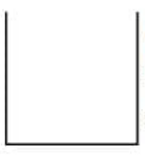
The Hilbert curve

The Hilbert curve is a space filling curve that visits every point in a square grid with a size of 2×2, 4×4, 8×8, 16×16, or any other power of 2. It was first described by David Hilbert in 1892. Applications of the Hilbert curve are in image processing: especially image compression and dithering. It has advantages in those operations where the coherence between neighbouring pixels is important (see Douglas Voorhies's contribution to the "Graphic Gems" series). The Hilbert curve is also a special version of a quadtree; any image processing function that benefits from the use of quadtrees may also use a Hilbert curve.

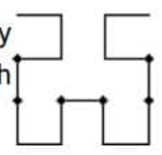
Note on the implementation: Niklaus Wirth published a recursive algorithm in "Algorithms + Data structures = Programs" second edition; 1976; Prentice Hall. I developed the algorithm presented here independently, mostly because I could not find Niklaus Wirth's book. David W. Brown, from Dana Corporation, later sent me photocopies of the relevant pages from the book (thank you again, David). From this, it is apparent that my approach closely follows what which Niklaus Wirth published over 20 years ago. The main difference is that this document describes a single large function and "Algorithms + Data structures = Programs" uses four small functions with indirect recursion.

Cups and joins

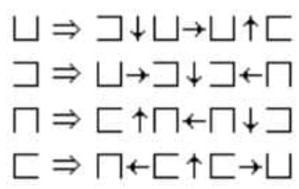
The basic elements of the Hilbert curves are what I call "cups" (a square with one open side) and "joins" (a vector that joins two cups). The "open" side of a cup can be top, bottom, left or right. In addition, every cup has two end-points, and each of these can be the "entry" point or the "exit" point. So, there are eight possible varieties of cups. In practice, a Hilbert curve uses only four types of cups. In a similar vein, a join has a direction: up, down, left or right.



A first order Hilbert curve is just a single cup (see the figure on the left). It fills a 2×2 space. The second order Hilbert curve replaces that cup by four (smaller) cups, which are linked together by three joins (see the figure on the right; the link between a cup and a join has been marked with a fat dot in the figure). Every next order repeats the process or replacing each cup by four smaller cups and three joins.



Cup subdivision rules



The function presented below (in the "C" language) computes the Hilbert curve. Note that the curve is symmetrical around the vertical axis. It would therefore be sufficient to draw half of the Hilbert curve.

Drawing the Hilbert curve

```
enum {
  UP,
  LEFT,
  DOWN,
  RIGHT,
};

void hilbert(int level,int direction=UP)
{
  if (level==1) {
    switch (direction) {
    case LEFT:
      move(RIGHT);      /* move() could draw a line in... */
      move(DOWN);       /* ...the indicated direction */
      move(LEFT);
      break;
    case RIGHT:
      move(LEFT);
      move(UP);
      move(RIGHT);
      break;
    case UP:
      move(DOWN);
      move(RIGHT);
      move(UP);
      break;
    case DOWN:
      move(UP);
      move(LEFT);
      move(DOWN);
      break;
    } /* switch */
  } else {
    switch (direction) {
    case LEFT:
      hilbert_level(level-1,UP);
      move(RIGHT);
      hilbert_level(level-1,LEFT);
      move(DOWN);
      hilbert_level(level-1,LEFT);
      move(LEFT);
      hilbert_level(level-1,DOWN);
      break;
    case RIGHT:
      hilbert_level(level-1,DOWN);
      move(LEFT);
      hilbert_level(level-1,RIGHT);
      move(UP);
      hilbert_level(level-1,RIGHT);
      move(RIGHT);
      hilbert_level(level-1,UP);
      break;
    case UP:
      hilbert_level(level-1,LEFT);
      move(DOWN);
      hilbert_level(level-1,UP);
      move(RIGHT);
      hilbert_level(level-1,UP);
      move(UP);
      hilbert_level(level-1,RIGHT);
      break;
    case DOWN:
      hilbert_level(level-1,RIGHT);
      move(UP);
      hilbert_level(level-1,DOWN);
      move(LEFT);
      hilbert_level(level-1,DOWN);
      move(DOWN);
      hilbert_level(level-1,LEFT);
      break;
    } /* switch */
  } /* if */
}
```

Conclusion

The Hilbert curve is easy to generate. When applied over a digitized photograph or a ray-traced image, it makes better use of the coherence of neighbouring pixels than the traditional scan-line based approach.

For those interested: there exists a 3-dimensional variant of the Hilbert curve: a curve that fills a cube.

References

Voorhies, Douglas; "Space-Filling Curves and a Measure Of Coherence"; Graphic Gems II, edited by James Arvo; Academic Press; 1991; pp 26-30.

Discusses the coherence level of the Peano and the Hilbert curves. The coherence level is the amount in which neighbouring pixels are at sequential positions on the space filling curve.