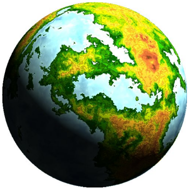
Faculty of Science

Planet Map Generation by Tetrahedral Subdivision

Torben Mogensen, DIKU



PSI 2009

Computer-generated maps

Used for:

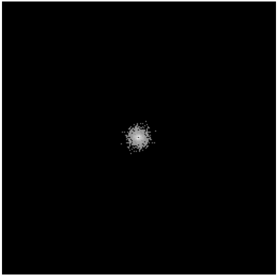
- Strategy games (Age of Empires, Civilization, . . .)
- Arcade games (Virus, Elite, . . .)
- Role-playing games (both computer and table-top)
- Movies (Star Trek II, . . .)
- Computer art

Methods for generation (2D)

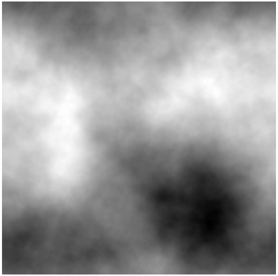
- "Pink" noise generated by Fourier transform
- Recursive mid-point displacement (diamond-square algorithm)
- Tectonic simulation

Fourier transform

Generate random points in frequency domain with amplitude inversely proportional to frequency, then Fourier-transform. Example:



Before FFT

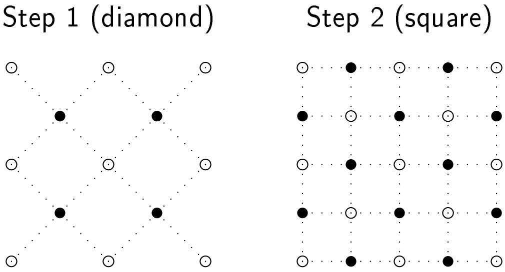


After FFT

Generates periodic maps, computationally expensive.

The diamond-square algorithm

Start with coarse grid and refine by recursive mid-point displacement:



Computationally cheap, can give artefacts.

Methods for generation (3D)

2D methods not suitable for full-planet maps, so we need methods tailored to spheres in 3D space.

- Fourier transform
- Generalized diamond-square algorithm
- Great-circle faulting
- Recursive polyhedral subdivision

Generalized diamond-square algorithm

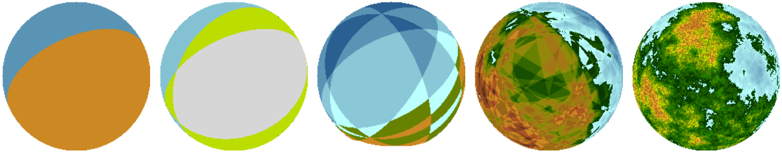
The diamond-square algorithm can also be modified to 3D using a cubical grid and three steps to get to a finer cubical grid:

- **Step 1** Find a value for the middle of each cube using the values at its eight corners.
- **Step 2** Find a value for the middle of each face of the cubes using the values at the four corners of the face and at the midpoints (from step 1) of the two adjoining cubes.
- **Step 3** Find a value for the middle of each edge of the cubes using the values at its two ends and at the midpoints (from step 2) of the four adjoining faces.
- **Repeat** from step 1 until the desired resolution is obtained.

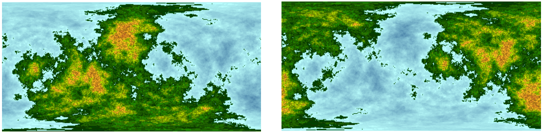
Requires full 3D grid, can have artefacts.

Great-circle faulting

1. Generate random great circle
2. Lower land on one side, raise on other
3. Repeat



Does not need 3D grid, limited zoom, can have artefacts.

Mirror effect: 

Polyhedral subdivision

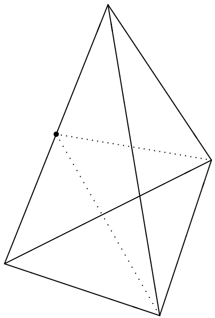
For each point of interest on the sphere do:

1. Embed the sphere inside a polyhedron, where each vertex is assigned a height value
2. Cut the polyhedron into two smaller polyhedra, generating height values for the new vertices from their neighbouring old vertices.
3. Select the polyhedron in which the desired point is located
4. Repeat from step 2 until the polyhedron is small enough
5. Use the average height values of the vertices of the final polyhedron as the height value of the desired point.

Our main contribution: Polyhedral subdivision using tetrahedra.

Tetrahedral subdivision

1. Cut longest edge to divide tetrahedron into two tetrahedra:



   Note that only one new vertex is generated.

2. Identify in which tetrahedron desired point lies and repeat.

Irregular tetrahedra reduce artefacts.
Unlimited zoom at low cost.

Algorithm description

Input: A target point p and four vertices $v_1, \ldots, v_4$ of a tetrahedron, each with the following information:

- Coordinates (x_i, y_i, z_i)
- Seed for pseudo-random number generator s_i
- Altitude value a_i

Repeat:

1. Re-order vertices $v_1, \ldots, v_4$ so the longest edge of the tetrahedron is between v_1 and v_2, i.e., such that $(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2$ is maximized.
2. Define new vertex v_m by

$$\begin{aligned}
(x_m, y_m, z_m) &= ((x_1 + x_2)/2, (y_1 + y_2)/2, (z_1 + z_2)/2) \\
l &= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \\
s_m &= \mathit{random}((s_1 + s_2)/2) \\
a_m &= (a_1 + a_2)/2 + \mathit{offset}(s_m, l, a_1, a_2)
\end{aligned}$$

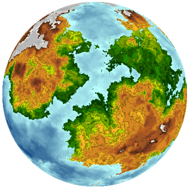
3. If p is contained in the tetrahedron defined by the four vertices v_m, v_1, v_3 and v_4, set $v_2 = v_m$. Otherwise, set $v_1 = v_m$.

Until: l is small enough
Return: $(a_1 + a_2 + a_3 + a_4)/4$

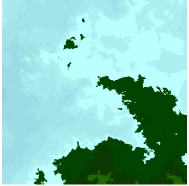
Comparison

400×400 pixels:

Tetrahedra

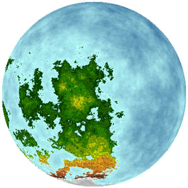


1.03 seconds



1.23 seconds

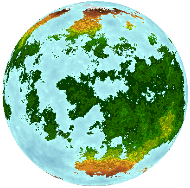
Circles (2000 cuts)



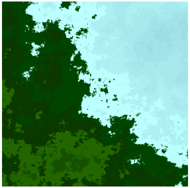
3.46 seconds

4.38 seconds

Circles (20000 cuts)



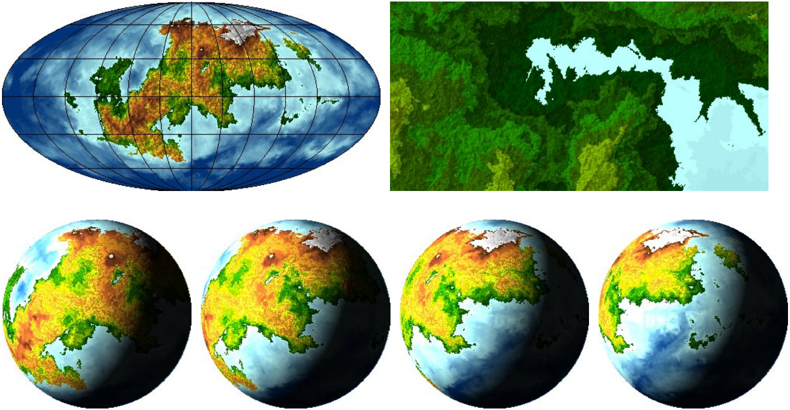
34.2 seconds



43.5 seconds

Sample pictures

Same planet, different views.



`www.diku.dk/~torbenm/Planet/`