Mycelium 3D

3D graphics with Context Free Art

www.fdsa.it/mycelium3d

Mycelium 3D 2014

Image in first page was made using Context Free Art and Mycelium 3D.

Requirements and use of the library

Mycelium 3D is available for download at http://www.fdsa.it/mycelium3d

To use this library, you need Context Free Art v. 3. All functions relating to the management of the colors and alpha channel that are used are those provided natively by CFA. It's asked, of course, a bit 'of familiarity with CFA. To include it, the import command is used. The mycelium3d.cfdg file must be in the same directory as the drawing file that we are doing. Before the import command, you must specify the level of pseudo-shading declaring the constant *shade*. Example:

```
shade = .5
import Mycelium3D_2014.cfdg
```

Basic Concepts

Local 3D reference system

The 3D local reference system is a vector (a list of 18 numbers) that contains all the information that specifies:

- the position in 3D space.
- the orientation of the three orthogonal axes.
- the scale of the 3 axes.
- pseudo-physical properties such as weight, stiffness, age.

You can imagine it as a cube rotated and deformed, or like three coordinate axes rotated and scaled in space. Almost all the operations that we perform involve creating, moving, rotating, scaling, duplicating local 3D reference systems.

Why local? Local system is opposed to absolute system. In the sense that once we changed it, subsequent changes always occur at the coordinate axes of the system itself (local system) and not with respect to the absolute coordinates of the 3D space.

Moving a object is like piloting it from a subjective point of view.
There is analogy with what happens normally in the use of Context free art in 2D.

3D space, 2D space, camera, primitives

The space in which objects created with M3D "exist" has 3 dimensions. For their representation (on a 2D canvas) you need to use a camera that looks at them for display on the screen.

CANVAS 2D

It is the two-dimensional space on which the design appears - the drawing area on the screen.

PRIMITIVE SHAPES

The basic primitive shapes are few and simple: sprites (circles), cubes, lines, tapes, pipes.
The purpose of the library is in fact create very complex designs formed by millions of elements that must be fast to calculate and represent.
Creating new primitive is however very simple, the code of the implemented primitives can be used as guide.

Working with 3D reference systems

Ex novo creation and duplication

A reference system is created with the 3D function Sys ()

```
W = Sys()
```

W is initially aligned with the principal axes and has unitary size.

to duplicate W we use the = operator

```
duplicate =  W
W1 = W
```

Movement

```
W = MoveX( 1, W )  // move W of 1 along X axe
W = MoveY(-1, W )  // move W of -1 along Y axe
W = MoveZ(.5, W )  // move W of 0.5 along Z axe

W = MoveXYZ(mx, my, mz , W )  // generic movement
```

Rotation

```
REF = RotX( 10, W )  // rotate W   10° around X axe
REF = RotY(  4, W )  // rotate W    4° around Y axe
REF = RotZ(-18, W )  // rotate W  -18° around Z axe
```

Scale and symmetry

```
W = Scale (.9, W )     // homogeneous scale the 3 axes

W = ScaleX(.5, W )    // scale only X axes
W = ScaleY(.7, W )    // scale only Y axes
W = ScaleZ(.5, W )    // scale only Z axes

W = ScaleX(-1, W )    // scale can be negative with symmetry effect
                      // in this case symmetric to YZ plane
```

Cameras

Presets

There are preset cameras whose names say a lot about…:

- FRONT
- BACK
- TOP
- BOTTOM
- LEFT
- RIGHT

CAM equals FRONT

Customized cameras

It’s easy to make custom a Camera with Camera function:

```
CAM = Camera( rotationY, rotationX, distance )
```

The camera observes the default coordinates (0,0,0) .
To translate the camera (and the central point) function we use *MoveCamera* function:

```
CAM = MoveCamera( mx, my, mz, CAM )
```

The coordinates mx, my, mz are absolute.

Primitive 3D objects

3D objects are primitive shape with 2 or 3 parameters that specify always at least one reference system and the camera to use.
You can specify the color with the normal parameters of the CFA, (in the final square brackets).

The sprite

The sprite is the simplest 3D element available. It is a circle drawn always perpendicular to the observer. (It always displays as a scaled circle).
To draw a sprite must use a reference system (W) specifying the center and scale (spritesize) of the circle and the camera from which we observe it (CAM)

```
SPRITE(spritesize, W , CAM)[ b 1 h 120 sat .5 a -.9 ]
```

3D Square (Face)

Square rotated in space, oriented according to the axes of reference system, perpendicular to Y. To draw it you must always specify a reference and a 3D camera:

```
FACE(cubesize, W , CAM)[ ]
```

Cube

Also called a hexahedron ... To draw it you must always specify a reference and a 3D camera:

```
CUBE(cubesize, W , CAM)[ ]
```

Lines, tapes and tubes

Sometimes, for linear elements, lines, strips or tubes can give better results than sprite and cube.
To draw lines and tubes you must specify not one but two reference systems (W1, W2) which correspond to the two ends of the segment.
Usually, we will duplicate the reference before modifying it. The not-changed-duplicate will be used later as first end of the line. The changed item is used as the other end

Note: When you use tapes or tubes movements must take place along the Y axis

```
TRACE(pensize, W1, W2, CAM)[ ]  // thickness of pensize scale independent.
MARKER(pensize, W1, W2, CAM)[ ] // marker pen stroke.
LINE(pensize, W1, W2, CAM)[ ]   // thickness of pensize variable according to scale
TAPE(pensize, W1, W2, CAM)[ ]   // tape that follows movements along Y.
TUBE(pensize, W1, W2, CAM)[ ]   // tube that follows movements along Y.
```

W1 and W2 are the two reference systems that specify the 2 ends of line.
CAM must be a camera.

Ricorsion control

Ending the execution of recursive shape

In many cases we want to use recursive shapes as well as we normally do with CFA in 2D.
Since there is no automatic way to do it, we rely on one of 2 functions to be associated with [s]:

FIRST WAY: MINSIZE()

```
RecursiveShape(W)[ s MinSize( 0.1 , W) ]
```

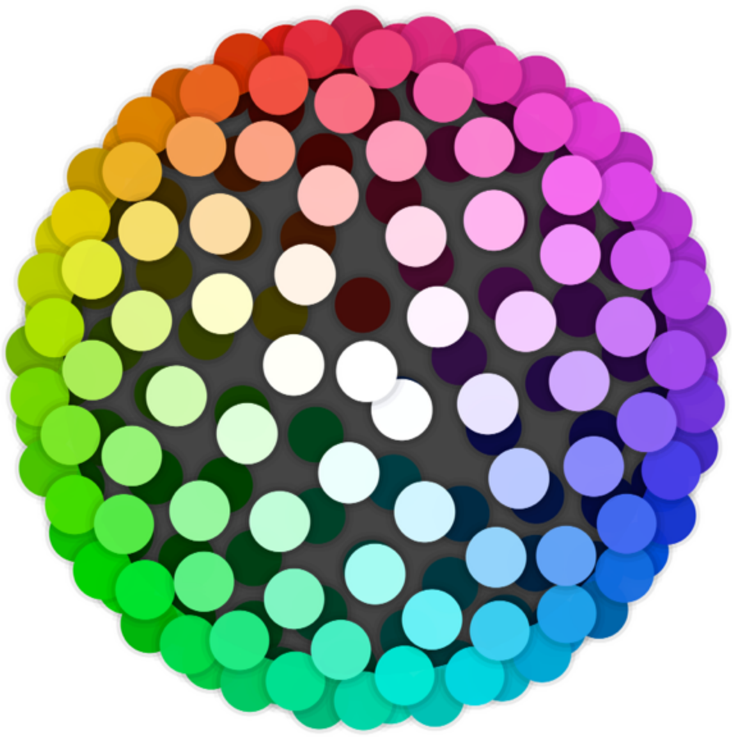
RecursiveShape will be terminated if W is smaller than 0.1.

SECOND WAY: MAXAGE ()

```
RecursiveShape(W)[ s MaxAge( 10 , W) ]
```

RecursiveShape terminates automatically if age > = 10; age is a variable in reference system W that is automatically incremented every time we use the Move function.
In most cases it is therefore linked to the number of iterations of the recursive shape

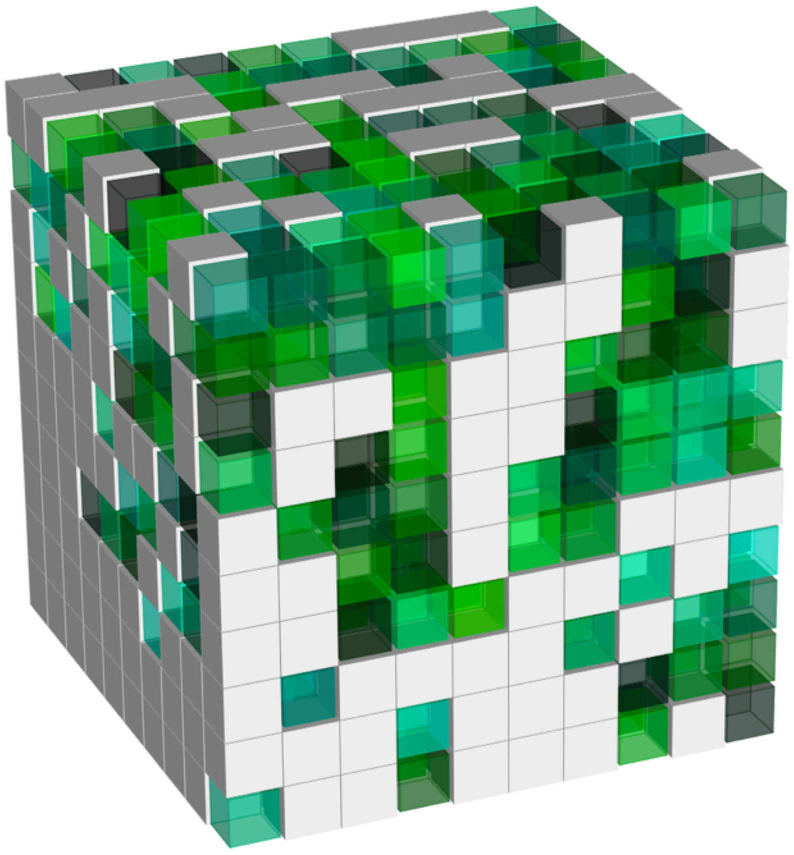
Examples



Sprite ball

```
shade = .45
import mycelium3D-2014.cfdg

CF::Background = [b 1 ]
CF::Size = [s 60]

K=Camera(0, 90, 400)  // define the camera

startshape Sphere(180, Sys()) // here we create the original Reference System

shape Sphere (count, vector18 W)  // remember to specify vector18
{
       SPRITE(36.5, W, K)[a -.2]

       loop i = count/1  [ h phi  sat .03 ]
       {
              kk = asin(-1 + 2*i/count)
              W1=RotX( (phi*i) , W)
              W1=RotZ( (kk) , W1)
              W1 = MoveY(20 , W1)
              SPRITE(3.5, W1, K)[ b 1  a -.03   ]  // Colored dot
              SPRITE(4, W1, K)[ b -1 a -.9 z -.001      ]   // shadow 1
              SPRITE(4.5, W1, K)[ b -1 a -.95 z -.002   ]   // shadow 2
       }
}
```

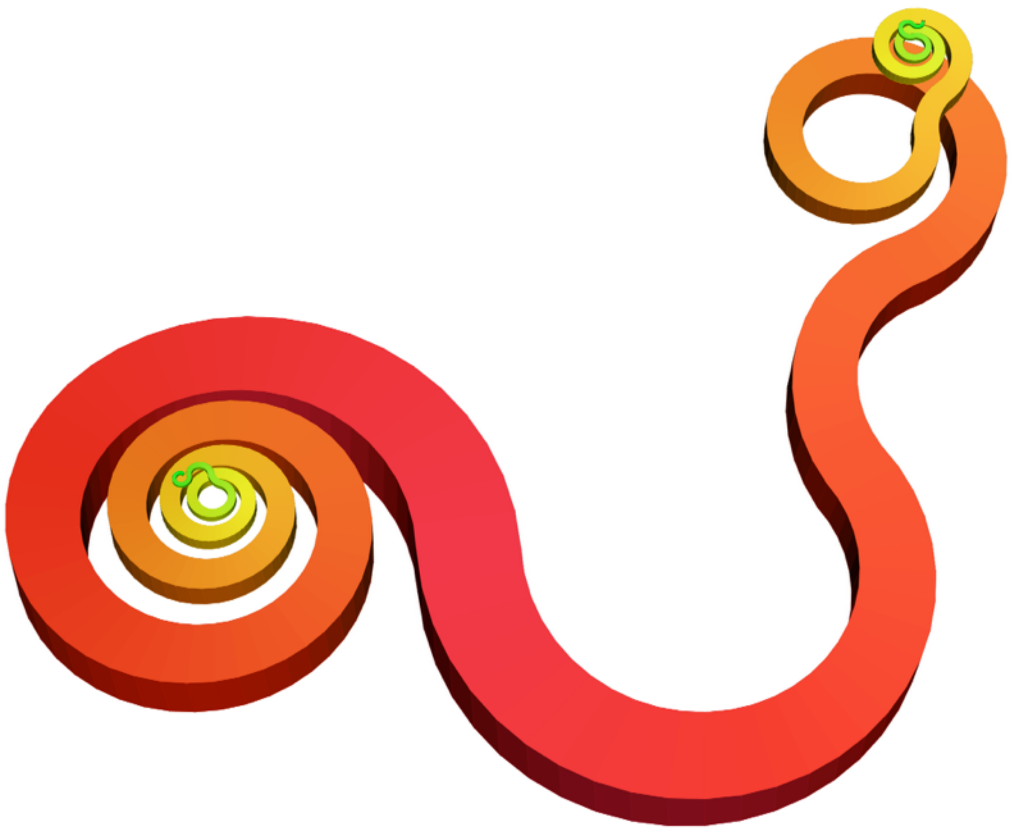
```
}
```



Cube of cubes

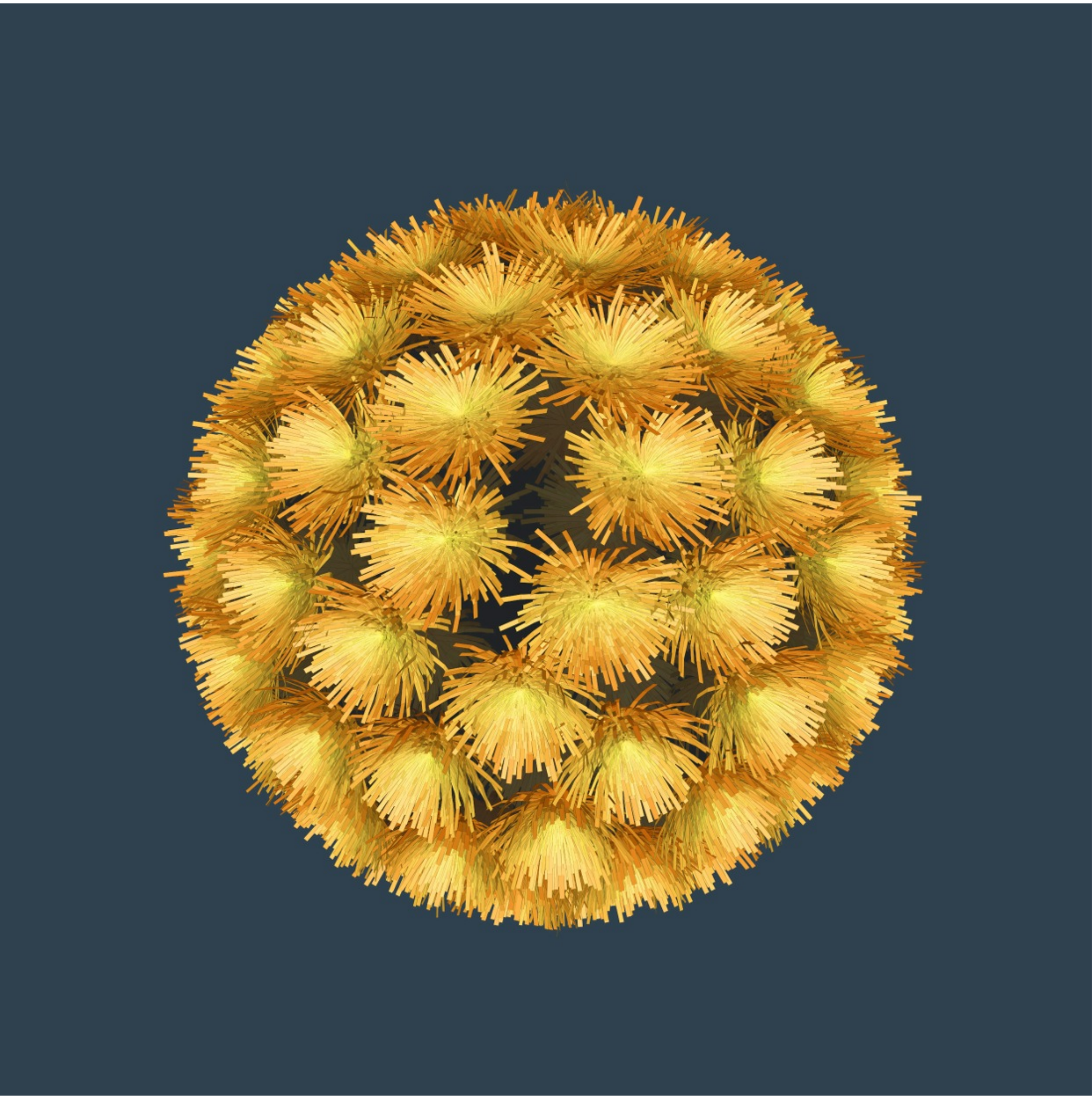
```
shade = 1
import mycelium3D-2014.cfdg

CF::Background = [b 1]

K=Camera(25, 66, 90)

startshape Matrix(10, Sys())[h 120]

shape Matrix( NN  , vector18 W)
{
	loop i=NN [] loop j=NN [] loop k=NN [] {
		W = MoveXYZ(i, j, k , W)
		if (rand(NN*2)>(i+j))
			CUBE(.97, W, K )[b 1 ]
		else
			CUBE(.88, W, K )[b rand(1) sat 1 a -.5 h rand(60) ]
	}
}
```

```
}
```



Spiral tubes

```
shade=.95
import mycelium3D-2014.cfdg

CF::Background = [b 1]
K=Camera( -25, 0, 50)

startshape Two(Sys())

shape Two (vector18 W) {
       RandomPath(W)[]
       W = RotZ(180, W)
       W = ScaleZ(1, W)
       RandomPath(W)[]
}

shape RandomPath (vector18 W) {

       W1 = RotZ(8, W)
       W1 = RotY( .5 , W1)
       W1 = MoveY(.5, W1)
       W1 = Scale(.985, W1)

       if (fuzzy(.015))     //   fuzzy(N) = (rand() < N)
       {
              W1=ScaleX(-1,W1)
              RandomPath (W1)[s MinSize(.02, W) h .5]
       } else
              RandomPath (W1)[s MinSize(.02, W) h .5]

       TUBE(.75, W, W1, K)[ b 1 a -.0 sat 1 ]
}
```



Dandelion ball

(image in previous page)

```
shade = .7
import mycelium3D-2014.cfdg
K=Camera(0, 90, 1580)

CF::Background = [b -.7 sat .5 h -140 ]
CF::Size = [s 110 ]

startshape Centro (Sys())[b 1]

shape Centro(vector18 W)
{
	SPRITE(65, W, K)[b -.7 a -.15]
	count=80
	loop i=1,count/1 [] {
		kz=asin(-1 + 2*i/count)
		W1=RotX( (phi*i) , W)
		W1=RotZ( (kz) , W1)
		W1 = MoveY(35 , W1)
		W1=Scale(.15, W1)
		Flo (400,   W1)[]
	}
}

shape Flo (Q, vector18 W)
{
	W=Stiff(.95,W)
	loop i=Q [] {
		W=RotY(i*phi, W)
		W=RotZ(-90, W)
		W=Scale(i/Q+.1, W)
		W=MoveX((i)/30, W)
		Hair(W)[h -(20*i/Q)]
	}
}

shape Hair(vector18 W)
{
	W1=W
	W=RotZ(rand(0,25), W)
	W=MoveY(16, W)
	TAPE(1.5, W, W1, K )[ a 1 b 1 sat 1 h 60]
	TAPE(1.7, W, W1, K )[ a -.5 b -1 sat 1 h 60 z-.1]
	Hair(W)[s MaxAge(6, W)]
}
```

3D in context free art

Context Free Art is a 2D drawing program, this means that we can not perform the typical operations of 3D modeling program such as rotate a view in real time or choose a rendering algorithm

3D context-free means having a 3D geometry that will be represented with the "painter's algorithm". Lights and shadows are therefore also to “draw".

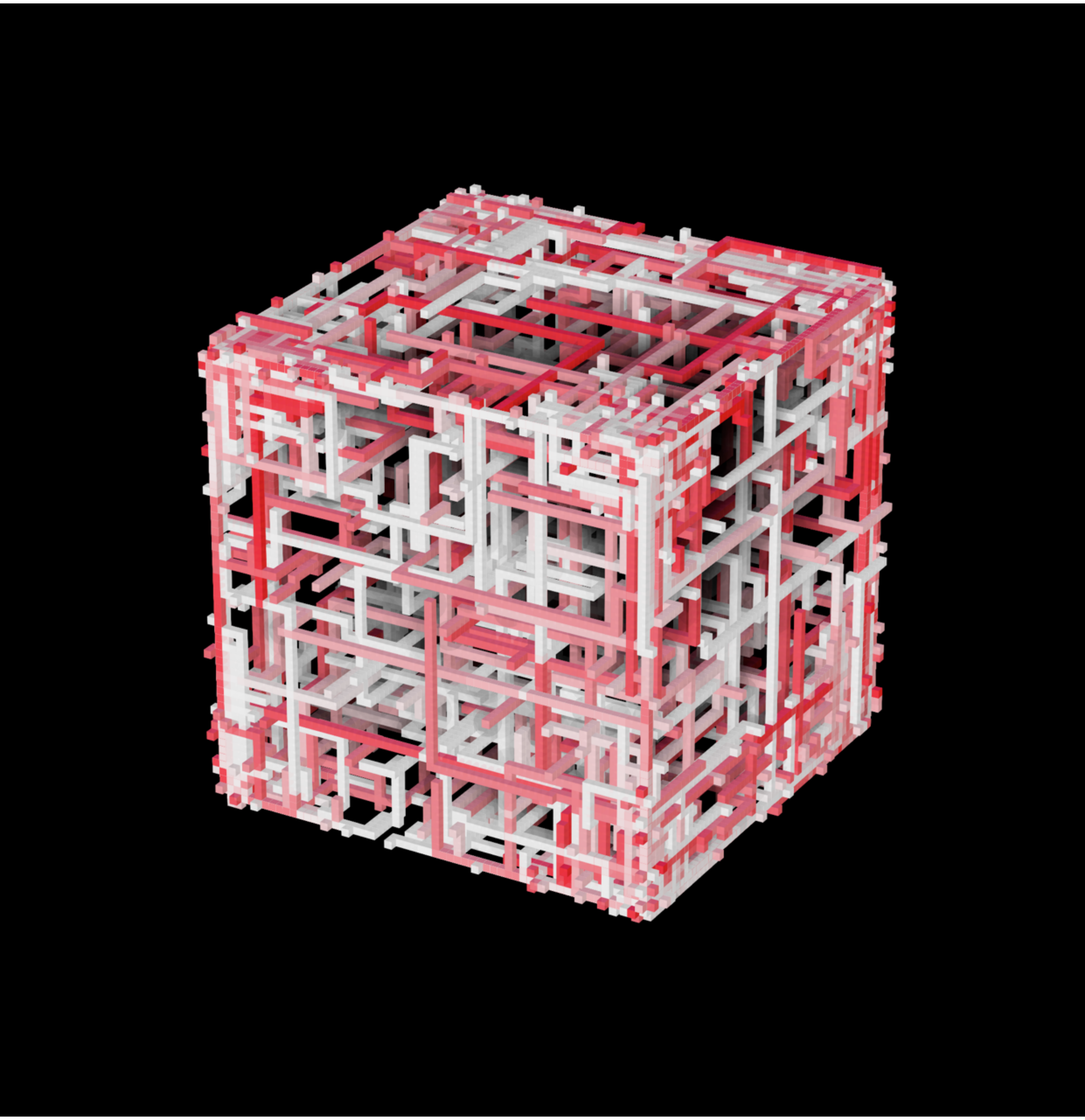
These tools, which seem very limiting for traditional uses of 3D graphics can be rather powerful and unique in applications that require computation of a large number of elements, high-speed rendering, or massive use of transparencies.

Possible applications are for example: generative graphics, fractals, simulations of natural textures, animations of complex structures.

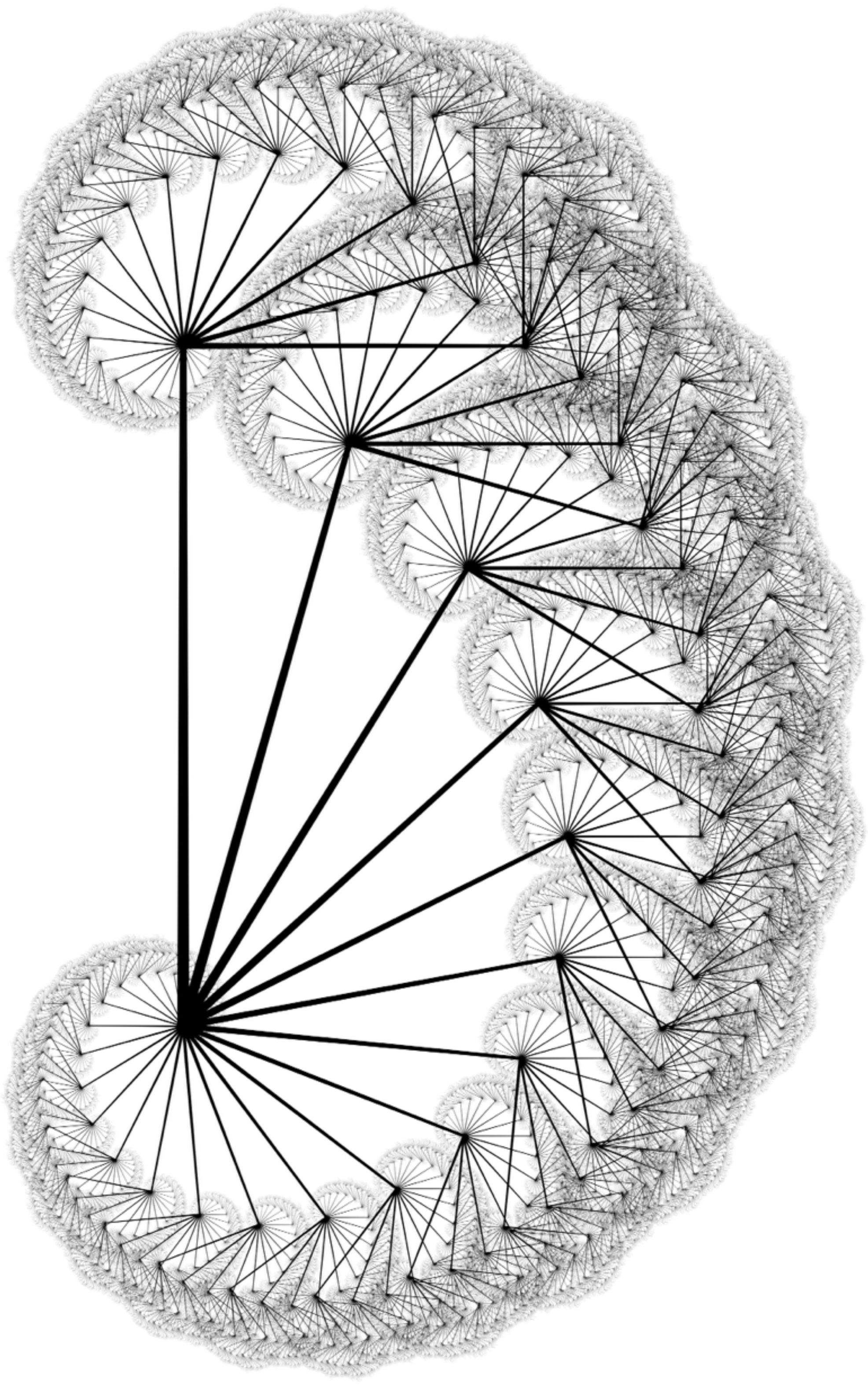
Painter's algorithm and ambient occlusion

The painter's algorithm puts the elements to draw in a row, from the most distant to the closest and diligently represent them overlapping.
It is not particularly optimized but has the unique feature to manage infinite levels of transparency. The 16 bits channel of opacity of CFA enhances this feature: it allows, for example, to make very expressive effects of shadows and sometimes similar to so called "ambient occlusion".

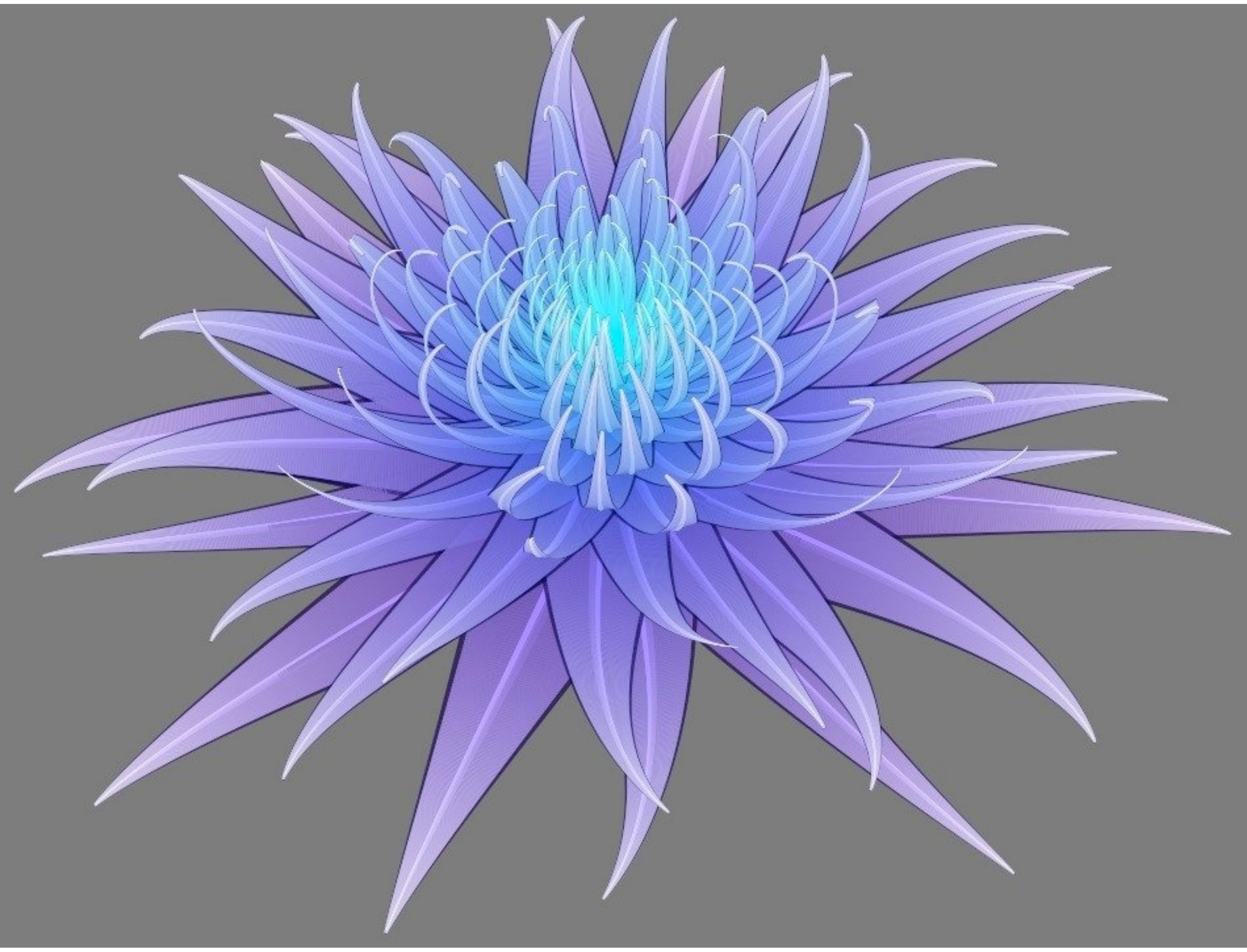
Images created with 3D Mycelium



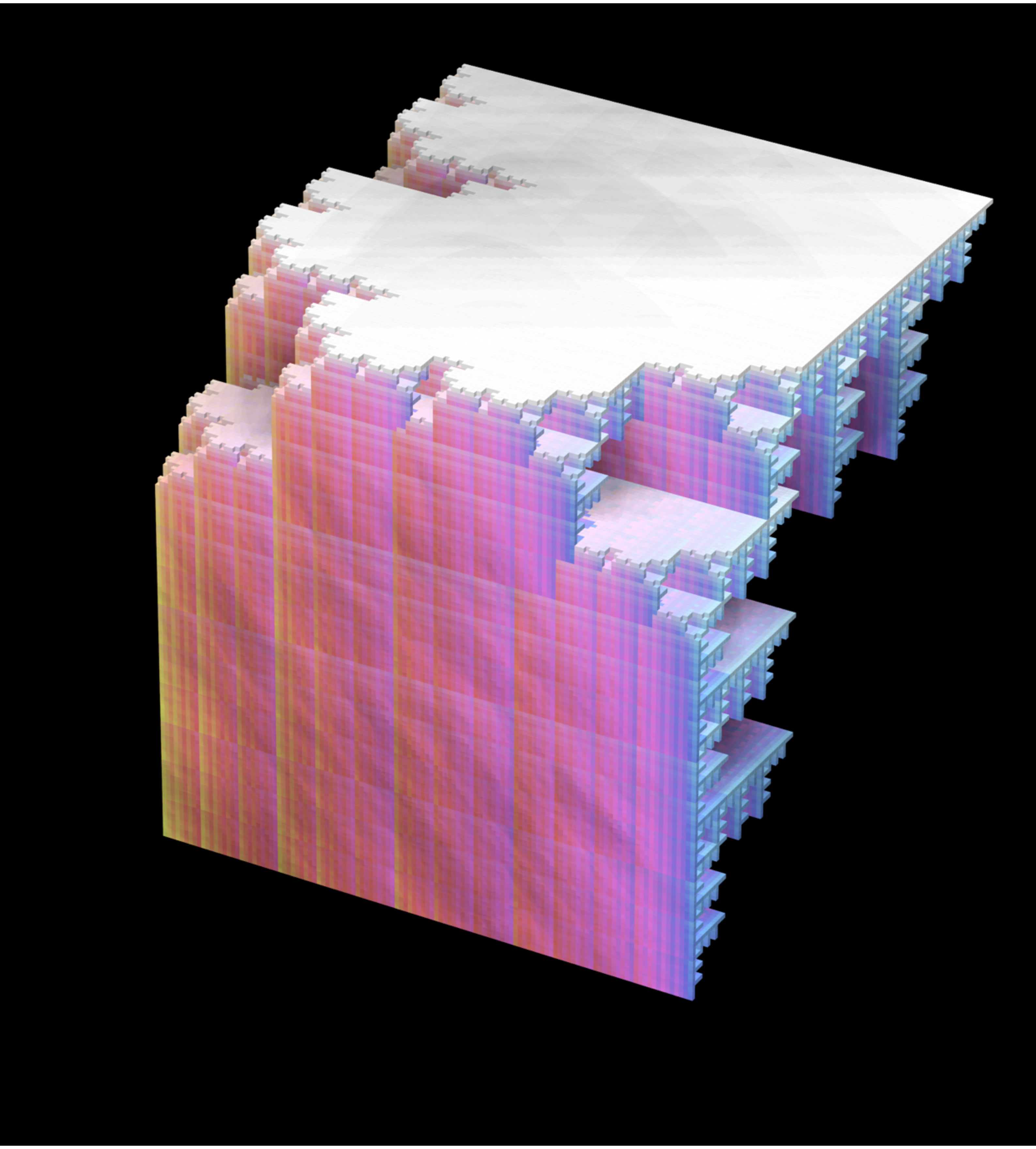
Ambient occlusion effect is achieved by superimposing translucent Sprites



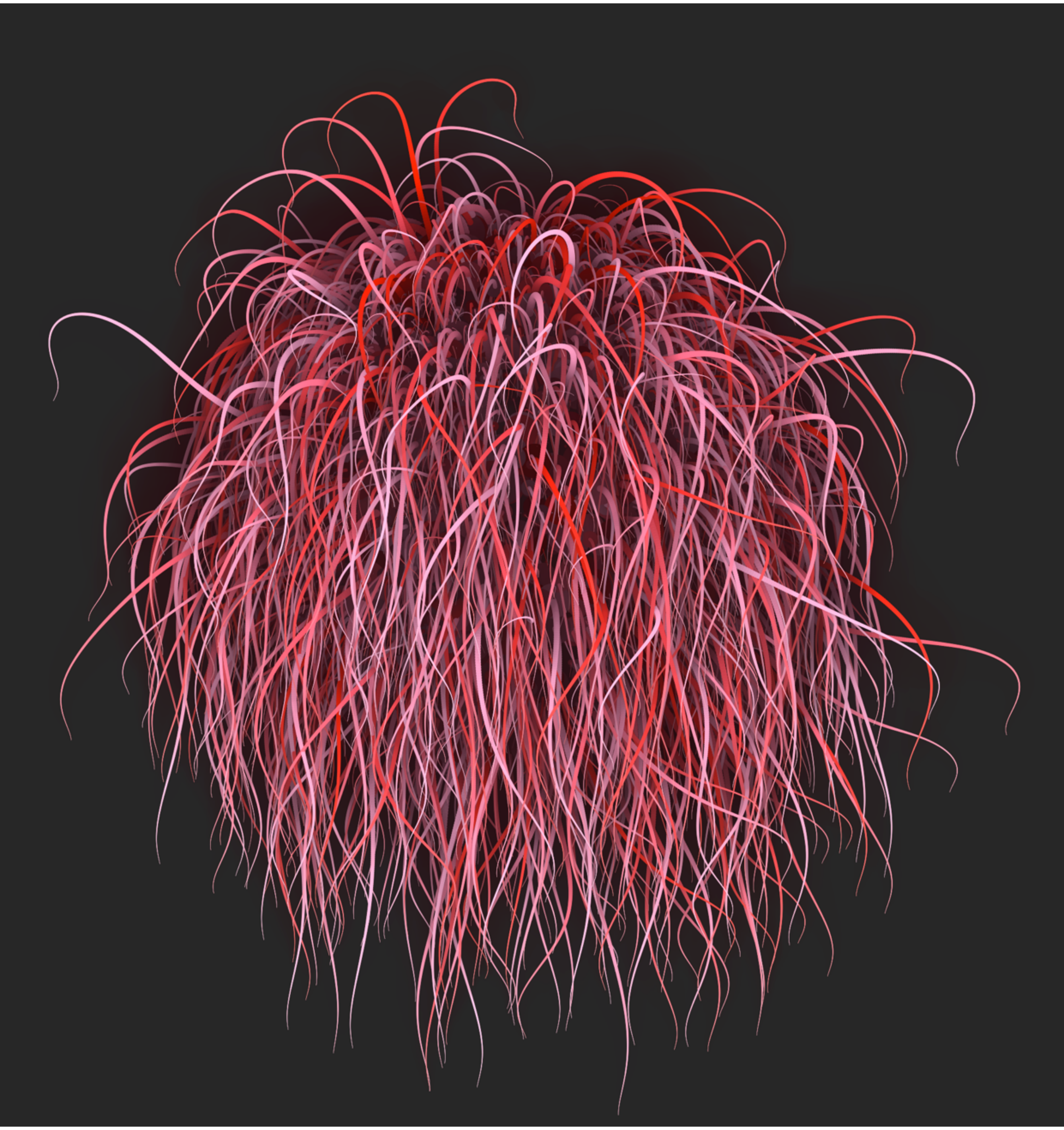
Lines draw a 2D fractal.



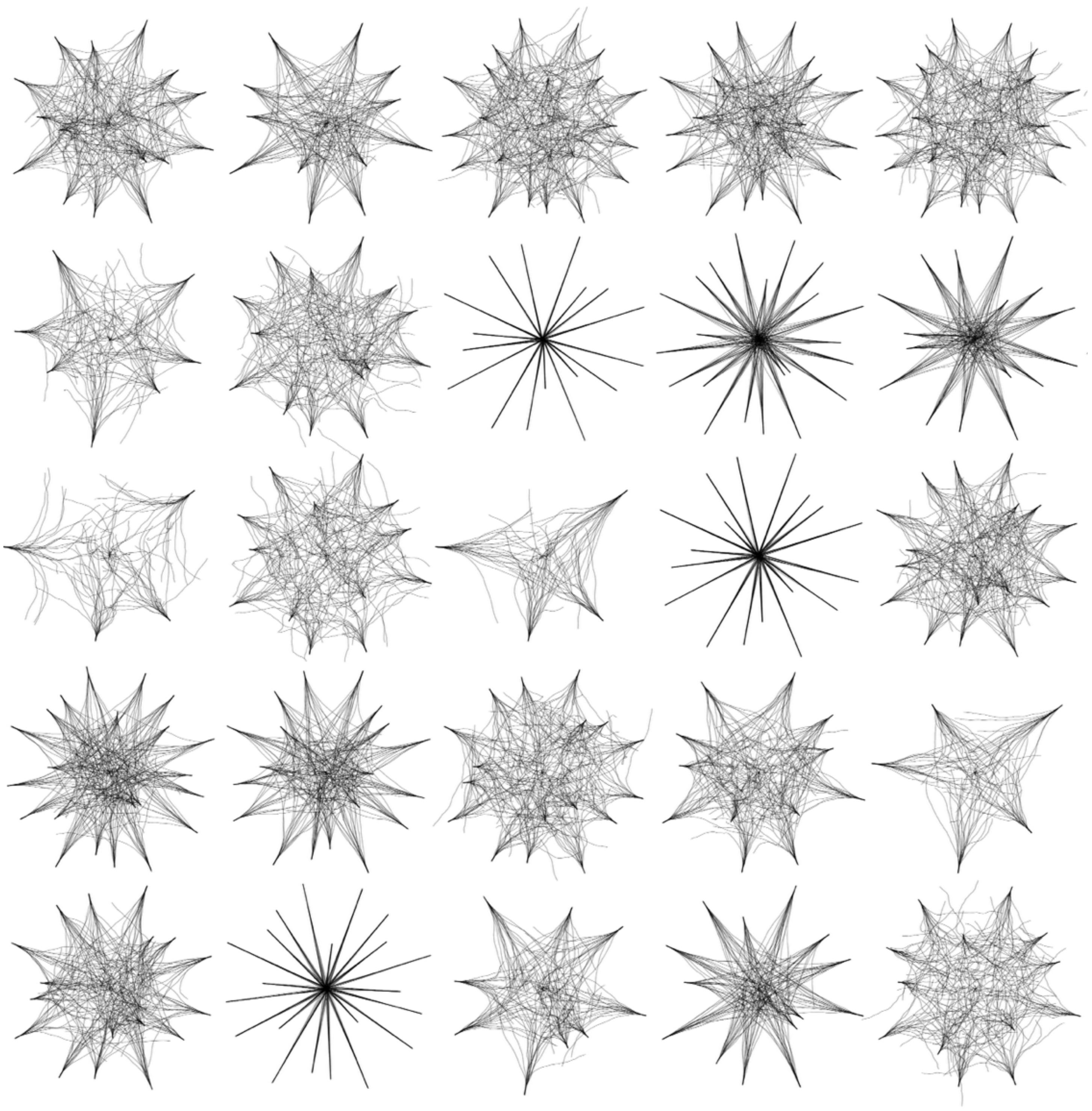
Natural structure



3D Fractal (Menger cube generalized)

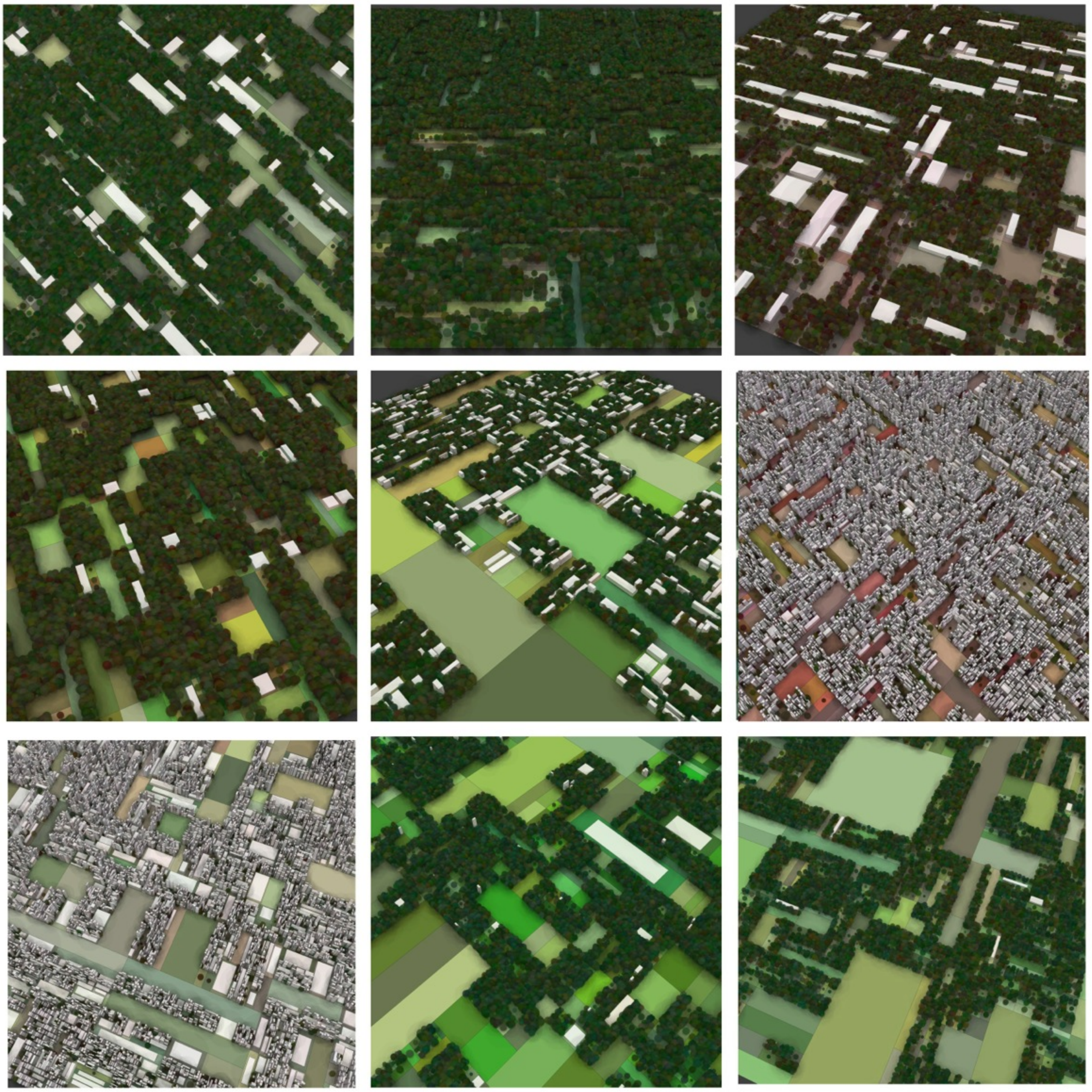


10.000 hairs / 1.500.000 shapes



© arch. Federico Donelli 2014
Three-dimensional stars based on golden angle phi.

Natural simulations



© arch. Federico Donelli 2014
Landscapes generated with a single iterative algorithm.

Creative common license

Standard license

Different licenses