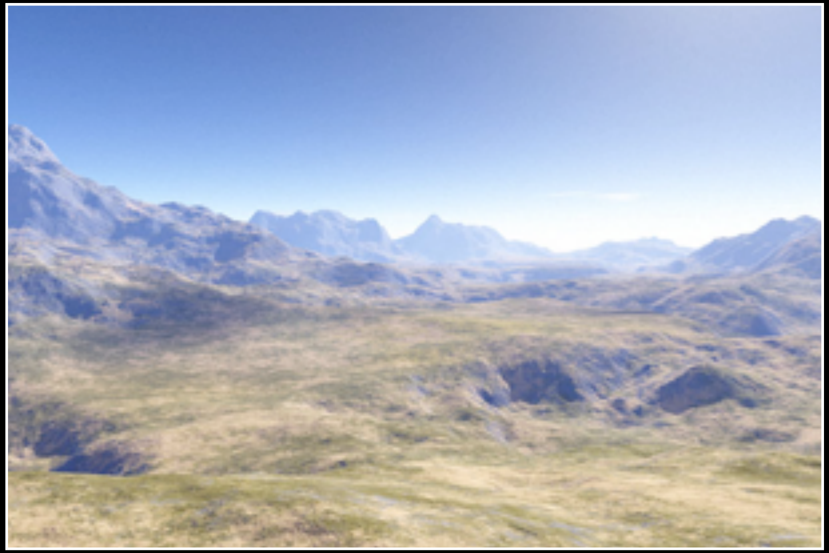
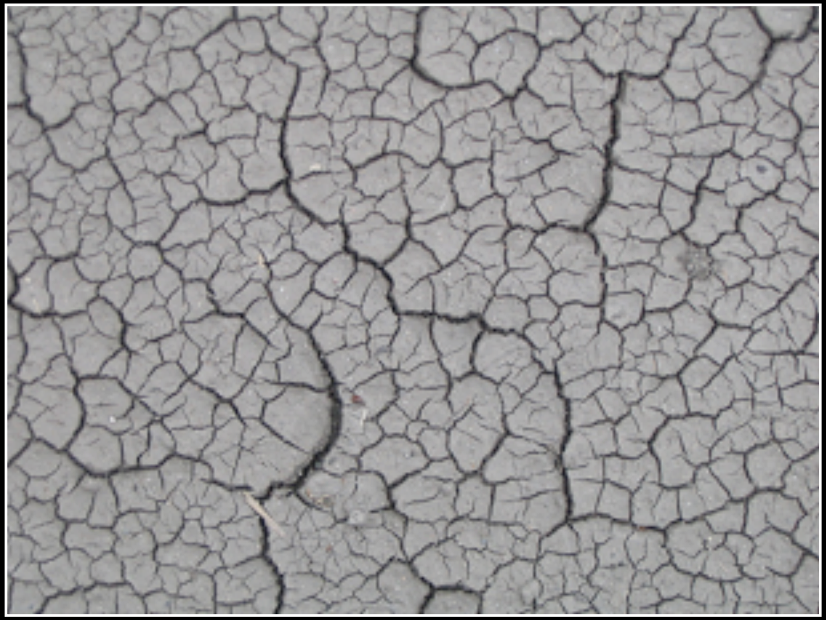
Space filling: A new algorithm for procedural creation of (some) game assets

Paul Bourke
iVEC@UWA

Random and/or fractal generators

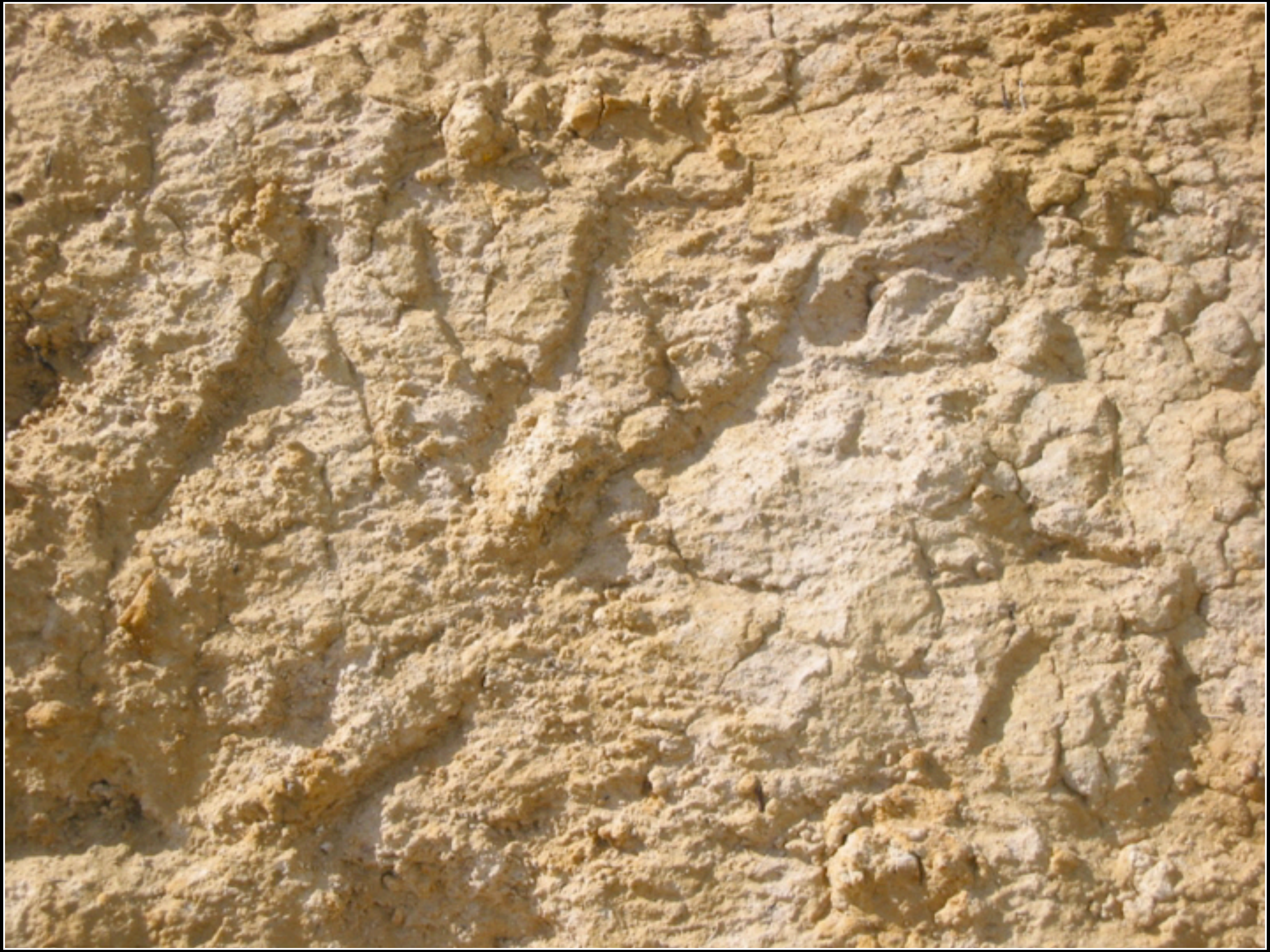
- Parametric generation of form and/or texture.
- Random and fractal generators have been widely used to create game assets, objects for virtual environments, and in the animation/movie industry.
- Well know examples include
  - Perlin noise for clouds
  - landscape and terrain generation
  - fractal plant creation
  - ocean waves
  - texture generation





Example: fractals in nature

- Why is this fractal?
- Why is this useful for parametric models?

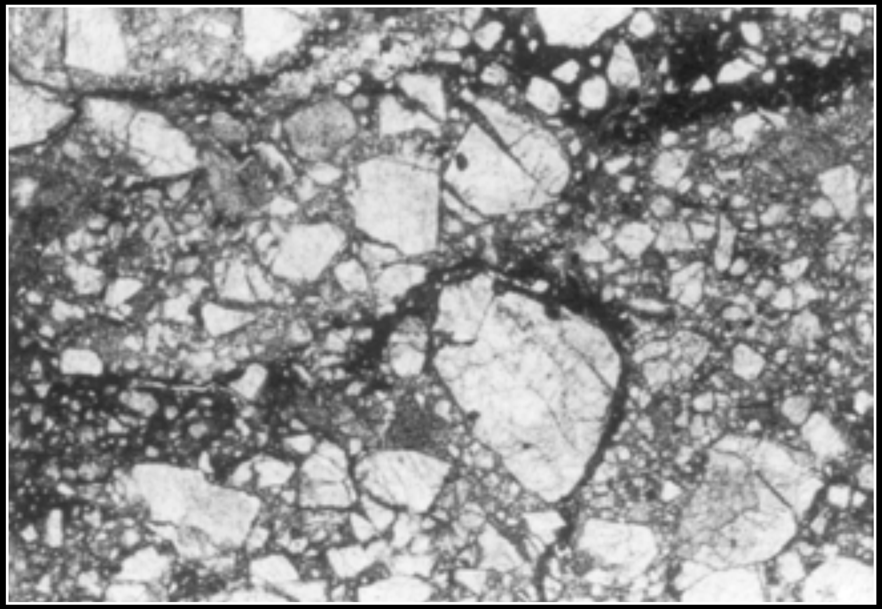


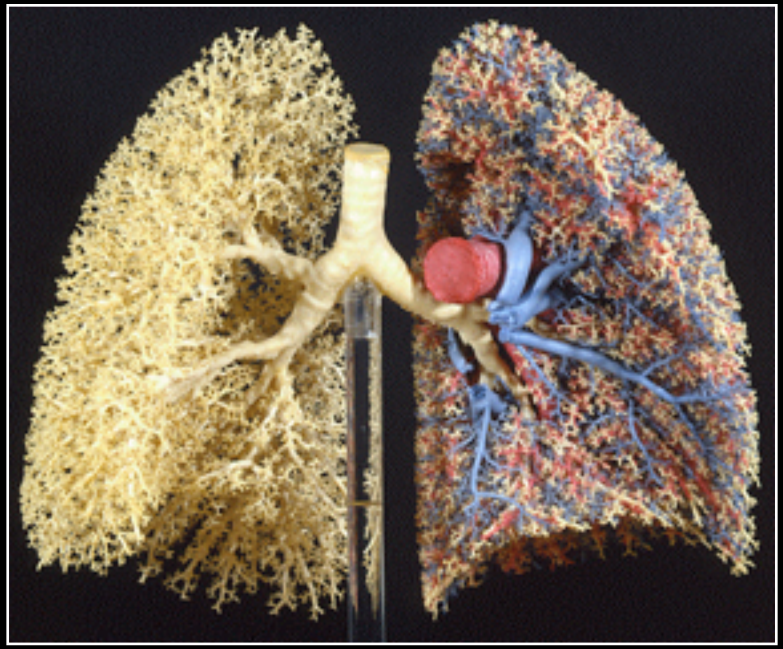
Advantages

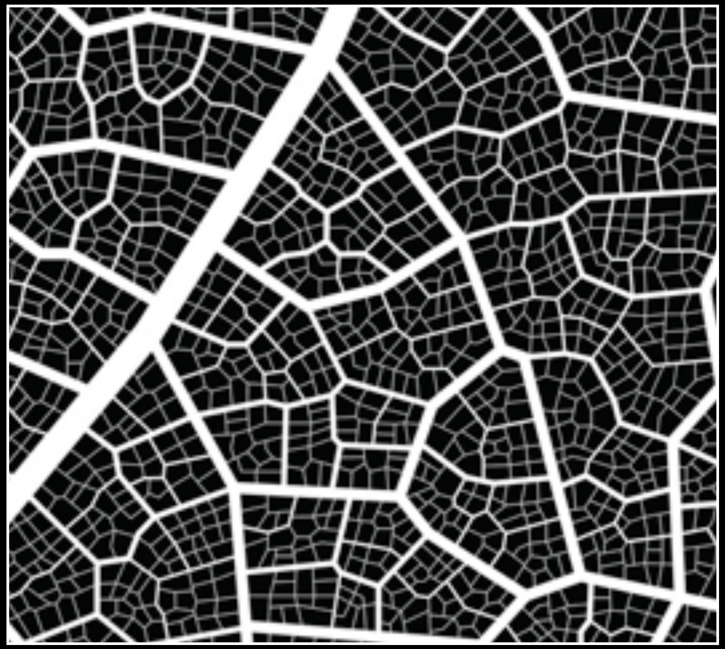
- Reduced delivery payload, generate rather than store.
- Variation. Random processes can result in different environments each time.
  eg: clouds, mountain ranges, ocean waves.
- Fractal processes result in image/geometry that appears natural.
  Nature is often self-similar across scales.
- Can often be formed with infinite range or seamlessly tile.
- Can usually be formed at multiple resolutions allowing control of performance and level of detail.
- Algorithms are generally very fast and based upon simple iterative or recursive rule sets.
  Simple rules that generate complicated structures.

Models for space filling

- A range of natural phenomena have the property of being space filling.
- As with all fractals this is often over a limited range of scales.
- Results in high fractal dimension.
- Space filling is a well known phenomena in geology and biology.

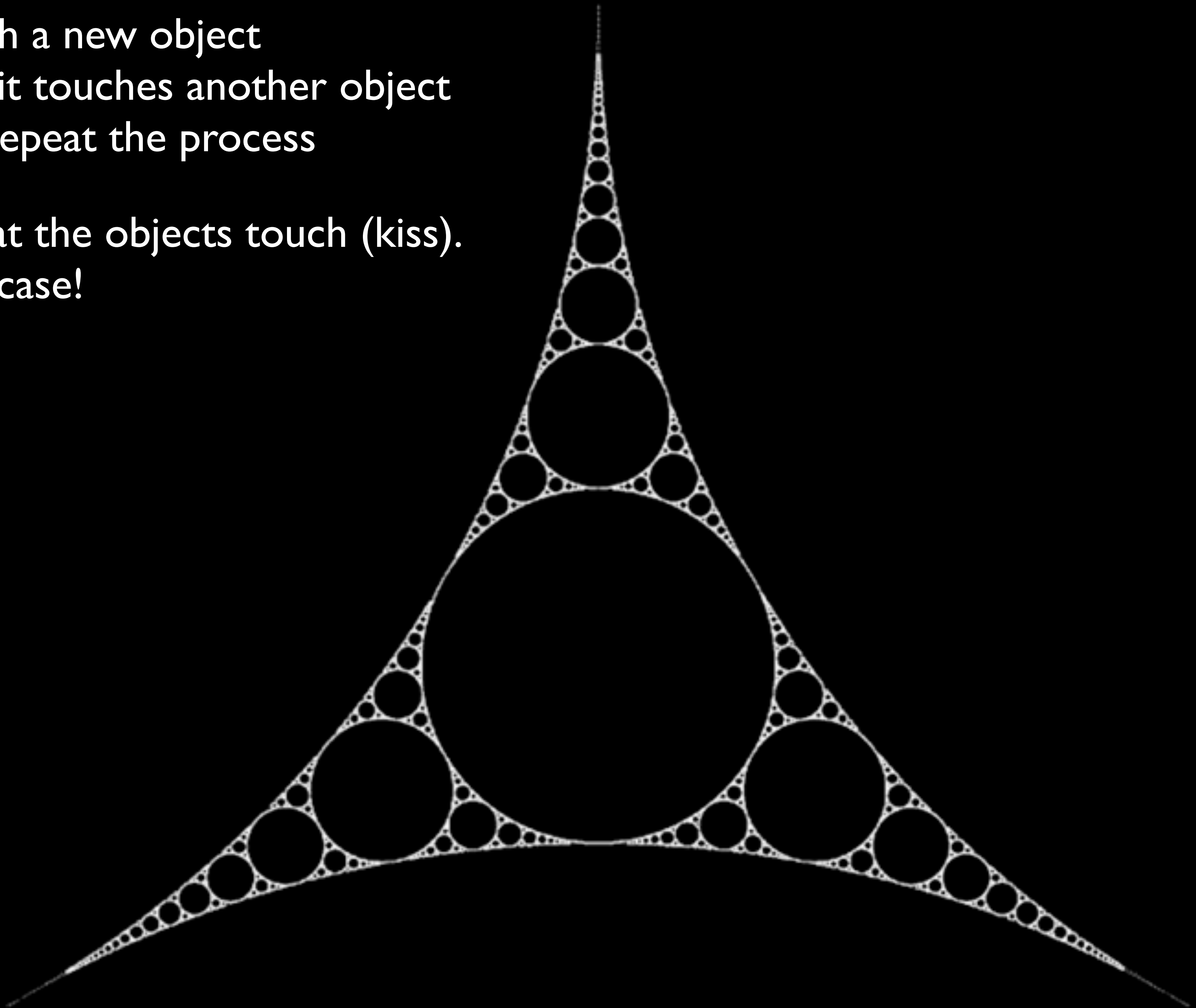






Previous model: Apollonian packing

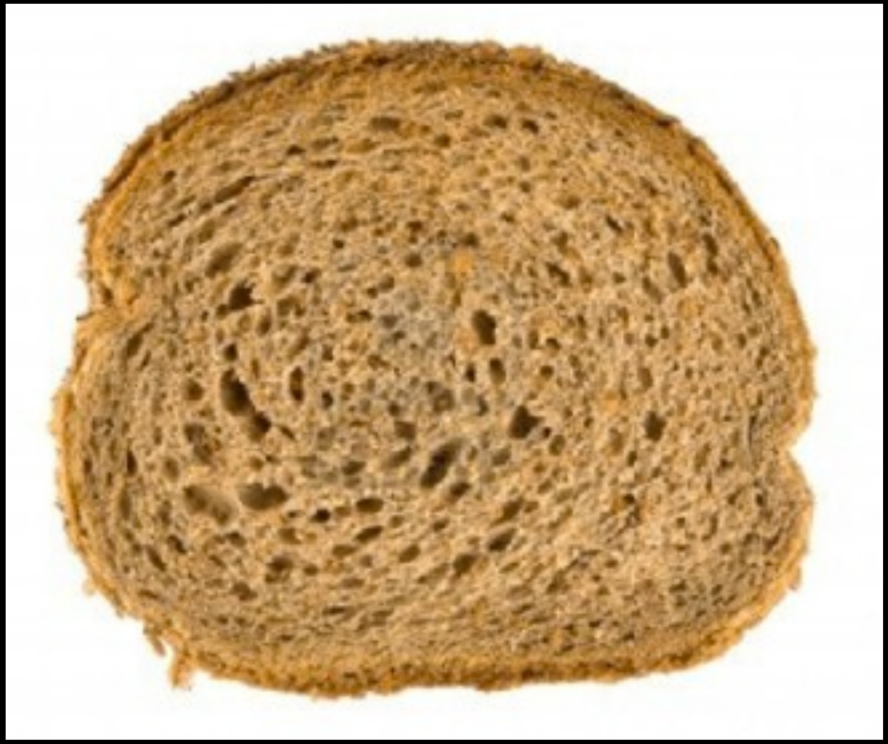
- A previous model for creating these space filling has been so called Apollonian packings.
- Algorithm (or variations)
  - find some empty space
  - seed that location with a new object
  - grow the object until it touches another object
  - place the object and repeat the process
- Key characteristic is that the objects touch (kiss).
  Clearly not usually the case!



Non-Apollonian packings

Bubbles in a plate

Rocky river shore

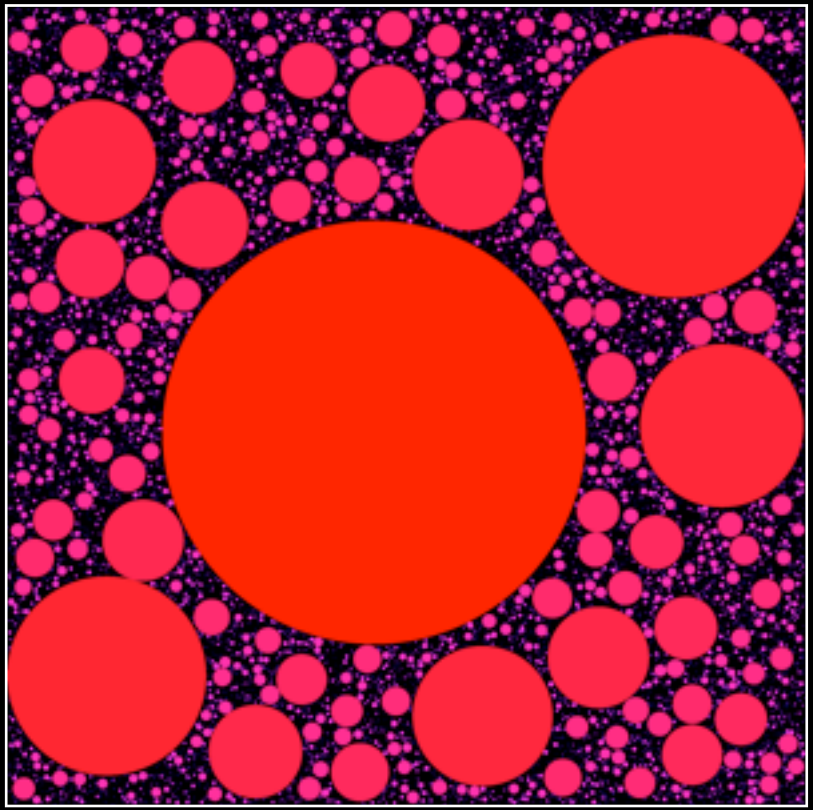

Bread air pockets

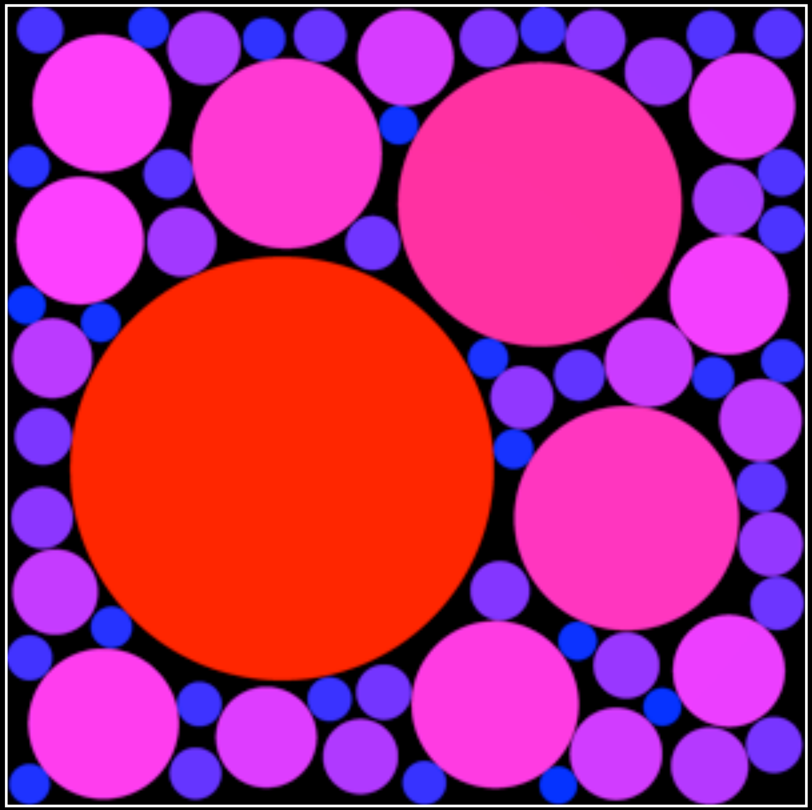
Lily pond

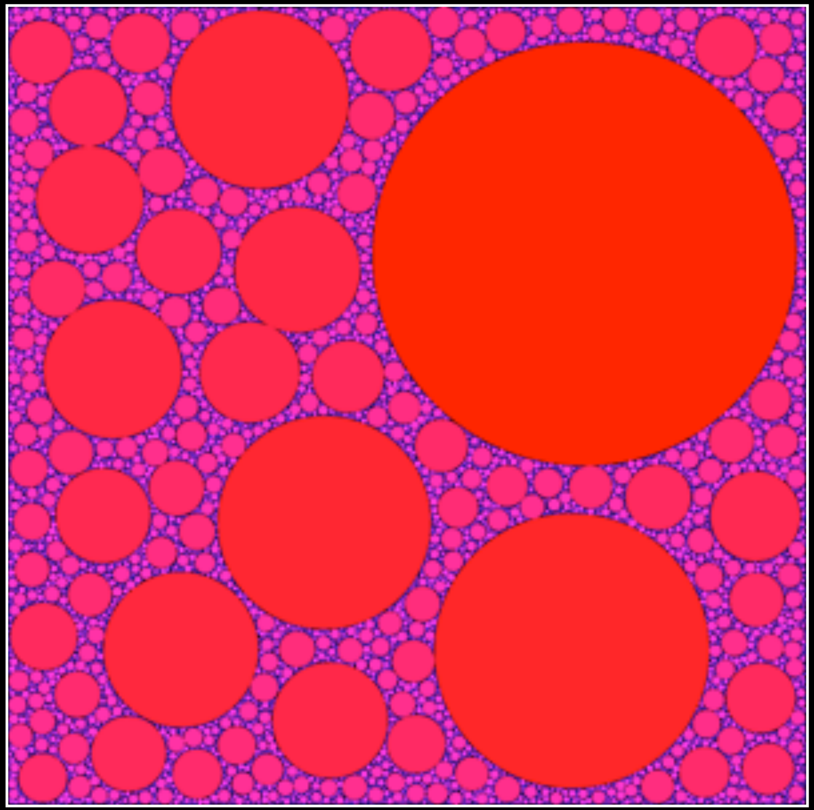
New model for such processes

- Algorithm:
  - randomly find a place to locate the next object at a particular size
  - add that to the database of shapes
  - reduce the size of the object
  - repeat until some object count is reached or space is filled to some pre-specified level

- The trick ("magic") is how to reduce the size of the object such that the process never stops, fills space (without gaps), and is self similar across scales.


If size is reduced to fast space is not filled


If the size is not reduced fast enough there will be no space for the next object of the intended size


If reduced exactly right then space is filled and fractal.

The magic size relationship

- Ao is the area of the first shape and g() determines the area reduction on each iteration

$$A_0, A_0\,g(1), A_0\,g(2), \ldots, A_0\,g(n), \ldots$$

- The total area A for space filling is then given by

$$A = A_0 \sum_{i=1}^{\infty} g(i)$$

- This is satisfied by the Reimann Zeta function that is known to converge for c > 1

$$g(i) = \frac{1}{i^c}$$

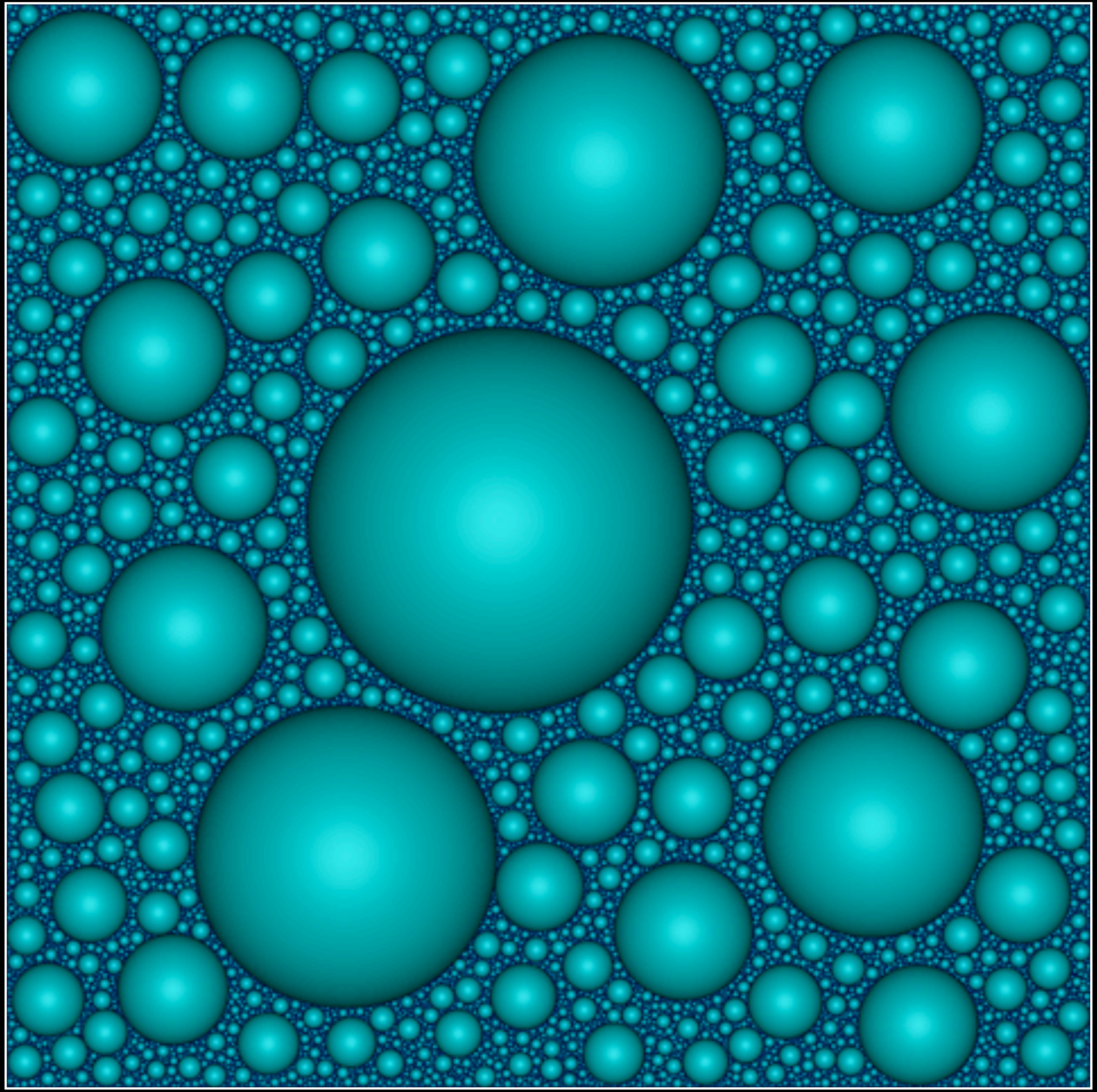
- Can apply to other dimensions, in one dimension use length, for three dimensions use volume.

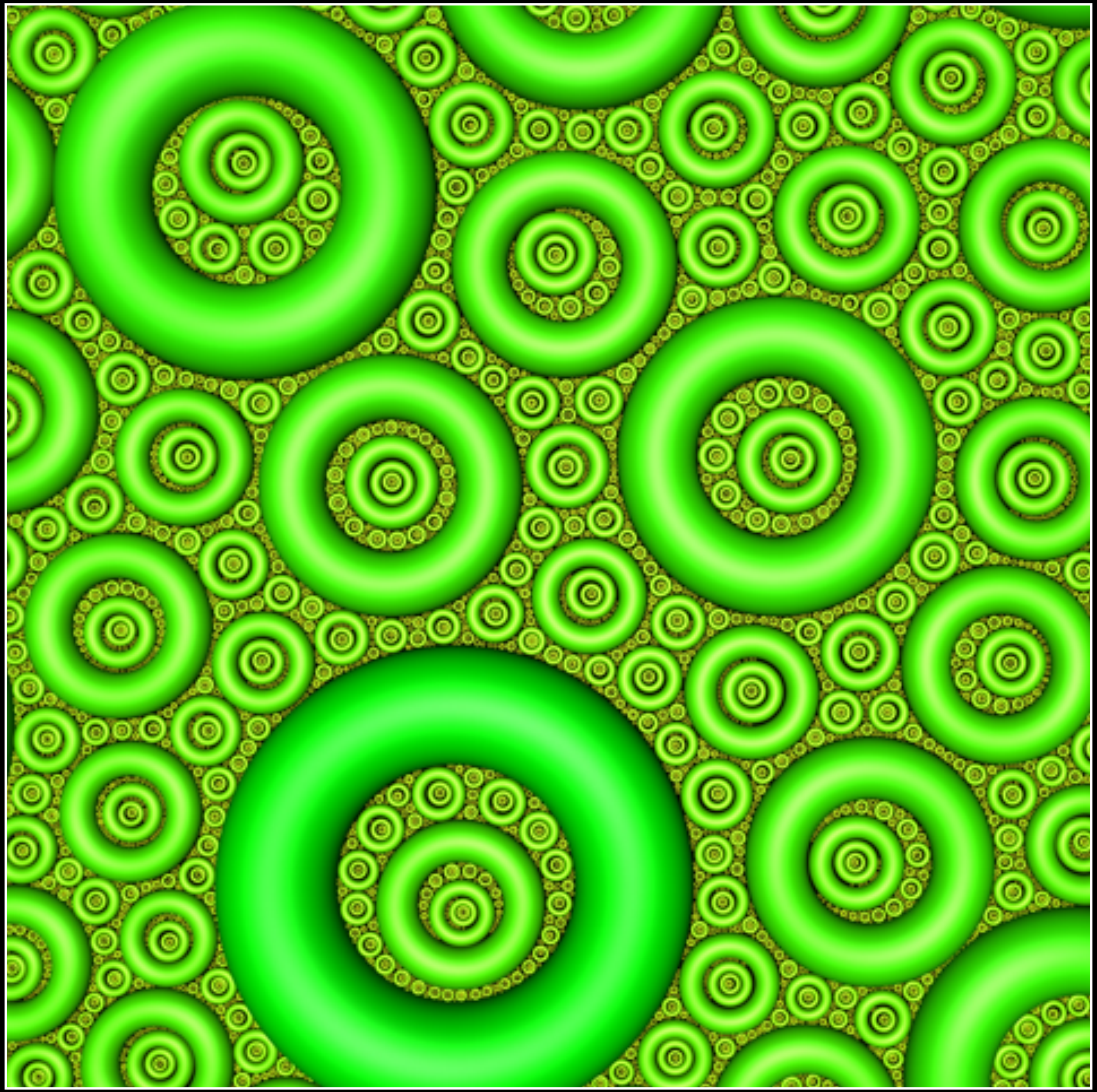
- Turns out one can control fractal dimension “d” of the result, where D is the embedding dimension (1,2, or 3)

$$d = D / c$$

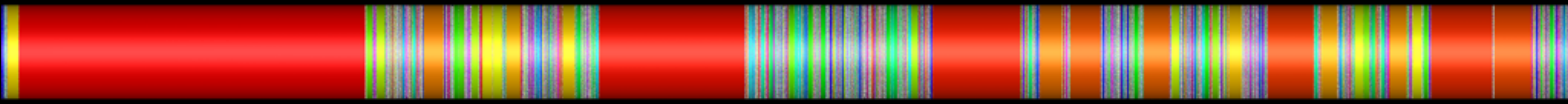
Examples in 2 dimensions

- Large rocks - Small rocks - pebbles - sand

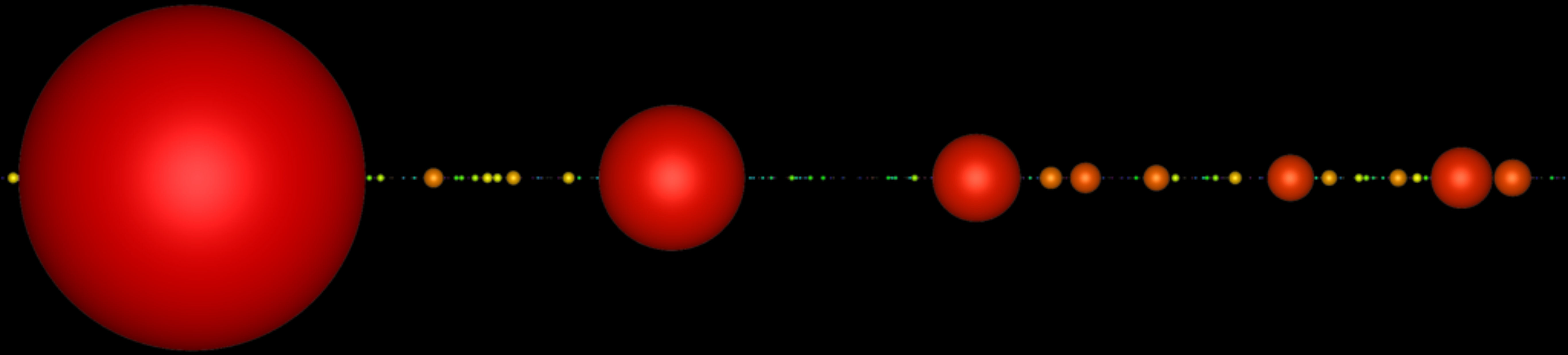




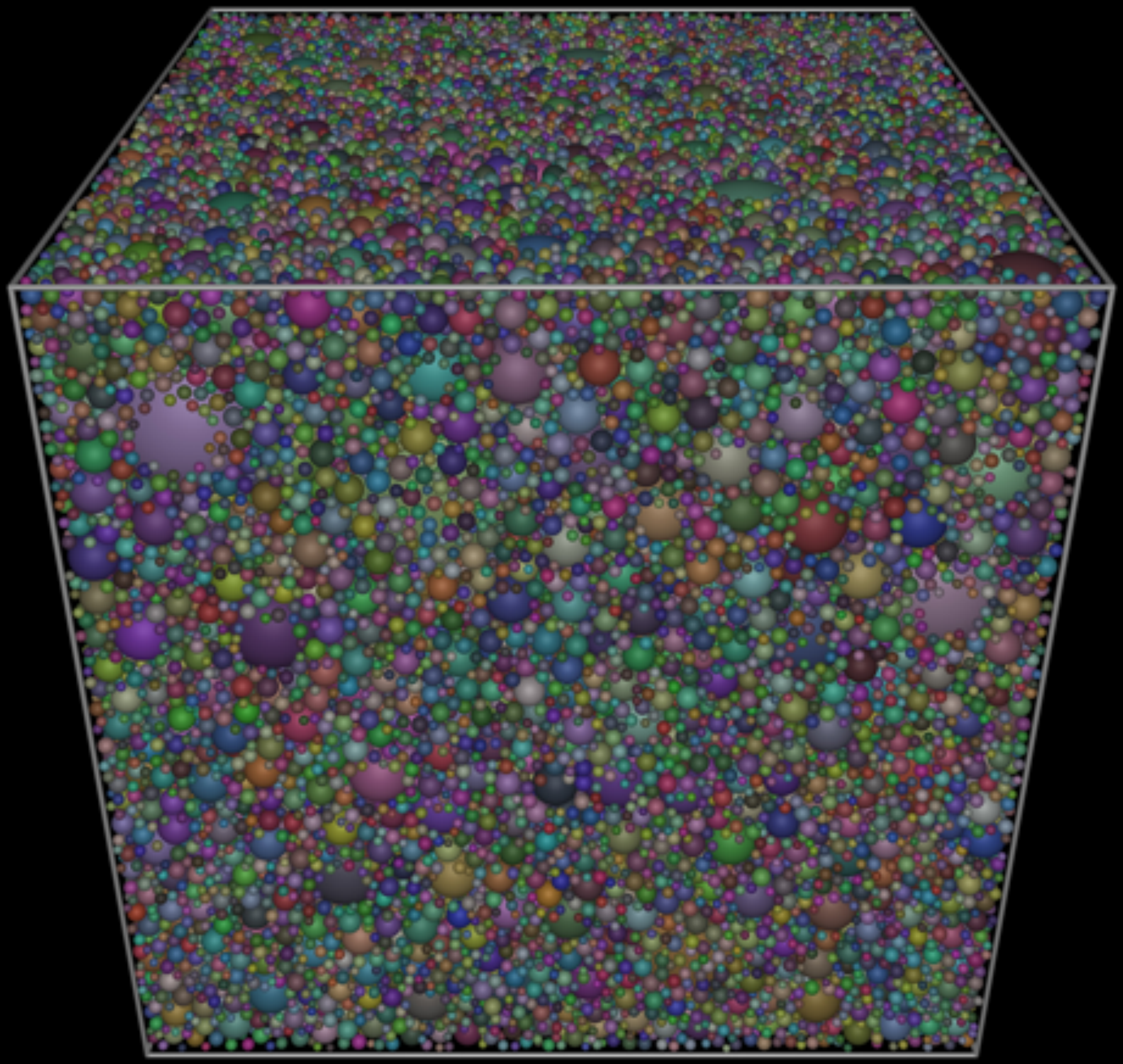
Example in 1 dimension



- Not many different shapes possible, both for the filling shape and shape to be filled can only be line segments.
- ... or curves.

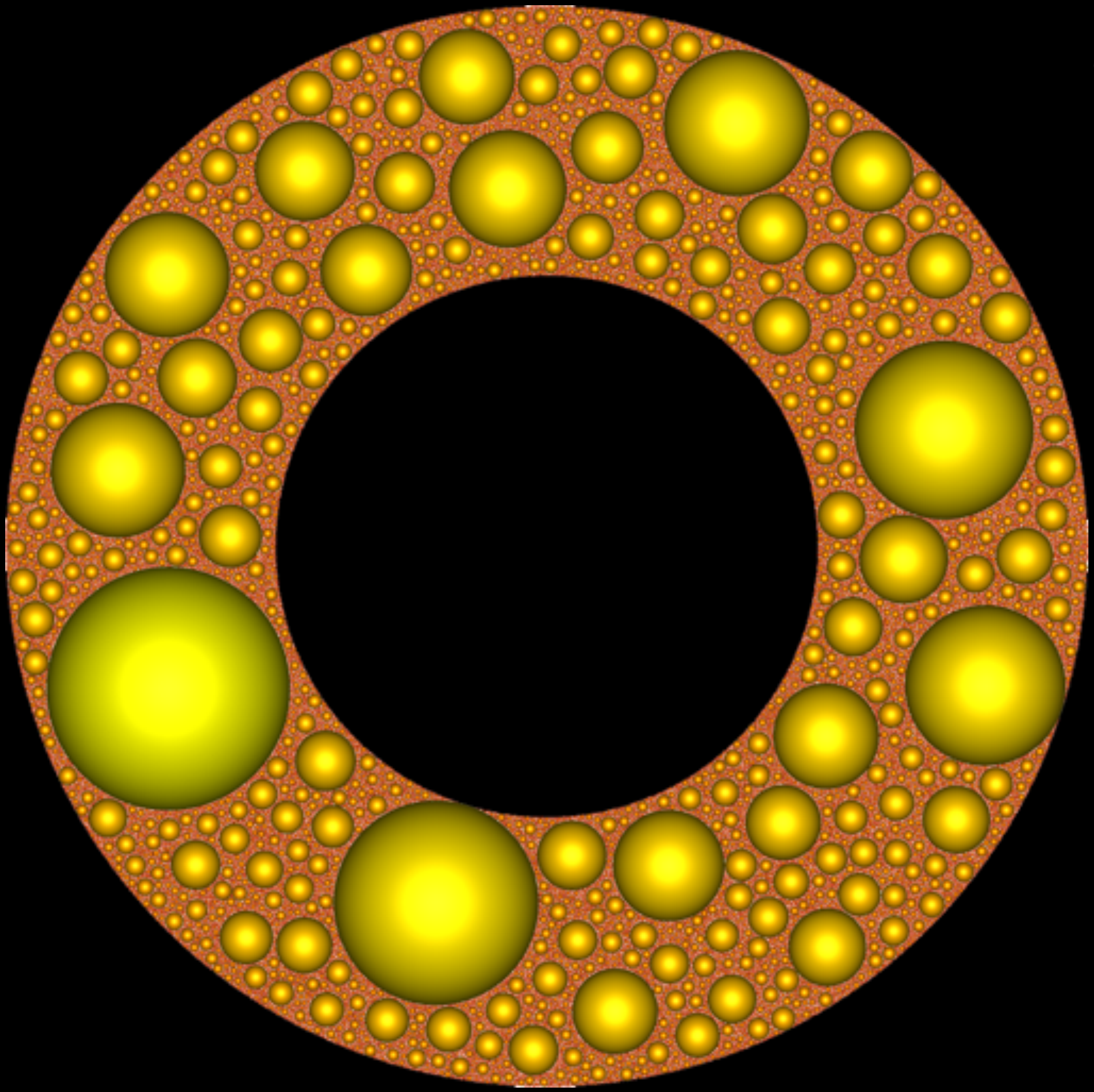


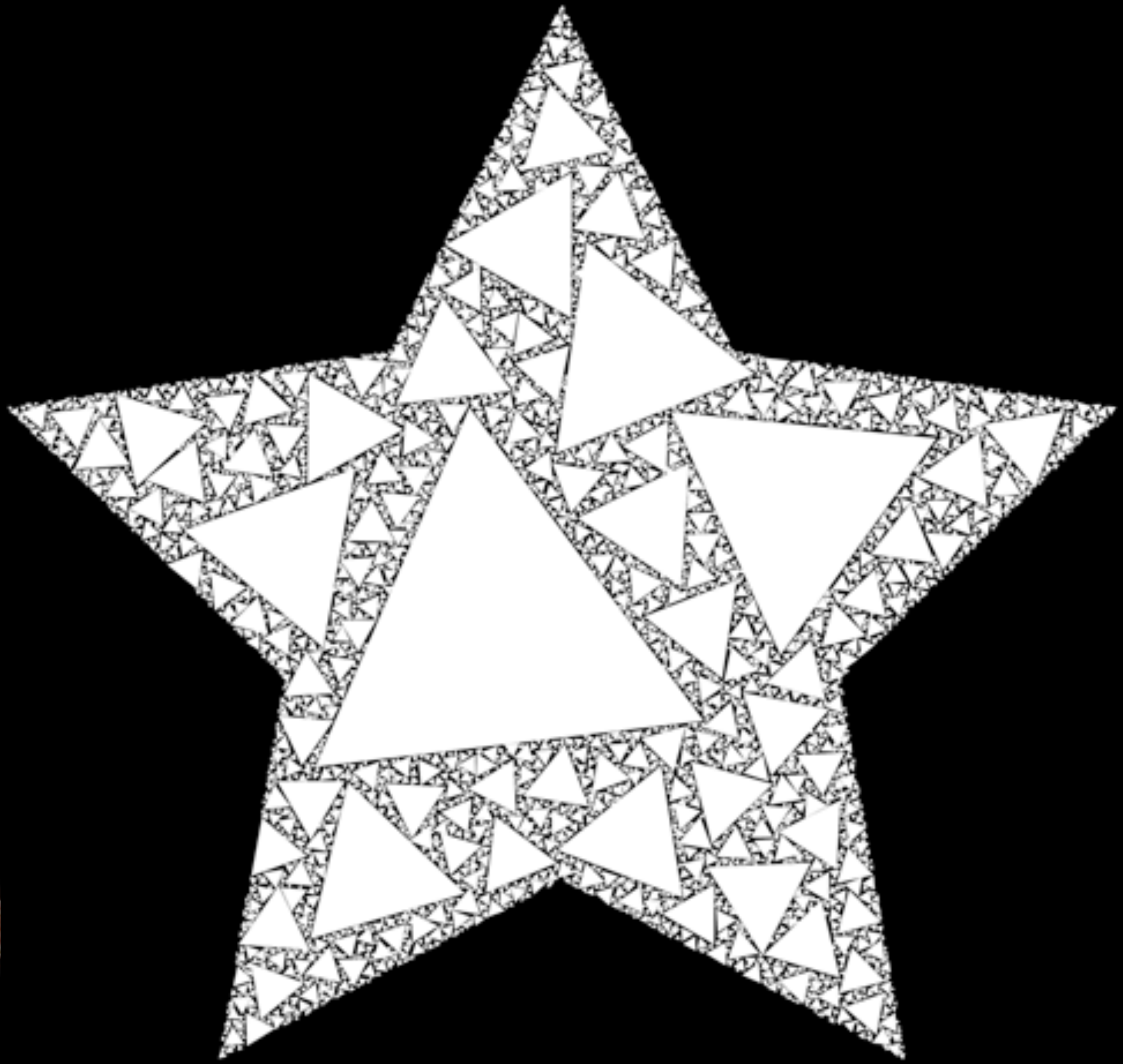
Examples in 3 dimensions


100,000 spheres

10,000 torii

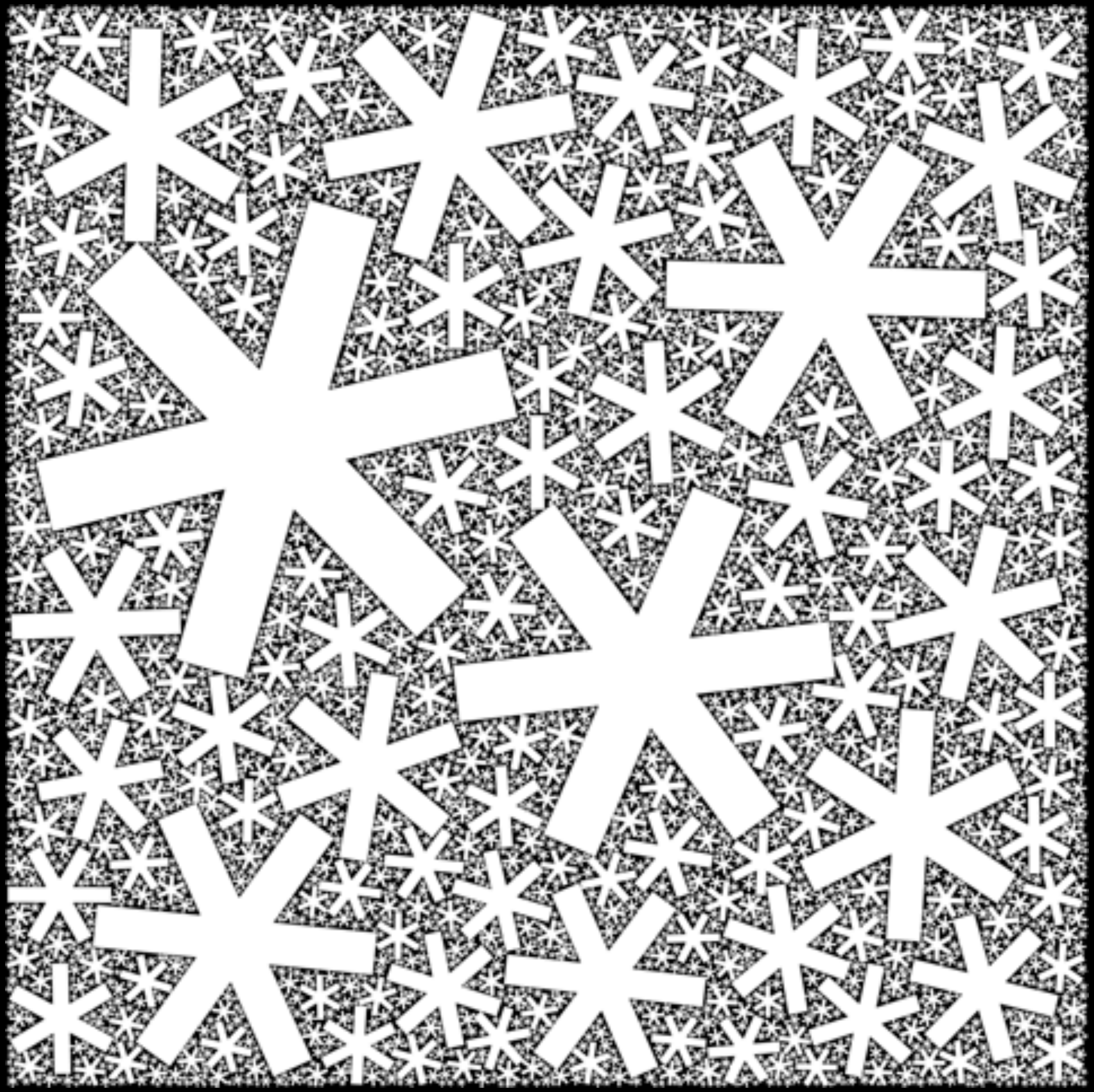
Filling shapes

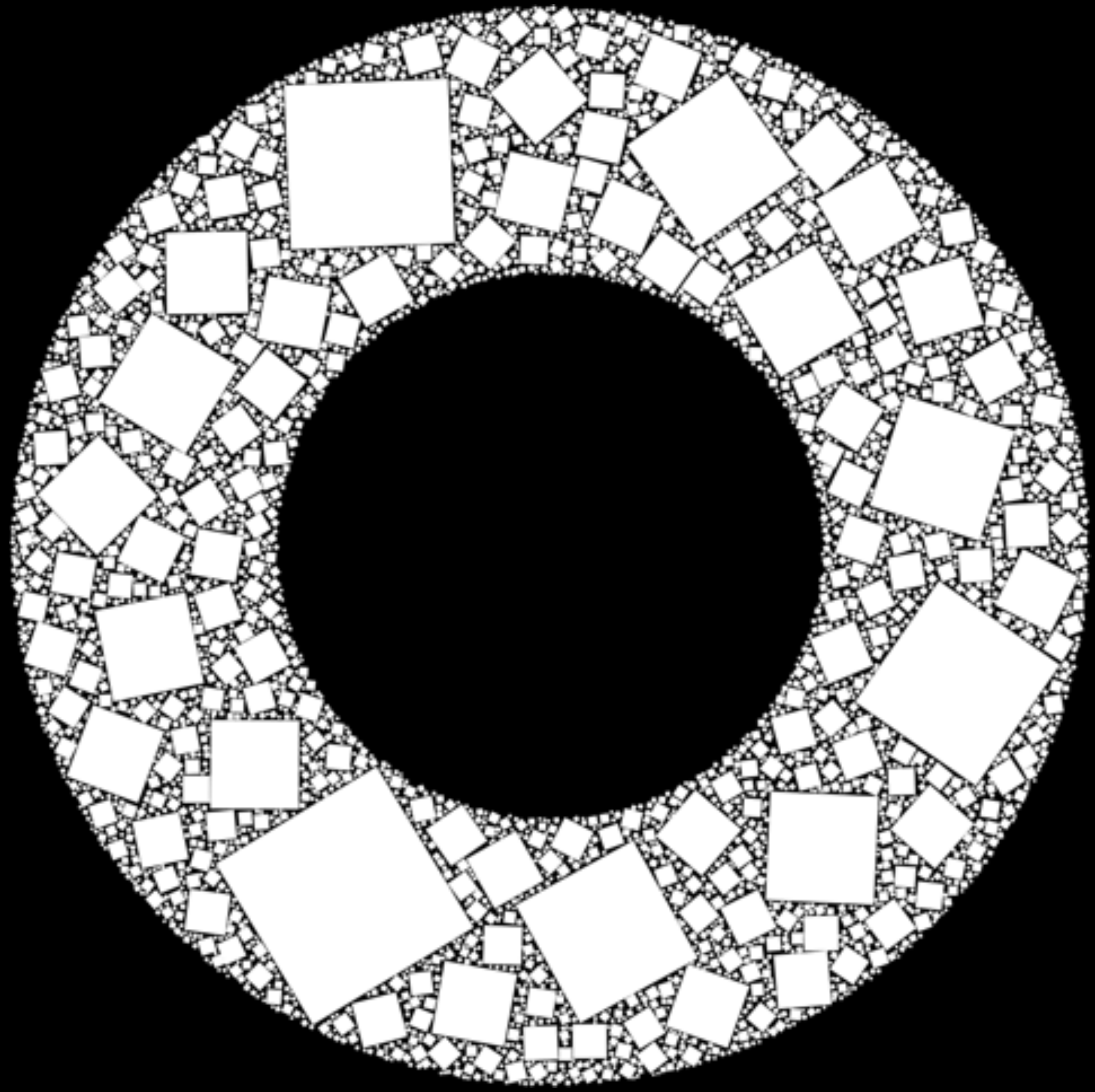




Characteristics

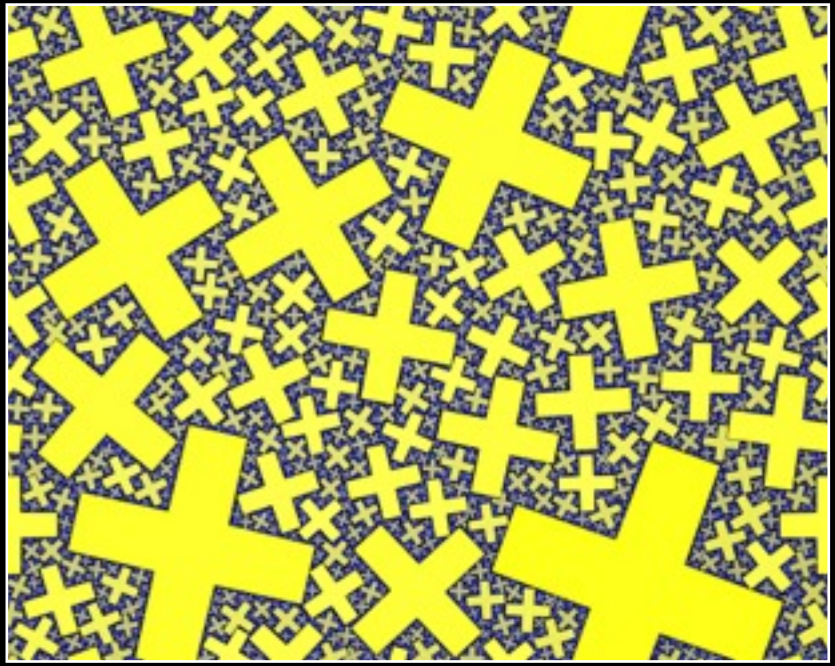
- Any shape can do the filling.
- The formulation only depends on the area not the shape itself.
- Mixed shapes can do the filling, again, the algorithm depends only on the area of the shape.

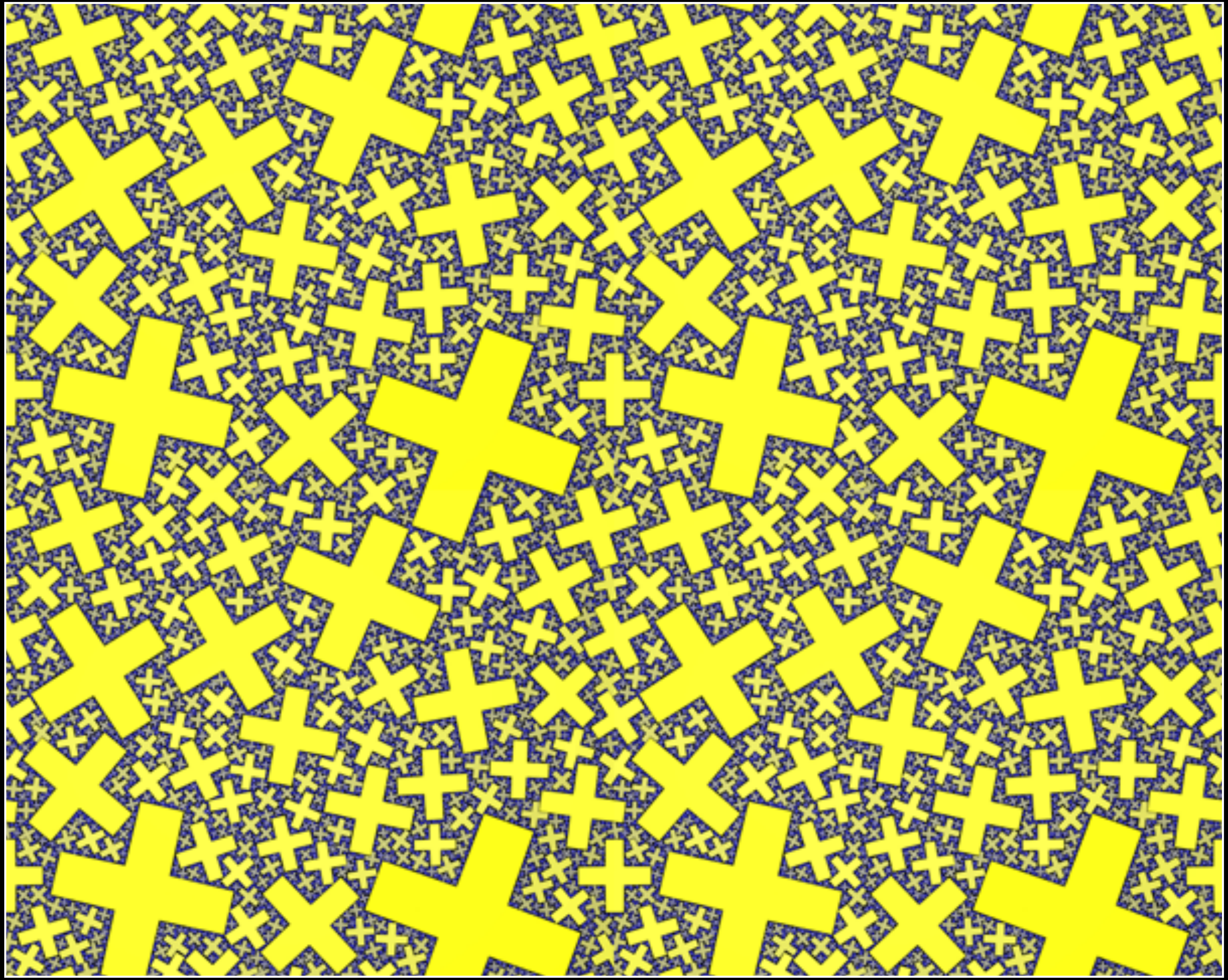




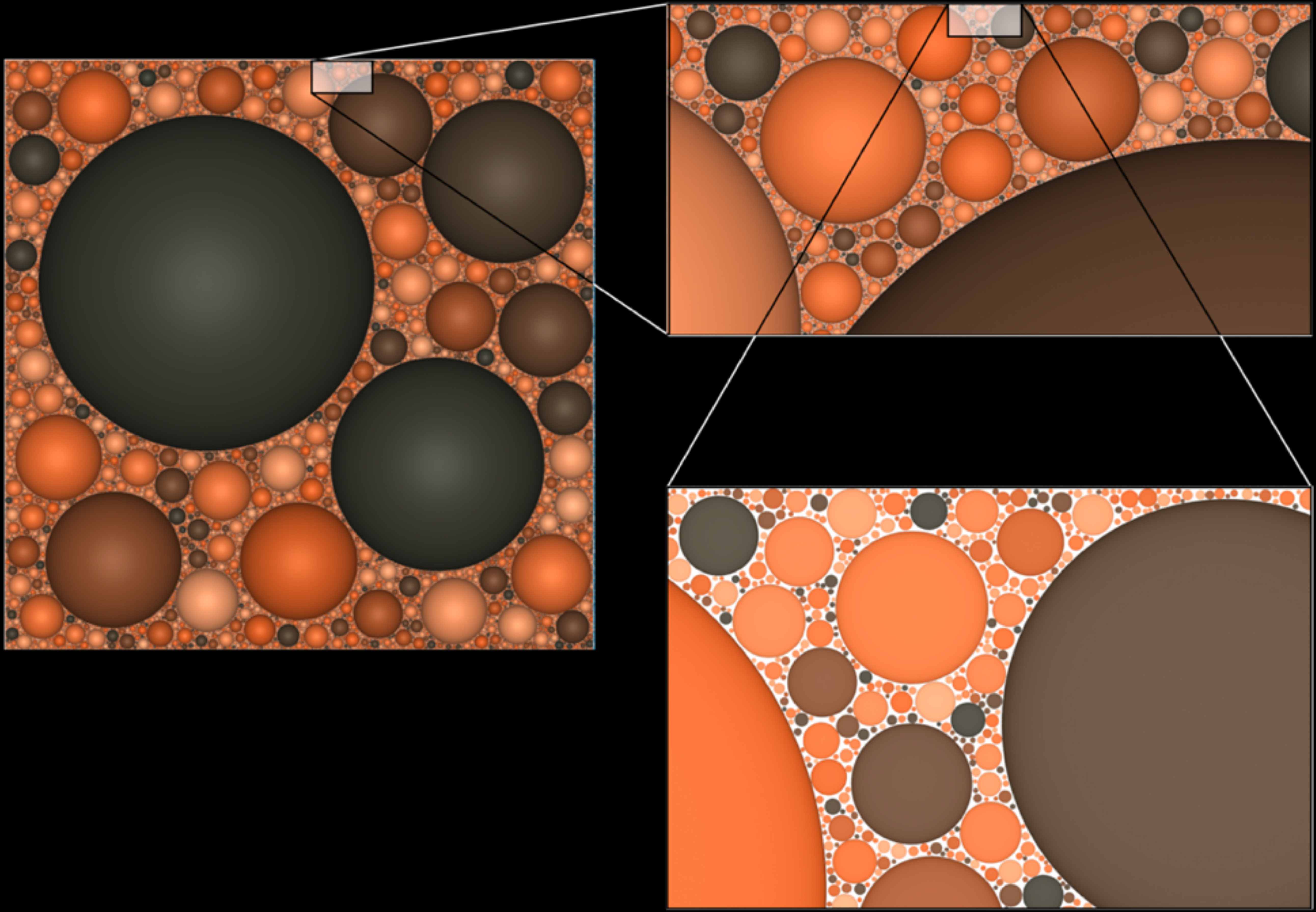
Characteristics

- Tileable rectangles can be filled by using periodic boundary conditions.
- Requirement for seamless textures.


Tile shape


2x2 tile

Self similar (fractal)



Conclusion

- Developed another algorithm for creating parametric geometry for virtual worlds.
- Supports the characteristics one might expect
  - variable levels of detail, to infinity even
  - produces identical or variation of imagery depending on random number seed
  - exhibits self similarity across scales (fractal) and therefor may be suited for representing natural phenomena
  - simple algorithm and fast to implement
  - supports periodic boundary conditions for tileable texture generation