised Balanced Ternary cell address system.

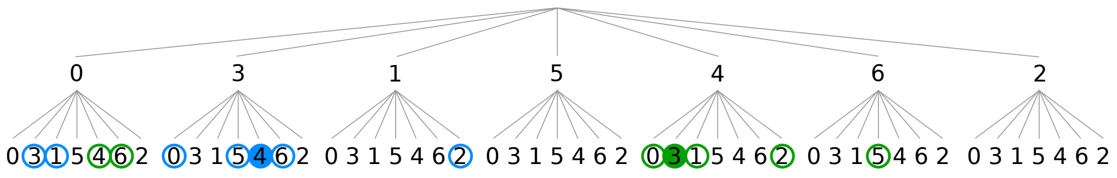
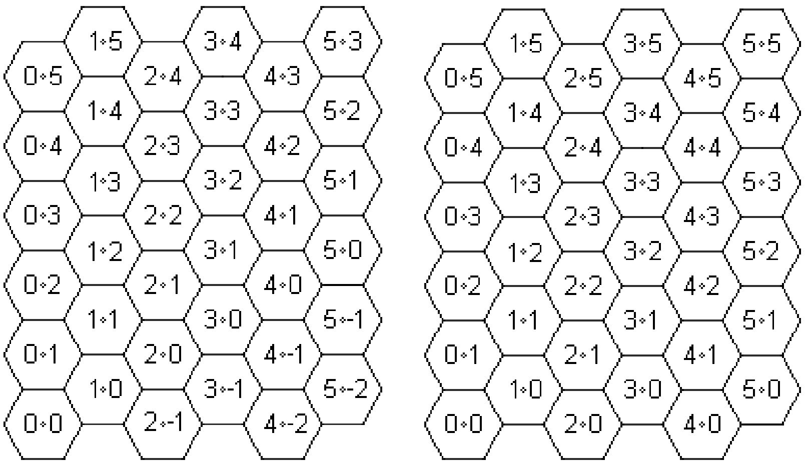


Figure 9: Two level GBT as a tree; circles indicate the neighbourhood for cells 34 (blue) and 43 (green).



(a) Natural or *h2* coordinates. (b) "Squared-up" coordinates.

Figure 10: Individual cell coordinates according to the natural and alternative discrete coordinate systems on hexagonal meshes.

load increasing to the square of the number of digits. While this may not be an issue for applications where geo-location can be entirely governed by the GBT, it becomes relevant when combining hexagonal *rasters* with other data types.

A revival of this addressing system took form after the turn of the century, with the need for hierarchical addressing systems re-emerging in GDGGS. Sahr (2008) introduced the Icosahedral Modified Generalized Balanced Ternary (MGBT), in which a geo-location is coded in a top down fashion. It starts with a digit for the coarser hexagons and pentagons of the ISEAG and is refined further with digits for the permutahedron in the following resolution intersected by the cell in the previous resolution. Vince and Zheng (2009) presents a similar approach, starting from the same ISEAG, while Tong et al. (2013) proposed the Hexagonal Quad Balanced Structure (HQBT) that starts with two different geometric partitions of the hexagon, complementing each other after a second sub-division. All these systems have in common the balanced septenary notation at the lowest level, then building up different hierarchies to the spherical surface.

3.4 Alternative hexagonal coordinate system

In order to address the shortcomings of the systems presented above, an alternative hexagonal coordinate system has been used in informal fashion in various software tools; a good example is the MASON simulation tool-kit (Luke et al. 2005). This system consists in "squaring up" the cell address mesh into a form where it can be trivially stored and handled as a matrix. Figure 10 exemplifies this system.

There are however some disadvantages to this alternative coordinate system. In first place, it does not provide an axes system capable of defining a basis for $\mathbb{R}^2$. More importantly, as with the GBT, this system does not allow for singular and unequivocal neighbourhood definition. In a North-South oriented mesh, the coordinates of the neighbours of a cell in an odd column are different from those for a cell in an even column. Being m and n the coordinates of a cell within an odd column of this system, its neighbourhood is given by:

$$N_O(p) = \{(m, n+1); (m+1, n+1); (m+1, n); (m, n-1); (m-1, n); (m-1, n+1)\} \quad (3)$$

While for a cell in an even column the neighbourhood is:

$$N_E(p) = \{(m, n+1); (m+1, n); (m+1, n-1); (m, n-1); (m-1, n-1); (m-1, n)\} \quad (4)$$

This contrasts with the unequivocal neighbourhood defined by the natural coordinate system:

$$N_H(p) = \{(m, n+1); (m+1, n); (m+1, n-1); (m, n-1); (m-1, n); (m-1, n+1)\} \quad (5)$$

The neighbourhood duality of this alternative system again raises issues for the realisation of map algebra operations on hexagonal meshes. This is obvious for those operations relying on direct cell neighbourhoods, but it also requires an intricate definition for other operations, such as those based on distances. From a strict computational perspective, there is an evident extra step required to identify the column in which a cell lays before computing its neighbourhood.

3.5 Combining hexagonal coordinate systems

This overview of different discrete hexagonal coordinate systems makes clear the lack of a perfect solution, with each system imposing important shortcomings, be it on practical or computational grounds. A pan-optic approach to the hexagonal *raster* domain is to benefit if various of these systems are combined, taking up their advantages and compensating their weaknesses.

The combination of the natural, or *h2*, system with the alternative "squared-up" system is particularly simple to obtain. Being (m_h, n_h) the coordinates of a cell in the natural system, and (m_a, n_a) its counterparts in the alternative system, the relationship between the two is given by:

$$\begin{cases} m_h = m_a \\ n_h = n_a + (m_a/2) + (m_a \mod 2) \end{cases} \quad (6)$$

The natural system is more appropriate for the definition of map algebra operations, while the alternate system facilitates mesh storage and mapping to digital data structures. These two systems are quite complementary, as demonstrated in the development of a prototype hexagonal map algebra (de Sousa 2006).

Combining the GBT system with these two systems is considerably more challenging, requiring an intricate and computationally heavy algorithm. This is an important disadvantage of the GBT system.

4 The HexASCII Grammar

This section starts by identifying the minimum information required to define and position an hexagonal mesh in the Euclidean space. Following are outlined a number of assumptions and options taken towards a concrete *raster* file format. Finally the HexASCII file format is specified through a context free grammar expressed in the Bakus-Naur Form (BNF) (Knuth 1964).

4.1 Basic hexagonal *raster*

In light of the aspects previously discussed, the minimum information required to define an hexagonal *raster* can be summarised to:

- **Cell side** - determining the dimensions of each cell. Since all cell sides are equal in a regular hexagon a single figure suffices.

- **Mesh dimensions** - the number of cells in the mesh. Since cell centroids are not orthogonally aligned against both axes, this aspect is not straightforward as in square grids. However, it is also possible to define an hexagonal mesh by simply stating a number of lines and a number of columns.
- **Mesh orientation** - defining how cells are oriented relative to the cartographic referential axes. This specification is required since hexagons may have only two sides parallel to only one of the axes.
- **Mesh position** - with the previous characteristics defined, positioning the mesh in the Euclidean space requires the coordinates of a single point. The centroid of a corner cell is most suited for this purpose.
- **Cell indexation** - determines the order by which cell values are indexed. In hexagonal meshes this is a particularly important aspect since their cells are not orthogonally aligned with the cartographic axes. This aspect determines how cells are read from and saved to disk.

4.2 Assumptions and other options

In order to define a concrete file format various options must be taken to address the ambiguities identified in Section 3. In the HexASCII format meshes are assumed to be oriented in a North-South fashion by default. The area defined by the mesh stored in this format comprises a rectangle, defined simply by a number of rows and a number of columns, as in the "squared-up" hexagonal coordinate system. Even columns are those with cells shifted to the South, i.e. the lower-left cell marks the southern end of the mesh.

For alternative cell orientations the HexASCII file format allows the definition of rotation angle that applies to the whole mesh. Specifying a 90° angle results in a East-West mesh. Any other orientations are admissible, allowing for meshes unaligned with both cartographic axes. The HexASCII format also supports the definition of a special character sequence specifying null or empty cells.

Cell values are stored in the top-down, left-right order, traditional in Computer Science. This makes their reading into a matrix and addressing with the "squared-up" coordinate system straightforward. This option does not limit the employment of the other coordinate systems. If such is a requirement of the software using HexASCII files, a different system may be employed internally by applying the conversions referred in Section 3.

4.3 BNF Grammar

The HexASCII grammar starts by defining special characters that separate information bits in the ASCII file. They determine where keywords and values start and end, and provide the separation between different sections in the file. These first rules specify what is a blank space and a line break; a blank space can actually be a continuous string of such characters.

⟨blank⟩ → | **t** | ⟨blank⟩ | \t ⟨blank⟩

⟨line-break⟩ → \n | **r**

Values in the HexASCII file format are expressed with numbers. The following rules specify first a digit: a numeral from 0 to 9; numbers are sequences of digits. Then is defined an integer, a number that may be preceded by a plus or a minus sign.

$\langle$digit$\rangle \rightarrow$ **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

$\langle$number$\rangle \rightarrow \langle$digit$\rangle$ | $\langle$digit$\rangle\langle$number$\rangle$

$\langle$integer$\rangle \rightarrow$ **+**$\langle$number$\rangle$ | **-**$\langle$number$\rangle$ | $\langle$number$\rangle$

A real number can now be specified. A first rule describes a fraction, a dot followed by a number. Then is defined an exponent, a sequence starting with an "e" character (capital or small) and followed by an integer. Finally a real: it can be just an integer, but it can also have a fraction component and an exponent.

$\langle$fraction$\rangle \rightarrow$ **.**$\langle$number$\rangle$

$\langle$exponent$\rangle \rightarrow$ **E**$\langle$integer$\rangle$ | **e**$\langle$integer$\rangle$

$\langle$real$\rangle \rightarrow \langle$integer$\rangle$ | $\langle$integer$\rangle\langle$fraction$\rangle$ | $\langle$integer$\rangle\langle$exponent$\rangle$ | $\langle$integer$\rangle\langle$fraction$\rangle\langle$exponent$\rangle$

The general structure of an HexASCII file is defined: a mandatory header, then a number of optional fields and finally the cell values themselves.

$\langle$hasc-raster$\rangle \rightarrow \langle$mandatory-header$\rangle$ $\langle$optional-fields$\rangle$ $\langle$cell-values$\rangle$

The mandatory header is composed by a number of *(keyword, value)* pairs. These form the minimum information required to: (a) define the mesh dimensions and (b) position the mesh in the Euclidean space. Considering the aspects outlined in Section 4.1 this information can be narrowed down to:

- number of columns;
- number of rows;
- cell dimension - given by its side in the HexASCII format;
- coordinates of a cell - in this case, the lower left cell centroid;

Each of these parameters is defined in a single line of the file with a specific keyword.

$\langle$mandatory-header$\rangle \rightarrow \langle$ncols$\rangle$ $\langle$nrows$\rangle$ $\langle$xll$\rangle$ $\langle$yll$\rangle$ $\langle$side$\rangle$

$\langle$ncols$\rangle \rightarrow$ **ncols** $\langle$blank$\rangle$ $\langle$number$\rangle$ $\langle$line-break$\rangle$

$\langle$nrows$\rangle \rightarrow$ **nrows** $\langle$blank$\rangle$ $\langle$number$\rangle$ $\langle$line-break$\rangle$

$\langle$xll$\rangle \rightarrow$ **xll** $\langle$blank$\rangle$ $\langle$real$\rangle$ $\langle$line-break$\rangle$

$\langle$yll$\rangle \rightarrow$ **yll** $\langle$blank$\rangle$ $\langle$real$\rangle$ $\langle$line-break$\rangle$

$\langle$side$\rangle \rightarrow$ **side** $\langle$blank$\rangle$ $\langle$real$\rangle$ $\langle$line-break$\rangle$

The "no_data" parameter is optional and allows the definition of a particular value to represent cells for which no data is available or known. An angle can also be optionally defined, determining a rotation of the mesh relative to the easting axis. These two parameters can be entirely missing from an HexASCII file. In such cases the rotation angle is assumed to be zero degrees and that no missing values exist in the mesh.

$\langle$optional-fields$\rangle \rightarrow \quad | \; \langle$no-data$\rangle \; | \; \langle$angle$\rangle \; | \; \langle$no-data$\rangle \; \langle$angle$\rangle$

$\langle$no-data$\rangle \rightarrow$ **no_data** $\langle$blank$\rangle$ $\langle$real$\rangle$ $\langle$line-break$\rangle$

$\langle$angle$\rangle \rightarrow$ **angle** $\langle$blank$\rangle$ $\langle$real$\rangle$ $\langle$line-break$\rangle$

The final section of an HexASCII file contains the cell values in a sequence. Values are separated by one or more blank characters or by a line break. As common to other *raster* formats, values are coded from left to right and top to down. This means the first value encoded corresponds to the top left cell, the second to the cell immediately to its right and so forth; the last cell value encoded is that of the lower right corner.

$\langle$cell-value$\rangle \rightarrow \langle$real$\rangle$ $\langle$blank$\rangle$ | $\langle$real$\rangle$ $\langle$line-break$\rangle$

$\langle$cell-values$\rangle \rightarrow \langle$cell-value$\rangle$ | $\langle$cell-value$\rangle$ $\langle$cell-values$\rangle$

Figure 11 presents an example of an hexagonal *raster* and the corresponding HexASCII file. The HexASCII grammar is available at a GitHub repository [3] and may be tested in a web tool such as that provided by Icosaedro [4].

5 Prototype

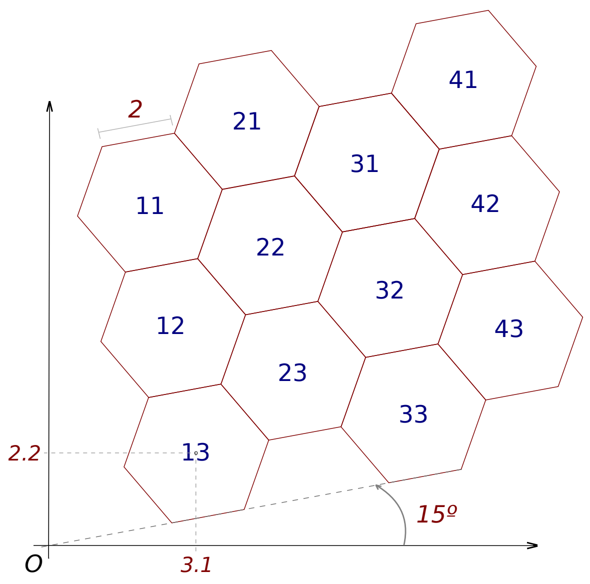
As a demonstration use case, a set of tools was developed providing basic functionality for HexASCII *rasters*. This is a software prototype consisting of a simple API and a series of command line utilities.

These tools were developed in the Python programming language (Sanner et al. 1999) and are gathered in a project named `hex-utils` (de Sousa and Leitão 2017), released under an open source licence and available at GitHub [5].

At the core of these tools an object-oriented domain model concepts describing *rasters* in general and then specific file formats. Both the HexASCII format and the squared ASCII format specified by ESRI are included in this model. The classes in this model provide a wrapper around the two file formats, with specific properties to host the meta-data and the cell values array. They also expose various methods for basic interaction with a *raster* instance, such as reading and writing from disk.

A number of command line tools where developed that perform basic operations with HexASCII *rasters*:

[3] https://github.com/ldesousa/HexAsciiBNF
[4] http://www.icosaedro.it/bnf_chk/bnf_chk-on-line.html
[5] https://github.com/ldesousa/hex-utils



(a) Mesh schema.

```
ncols 4
nrows 3
xll   3.1
yll   2.2
side  2
no_data 9999
angle 15
11 21 31 41
12 22 32 42
13 23 33 43
```

(b) HexASCII encoding.

Figure 11: Example of an hexagonal *raster* encoded with the HexASCII grammar.

- `csv2hasc` - interpolates a new hexagonal *raster* from a set of sparse samples provided as a ternary *(easting, northing, value)*.
- `asc2hasc` - interpolates a new hexagonal *raster* from an input squared *raster*.
- `surface2hasc` - creates a new HexASCII *raster* from a continuous two-dimention surface (provided as a Python function).
- `surface2asc` - same as above, but producing an ESRI ASCII *raster*.
- `hasc2gml` - given an input HexASCII file, creates a vector representation in the Geographic Markup Language (GML) (Burggraf 2006).
- `hasc2geojson` - given an input HexASCII file, creates a vector representation with the GeoJSON syntax (Butler et al. 2008).

These tools provide basic functionality for the HexASCII file format: the creation of new *rasters* and their visualisation with traditional GIS software. The GML or GeoJSON files produced with these tools can be seamlessly loaded into one of such programmes and there be subject to styling with choropleths, feature selection, feature inquiring, geometrical assessment and so forth. Figure 12 portraits a GML file created from an HexASCII *raster* displayed in QGIS (Team et al. 2013).

6 Summary and Future Work

This article presented the HexASCII file format, a text based medium to store and port hexagonal *rasters*. This file format is specified through a formal context free grammar. This grammar is founded on a set of parameters identified as the minimum information required to describe

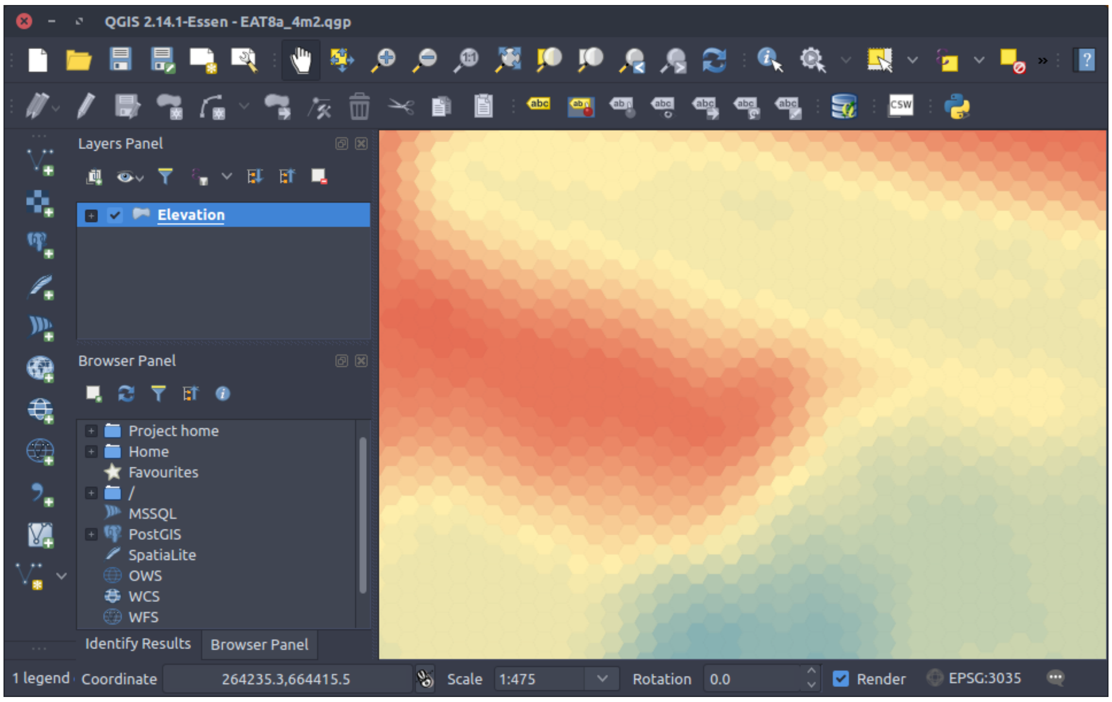


Figure 12: An HASC *raster* displayed through a GML layer in QGIS.

an hexagonal *rasters*: (i) cell geometry, (ii) mesh dimensions, (iii) mesh orientation, (iv) mesh positioning and (v) cell indexation.

A prototype tool-kit named `hex-utils` was developed to provide essential functionality to manipulate HexASCII *rasters*, including their creation and visualisation in GIS software. This prototype can be the basis for the development of further software supporting (or making use of) hexagonal *rasters*.

Despite the known advantages of hexagonal meshes, their wide adoption will not pick up if tools allowing their seamless use are not made available. The HexASCII specification and the accompanying prototype tools are an initial step in this direction, with the ultimate aim of making hexagonal *rasters* as easy to use as their squared counterparts, be it for GIS users, spatial analysts, researchers or software developers.

Further steps towards this end are still necessary, among which data acquisition comes in first place – the translation of common spatial data sources formats into hexagonal *rasters*. Tools such as *LiDAR to HASC* are therefore obvious extensions to the prototype presented here.

Improving user interfaces is an equally important aspect. The tools presented here, as those to follow, would be ideally invoked through a graphical user interface. The plug-in mechanism provided by QGIS (Team et al. 2013), simply and also reliant on Python, poses itself as a strong candidate for this evolution.

Afterwards this set of tools must grow to provide an hexagonal map algebra. A prototype was developed on the C# language that included hexagonal *rasters* (de Sousa 2006). Even if this tool set did not endorse a definitive hexagonal *raster* file format, it provides a code base to be integrated with the HexASCII specification.

The work presented in this article can also be a relevant step in the adoption of global grids of hexagonal nature (de Sousa et al. 2007). The development of a geo-computation tool chest for cartographical hexagonal *rasters* is an ante-chamber to geo-computation in the geographic domain.

References

Brimkov VE, Barneva RP. 2001. Honeycomb vs Square and Cubic Models. Electronic Notes in Theoretical Computer Science. 46.

Burggraf DS. 2006. Geography markup language. Data Science Journal. 5:178–204.

Butler H, Daly M, Doyle A, Gillies S, Schaub T, Schmidt C. 2008. The geojson format specification. URL: `http://wwwgeojsonorg/geojson-spechtml/` (accessed 29-11-2016).

Carr D, Kahn R, Sahr K, Olsen T. 1997. ISEA Discrete Global Grids. Statistical Computing & Statistical Graphics Newsletter. 8(2/3).

de Sousa LM. 2006. Álgebra de Mapas em Grelhas Hexagonais [master's thesis]. Universidade Técnica de Lisboa.

de Sousa LM, Gibson M, Chen A, Savíc D, Leitão JP. 2016. Hexagonal cellular automata for flood modelling. In: Cellular Automata for Urban Systems Modelling - CAMUSS.

de Sousa LM, Leitão JP. 2017. Hex-utils: a tool set supporting HexASCII hexagonal rasters. In: Third International Conference on Geographical Information Systems Theory, Applications and Management.

de Sousa LM, Nery F, Sousa R, Matos J. 2006. Assessing the accuracy of hexagonal versus square tilled grids in preserving dem surface flow directions. In: Proceedings of the 7th International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences.

de Sousa LM, Sousa R, Nery F, Matos J. 2007. Grelhas geodsicas. In: IV Conferncia Nacional de Cartografia e Geodesia.

Frisch U, Hasslacher B, Pomeu Y. 1986. Lattice-Gas Automata for the Navier-Stokes Equation. Physical Review Letters. 56(14):1505–1508.

Gibson L, Lucas D. 1982. Spatial data processing using generalized balanced ternary. In: Proceedings of the IEEE Conference on Pattern Recognition and Image Processing. p. 566–571.

Golay MJE. 1969. Hexagonal parallel pattern transformations. IEEE Transactions on Computers. 18(8):733–740.

Knuth D. 1964. Backus normal form vs. backus naur form. Communications of the ACM. 7(12):735–736.

Lucarini V. 2008. From symmetry breaking to poisson point process in 2d voronoi tessellations: the generic nature of hexagons. Journal of Statistical Physics. 130(6):1047–1062.

Luke S, Cioffi-Revilla C, Panait L, Sullivan K, Balan G. 2005. MASON: A Multi-Agent Simulation Environment. Simulation: Transactions of the society for Modeling and Simulation International. 82(7):517–527.

Mersereau R. 1979. The processing of hexagonally sampled two-dimensional signals. In: Proceedings of the IEEE; Vol. 67. p. 930–949.

Sahr K. 2008. Location coding on icosahedral aperture 3 hexagon discrete global grids. Computers, Environment and Urban Systems. 32(3):174 – 187. Discrete Global Grids; Available from: `http://www.sciencedirect.com/science/article/pii/S0198971507000889`.

Sahr K, White D. 1998. Discrete Global Grid Systems. Computing Science and Statistics. 30.

Sahr K, White D, Kimerling AJ. 2003. Geodesic Discrete Global Grid Systems. Cartography and Geographic Information Science. 30(2):121–134.

Sanner MF, et al. 1999. Python: a programming language for software integration and development. J Mol Graph Model. 17(1):57–61.

Snyder JP. 1992. An equal-area map projection for polyhedral globes. Cartographica. 29(1):10–21.

Snyder WE, Qi H, Sander WA. 1999. A Coordinate System for Hexagonal Pixels. Vol. 3661. p. 716–727. Available from: `http://dx.doi.org/10.1117/12.348629`.

Stauton RC. 1989. Hexagonal Image Sampling: A Practical Proposition. In: Proceedings of the SPIE; Vol. 1008. p. 23–27.

Stauton RC, Storey N. 1989. A comparison between square and hexagonal sampling methods for pipeline image processing,. In: Proceedings of the SPIE; Vol. 1194. p. 142–151.

Suess M, Matos P, Gutierrez A, Zundo M, Martín-Neira M. 2004. Processing of smos level 1c data onto a discrete global grid. In: IGARSS. p. 1914–1917.

Team QD, et al. 2013. Qgis geographic information system. Open Source Geospatial Foundation Project http://qgis osgeo org.

Thiem J, Hartmann G. 2000. Biology-inspired Design of Digital Gabor Filters upon a Hexagonal Sampling Scheme. In: 15th International Conference on Pattern Recognition (ICPR2000); Vol. 3.

Tomlin CD. 1990. Geographic information systems and cartographic modeling. New Jersey: Prentice-Hall.

Tong X, Ben J, Wang Y, Zhang Y, Pei T. 2013. Efficient encoding and spatial operation scheme for aperture 4 hexagonal discrete global grid system. International Journal of Geographical Information Science. 27(5):898–921.

van Roessel JW. 1988. Conversion of cartesian coordinates from and to generalized balanced ternary addresses. Photogrammetric Engineering and Remote Sensing. 54(11):1565–1570.

Vince A, Zheng X. 2009. Arithmetic and fourier transform for the pyxis multi-resolution digital earth model. International Journal of Digital Earth. 2(1):59–79.

White D, Kimerling AJ, Overton WS. 1992. Cartographic and Geometric Components of a Global Sampling Design for Environmental Monitoring. Cartography and Geographic Information Systems. 19(1):5–22.

Yellot JI. 1981. Spectral consequences of photoreceptor sampling in the rhesus retina. Science. (212):382–385.

Zhu H, Thorpe S, Windle A. 2001. The geometrical properties of irregular two-dimensional voronoi tessellations. Philosophical Magazine A. 81(12):2765–2783.