TECHNICAL NARRATIVE

The emergence of Narayana is a manifestation beginning in the realms of the mind, built from the materials of logic, maths, data and imagination. Before we can begin, we must have in place the essences of mind, the tools and items of data that a mind can manipulate and explore within the creative enterprise of the Universe. As a part of this we undertake an exploration of the essences of colour, pattern and form, that we can manipulate both in our minds and by using a computer. The images so generated can be relayed back to human beings, both as a source of pleasure and interest and as a further stimulation to extend the faculties of the computer environment.

It is not a necessary feature that our designs should represent anything in the material Universe, only that they should explore the fundamentals of pattern and form that might appear as manifestations within it. The digital computer itself has square and list defined structures in a way that only otherwise occurs in human (or perhaps other sentient) minds. The Universe itself is composed of spheres and ellipses, straight line segments generally only arise in limited contexts from arrays of such spheres, as in crystals. Light beams also appear to be straight. This means that the basic patterns derived from computers can be shared with minds, but are already strangers to the rest of the Universe. The computer uses straight line segments to describe curves but the Universe uses curves to build straight lines.

The fundamental elements of pattern and form are dimensions (in this case 2 or 3), points, lines, number and colour.

COLOUR

The computer screen consists of a rectangular grid of square points which can be set to black (off) or a combination of the light colours green, red and blue (primary colours) which the eye translates into the rest of the colours. The point of light or colour is called a pixel. These colours are represented by three numbers one for each of the primary colours. The numbers range, to represent intensity, from 0 to 255 so that there are 256^3 possible colours, far more than the eye can decipher. This range is simply the band of light frequencies to which the human eye is sensitive.

A pattern is then a distribution of these colours over the two dimensional surface of the screen or perhaps a larger area (like an A2 poster) which we can scroll around to view.

Generally, a pattern consists of areas or patches of colour than can be distinguished by the eye, but note that we can wash a range of colours over the display area and allow the eye to resolve areas for itself as in the colour transition example in Figure 1.

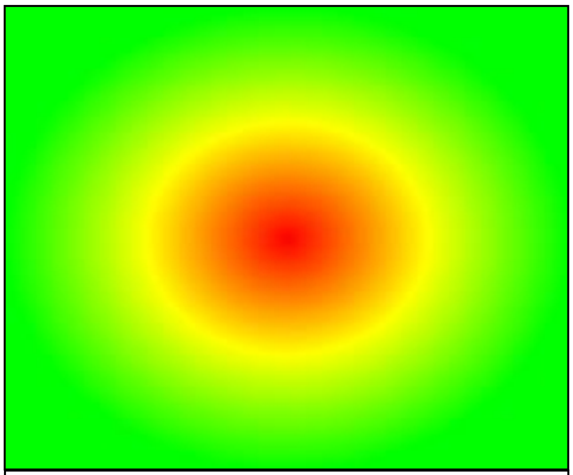
Figure 1: Colour Transition

PATTERN AND SHAPE

We can divide the surface into regular areas or we can divide it into irregular areas, so that the eye can choose shapes and patches of colour to focus on.

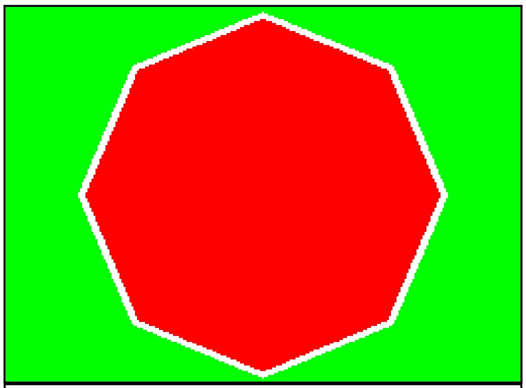

Figure 2: Polygon Filled with Colour

We can create a single, bounded area by defining the beginning and end nodes of a closed set of lines (actually, each node doubles-up as a beginning and an end node of a line). This forms a polygon if the nodes are equiangular and equidistant about the centre of the shape, otherwise we call it a polyline. In this way we can define such shapes as rectangles, hexagons, octagons etc.

These bounded areas can be filled with colour or with other patterns and images from rectangles of patterns as in Figure 2.

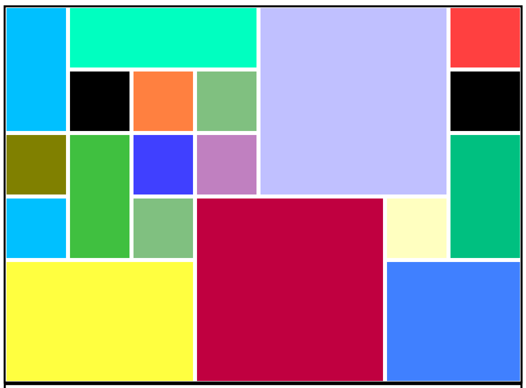

Figure 3: Random Rectangles

We can make patterns from some of these basic shapes by arranging them into arrays across our drawing area.

Figure 3. divides the surface into rectangles of different colours based on squares of unit size. The Pattern Grids gallery has images like this.

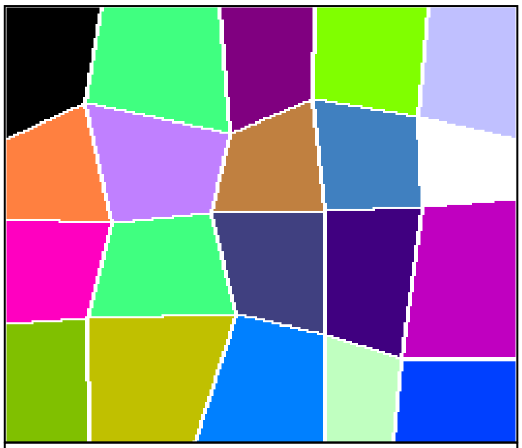

Figure 4: Irregular Quadrilaterals

Also derived from a rectangular grid we can vary the positions of the corners of the rectangles to yield a more irregular pattern, as in Figure 4.

Notice how we can use random numbers to determine certain of the parameters for the pattern, this means that when we design the general form of the pattern we are in fact describing a range of possible images. The variation of the nodes from rectangles in Figure 4 is determined in this way.

We can divide up the surface with other polygon shapes as well. Hexagons can be used to completely cover the surface, or we can use a mixture of polygon and other shapes to complete the pattern. Figure 5 is an example using octagons and squares (as diamonds).

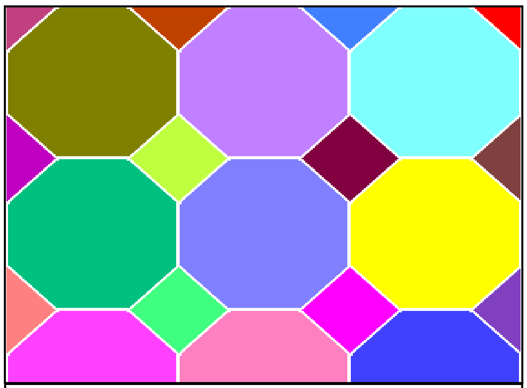
Figure 5: Octagons Grid

We do not have to confine ourselves to determining the shapes of the areas in this way. Instead we can associate characteristics such as pattern and colour to a central node of some area and then use some method to determine which pixels belong to which central node.

We can also divide up the drawing area with criss-crossing lines, as in Figure 6 where the lines are made by the tracks of an orbiting particle.

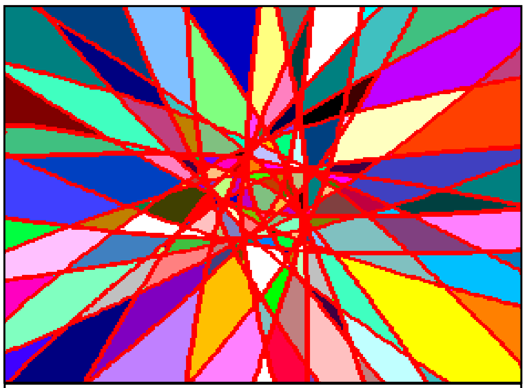
Figure 6: Particle Motion Tracks

If we scatter the central nodes at random and associate the neighbouring pixels with the nearest central node, this produces irregular zones as in Figure 7.

Freeing ourselves even further from the idea of a predefined shape, we can begin with a random scatter of coloured pixels and apply a method to scan the area around each pixel in random sequence to determine the most frequent occurrence of a colour and set the central pixel of the area to that colour. In this way we can create patches of different colour as in figure 8. In this case we can vary the range of colours and their probabilities as well as the area scanned. Notice how randomness introduces curves into the pattern in spite of the fact that we are using a rectangular scan area on a square grid.

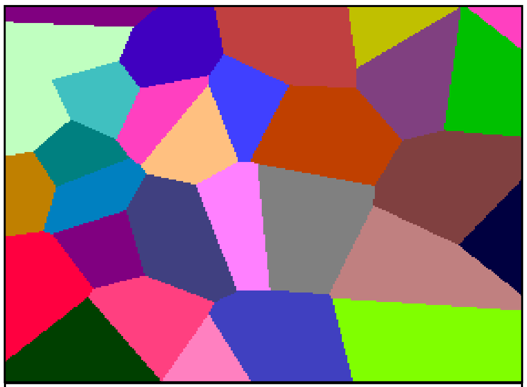
Figure 7: Irregular Zones

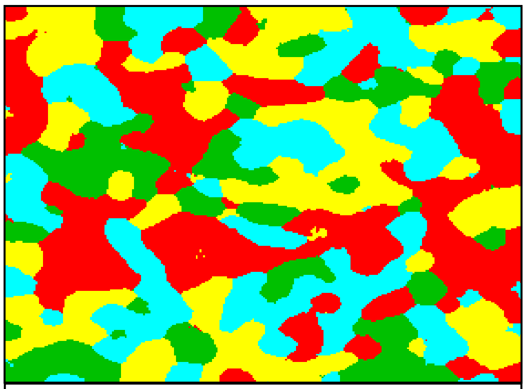
Figure 8: Random Patches

Returning to the idea of a set of scattered points on the surface (as in Irregular Zones), we can introduce the idea of a set of cells growing from each point with periodic or irregular colour changes as the set of cells grows outwards. Furthermore we can vary the rates of growth in each of 8 (orthogonal or diagonal) directions periodically or irregularly. This yields patterns like those in the Cellular Growth gallery and those in Figures 9 and 10.

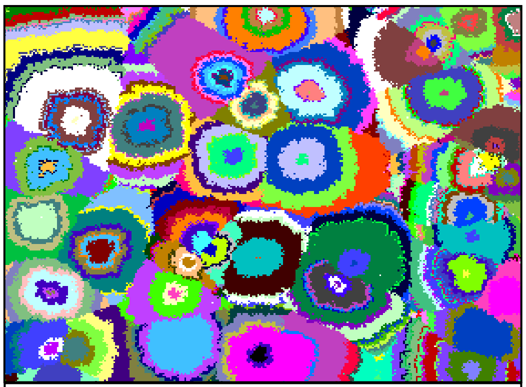
Figure 9: Cellular Growth with Randomised Direction Growth

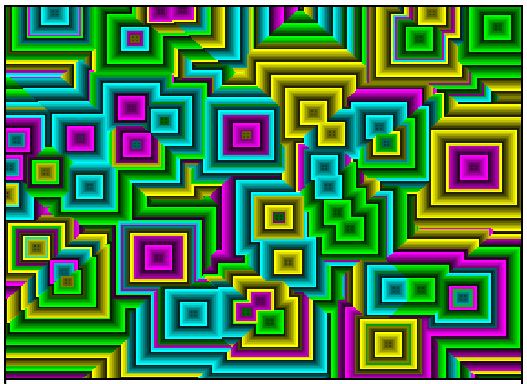
Figure 10: Cellular Growth with Periodic Direction Growth

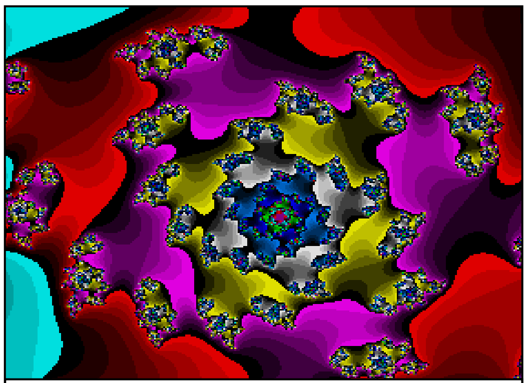
Figure 12: Fractal Pattern

Following on with the idea of cellular growth, we can develop the idea of linear growth with branching, and extend even that idea with side-growth so that we get tree-like structures as in Figure 11.

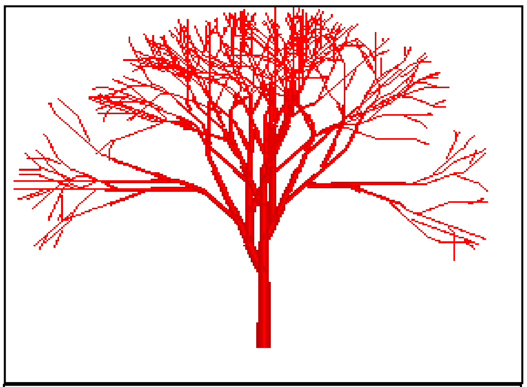
Figure 11: Tree Structure

In this case we cannot represent the branch angles directly in terms of the 8 orthogonal and diagonal directions of the square grid, some of the angles are at say 30 degrees, so we have to keep a record of the original branch bud direction and plot the lines so that they LOOK as though they are at the angle. This is an example of the difference between the structure of the computer and reality and the mathematical universe that we use to model it. We plot curves and lines that fool the eye. Nevermind, the Universe plays the same game by building structures from tiny particles.

We do not have to start with the idea of areas or shapes or points to generate a pattern. Instead we can plot patterns whose colours are a function of the X and Y coordinates of the drawing area. Figure 11 is a fractal pattern generated by a complex number function of the form $z \to z^n + u$ where z and u are complex numbers where the result is iterated back into the function until some limit is reached. The initial value of z is a derivative of X and Y and these are the real and imaginary parts of the complex number. The colour of the pixel at X, Y is then derived from a count of the iterations. Notice how patterns within this repeat at different scales, we can also zoom into them by reducing the range derived from X, Y and the patterns continue at finer scales.

We can also use the idea of the count of the number of times a pixel is entered by a function operating within a range of X,Y values. The attractor pattern in Figure 13 uses a set of trigonometrical functions with feedback operating on parameters to derive a shaded pattern in this way.

We can also create a visual effect by simply scattering colour and tone over the pixels at random, though this is not strictly a pattern, Figure 14 demonstrates this effect.

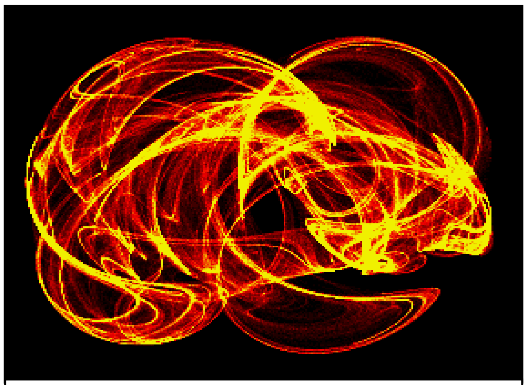
Figure 13: Attractor Pattern

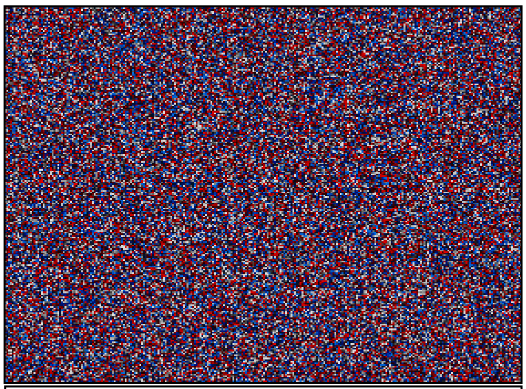
Figure 14: Random Tones

NESTED PATTERNS

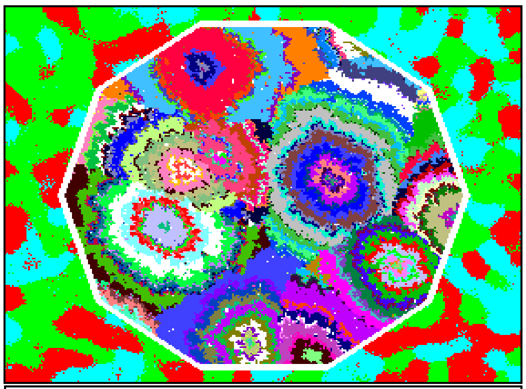
Figure 15: Pattern Filled Polygon

Once we have the idea of rectangles of pattern we can use them to fill areas of other patterns to create interesting effects. See Figures 15 and 16.

There are two types of area that we can fill, those of a particular colour already on the image and areas defined as polygons or polylines. These are raster and vector fill routines respectively. Note that with vector fills we are inside the area when we have crossed an odd number of lines, though we need to account for horizontal lines while we are scanning the area. The raster fill routine begins anywhere on the area of colour and scans from left to right then right to left to complete the line scan. It keeps track of where it is to scan subsequently on a stack, in this way it deals with the vertical scan. Note that it scans from the start position to the end position of the preceding line to take account of vertically connected areas.

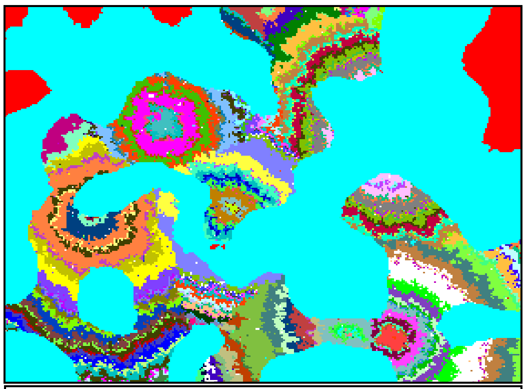
Figure 16: Raster Filled Pattern

We can extend the idea of filling by adding in fills of file images so that we can include photographic and hand drawn picture elements and images from other sources.

We can also use rectangles of pattern to create reflected effects. These can be either circular reflection, or reflection into two or more rectangular areas. Notice how a new pattern seems to emerge around the lines and centres of reflection. See Figure 17.

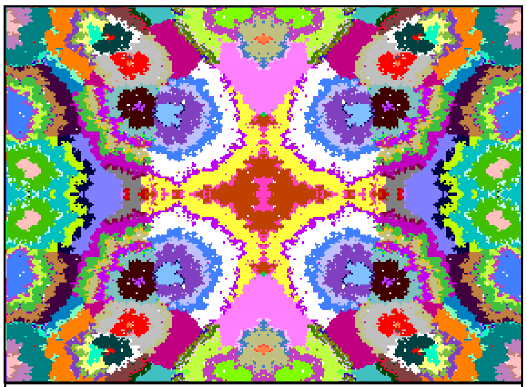
Figure 17: Reflections of Cellular Growth

OTHER POLYLINE SHAPES

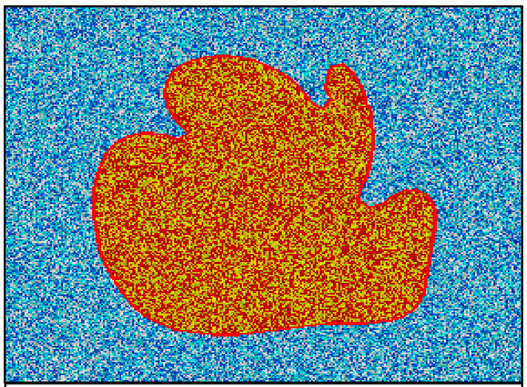
Figure 18: Polyline Chain

We do not have to make simple shapes bounded with polylines. We can make things more interesting by using other methods than rotation about a centre to define them. Here we have Polyline Chains, in which we begin with an ellipse of nodes and then push nodes inward or outward and drag adjacent nodes with them, like pulling on a link in a ring of chain. We can arrange things so that random nodes are moved, this produces a wide variety of shapes. The result can look rather like an amoeba. See Figure 18.

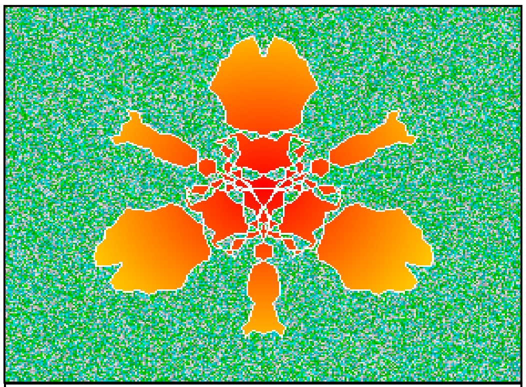
Figure 19: Drunken Walk Rotated and Reflected

We can make use of the idea of a "drunken walk" in which we stagger from point to point in a general direction. This makes a complex line composed of the points which we reach along the way. We can then take this line and join it end to end with copies around a central point. This provides shapes that can be filled like that in Figure 19.

We can also plot complex curves by rotating circles within circles (and again a circle within that). The maximum sensible nesting level is about 4 in which only a limited number of forms are coherent. The most interesting level of nesting is 3.

These shapes are called Hypotrochoids and were made familiar to a generation by the Spirograph toy. See Figure 20. Hypotrochoids are best plotted by using circles with fractional ratios of sizes, the number of loops is then a derivative of the sum of the

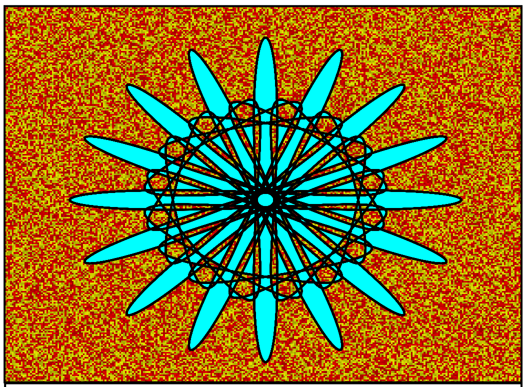
Figure 20: Hypotrochoid demonstrating Cyclical Periodicity

divisor and denominator of the fraction, so a radius ratio of 2/3 will produce five loops, for example. In Figure 20, there are sixteen major loops, so that pattern is produced by a ratio like 5/11 using 3 circles.

These cyclical patterns were produced using a VBA macro to plot the polylines, rather than XBASIC.

3D SHAPES (FORMS)

Since we can vary the brightness of pixels, we can see that we can represent the illumination of a surface in the same way that we shade areas in a drawing. Clearly, we need to be able to represent a surface so that its elements can be illuminated accordingly. To do this we can divide the surface into an array of triangles since these unambiguously represent a plane which can have an angle with respect to incident light. The facets of the surface so represented have three sets of three coordinates (x,y,z) and a maximum colour attribute which can be varied according to the incident light. The angle of the triangular planes is best described by the normal (n) to the plane, which is vector perpendicular to the plane and can be found by cross-multiplying the two vectors (v and w) derived from the triangle coordinates.

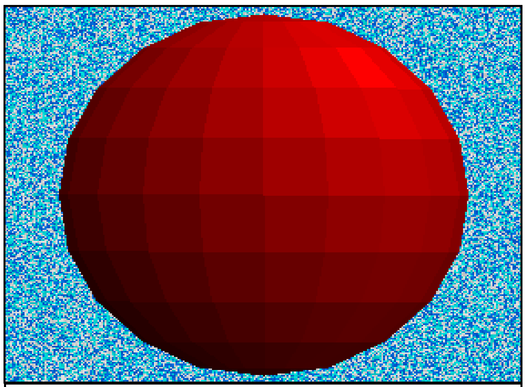
Figure 21: Spheroid demonstrating planar illumination

```
n.X = (v.Y * w.Z) - (v.Z * w.Y)
n.Y = (v.Z * w.X) - (v.X * w.Z)
n.Z = (v.X * w.Y) - (v.Y * w.X)
```

The illumination of the facet is then derived from the angle between this normal and that of the "light" beams.

Figure 21 shows a hemispherical form where each quadrilateral face is comprised of two such triangles with the light coming from the top right.

The triangles can be determined from the nodes of a sphere or other shape as we plot them at angular and linear intervals. The smaller we make the triangles, the smoother our curved surfaces will be, although if we use arrays to store the results, we will also use more memory.

We are not confined to using a single colour for our 3D shapes, we can either assign colours from a rectangular pattern wrapped around the shape (See Figure 22) or we can apply colour as we are building the shape. Not every 3D shape can be conveniently wrapped by a rectangular pattern. See also the gallery "Primitives of Pattern and Form".

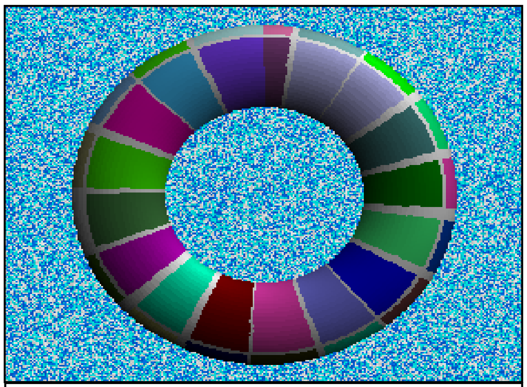
Figure 22: Torus wrapped in rectangular pattern

We may also want to represent patterns on the faces of an object like a cube, which cannot be so easily plotted by reference to central or linear point.

The Cubic Lattice overcomes this problem by having a node coordinate system (x, y) on each of its six faces. Triangular facets can be derived from these nodes so that a pattern can be superimposed on it and represented at the appropriate angle in space. See Figure 23 and the gallery "Primitives of Pattern and Form". Since the cube cannot be wrapped in a rectangular pattern without omission, the grid coordinates of the cube can be used to tailor patterns for the whole surface.

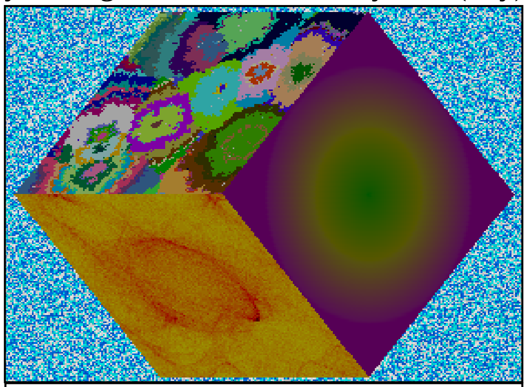
Figure 23: Cubic Lattice with pattern on each face

In addition to providing more scope for patterns, the nodes of the cubic lattice can be manipulated to produce other shapes whilst still retaining their relative co-ordinate system. This allows us to produce forms like the spikey cubic lattice in Figure 24. Here the nodes are pulled outward in concentric squares towards a central node in each face.

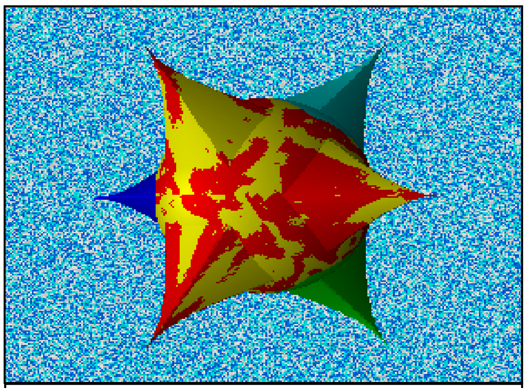
Figure 24: Spikey Cubic Lattice with pattern on each face

SUMMARY

This work provides some basic patterns and forms which can produce interesting visual effects. There is great deal of scope to extend the work further. If you are interested in doing the development yourself, email me at info@narayana-art.co.uk and I can arrange to send you my XBASIC source code. You can obtain the XBASIC Program Development Environment (PDE) via www.xbasic.org.