:

The SPOT System

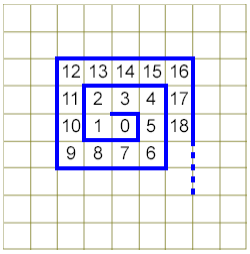

Figure 1

SPOT is an acronym of ***S**piral **P**osition and **O**rientation **T**ranslator*. It is a method of denoting digital vectors (movements, linear displacements) in a row and column matrix of elements, such as the pixels of a computer display or tiles of a mosaic.

On a rectangular grid of tiles, select one to serve as a reference. Label it '0'. Moving out and around this reference tile in a clockwise rectilinear spiral pathway, count each tile on the path, as shown in the illustration.

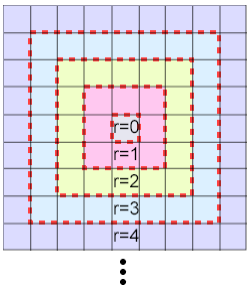

Figure 2

Each complete circuit of the spiral comprises a ***range*** of tiles. The center tile, 0, is Range r=0; tiles 1...8 constitute Range r=1; tiles 9...24 constitute Range r=2; and so on. Each range comprises a block of tiles all of which are 'r' distance from the center tile '0'. The tile numbers are called ***tallynumbers***.

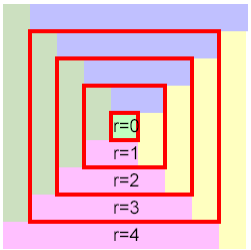

Figure 3

A range comprises four ***legs***, each of which has 2r tiles.
The first tallynumber, T, of a range is
T = (2r - 1)^2 = 4r^2 - 4R + 1 . ['^2' means to square a number; example 7^2 = 49.] For Range r=1, 1^2 = 1; for r=2, 3^2 = 9; for r=3, 5^2 = 25; and so on.
The last tallynumber of a range can be expressed
T = 4r(r+1) = 4r^2 + 4r .
The first tallynumber of Range r+1, the next range, is T = 4r^2 +4r + 1

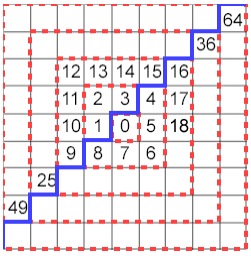

Figure 4

The odd square tallynumbers are aligned along a diagonal down and to the left of the center; the even square tallynumbers, up and to the right of the center tile. The stepped boundary common to the odd and even square tallynumbers is the ***major diagonal***, drawn here in blue.

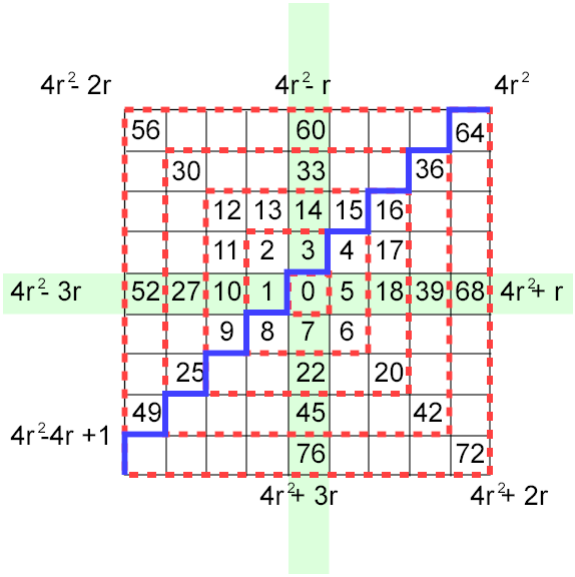

Figure 5

Tallynumbers can easily be expressed as a function of their range, r. For example, in Range r=3, on the horizontal axis, tallynumber T = 39 = 4*3*3 + 3

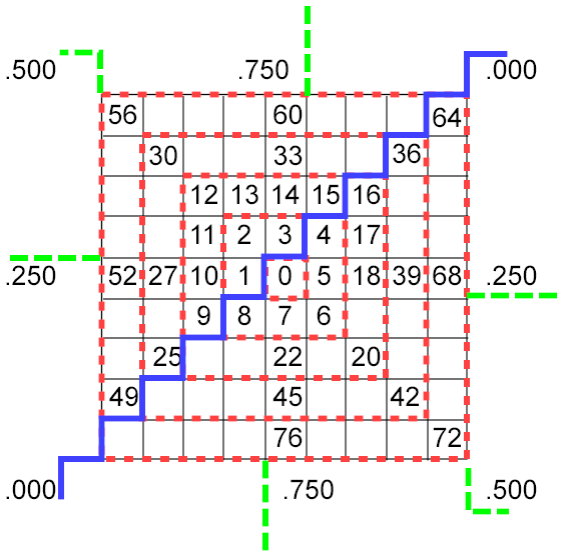

Figure 6

The square root of a tallynumber is its ***tallyroot***; the fractional portion of the root value approaches limits associated with lattice diagonals and axes: .000, .250, .500, .750, and back again to .000 at the major diagonal. Thus the tallynumber (and tallyroot) convey positional information in their numeric values. Further, the integer part, M, of the square root value will be **odd**, M=2r-1, if the target tile is to the left of the major diagonal and **even**, M=2r, if to the right. This is where the SPOT System derives its name; the positional character of a vector displacement from the reference, '0' to some target tile is expressed in its tallyroot, which also is called its ***SPOT value***.

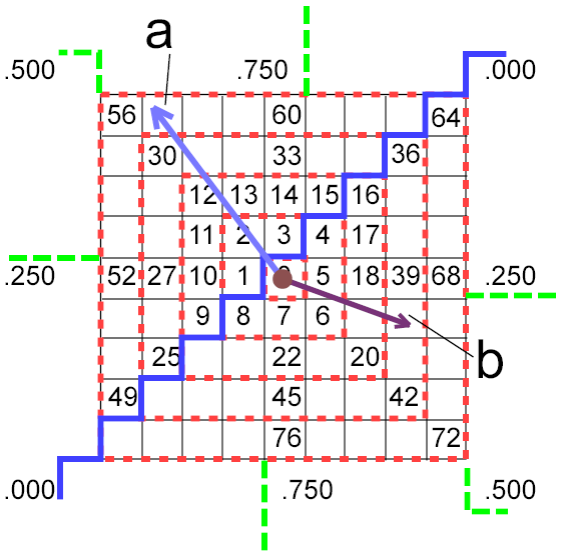

Figure 7

The integer part of the tallyroot gives the magnitude (length), M, of the displacement (line). The fractional part of the tallyroot gives the gradient (angle of inclination), G, of the vector in relation to the major diagonal (.000). Two vector examples are shown in Figure 7.

Vector a: Range = 4; Tallynumber = 57; tallyroot = 7.54983...; M= 2r-1 = 7; G= .54983....

Vector b: Range = 3; Tallynumber = 40; tallyroot = 6.32456...; M= 2r = 6; G= .32456....

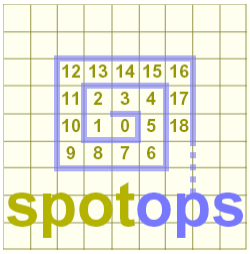

Figure 8

In digital graphics the axes and diagonals of the SPOT lattice echo sentient operations within human visual perception. Further, myriads of regularities in the SPOT numeric field enable quick and convenient analysis of fundamental factors of visual form such as 'positive/negative' shape and rhythms of pictorial composition. [See *Expressing Shape and Composition by LAVA/SILK*, Howard Jones, 2004]

Cartesian/SPOT Conversion Routines

```
#! /usr/bin/env python
"""
SPOT Translator: converts between Cartesian & SPOT digital vectors.
On left-handed cartesian row/column matrix (z-axis pointing out from screen,
as in conventional computer graphics display screens) the Spiral Position
Orientation Translator (SPOT) pathway spirals counter-clockwise about
tallynumber=0 at cartesian element (x,y).
Routines:
    carSpotT(dx,dy): 2D Cartesian displacement to SPOT tallynumber
    carSpotV(dx,dy): 2D Cartesian displacement to SPOT value (tallyroot)
    spotVcar(spotValue): SPOT value (tallyroot) to 2D Cartesian displacement
    spotTcar(tallynumber): SPOT tallynumber to tallyroot to 2D Cartesian displacement

version date: 20040329

"""

import math

# convert 2D Cartesian displacement to SPOT tallynumber
def carSpotT(dx, dy):   # returns SPOT tallynumber
    x = int(dx)         # dx,dy  must be signed integers!!!
    y = int(dy)
    xneg = x<0
    yneg = y<0

    absx = abs(x)
    absy = abs(y)

    if(absx == absy):   # at corner of SPOT Range square?
        r = absx
        if(xneg and not(yneg)):
            spotT = 4*r*r + 4*r
        elif(xneg and yneg):
            spotT = 4*r*r - 2*r
        elif(not(xneg) and yneg):
            spotT = 4*r*r
        elif(not xneg and not(yneg)):
            spotT = 4*r*r + 2*r

        return spotT

    else:               # at which Leg ("side") of Range?
        r = max(absx, absy)
        if(r == absx):
            if(xneg):
                spotL = 0
            else:
                spotL = 2
        else:
            if(yneg):
                spotL = 1
            else:
                spotL = 3

    if(spotL == 0):     # calculate SPOT TallyNumber on Range
        spotT = 4*r*r - 3*r - y
    elif(spotL == 1):
        spotT = 4*r*r - r + x
    elif(spotL == 2):
        spotT = 4*r*r + r + y
    elif(spotL == 3):
        spotT = 4*r*r + 3*r - x

    return spotT

# convert 2D Cartesian displacement to SPOT value (tallyroot)
def carSpotV(dx,dy):
    spotV = math.sqrt(carSpotT(dx,dy))

    return spotV
```

```
# convert SPOT value to 2D Cartesian displacement
def spotVcar(spotValue):   # accepts SPOT value float

    # extract integer and fractional components of SPOT value
    sv = abs(spotValue)    # must be positive float
    iSpot = int(sv)         # integer portion
    fSpot = sv - iSpot      # fractional portion

    # determine vector orientation relative to major_diagonal = 0.0
    iodd = iSpot % 2        # is the integer portion odd?
    spotT = int(round(sv*sv))
    spotL = 0               # which leg of SPOT range ( 0, 1, 2, or 3) ?

    # find SPOT range
    r = (iSpot + 1) / 2

    # isolate which leg of SPOT range
    if iodd:
        if fSpot < 0.5 :
            spotL = 0
        else:
            spotL = 1
    else:
        if fSpot < 0.5 :
            spotL = 2
        else:
            spotL = 3

    if spotL == 0 :
        dX = -r
        dY = 4*r*r - 3*r - spotT
    elif spotL == 1 :
        dY = -r
        dX = spotT - 4*r*r + r
    elif spotL == 2 :
        dX = r
        dY = spotT - 4*r*r - r
    elif spotL == 3 :
        dY = r
        dX = 4*r*r + 3*r - spotT

    return (dX, dY)

# convert SPOT tallynumber to tallyroot to 2D Cartesian displacement
def spotTcar(tallynumber):
    return(spotVcar(math.sqrt(tallynumber)))
```