Splitting the sky - HTM & HEALPix.

William O'Mullane
Anthony Banday
Kris Gorski
Peter Kunszt
Alex Zsalay

For "Mining the Sky" Garching July 31 - Aug. 4 2000

Splitting the sky - HTM & HEALPix.

Motivation

HTM - searches in spherical space

- Index the sphere
  - support spherical trigono-metrics.
- Support data binning
  - Hierarchical for storage

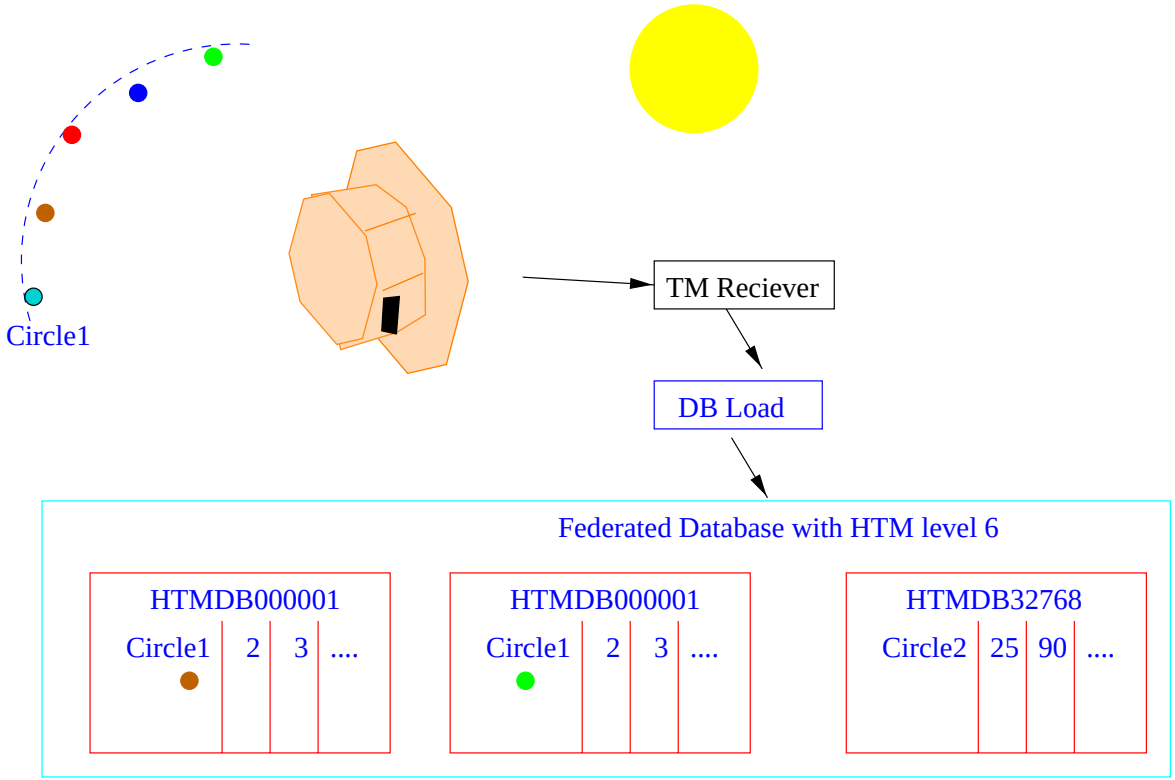
HEALPix - spherical images

- Numerical analysis -
  - convolutions local/global kernel
  - Fourier analysis with spherical harmonics
  - power spectrum estimation
- Topological analysis
  - extreme searches
  - neighbour searches
  - Minkowski functionals
- To aid above should have
  - equal area pixels
  - iso latitude distribution
  - Hierarchical for storage

GAIA

- Point sources
- large data volume (>10TB)
- complex processing

Could use a scheme for data partitioning to facilitate storage and distributed processing.



HTM candidate.

consider also interaction with catalogues

Might we need some HEALPix type qualities later?

Currently use HTM in Simulator

See the poster on GAIA!

Planck

- Cosmic Microwave Background - Signal maps
- Significant data volume
- complex processing

HEALPix in wide use for processing - Science Team adopted standard.

Geometric shape extraction would be good (e.g. as in HTM)

Already added box extraction for map filtering.

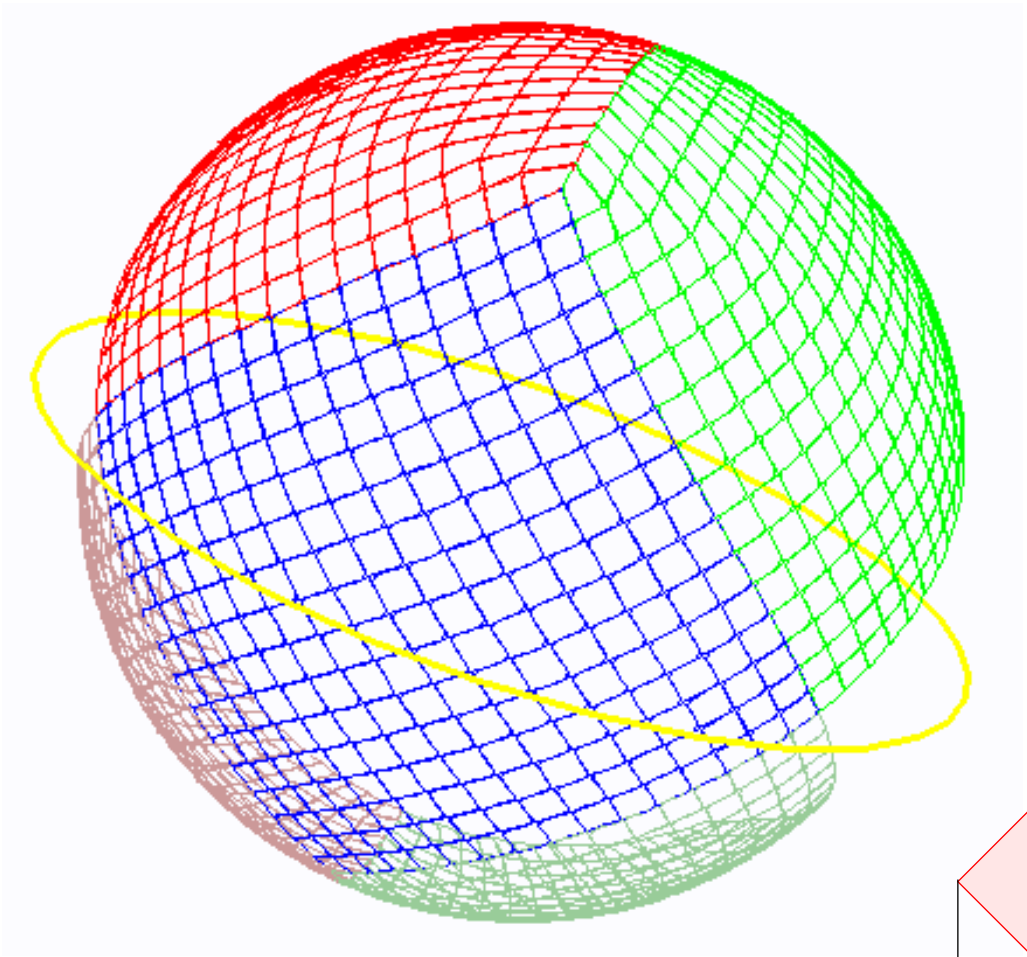
Talk earlier in the week be G. Giardino.

HEALPix

Hierarchical **E**qual **A**rea iso**L**atitude **Pix**elization

- Construction
  - 12 base pixels subdivided Nside times to desired resolution (Nside always power of 2)
  - Octahedron - may be seen as 4 identical zones ($\Pi/2$ in width)
- Hierarchical
  - Pixels sub-divided in 4 parts each time
  - all sub pixels fit inside parent
  - hierarchical numbering scheme works at all resolutions
  - good for neighbour searches, wavelets
- Rings
  - can simply number pixels from ring 1 spiralling downwards
  - not hierarchical
  - good for FFTs.

Base Pixels



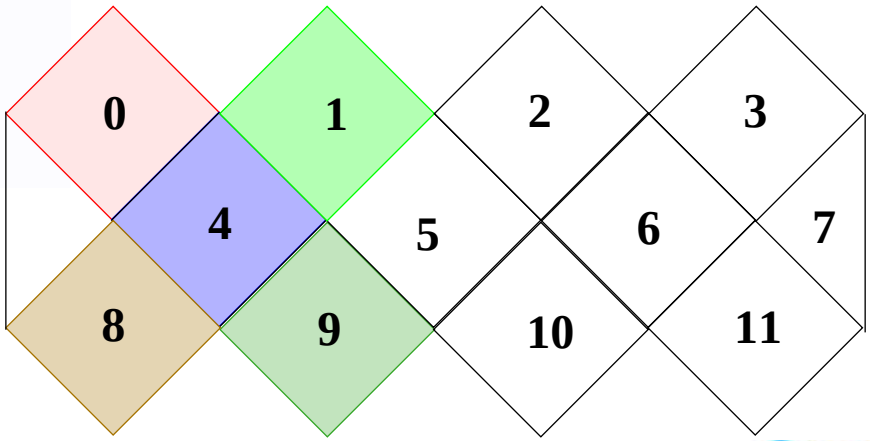
12 base pixels
4 North
4 Equatorial
4 South
Each subdivided Nside times.

Left:

Nside=16
5 base pixels in different colours are shown.

Below:

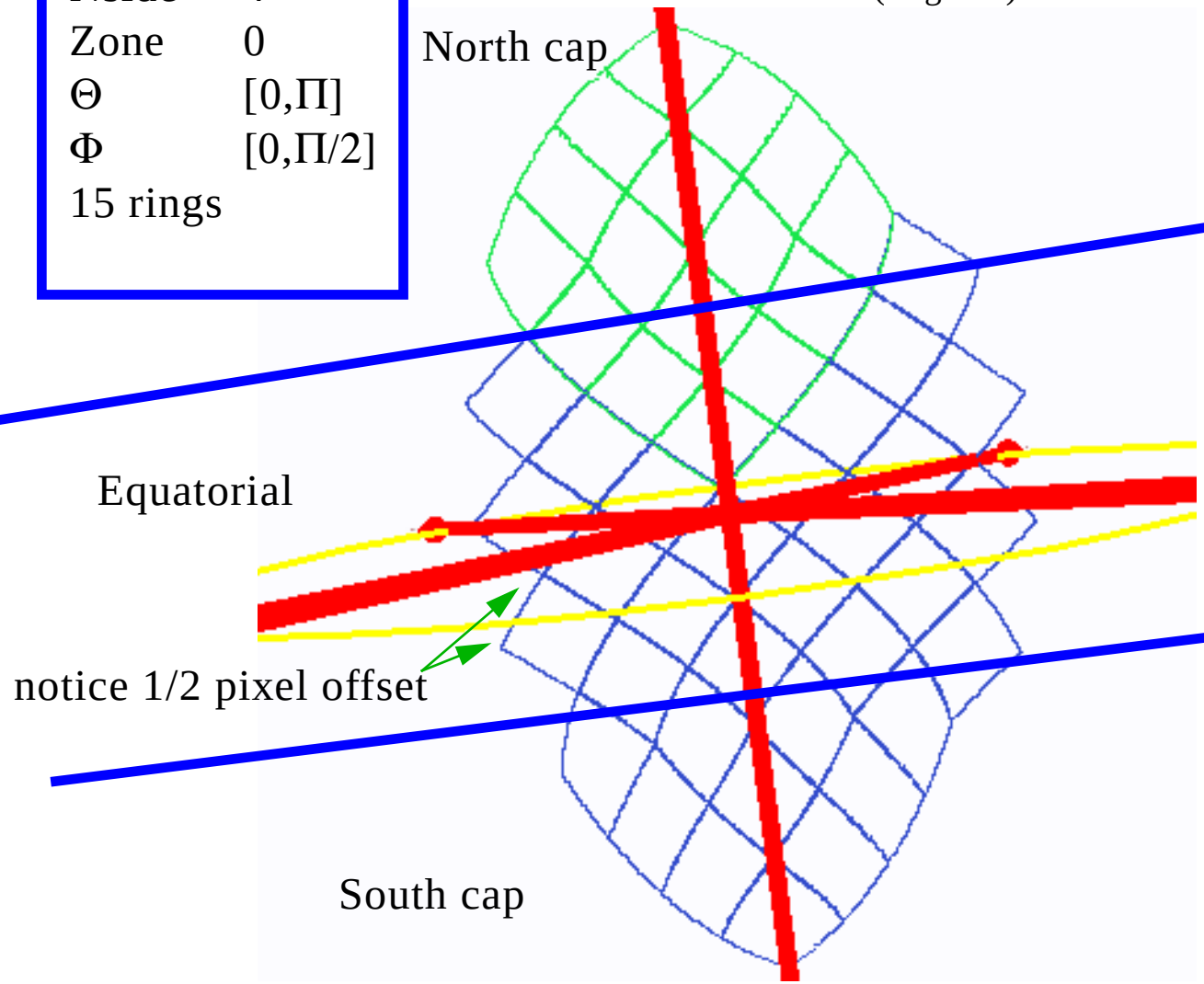
Flattened sphere



HEALPix zone

Two symmetrical quarters of the octahedron - in effect giving 3 regions

Nside	4
Zone	0
Θ	[0,Π]
Φ	[0,Π/2]
15 rings	



Θ divided into (4*Nside -1) rings

- constant latitude
- equal azimuth

4*Nside pixels per ring in equatorial (Nside per zone)
Vary by one pix per ring per zone in polar regions (Nside rings)

Boundaries non-geodesic nominally

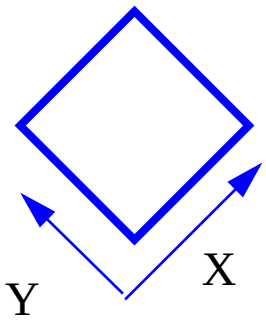
$\cos\Theta = a+b/\Phi^2$ in North cap
$\cos\Theta = a \pm b \cdot \Phi$ in Equatorial
$\cos\Theta = a+b/(\Pi/2-\Phi)^2$ in South cap

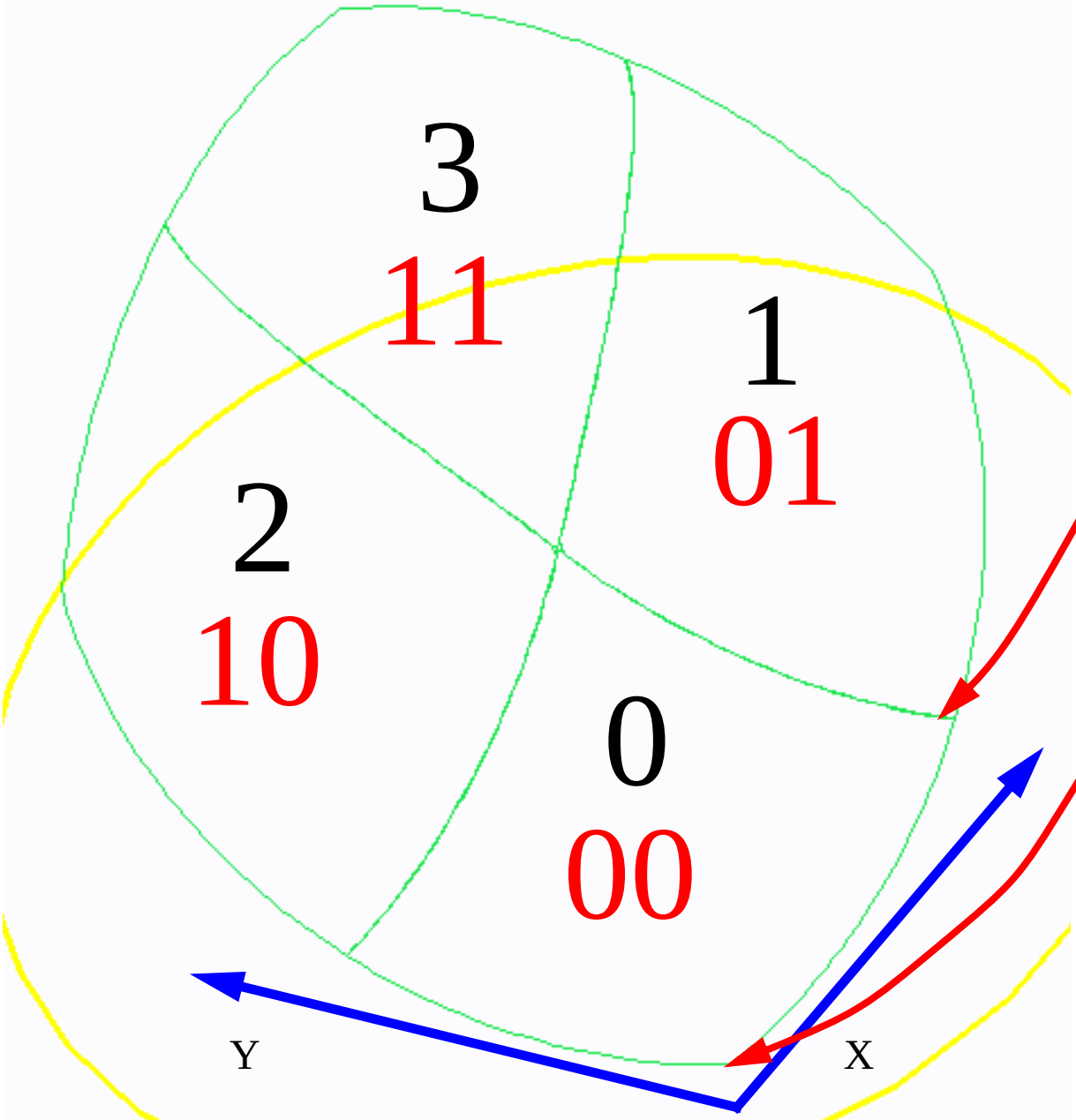
Programatically a little more complex!
But only a little

HEALpix Hierarchy

Within a face it is useful to use x and y offsets for pixels. Bottom pixel is given (0,0)

Particularly good for neigh-bour searches.





Face 0
(there are 12 faces)

NSIDE 2 (resolution)

=> 12* (NSIDE^2)

= 48 pixels of equal area.

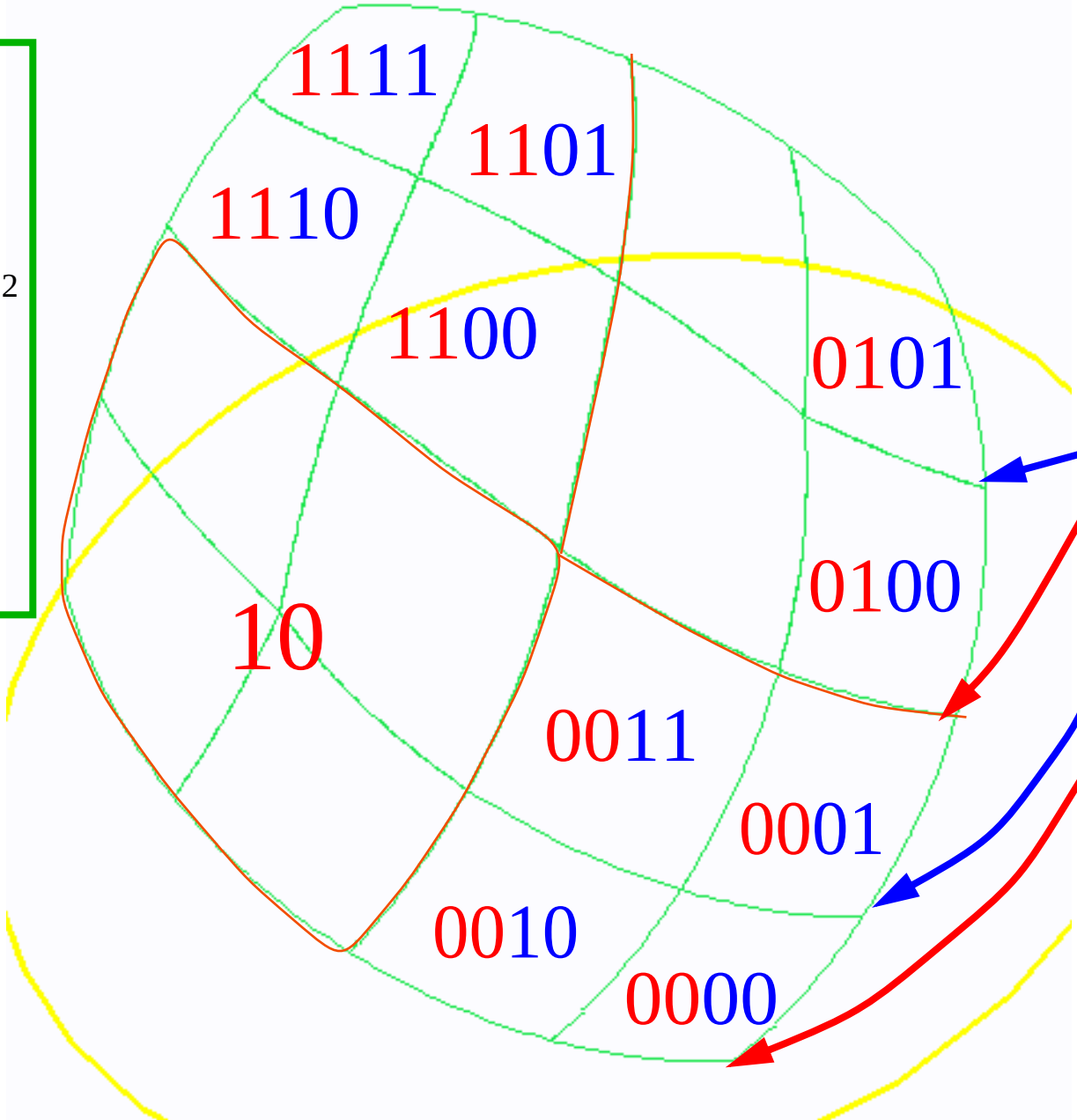
1 face shown =
(NSIDE^2) = 4 pixels

area=859.4 degrees^2

HTM ~

Depth = 2
= (4^depth)*8
= 128 pixels
area = 322.3 deg^2

Depth = 3
= 512 pixels
area = 80.6 deg^2



HEALpix Hierarchy

NSIDE 4
(always power of 2)
=> 12* (NSIDE^2)

= 192 pixels

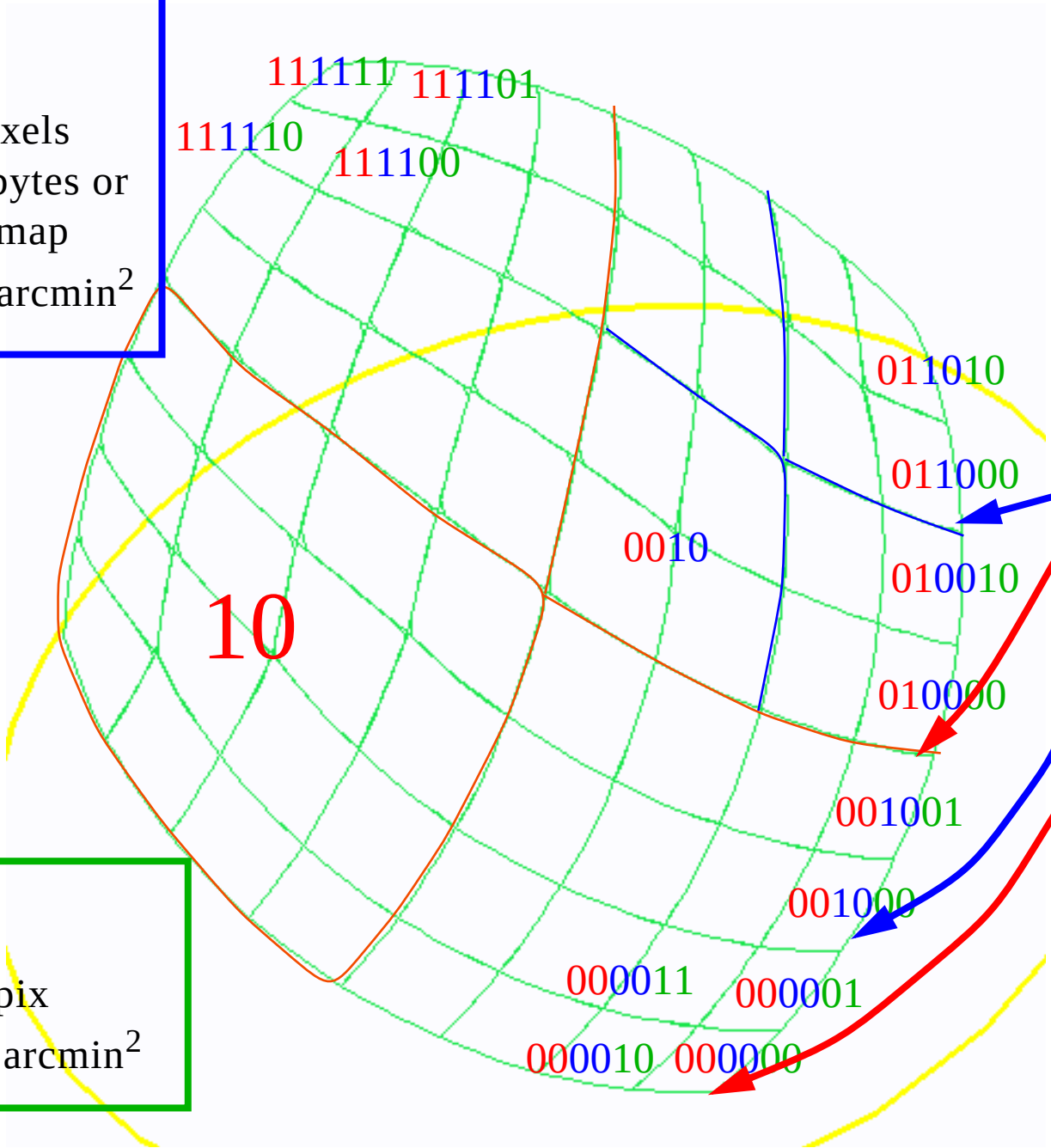
1 face shown =
(NSIDE^2) = 16 pixels

area=214.9 degrees^2

Planck needs

NSIDE=2048
=>50,331,648 pixels
=>402,653,184 bytes or 384 Mbytes per map

Pixel area=2.95 arcmin^2



HEALpix Hierarchy

NSIDE 8 = 768 pixels

1 face shown = 64 pixels

area= 53.7 degrees^2

HTM ~

Depth = 12
=>134,217,728 pix

Pixel area = 1.1 arcmin^2

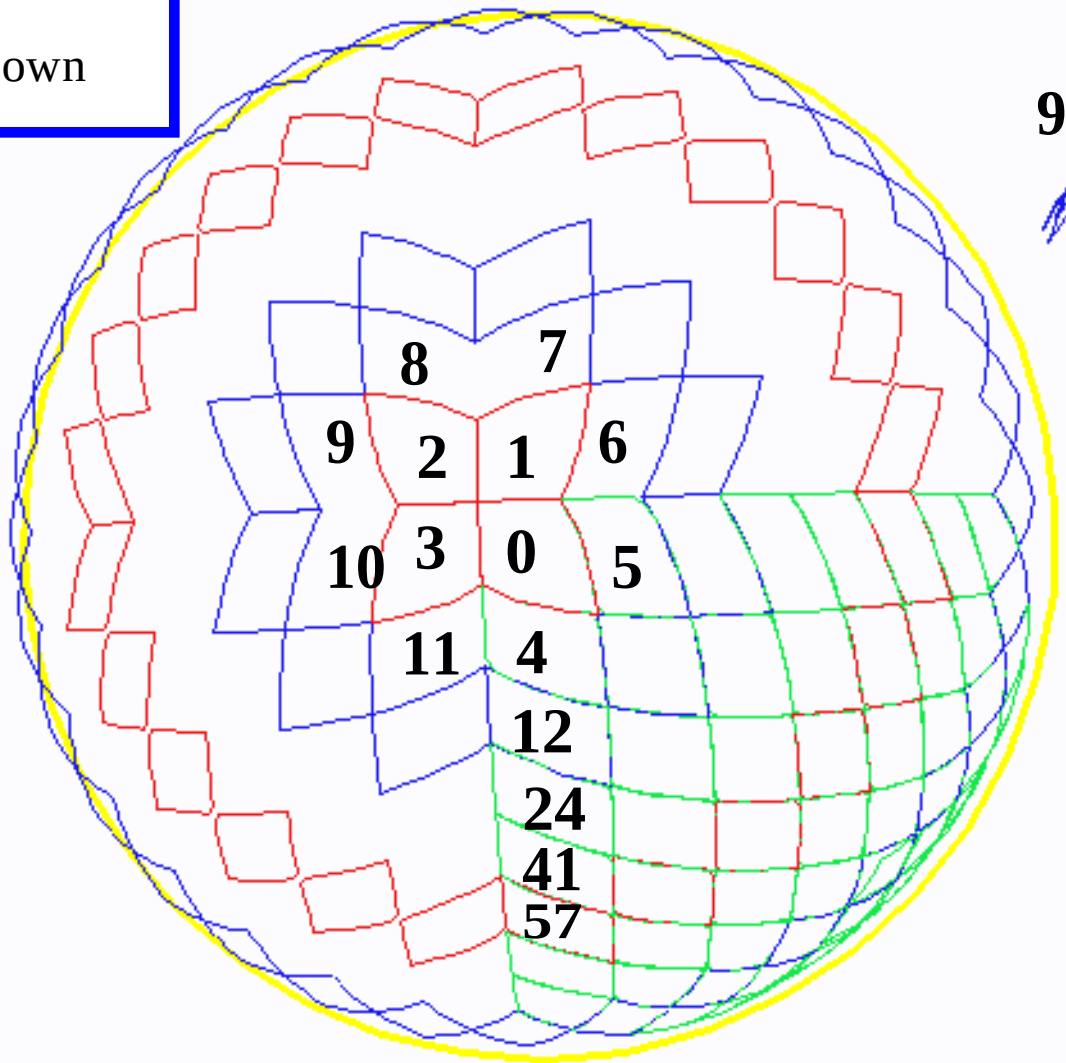
HTM ~

Depth = 4
= (4^depth)*8
= 2048 pixels

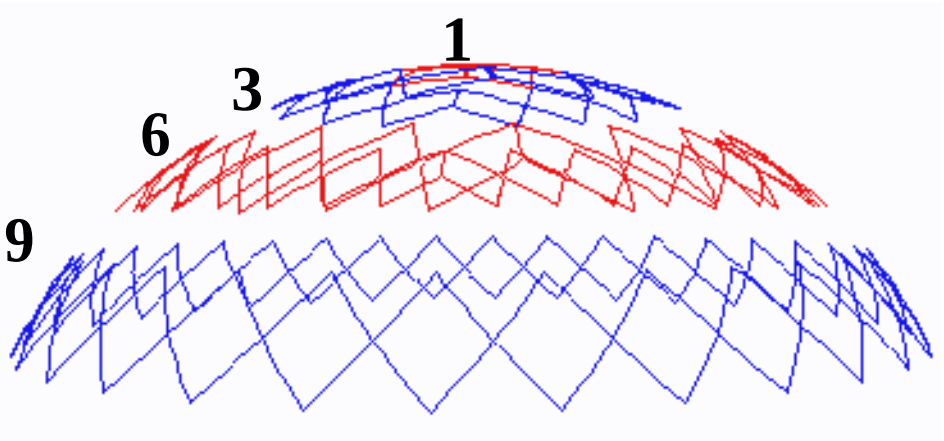
area = 20.1 deg^2

Rings

Nside 8
31 rings
not all shown



View from above giving pixel numbers



View from side giving ring numbers plotted

Numbers change with resolution
Pixels may be described by ring number and offset or i_theta and i_phi.

Special Considerations

Anomalies are inevitable with any tessellation.

HEALPix

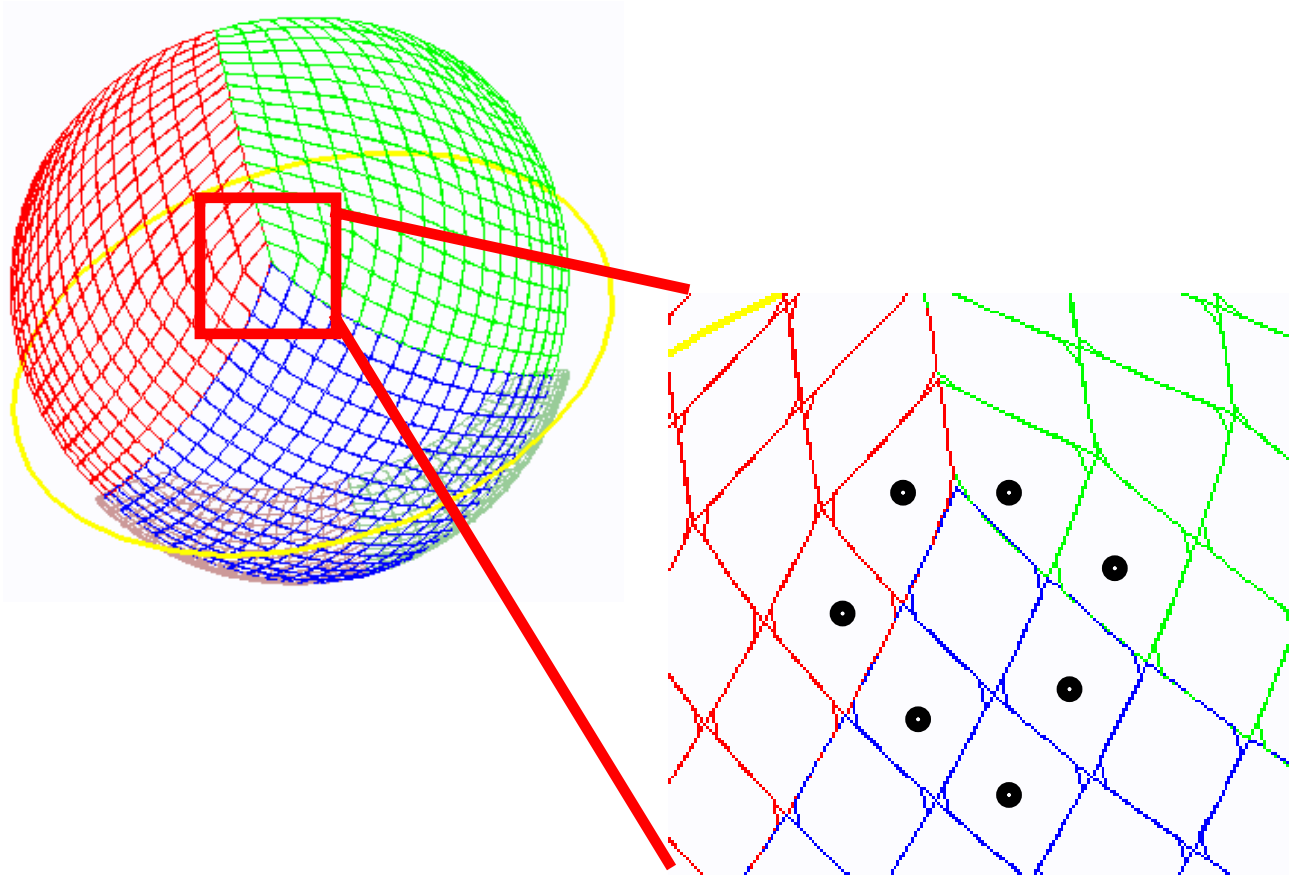
- Pixel shape varies - not a problem with high enough sampling
  - need to take account of position on the sphere (3 rules)
  - number of pixels in rings vary
- Most pixels have 8 neighbours - 8 have 7 neighbours
- Nested X,Y orientation of base pixel neighbours varies

HTM

- Pixel shape varies
- Pixel area varies - makes many numerical calculations difficult.
- no rings
- similar neighbour orientation problem

These are mainly problems for developers not particularly for users
Advanced libraries exist for both schemes although target audiences are different.

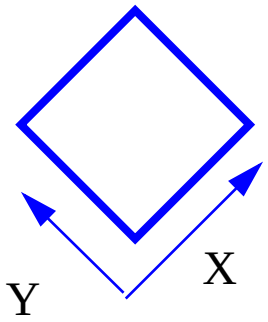
8 pixels have seven neighbours



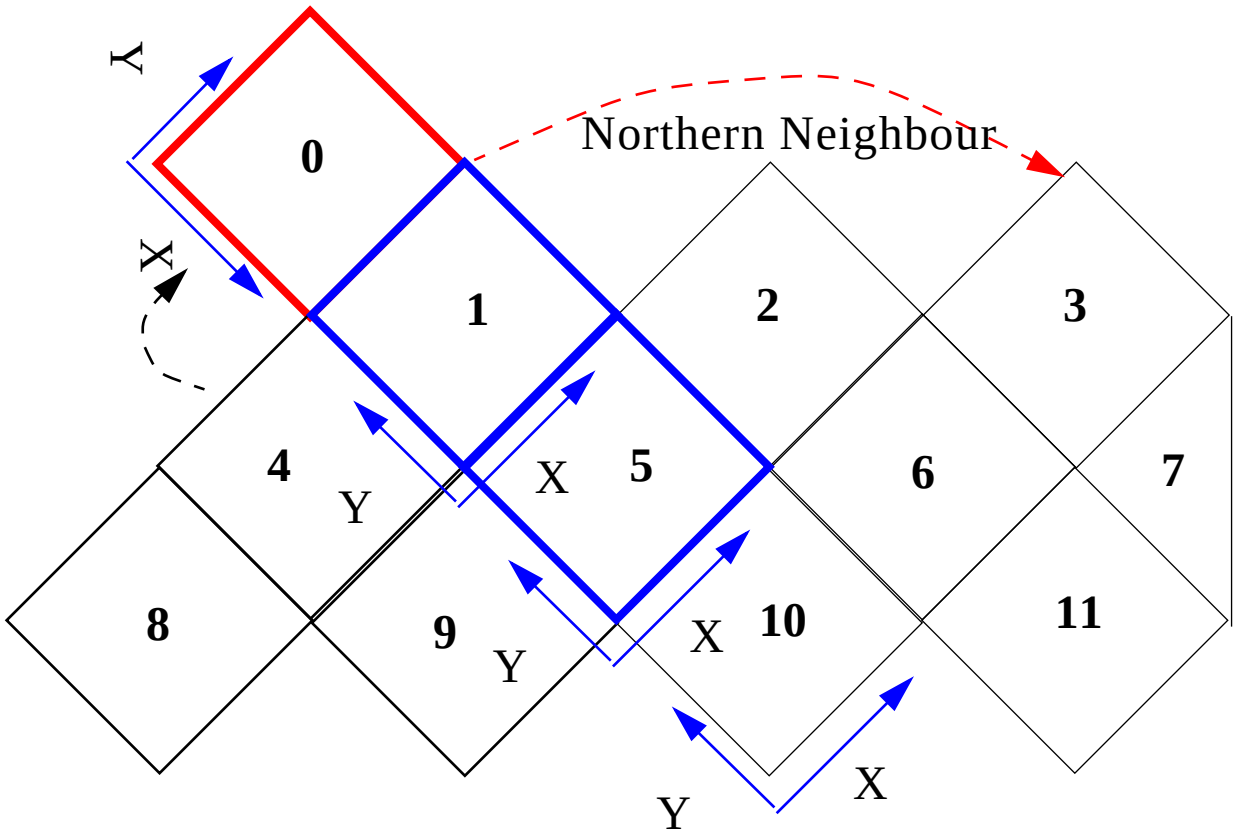
The north (nside-1,nside-1) and south (0,0) pixel of each Equatorial face

X,Y orientation of Base pixels

Within a face it is useful to use x and y offsets for pixels. Bottom pixel is given (0,0)



NW neighbour of southern and equatorial pixels have same X,Y orientation but for the northern pixels the NW neighbour is effectively rotated (depending on your definition of NW neighbour)



Resolution - No theoretical limit but

D	PixArea	NSIDE	PixArea	NPIX	HTM_leaves
10	1.77E1	1,024	1.18E1	12,582,912	8,388,608
11	4.43E0	2,048	2.95E0	50,331,648	33,554,432
12	1.11E0	4,096	7.38E-1	201,326,592	134,217,728
13	2.77E-1	8,192	1.84E-1	805,306,368	536,870,912
14	6.92E-2	16,384	4.61E-2	3,221,225,472	2,147,483,648
15	1.73E-2	32,768	1.15E-2	12,884,901,888	8,589,934,592
16	4.32E-3	65,536	2.88E-3	51,539,607,552	34,359,738,368
17	1.08E-3	131,072	7.20E-4	206,158,430,208	137,438,953,472
18	2.70E-4	262,144	1.80E-4	824,633,720,832	549,755,813,888
19	6.75E-5	524,288	4.50E-5	3,298,534,883,328	2,199,023,255,552
20	1.69E-5	1,048,576	1.13E-5	13,194,139,533,312	8,796,093,022,208
21	4.22E-6	2,097,152	2.81E-6	52,776,558,133,248	35,184,372,088,832
22	1.06E-6	4,194,304	7.03E-7	211,106,232,532,992	140,737,488,355,328
23	2.64E-7	8,388,608	1.76E-7	844,424,930,131,968	562,949,953,421,312
24	6.60E-8	16,777,216	4.40E-8	3,377,699,720,527,872	2,251,799,813,685,248
25	1.65E-8	33,554,432	1.10E-8	13,510,798,882,111,488	9,007,199,254,740,992

32 bit integer limit

64 bit double limit

Performance

Benchmarks are always difficult...
Need a generic interface.

Small example:
Took 20000 randomly generated Pointings and converted to Pixel Numbers and back.
Used SpatialVector as the input - added routine to Healpix to use this.
Used NSIDE=1024 and Depth=12

Index	Vec-> pix	pix -> Vec	comment
HEALpix(1024)	0.92 s	0.68 s	Mathematical operation with nside as parameter
HTM(12)	12.9 s	3.98 s	recursive nature (depth=6 takes just over 7 seconds).

HTM does not provide a pointing for an ID number - does return all vertices of the triangle though (used above).

Of course we can not test the nice domain features since this does not exist in HEALpix.

These tests were both done with java code - obviously much faster in C++ (HTM) but in relative terms this is fair.

Facilities

HEALPix

- For spherical coordinate (Θ,Φ) give pixel number for given NSIDE (and reverse)
- Fast simulation and analysis of full-sky maps of CMB temperature and polarisation anisotropy
- Read/write FITS formatted map data sets
- IDL-based display of results.
- Search the maps for pixel neighbours and extrema of a random field.
- An IDL toolkit for pixel manipulation and FITS file manipulation.
- A Fortran90 MAP2GIF visualisation.

HTM

- For a given point (x,y,z) on the sphere, return the node name it belongs to up to a given level
- For a given node name, return the 3 vertices
- Complicated geometric queries on the sphere, returning the nodes fully/partially contained.
- bitlist representation for large query handling
- Visualization in Java3D (only available in the Java package)

The systems are practically completely complimentary.
Use together does not seem simple - rather should try to build geometric facilities for HEALPix

Wish List

- Generic interface to allow interchange of tessellation schemes... tricky.
- Benchmarks?
- Geometric (Domain) type features in HEALPix... some work.
- Analysis type features in HTM... very tricky.
- Real 3D support (onion skin?) (both)
- Proper (un)balenced tree support (both) - requires a container /index split in code
    - partial tree support - incomplete maps, almost there.
        - this needs to be taken account of in the HEALPix processing software
- Bit List representation and operators for HEALpix - need this.
    - Masking out parts of the sphere
        - again this needs to be taken account of in the HEALPix software

Index Interface

A common set of signatures on an index would be nice for benchmarking and exchange of index in a project.

- HTM works with unit vectors. HEALpix uses spherical coordinates.
  - Provide both? e.g. pix2ang, pix2vec, pix2pointing -
  - add spherical coords to SpatialVector
- Uniform construction
  - HTM uses depth to specify resolution. HEALPix uses NSIDE. NSIDE = 2^{depth} would give some approximation
  - some form of factory?
- Container representation must be abstracted from the index
  - should be able to use index with different DB's and filesystems
  - possible now but only because the tree has uniform depth.
- Standardized geometric queries

The End... some web sites... Questions?

HEALpix website - http://www.tac.dk:80/~healpix/
HTM website - http://www.sdss.jhu.edu/
GAIA - http://astro.estec.esa.nl/gaia
Planck - http://astro.estec.esa.nl/Planck