Procedural Content Generation

Greg Turk
School of Interactive Computing
and GVU Center

How to Make Worlds using Random Numbers

Greg Turk
School of Interactive Computing
and GVU Center

Content

- Stuff in Games or Film
- Graphics
- Sounds
- Character Intelligence

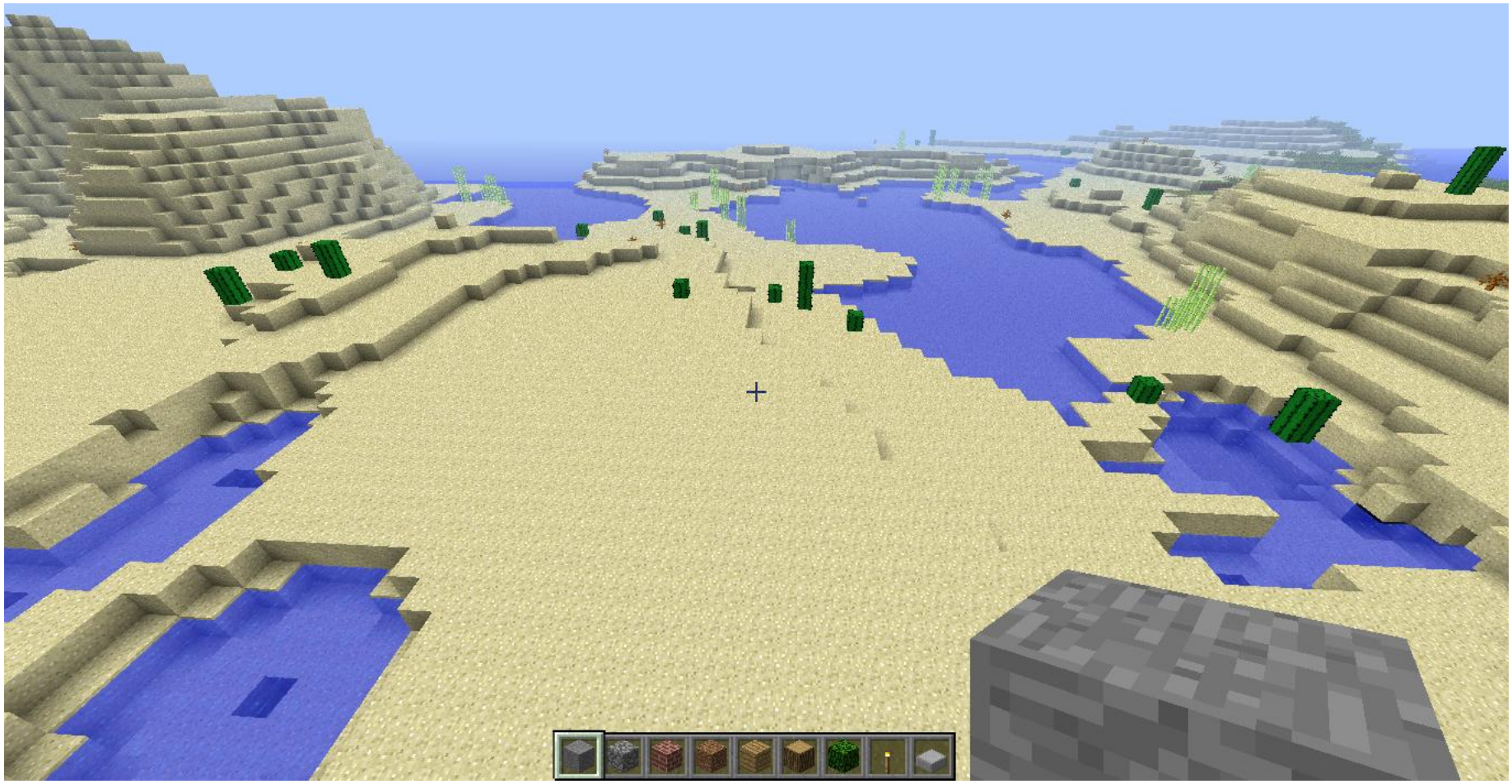
Problem: Human-creation of content slow

Graphics Content

- Mountains, rivers, valleys
- Trees, bushes, flowers, shrubs
- Streets, buildings, houses
- Cars, signs, telephone poles, hydrants
- Animals, humans

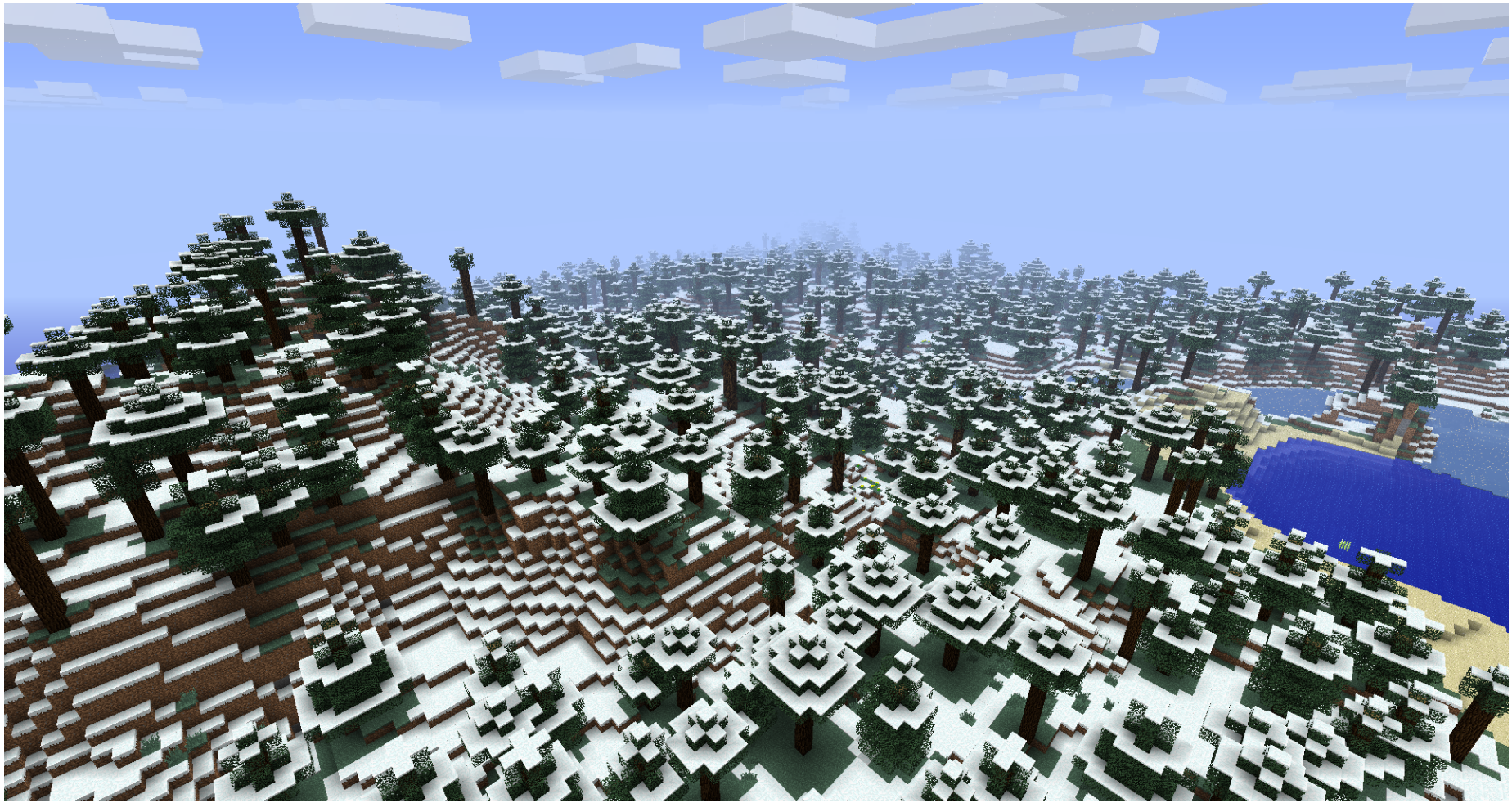
Minecraft

Minecraft



Minecraft

Minecraft



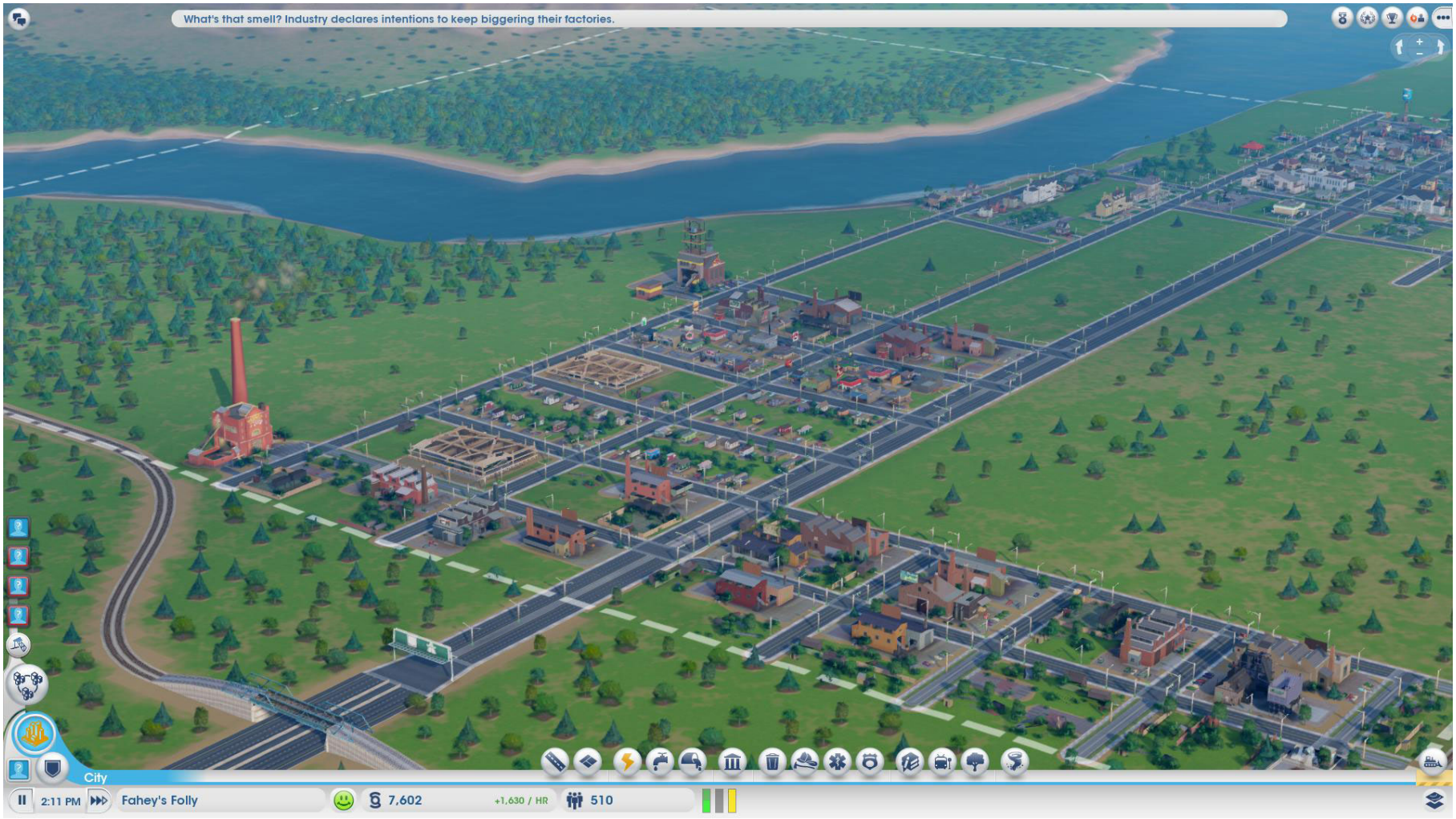
Minecraft

SimCity

SimCity

SimCity

SimCity


What's that smell? Industry declares intentions to keep biggering their factories.
City
2:11 PM
Fahey's Folly
7,602
+1,630 / HR
510

Spore

Spore


Base
1/3
Coat
Detail
Ictacular Grunflax
SPORE
creature creator

Spore

Spore

Spore

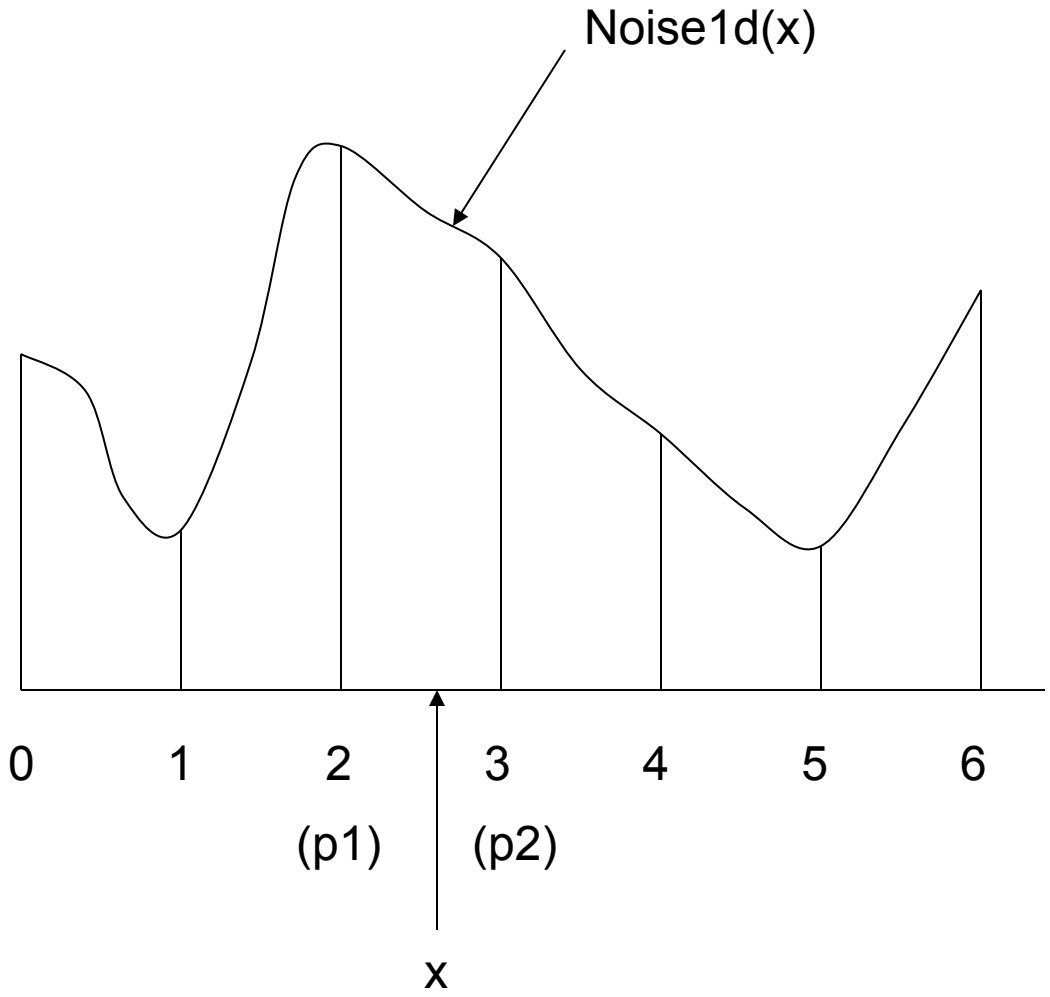
Procedural Content Creation

- Create content by algorithm (procedure)
- Minimize artist/designer time
- Algorithm must create variety of objects
  - Color variations
  - Shape and size differences
  - Include/exclude some features
- Variation from Random Numbers

Ken Perlin's Noise Function

- Controlled randomness
- Random number as function of position
- Numbers in range [0,1]
- Changing position slowly changes randomness slowly
- Position can be in 1D, 2D, or 3D
- Mostly used for texture creation

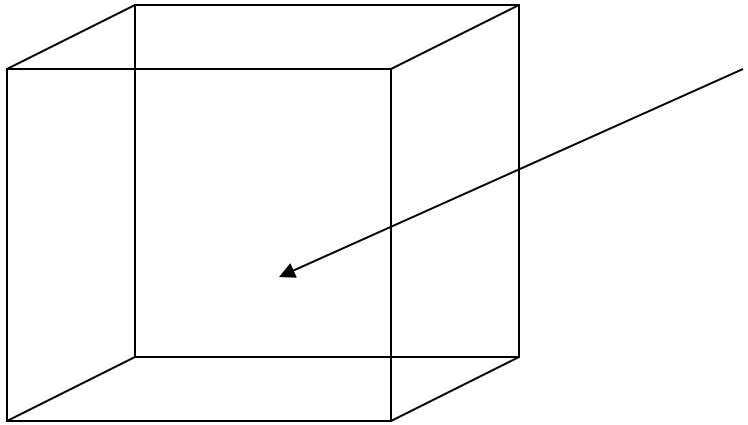
Perlin Noise



Noise Algorithm

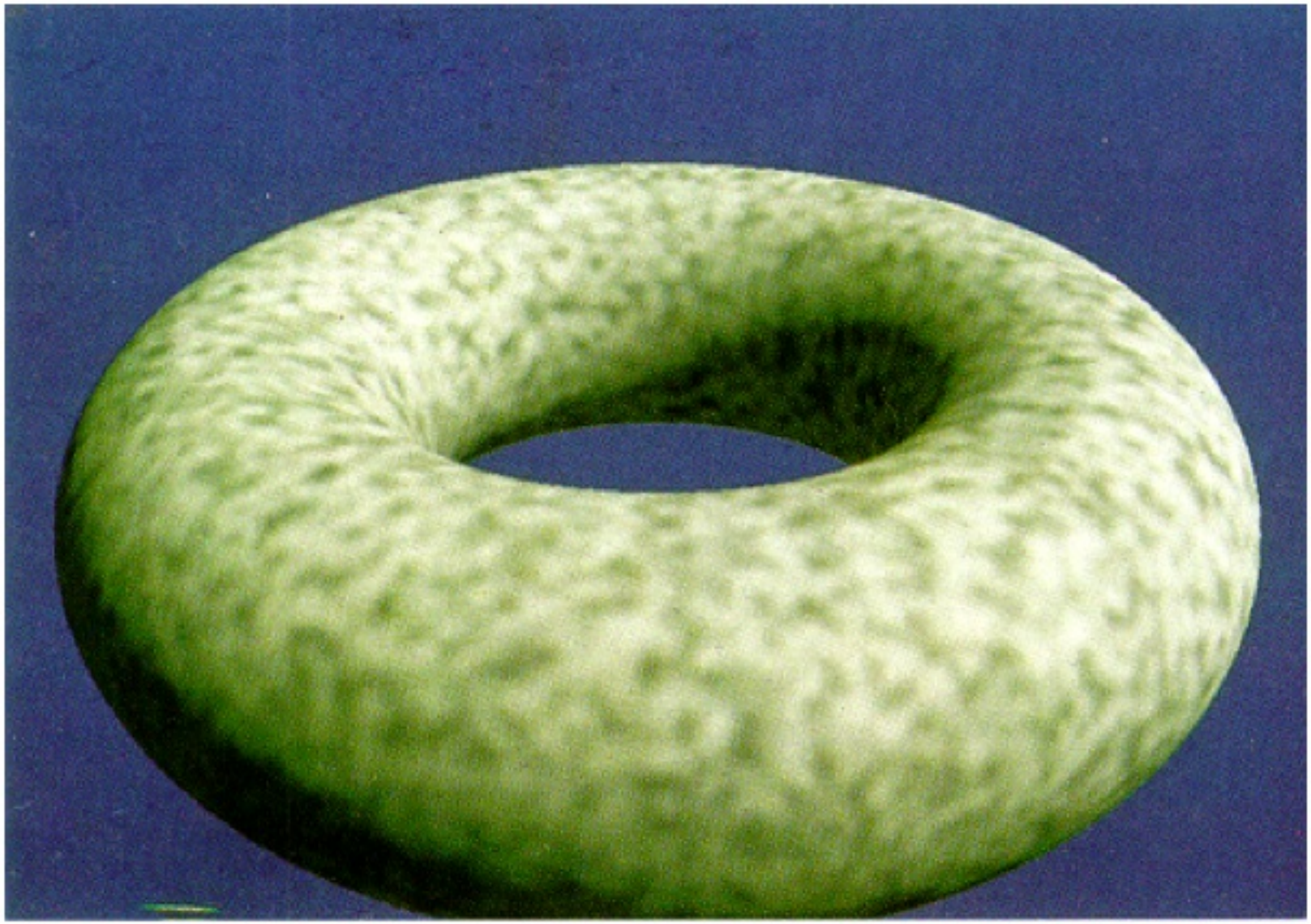
```
float Noise1d(float x) {
  p1 = floor (x)
  p2 = floor (x) + 1
  t = x - p1
  v1 = seeded_random(p1)
  v2 = seeded_random(p2)
  return (smooth_interp (v1, v2, t))
}
```

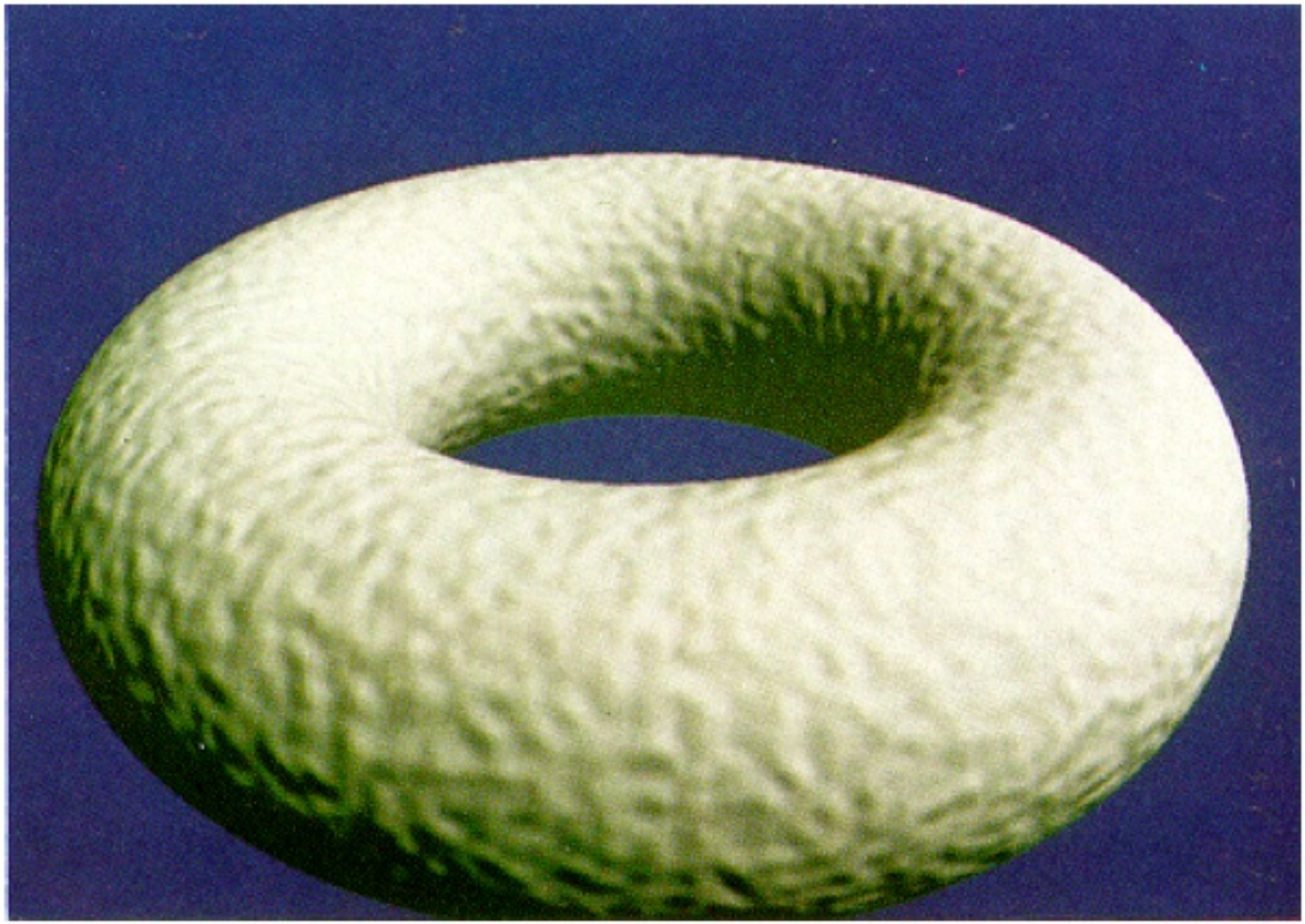
3D Noise

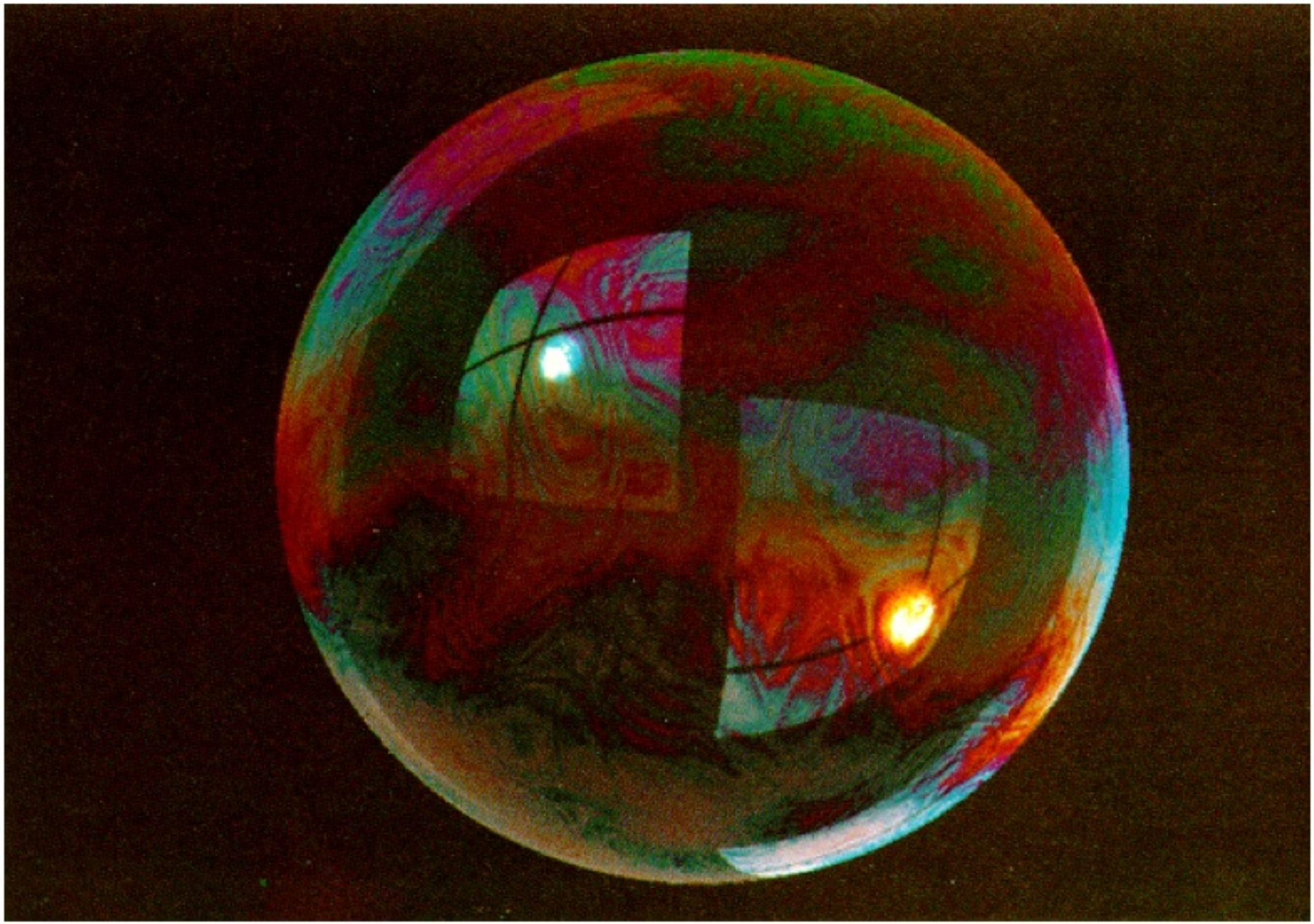


Interpolate values from 8 neighbors, all using seeded random numbers

ix = floor (x)
iy = floor (y)
iz = floor (z)
seed = (ix + prime1 * iy + prime2 * iz) mod seed_max







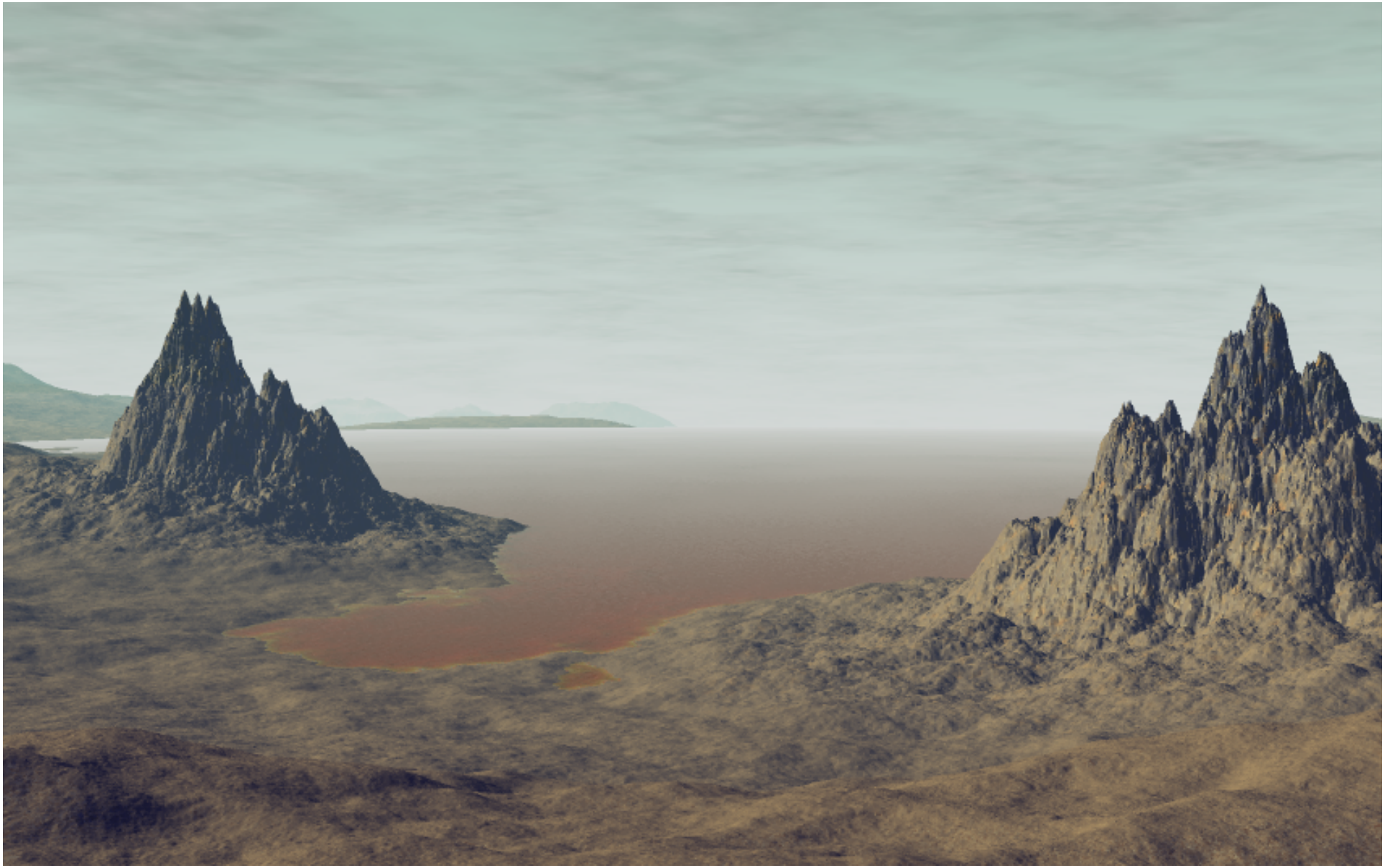
Terrain

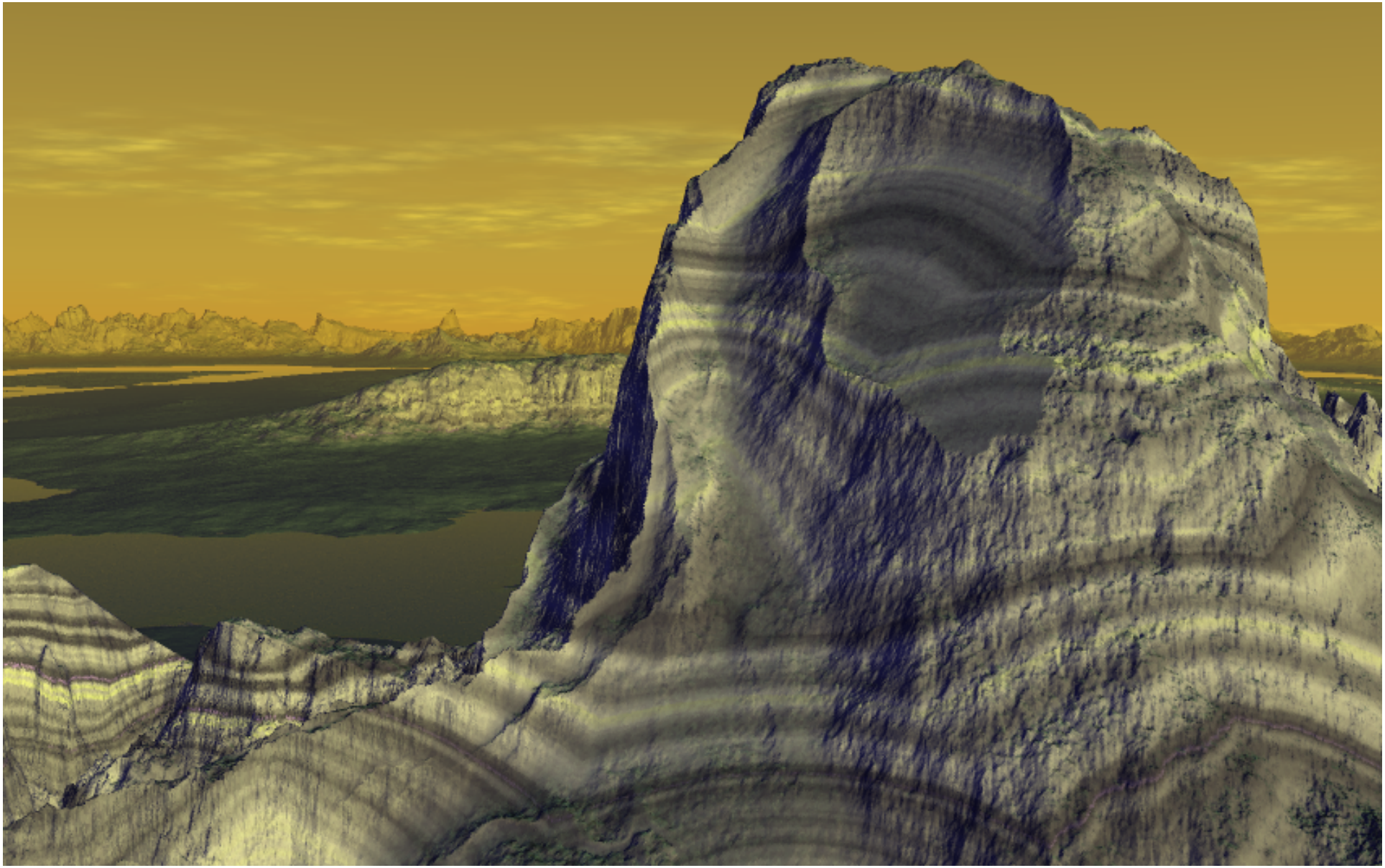
- Mountains, dunes, cliffs, valleys
- Use fractal subdivision
- Different subdivision parameters
  - Vary smoothness
  - Vary height ranges
  - Color and texture variation
  - Add water, snow, sand

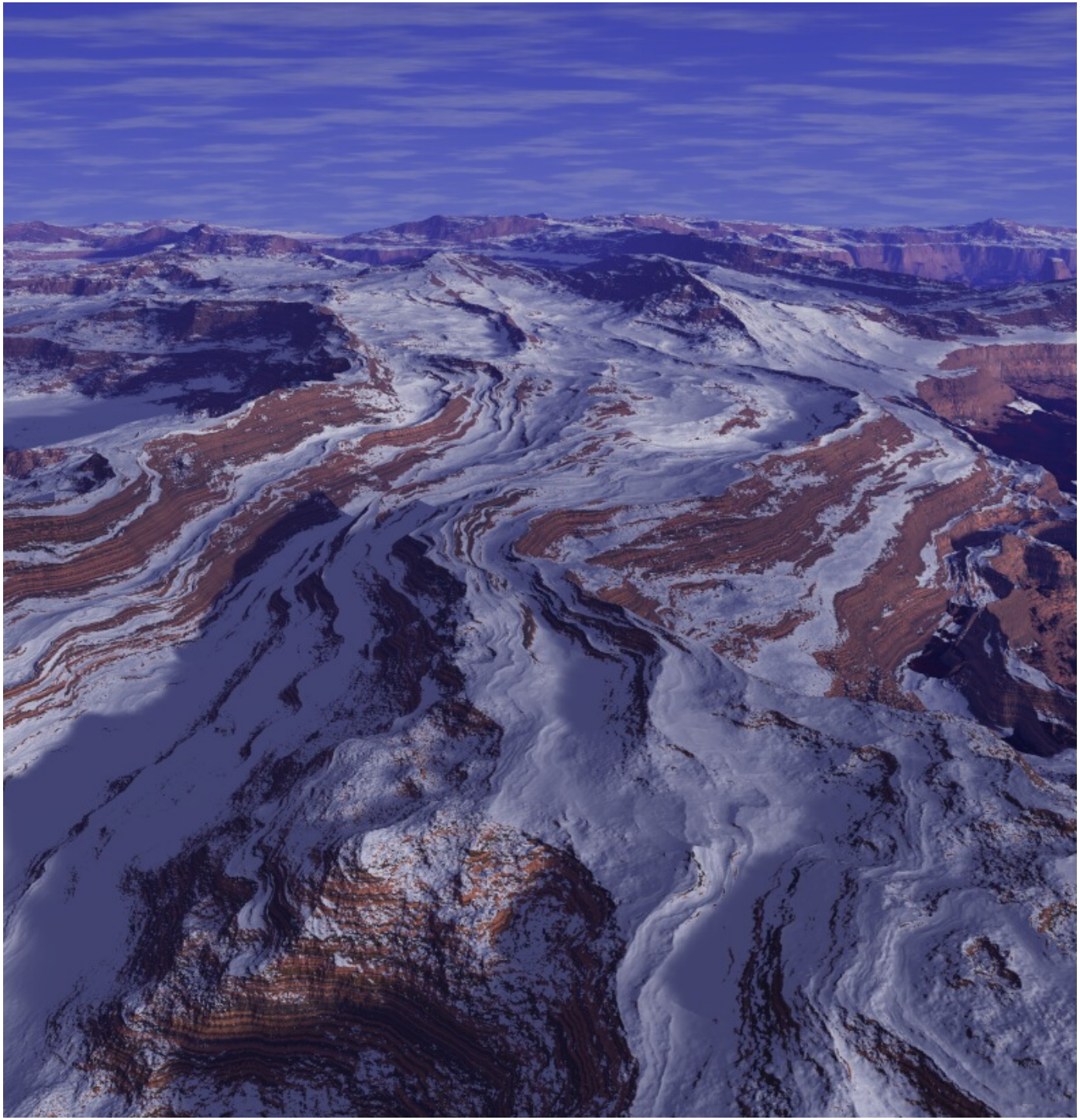
Ken Musgrave's Software

Pandromeda

www.terrascape.com



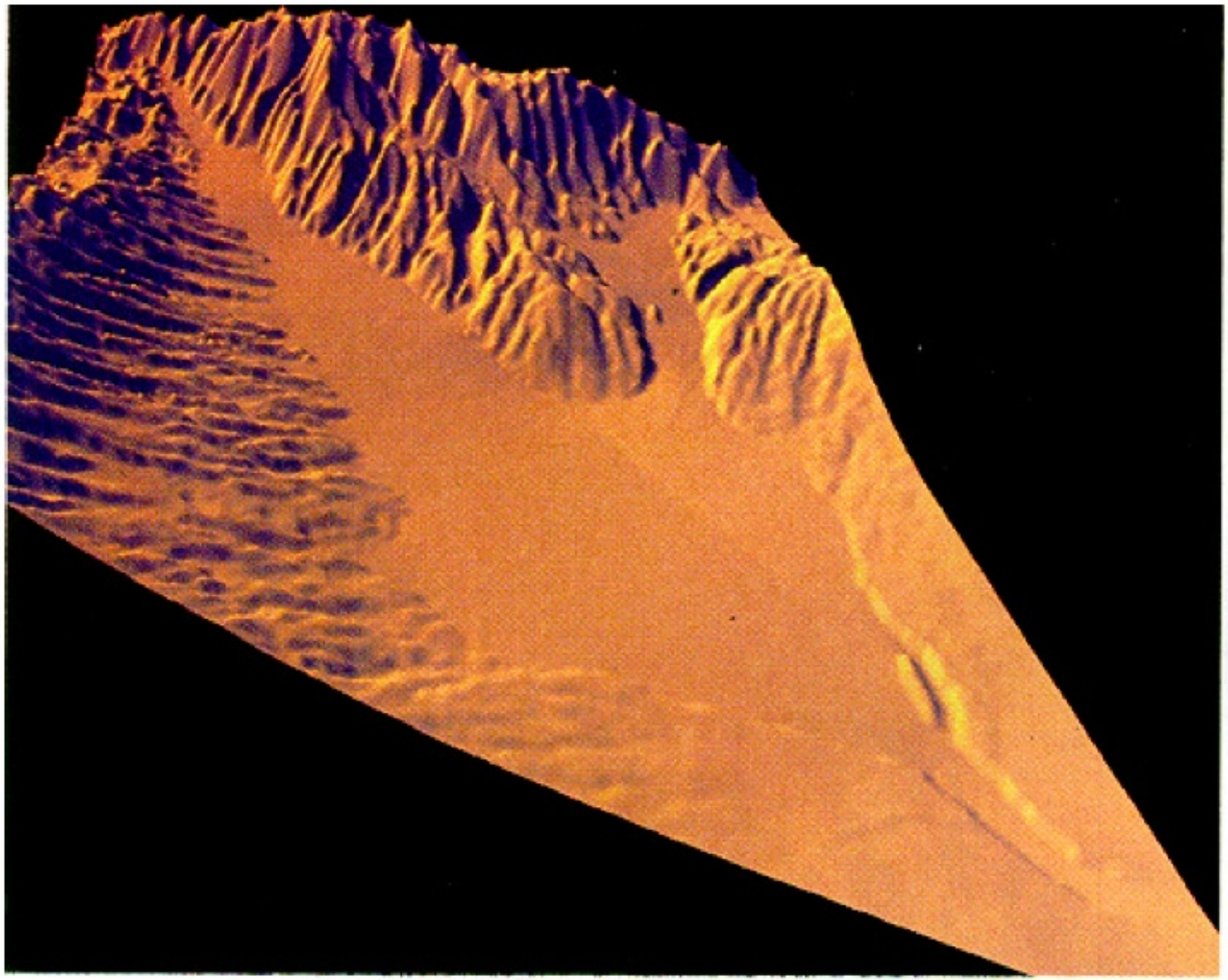




Simulated Erosion

- Rain deposited on grid
- Water moves to near cells based on height (downhill)
- Fast moving water picks up dirt and rock
- Slow water drops carried sediment

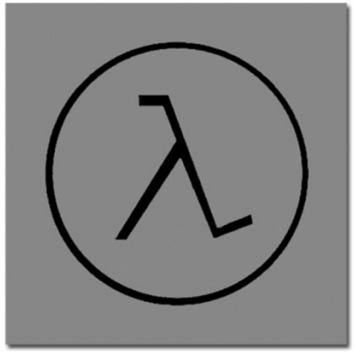
Erosion



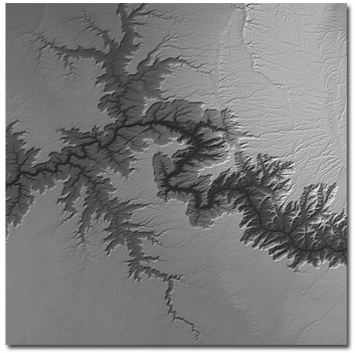
Erosion

User-Guided Terrain

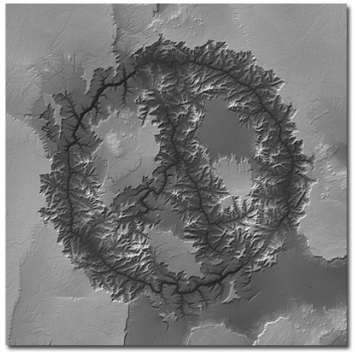
- Give users control over terrain
- User makes simple sketch
- User provides example terrain
  - Valley, mountain, lake, etc.
- Algorithm fills in details
- Work of Howard Zhou (Georgia Tech)



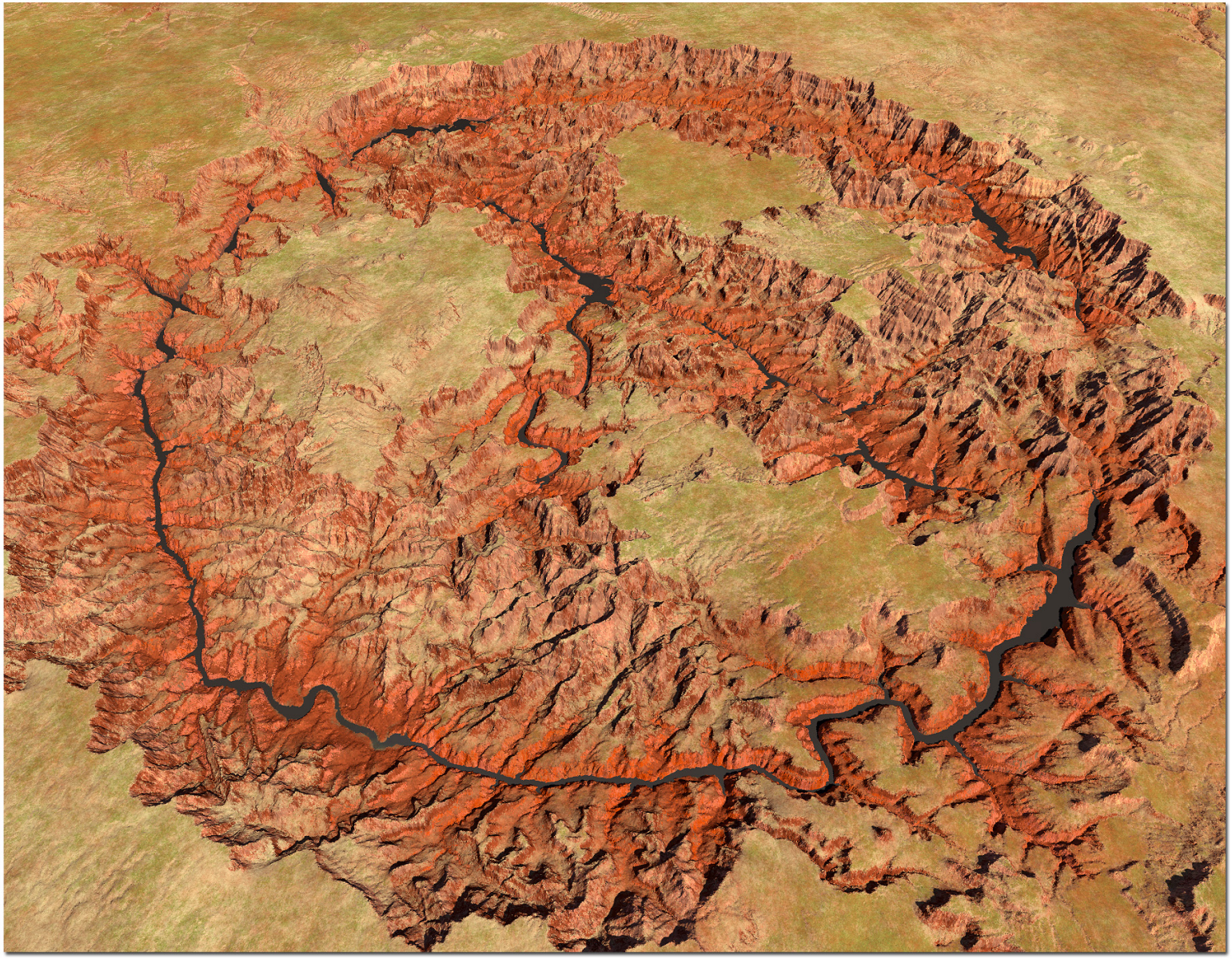
User Sketch



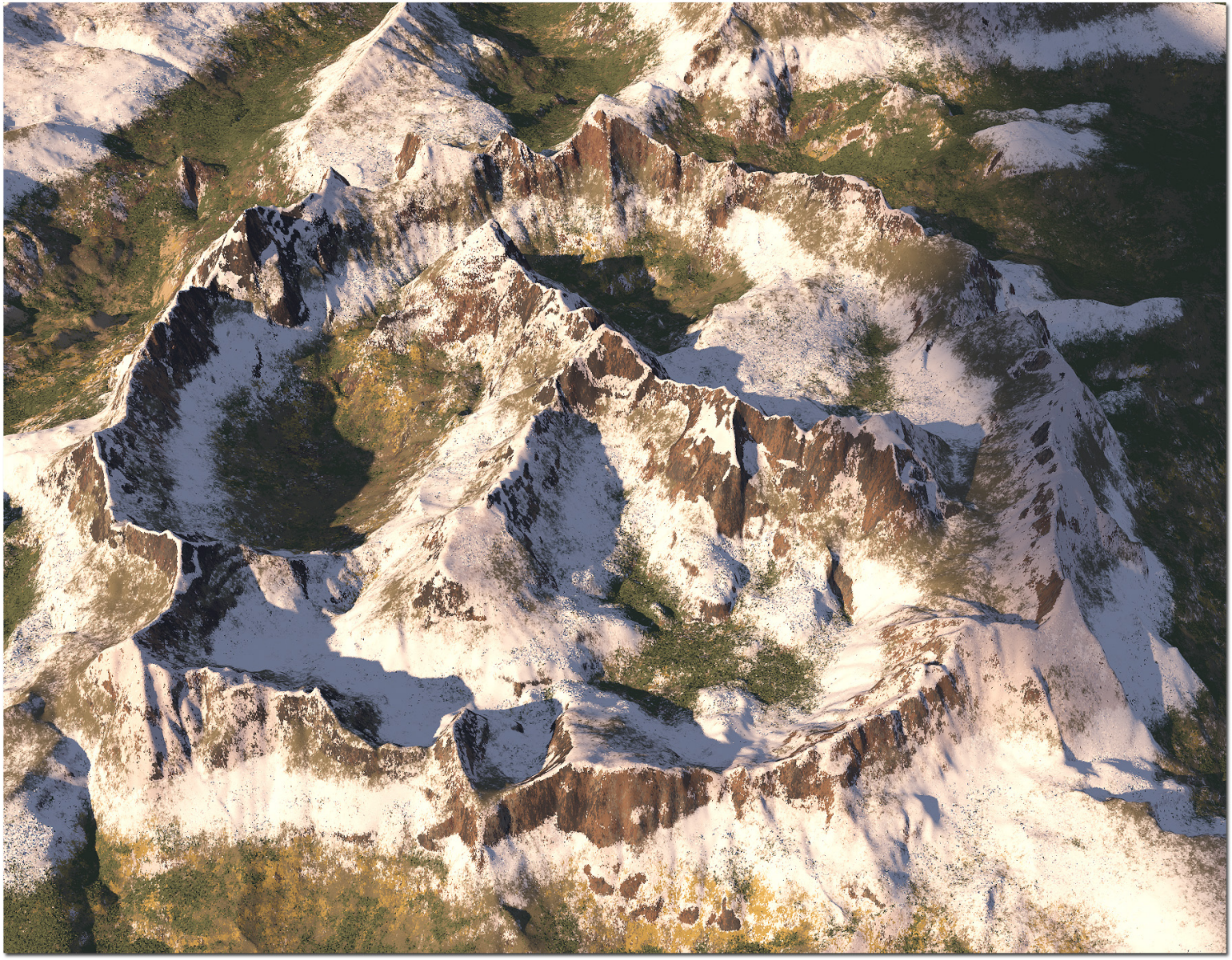
Grand Canyon (partial)



Synthesis result

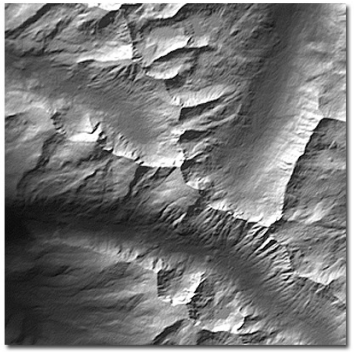


Rendered terrain synthesized from an elevation map of the Grand Canyon

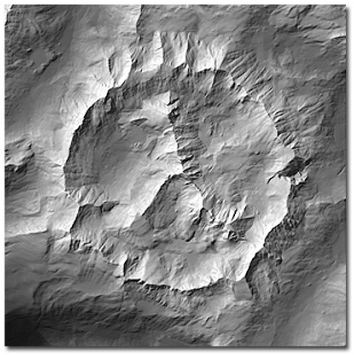


Rendered terrain synthesized from an elevation map of Mount Jackson, Colorado

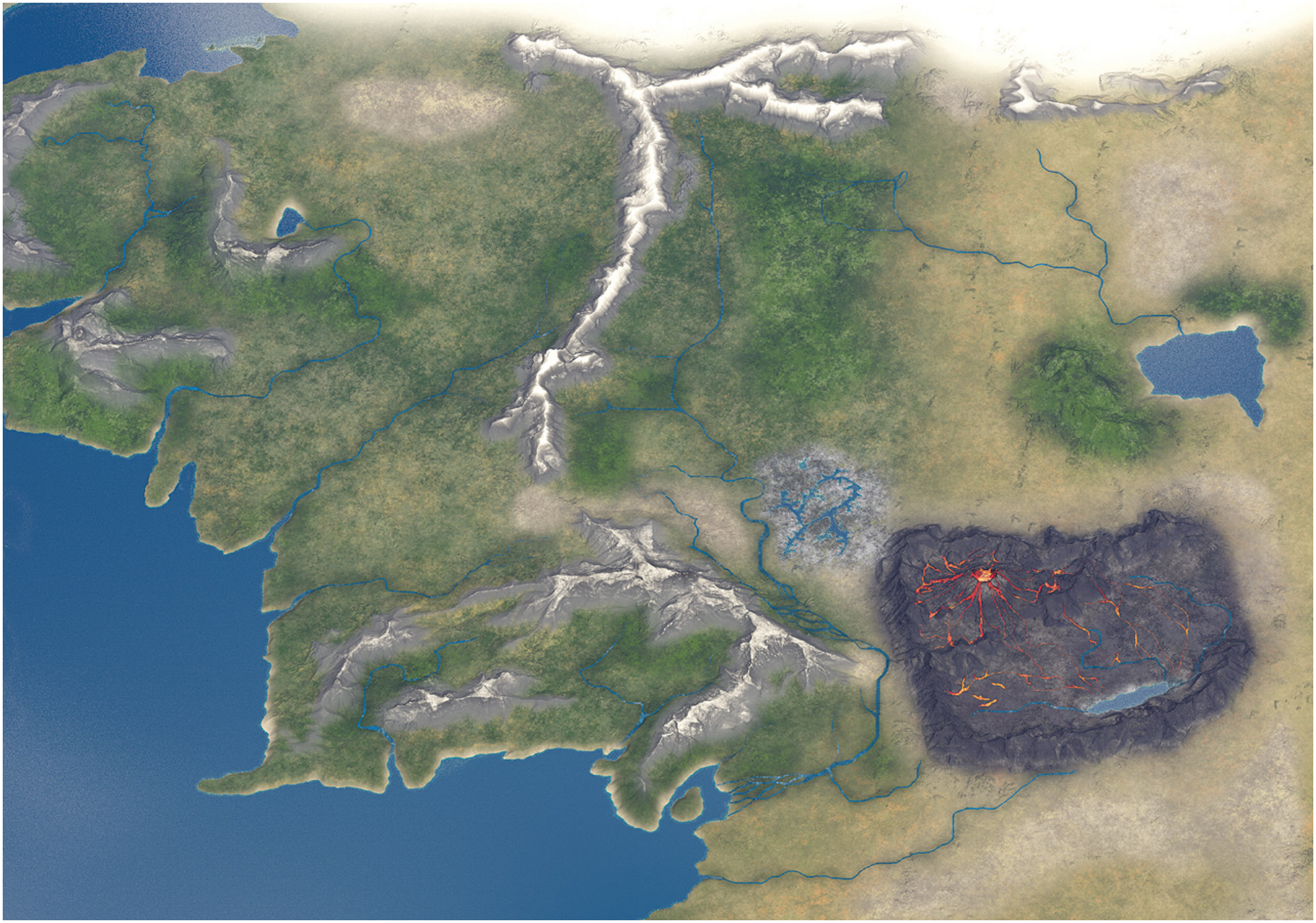
User Sketch



Mount Jackson



Synthesis result



Plant Modeling

- Use grammar (L-system)
- Rule guide segment changes/growth
- Specifies branching pattern
- Add lengths and angles of branches

- Work of Przemyslaw Prusinkiewicz

L-system (a)

ω: S

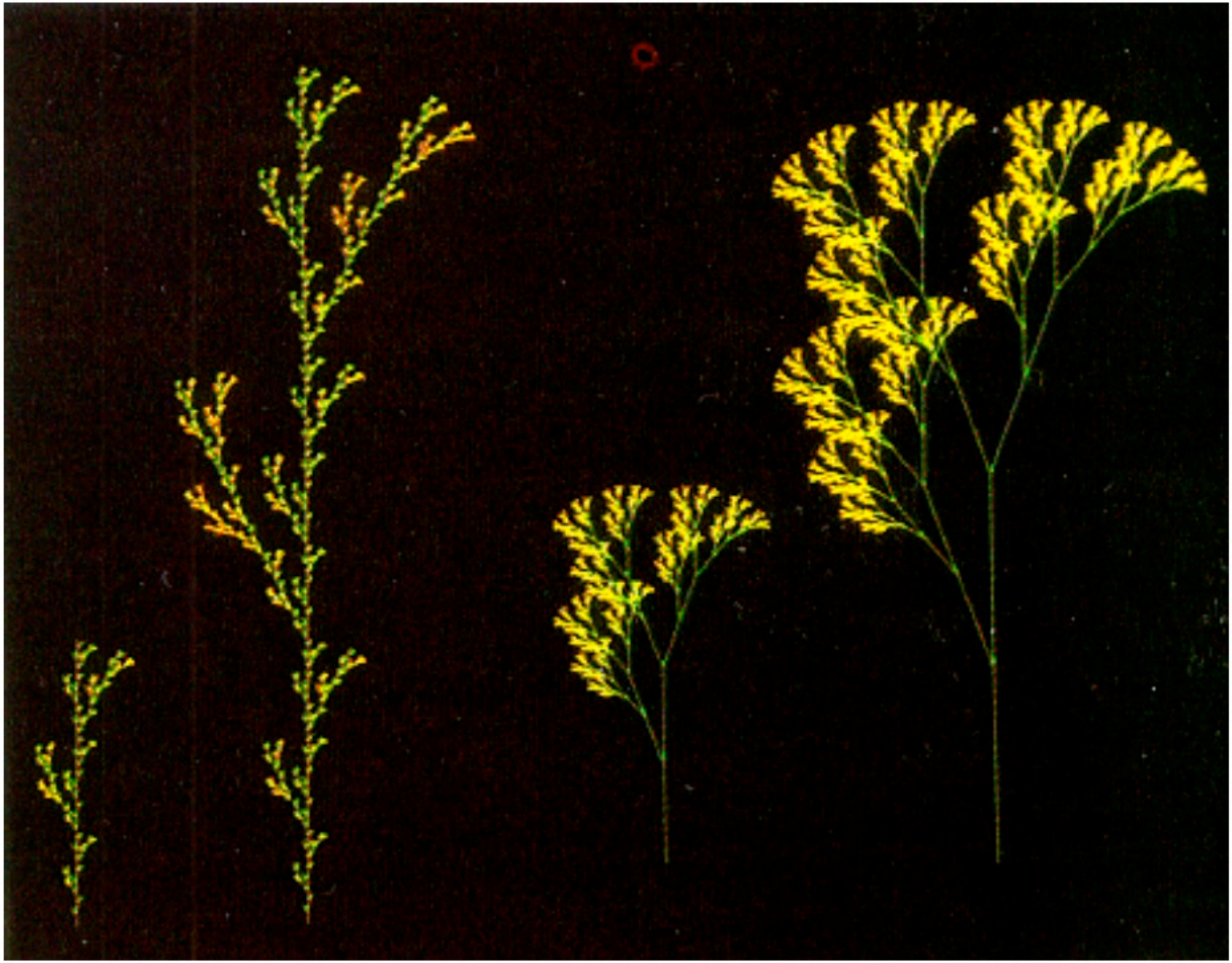
p: $S \rightarrow S[S]S[S]S$

L-system (b)

ω: A

p_1: $A \rightarrow S[A]S[A]A$

p_2: $S \rightarrow SS$



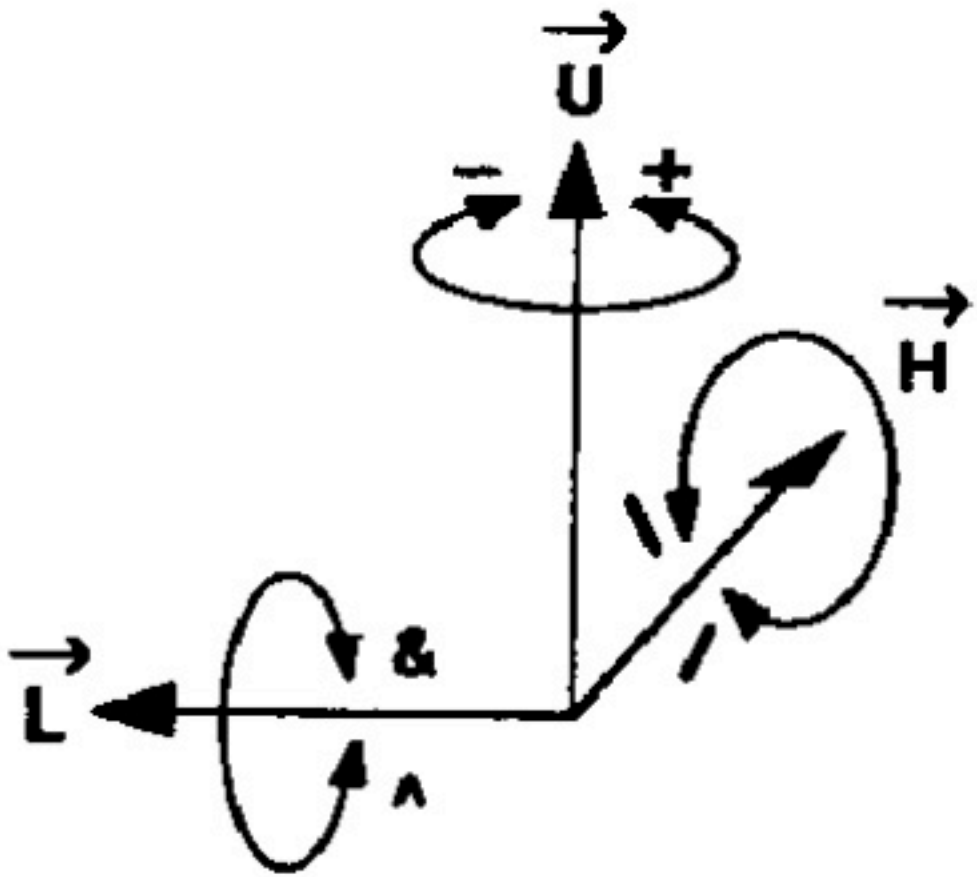


Figure 6. Turtle interpretation of geometric attribute symbols.

A more complex L-system generating the three-dimensional bush taken from [34] and shown in Fig. 7 is given below.

ω: $//a$

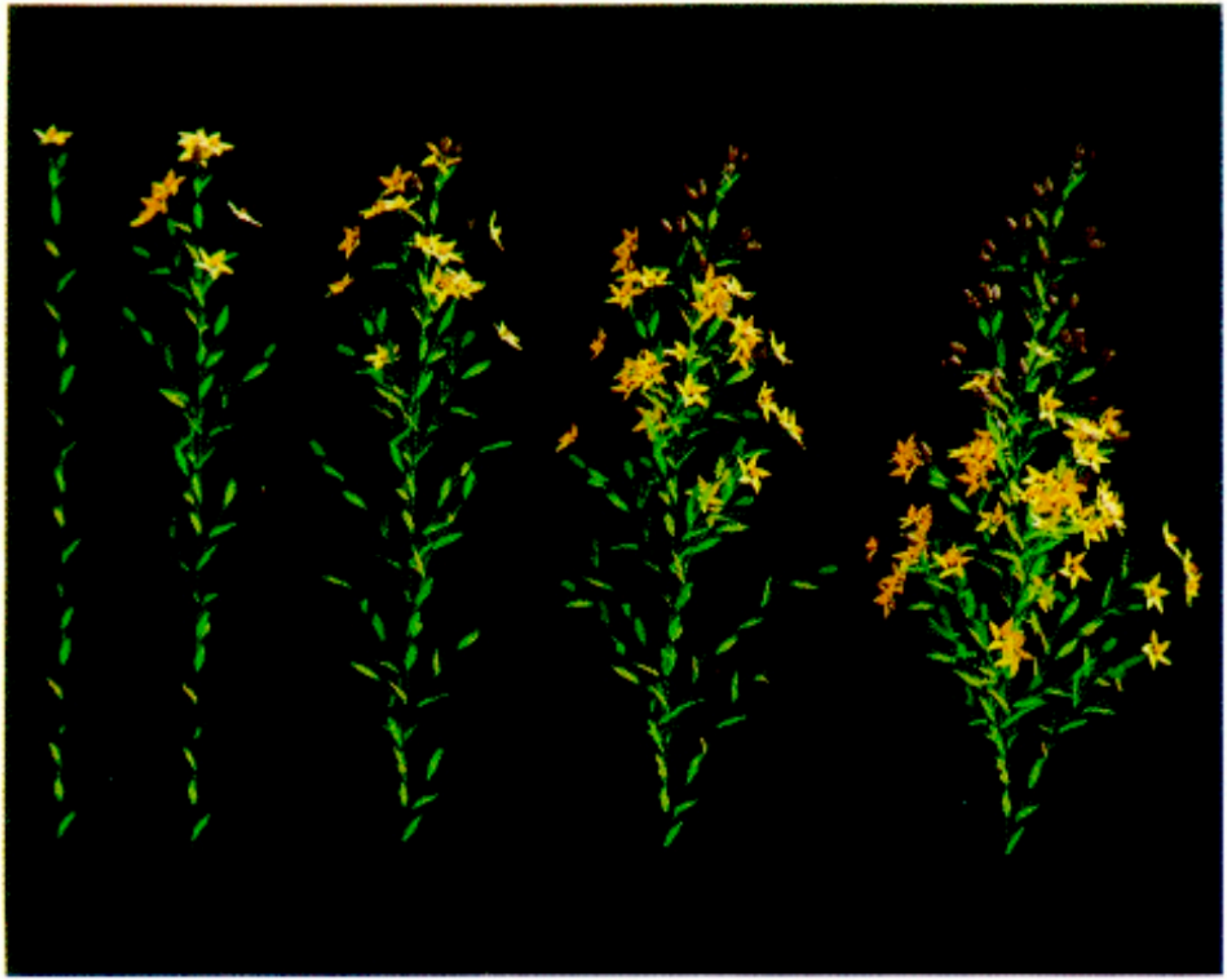
p_1: $a \rightarrow [[\&sl!a]/////'[\&sl!a]///////'[\&sl!a]]$

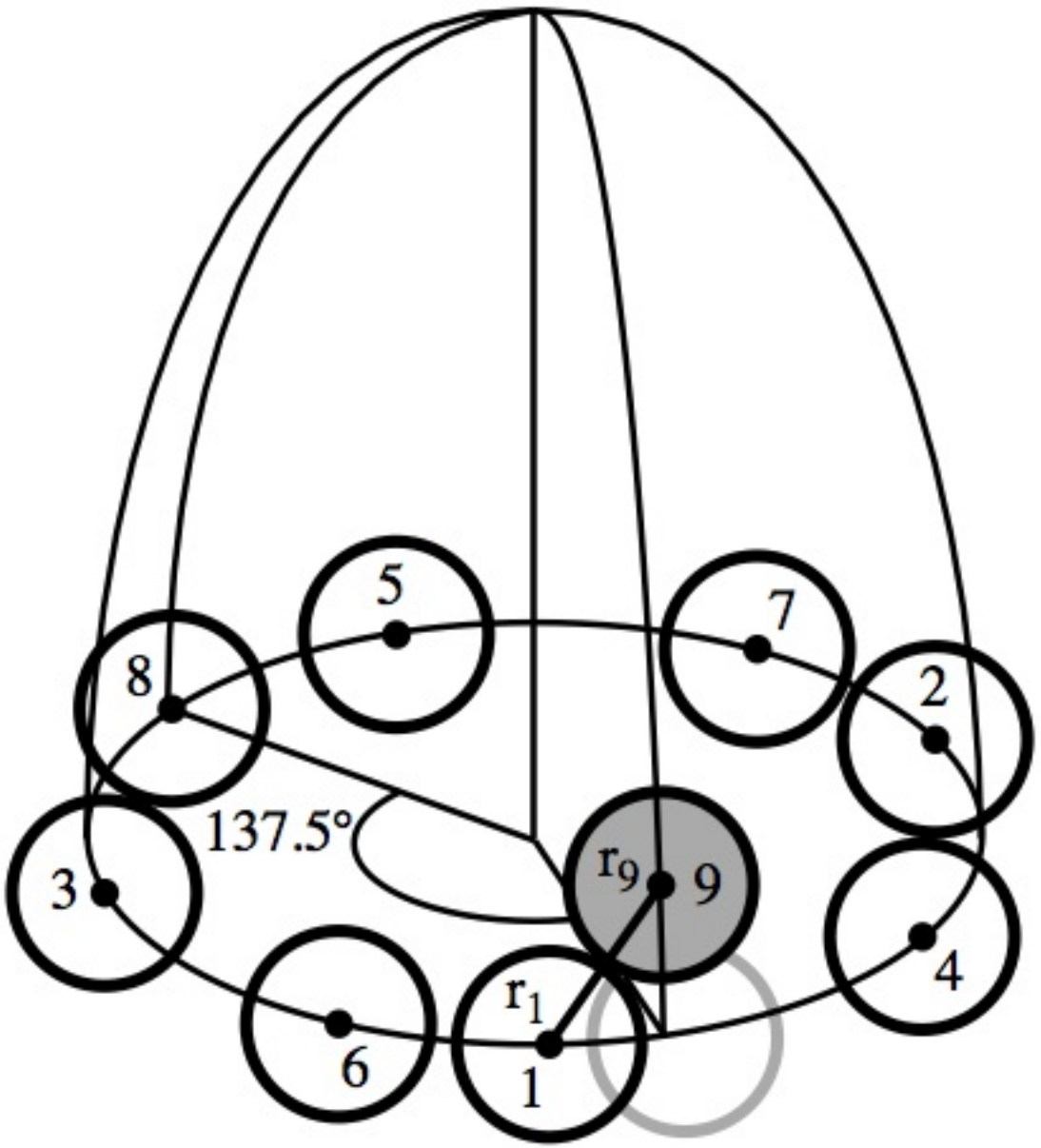
p_2: $s \rightarrow Sl$

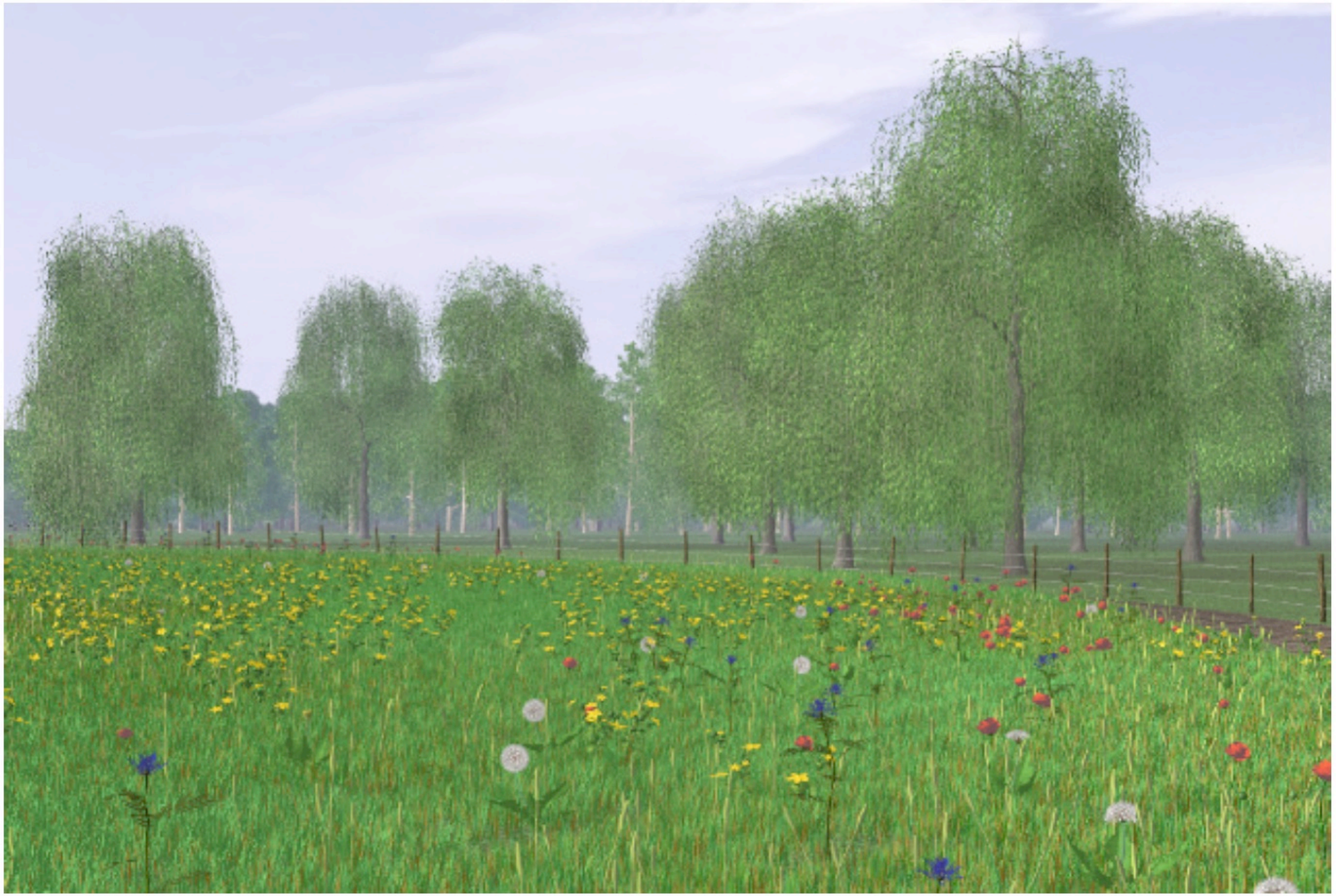
p_3: $S \rightarrow S/////s$

p_4: $l \rightarrow [''''\hat{}\hat{}(-S+S+S-l-S+S+S)]$

Figure 7. A bush.




5
7
8
2
137.5°
3
r_9
9
4
r_1
6
1



Buildings

- Use grammar for buildings
- Elements of buildings:
  - Overall shape (e.g. connected boxes)
  - Number of Floors
  - Window and door placement
  - Specific types of windows
  - Texture (brick, wood, tiles)

PRIORITY 2:

5: facade : Shape.visible("Street") == 0 $\rightsquigarrow$
Subdiv("Y",*groundfloor_height*,1r,*topfloor_height*)
{ groundfloor | floors | topfloors } fireescape

6: groundfloor $\rightsquigarrow$ Subdiv("X",1r,*entrance_width*,1r){ groundtiles |
entrance SnapLines("Y","entrancesnap") | groundtiles }

PRIORITY 3:

7: facade $\rightsquigarrow$ floors

8: floors $\rightsquigarrow$ Repeat("YS",*floor_height*){ floor Snap("XZ") }

9: floor $\rightsquigarrow$ Repeat("XS",*tile_width*){ tile Snap("Y","tilesnap") }

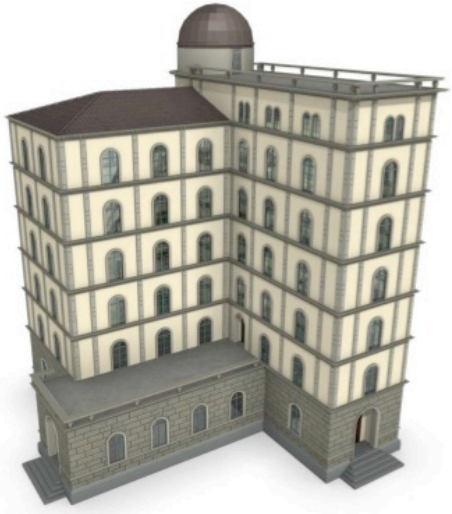
...

15: wall : Shape.visible("Street") $\rightsquigarrow$ I("frontwall.obj")

PRIORITY 4:

16: fireescape $\rightsquigarrow$ Subdiv("XS",1r,2**tile_width*,7r,"tilesnap")
{ epsilon | escapestairs | ε }

17: escapestairs $\rightsquigarrow$ S(1r,1r,*fireescape_depth*)
T(0,0,-*fireescape_depth*) Subdiv("YS",*groundfloor_height*,1r)
{ ε | Repeat("YS",*floor_height*){ I("fireescape.obj") } }



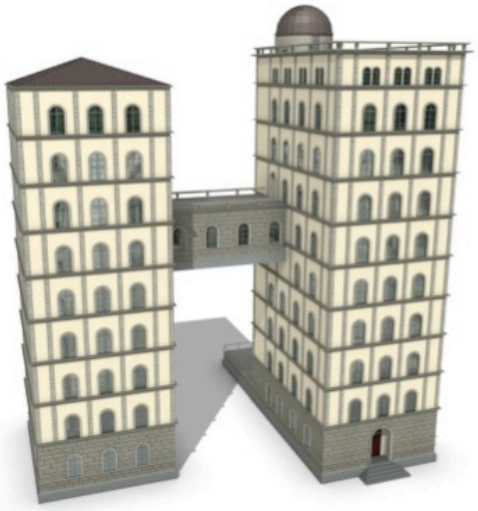
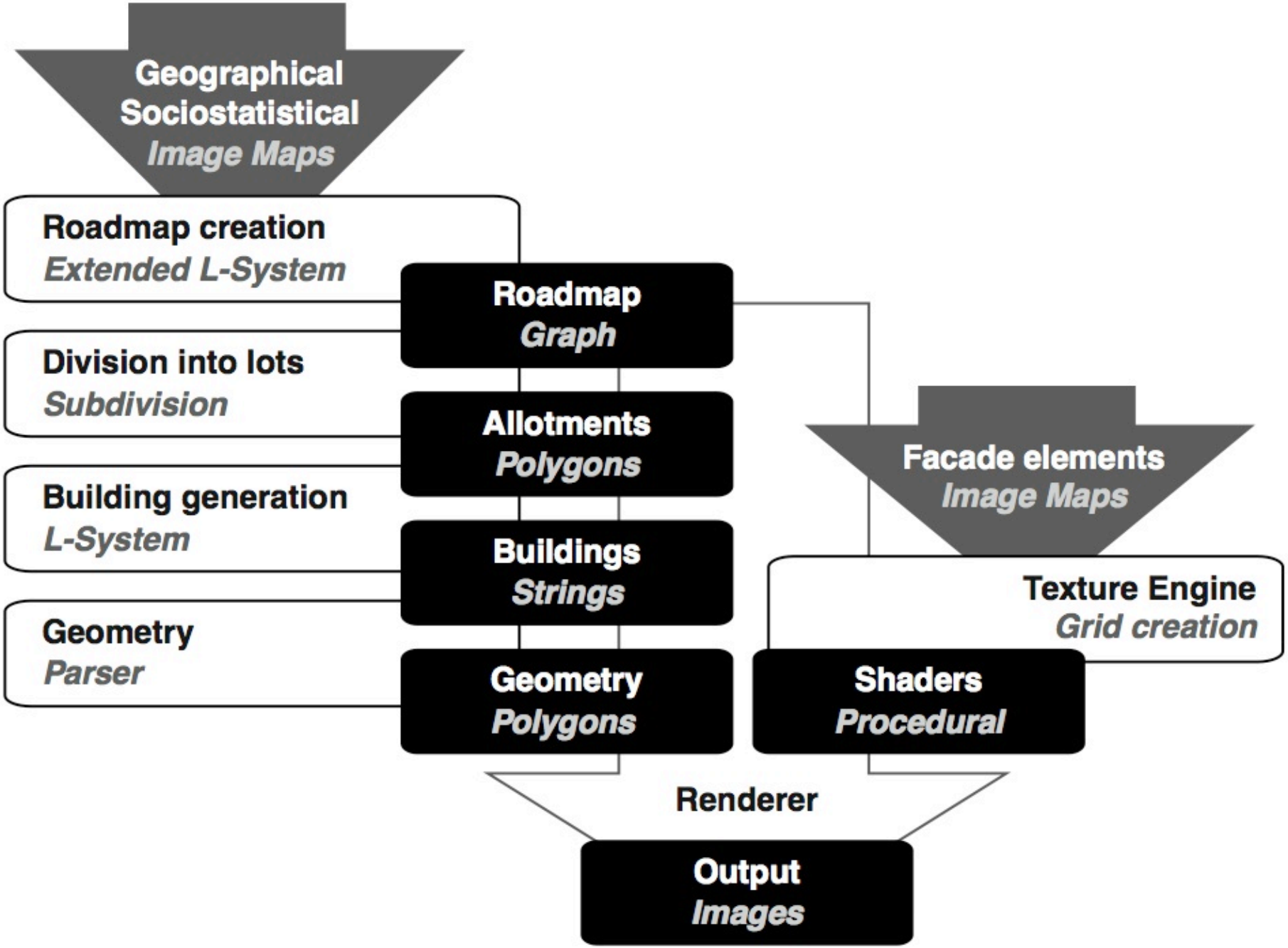
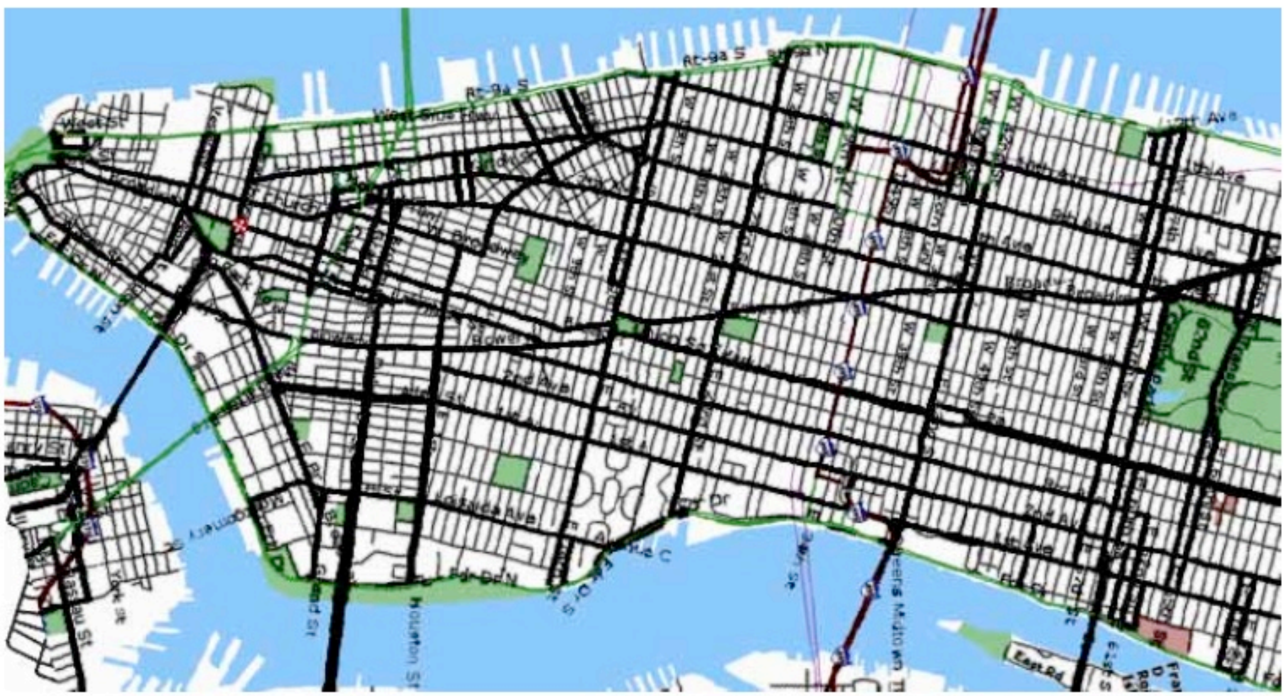


Figure 18: Various views of the procedural Pompeii model. Based on real building footprints, the city was generated with 190 manually written *CGA shape* rules. Hence, the whole model is a rule-based composition of 36 terminal objects (plus 4 tree types and the environment).

City Streets

- Given: map of land and water
- Create: street layout


Geographical
Sociostatistical
Image Maps
Roadmap creation
Extended L-System
Division into lots
Subdivision
Building generation
L-System
Geometry
Parser
Roadmap
Graph
Allotments
Polygons
Buildings
Strings
Geometry
Polygons
Facade elements
Image Maps
Texture Engine
Grid creation
Shaders
Procedural
Renderer
Output
Images



Rt-9A S
West Side Hwy
Houston St

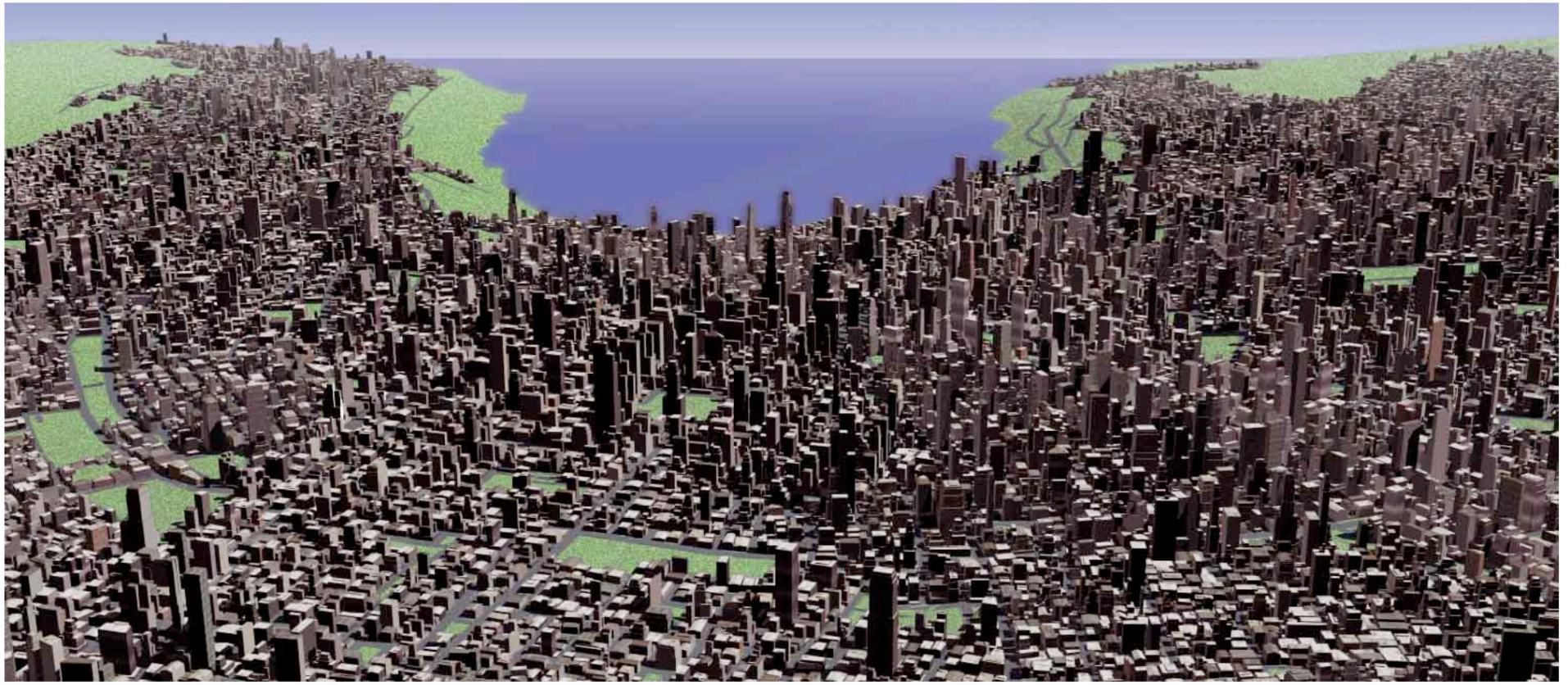


Figure 17. A virtual city modelled using the data from figure 2. Approximately 26000 buildings were created.

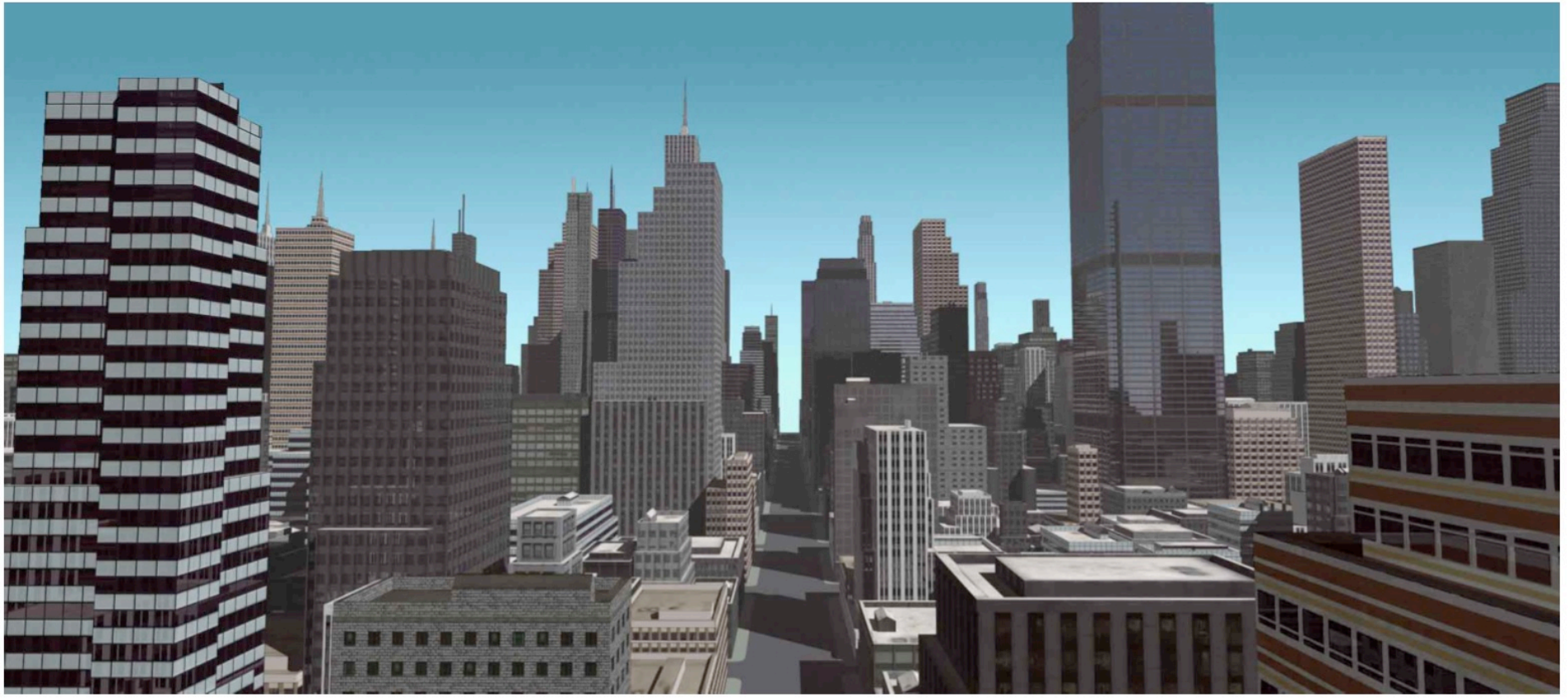
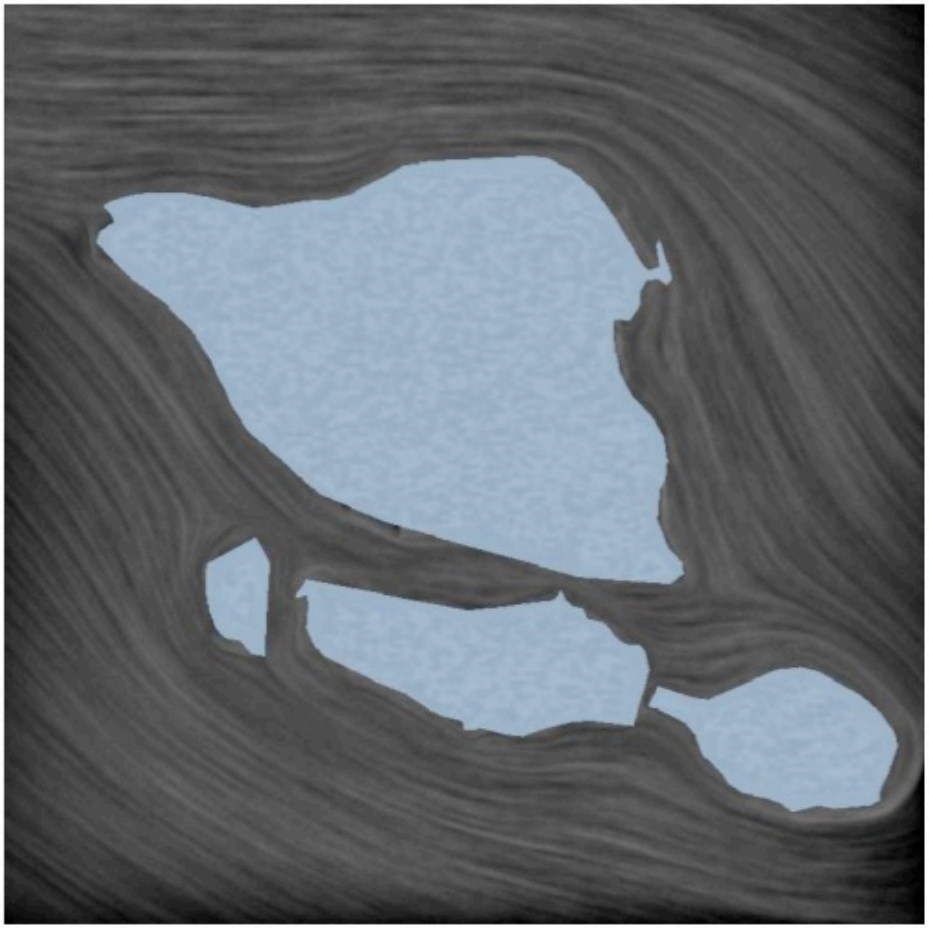
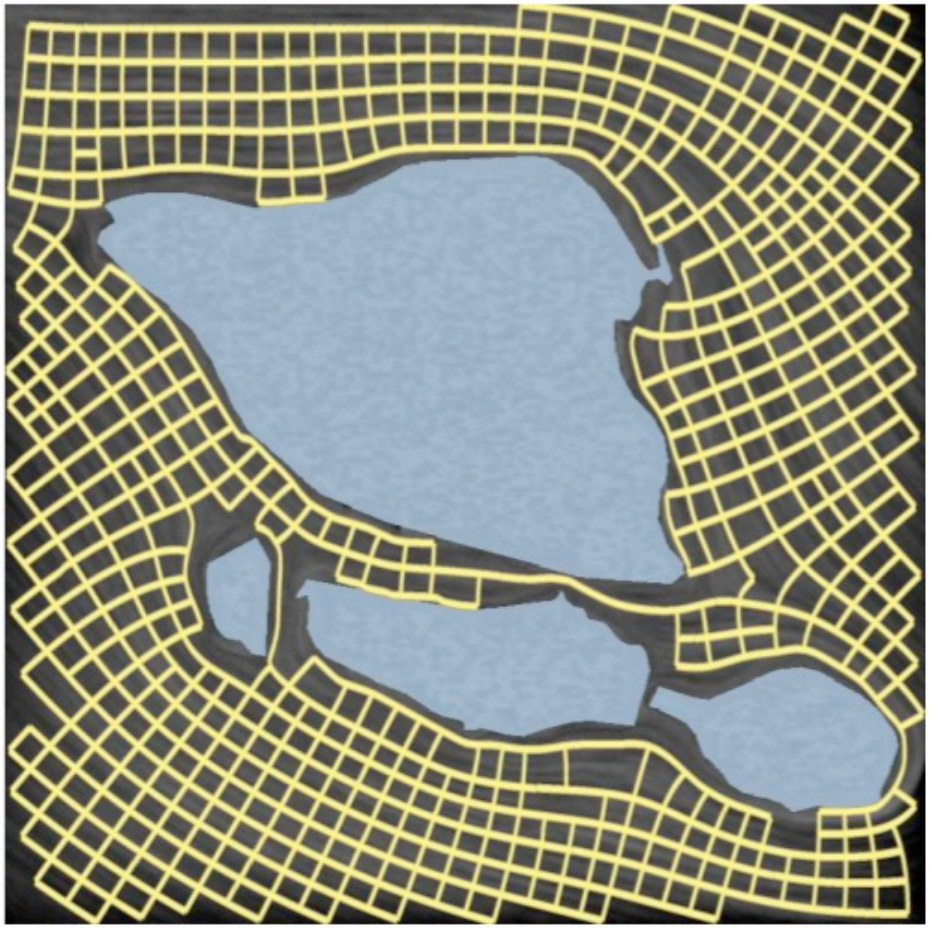


Figure 18. Somewhere in a virtual Manhattan.





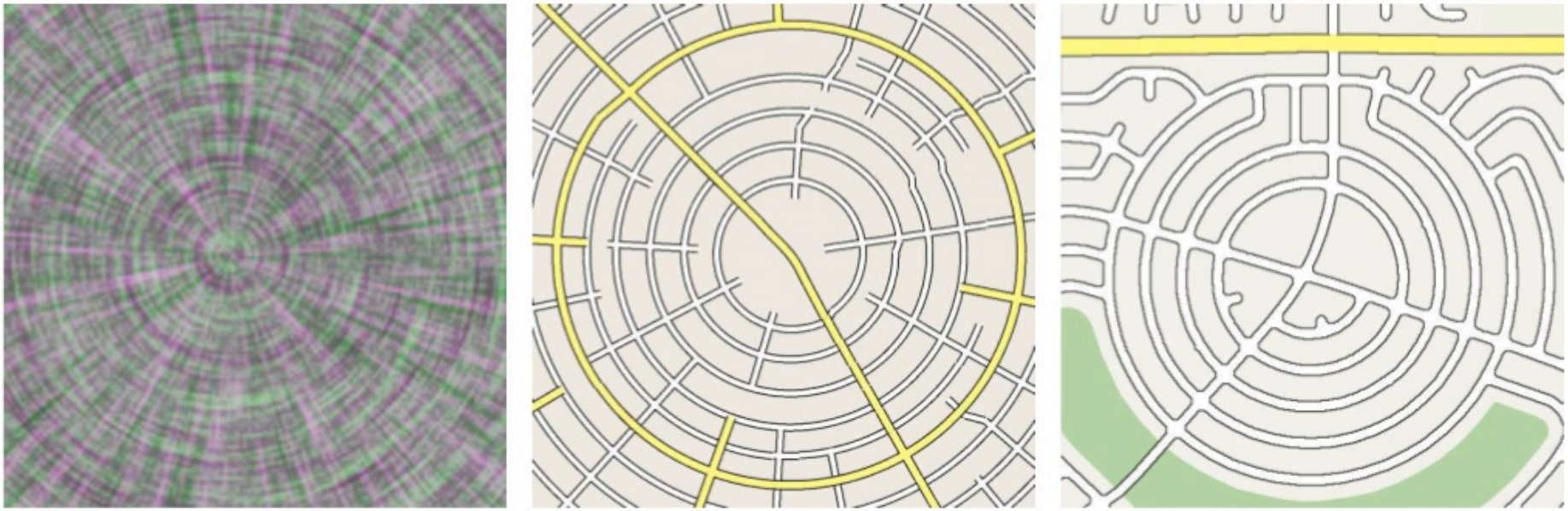
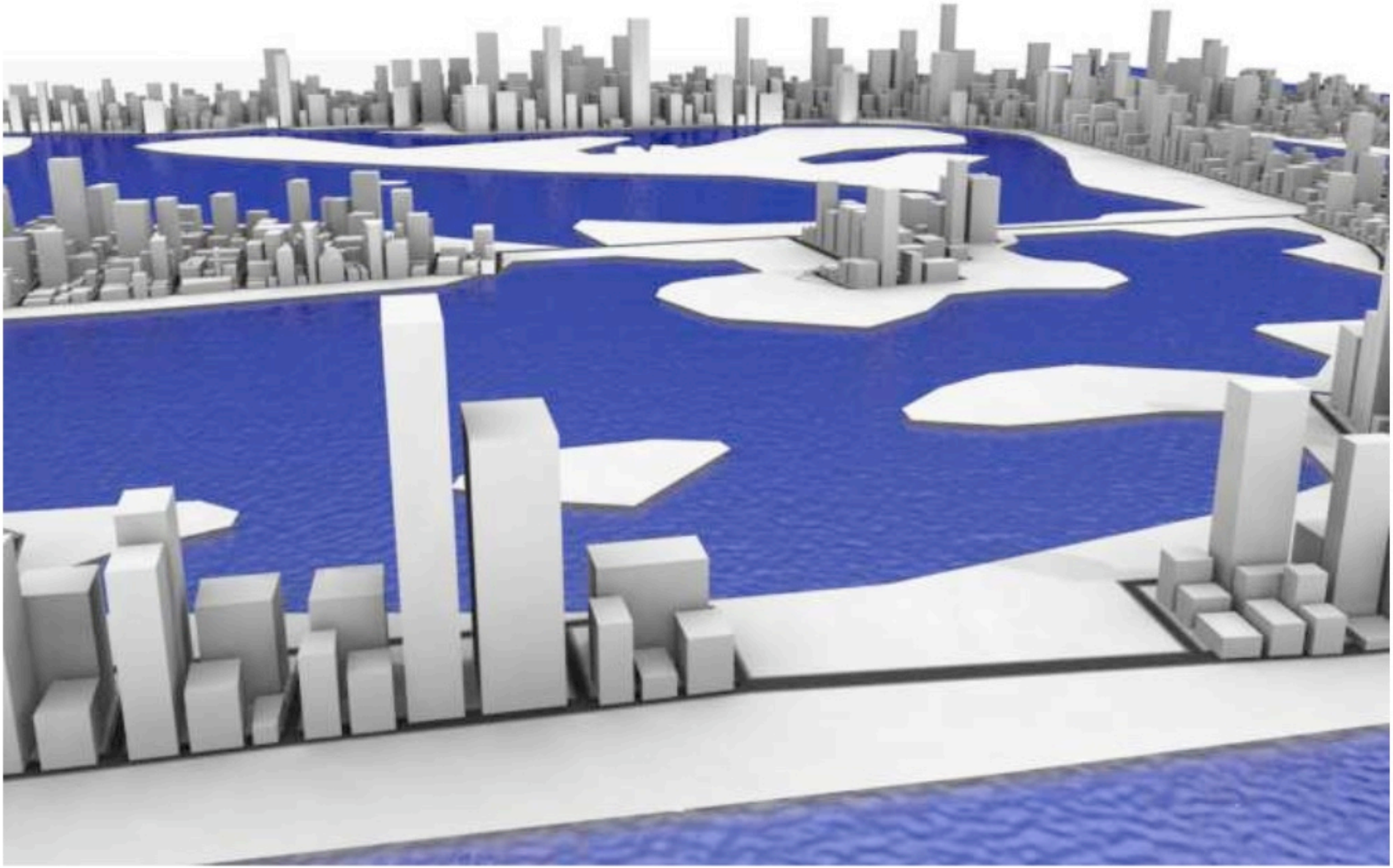
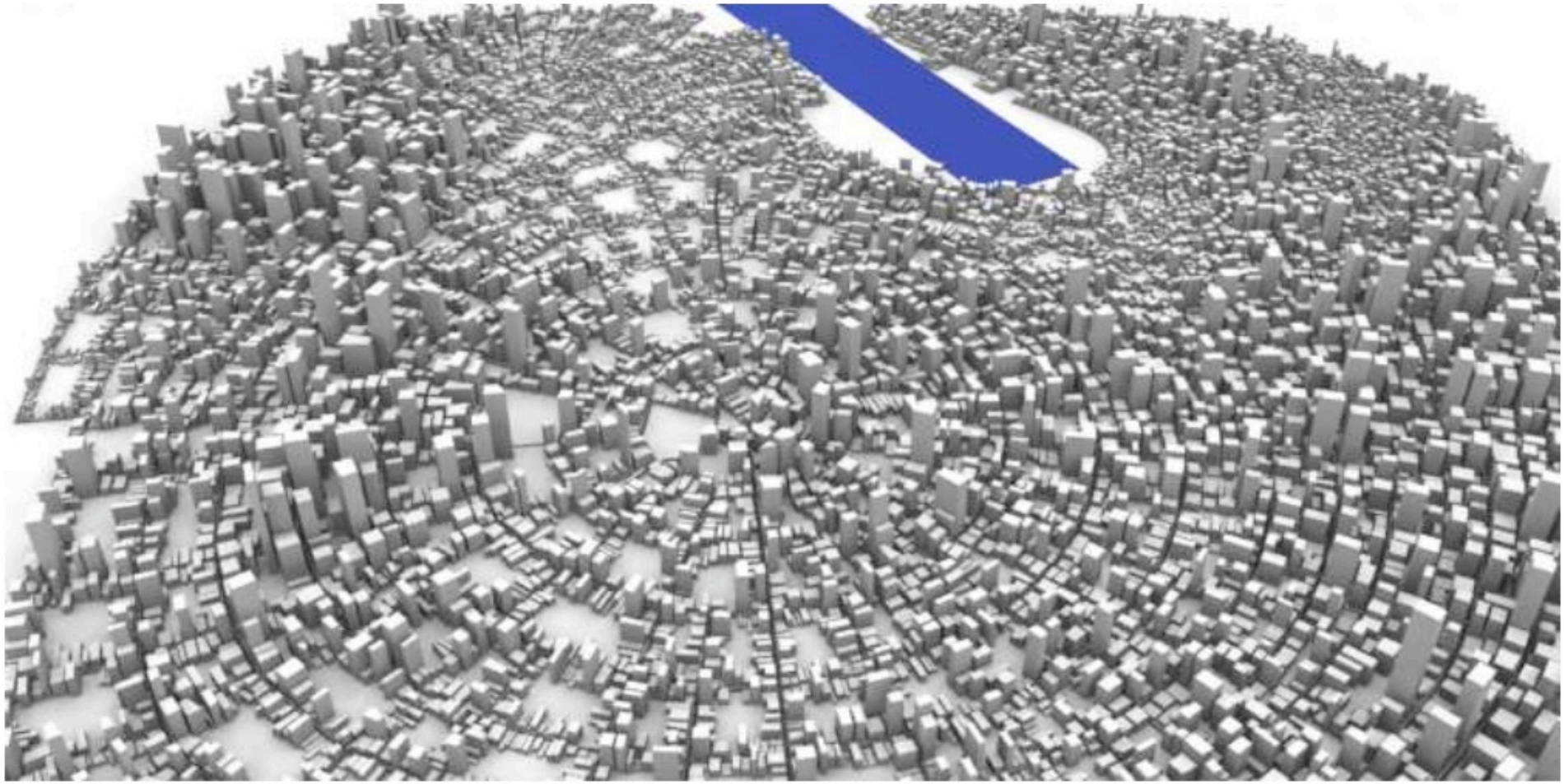


Figure 3: A procedurally generated radial pattern (middle) and its tensor representation (left). The image shown in the right is a radial pattern found in Scottsdale, Arizona.





Cities: Skylines

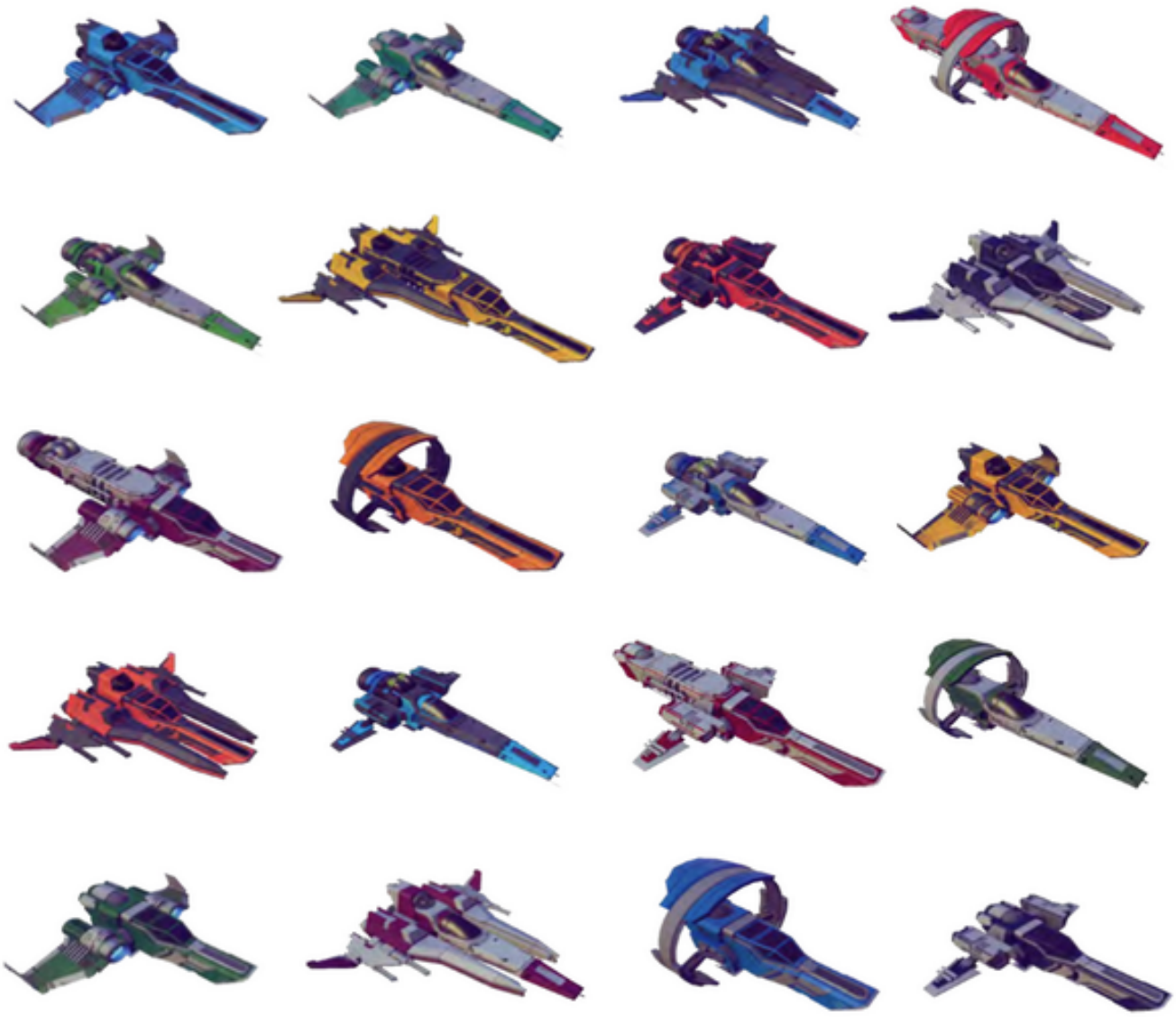
Cities: Skylines

Cities: Skylines

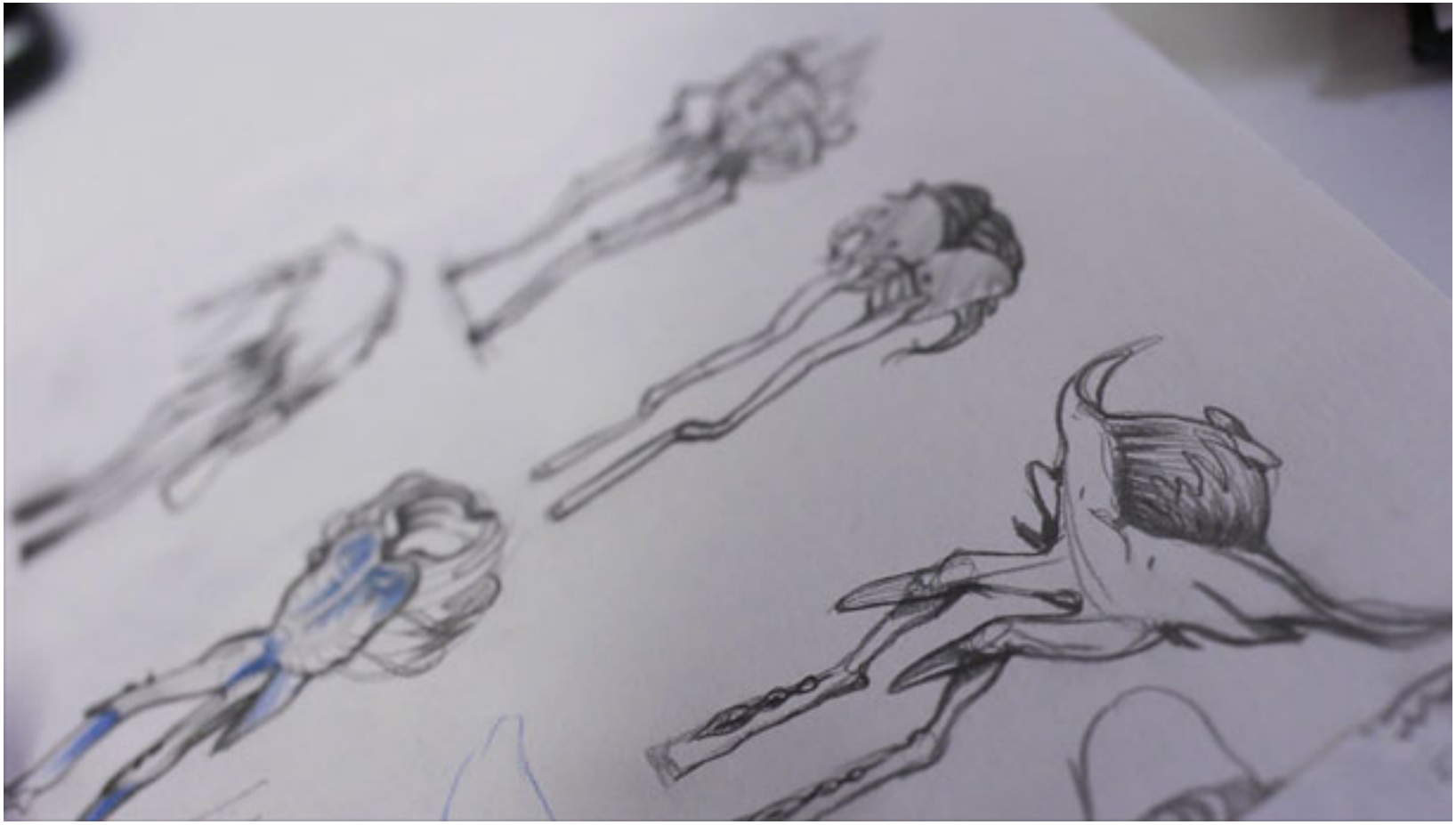
No Man’s Sky

No Man’s Sky

No Man’s Sky



No Man’s Sky



No Man’s Sky

No Man’s Sky

No Man’s Sky

End