3D Hilbertkurve

```
> restart;
  with(plots):
Warning, the name changecoords has been redefined
```

Arithmetisierung der 3D-Hilbertkurve

Definition der Operatoren H0 bis H7 (siehe Sagan, S. 28):

```
> H[0] := (x,y,z) -> ( x/2,     z/2,     y/2);
  H[1] := (x,y,z) -> ( z/2,     y/2+1/2, x/2);
  H[2] := (x,y,z) -> ( x/2+1/2, y/2+1/2, z/2);
  H[3] := (x,y,z) -> ( z/2+1/2,-x/2+1/2,-y/2+1/2);
  H[4] := (x,y,z) -> (-z/2+1,  -x/2+1/2, y/2+1/2);
  H[5] := (x,y,z) -> ( x/2+1/2, y/2+1/2, z/2+1/2);
  H[6] := (x,y,z) -> (-z/2+1/2, y/2+1/2,-x/2+1);
  H[7] := (x,y,z) -> ( x/2,    -z/2+1/2,-y/2+1);
```

$$H_0 := (x, y, z) \to \left(\frac{1}{2}x, \frac{1}{2}z, \frac{1}{2}y\right)$$

$$H_1 := (x, y, z) \to \left(\frac{1}{2}z, \frac{1}{2}y + \frac{1}{2}, \frac{1}{2}x\right)$$

$$H_2 := (x, y, z) \to \left(\frac{1}{2}x + \frac{1}{2}, \frac{1}{2}y + \frac{1}{2}, \frac{1}{2}z\right)$$

$$H_3 := (x, y, z) \to \left(\frac{1}{2}z + \frac{1}{2}, -\frac{1}{2}x + \frac{1}{2}, -\frac{1}{2}y + \frac{1}{2}\right)$$

$$H_4 := (x, y, z) \to \left(-\frac{1}{2}z + 1, -\frac{1}{2}x + \frac{1}{2}, \frac{1}{2}y + \frac{1}{2}\right)$$

$$H_5 := (x, y, z) \to \left(\frac{1}{2}x + \frac{1}{2}, \frac{1}{2}y + \frac{1}{2}, \frac{1}{2}z + \frac{1}{2}\right)$$

$$H_6 := (x, y, z) \to \left(-\frac{1}{2}z + \frac{1}{2}, \frac{1}{2}y + \frac{1}{2}, -\frac{1}{2}x + 1\right)$$

$$H_7 := (x, y, z) \to \left(\frac{1}{2}x, -\frac{1}{2}z + \frac{1}{2}, -\frac{1}{2}y + 1\right)$$

Algorithmus gemäß der Arithmetisierung der Hilbert-Funktion:

```
> hilbert3D := proc(t::numeric, depth::integer)
  # t: Argument der Hilbertfunktion
  # depth: Zahl der berücksichtigten Quartenärstellen von t
     global H;
     local q,r;

     if depth = 0 then return (0,0,0) end if;
```

```
    q := floor(8*t); # extrahiere die erste Stelle der
  Oktalzahl
    r := 8*t-q;      # und den Rest

    return H[q]( hilbert3D(r,depth-1) );

  end proc:
> hilbert3D(1/2,5);
```

$$1, \frac{1}{2}, \frac{1}{2}$$

```
> hilbert3D(1/8,9);
```

$$0, \frac{1}{2}, 0$$

```
> hilbert3D(1/3,5);
```

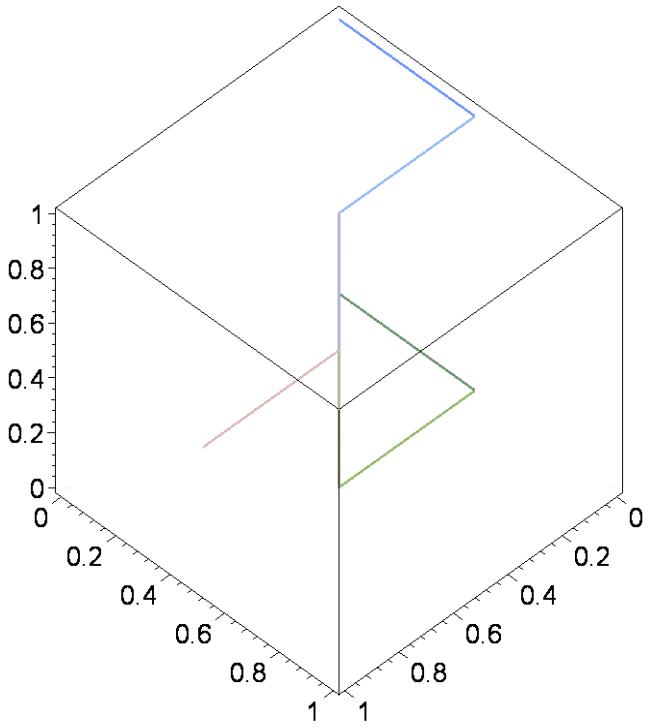
$$\frac{31}{32}, \frac{31}{32}, \frac{5}{16}$$

```
> hilbert3D(0.99999,10);
```

$$\frac{27}{1024}, \frac{9}{1024}, \frac{501}{512}$$

Plotten der approximierenden Polygone

```
> corners := pointplot3d([[0,0,0],[0,0,1],[0,1,0],[1,0,0]]):
> iter := 1:
  curve :=
  spacecurve( [ seq( [hilbert3D(i/8^iter,8)], i=0..8^iter-1),
  [0,0,1] ],
              axes=BOXED, scaling=CONSTRAINED, thickness=3):
  display3d(corners,curve);
```

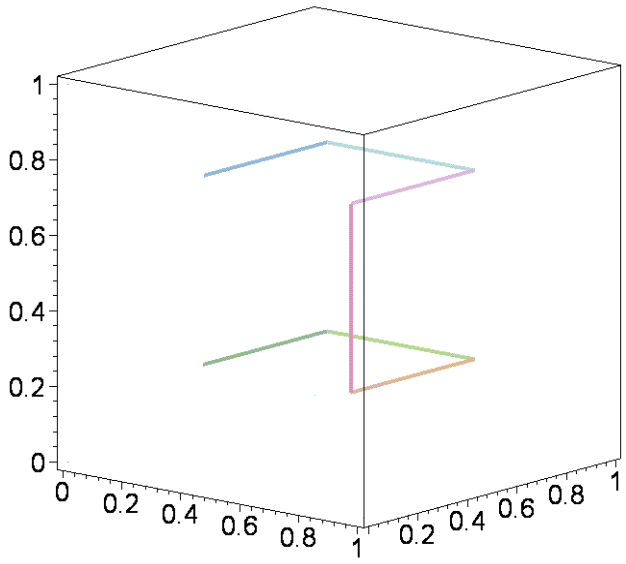



```
>
```

Plotten der Hilbertkurve

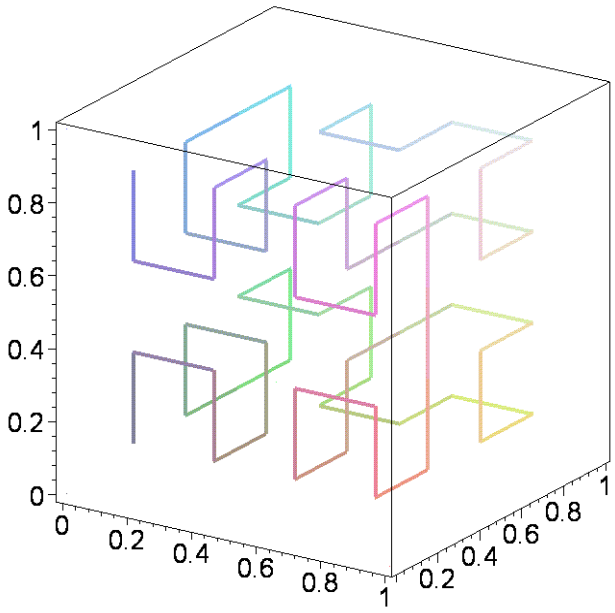
```
> iter := 1;
> curve :=
  spacecurve( [ seq( [hilbert3D((i+3/8)/8^iter,8)],
  i=0..8^iter-1) ],
              scaling=CONSTRAINED, axes=BOXED, thickness=4):
  display3d(corners,curve);
```

$$iter := 1$$




```
> iter := 2;curve :=
  spacecurve( [ seq( [hilbert3D((i+3/8)/8^iter,8)],
  i=0..8^iter-1) ],
                scaling=CONSTRAINED, axes=BOXED, thickness=4):
  display3d(corners,curve);
```

$$iter := 2$$




```
>
```

Parallelisierung

Wir benötigen einige Farben, um die Partitionen verschieden einfärben zu können:

```
> colors := [blue, red, green, yellow, cyan, magenta, black,
             gold, pink, brown, violet, gray, coral];
```

colors :=

[*blue*, *red*, *green*, *yellow*, *cyan*, *magenta*, *black*, *gold*, *pink*, *brown*, *violet*, *gray*, *coral*]

partition zerlegt eine Liste von Punkten (wiederum Listen) in number Einzellisten

```
> partition := proc(pts::list, number::posint)
    local parts,i;
    parts := [ pts[ (number-1)*floor(nops(pts)/number)..-1 ]
  ];
    for i from number-1 by -1 to 2 do
       parts := [ pts[
  (i-1)*floor(nops(pts)/number)..i*floor(nops(pts)/number) ],
                  op(parts) ];
    end do;
    parts := [ pts[ 1..floor(nops(pts)/number) ], op(parts)
  ];
    return parts;
  end proc:
```

Erzeugen der Partitionen der Hilbertkurve

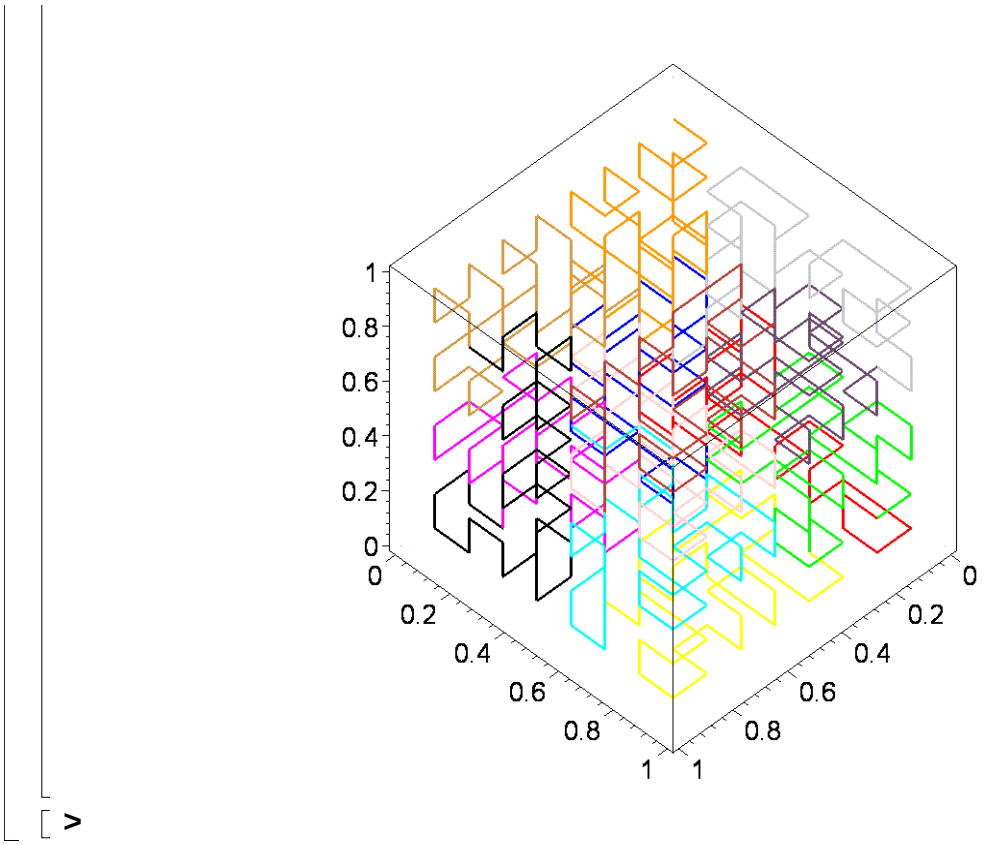
```
> iter:=3; anzpart := 13;
  parts := partition([ seq( [hilbert3D((i+3/8)/8^iter,8)],
  i=0..8^iter-1) ], anzpart):
```

iter := 3

anzpart := 13

Plot der Partitionen

```
> curves := seq( spacecurve( parts[i], scaling=CONSTRAINED,
  axes=BOXED,
                                        thickness=3, color =
  colors[i]),
                 i=1..anzpart) :
  display3d(corners,curves);
```


1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
1
0.8
0.6
0.4
0.2
0
v