NATURE MODELS

I.Kolingerová

Basic Types of Modelled Objects

- Dendrits, Corals
- Coastline
- Landscapes
- Planets
- Clouds
- Plant Ecosystems
- Fire, Smoke, Water

2 main approaches to modelling nature elements:

- Simulation of physical processes
- Emulation of resulting appearance

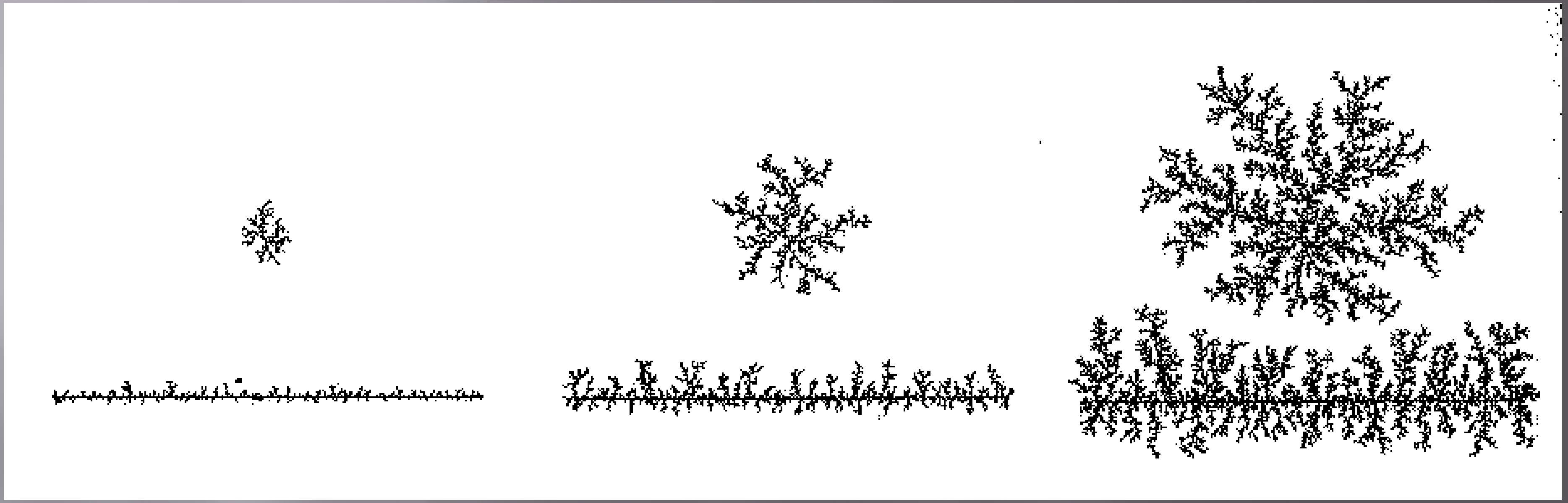
Dendrits, corals : Diffusion Limited Segregation (DLA)

- Straightforward application of Brownian movement
- Electrical discharge, patterns on frozen windows, corals…
- Dilution with a condensation core and flowing molecules
- The moving molecule caught by the core – becomes also a condensation core
- New cores and new shapes due to diffusion

DLA

- In 2D, fractal structures with dimension about 1.7 arise
- 2D: a 2D matrix at the beginning, non-zero elements – condensation cores
- Cycle: particles at the matrix boundaries, Brownian movement
- Particles after travelling to the condensation core connect to it, we set the matrix element as „occupied“
- A particle out of the screen: stop its tracing, a new particle

DLA



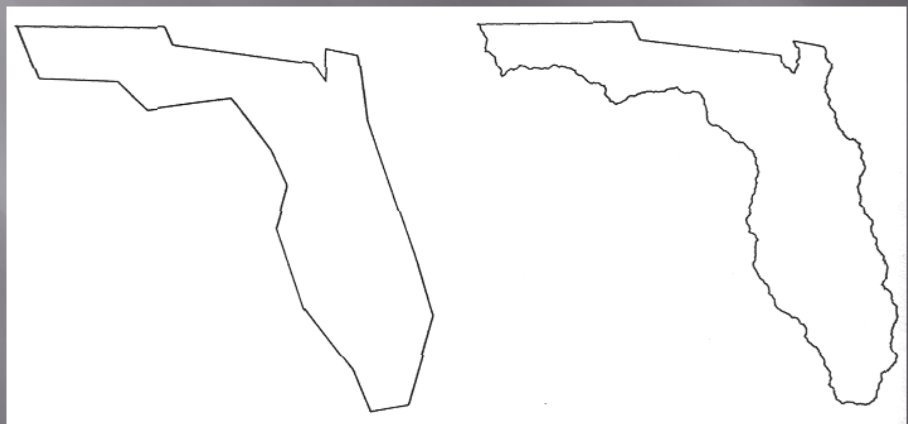
Condensation from a point, condensation from a line segment [Zar04]

DLA

- Modification:
  - To attach the molecule to the core only as late as after several touches (a counter of touches on the surface)
  - To attach the molecule with some probability
- Lengthy (exponential complexity), although acceleration with gradual area filling
- Speedup: outside the minmax box a faster movement

Coastline: random (mid)point relocation

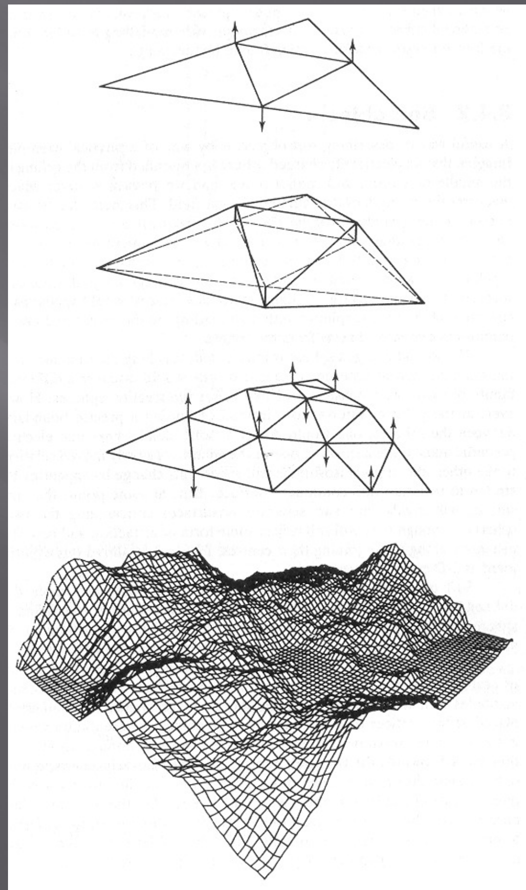
- Islands nearby the coastline – independent objects
- Details see in the fractal lecture



Landscapes

- Random (mid)point relocation: in the fractal lecture
- Other possibilities:

  see the planets

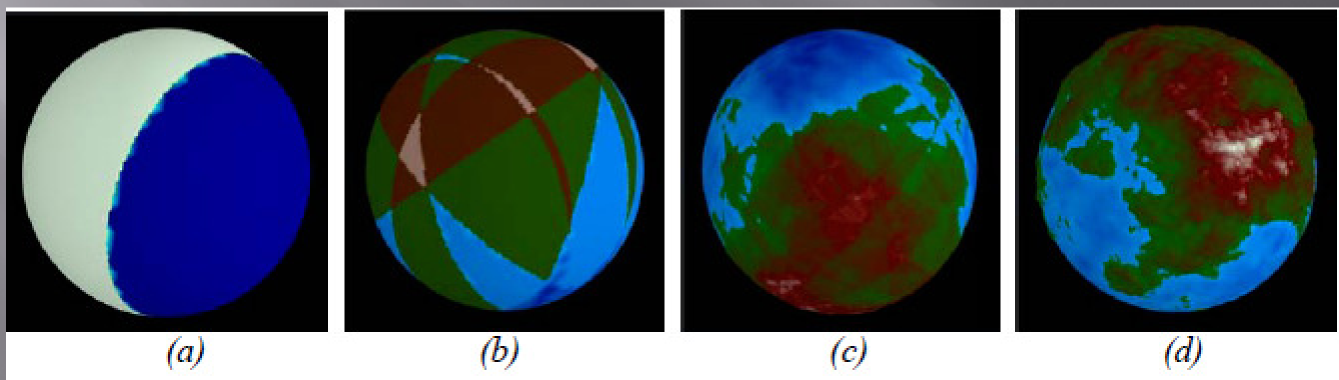


Landscapes

- Results from the Terragen program [TerG]


Planets: random error

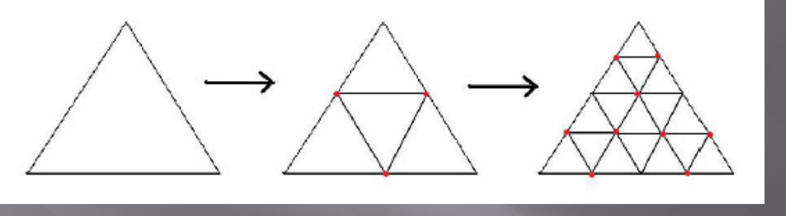
- Input: a sphere-shaped mesh
- One iteration: the sphere is cut by a randomly chosen plane into two hemispheres, the radius of one is randomly increased, of the other decreased



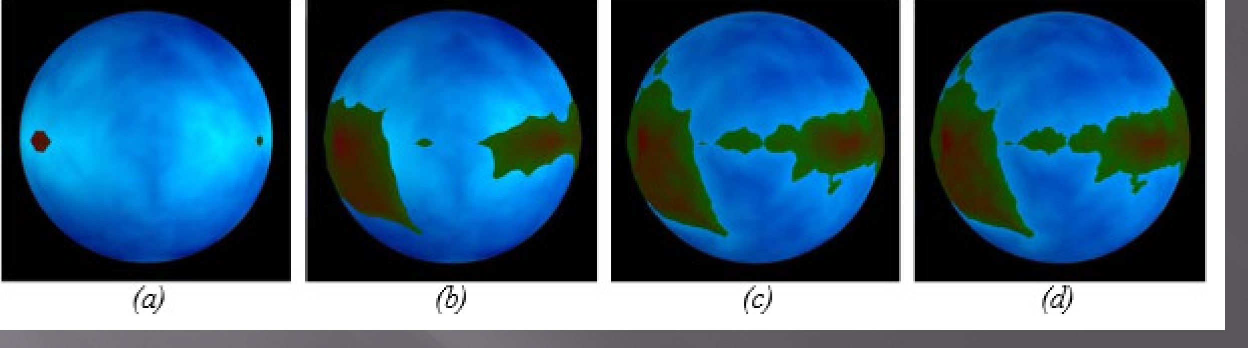
a) after 1 iter., b) after 10 iter., c) after 100 iter., d) after 1000 iter. [Lin07]

Planets: random (mid)point relocation

- Input: a sphere-shaped mesh (2 tetras will do)
- 1 iteration: the triangle is subdivided into 4 smaller,



the new vertex gets the height ~ the average of its parents + random relocation decreased according to the already done number of subdivions



a) after 1 iter., b) after 3 iter., c) after 5 iter., d) after 7 iter. [Lin07]

Planets: multifractal random (mid)point relocation

- To look more authentic, a different level of detail in different parts – relocation is not only decreased according to the number of subdivisions but also multiplied by the parents' heights average => bigger changes in bigger heights

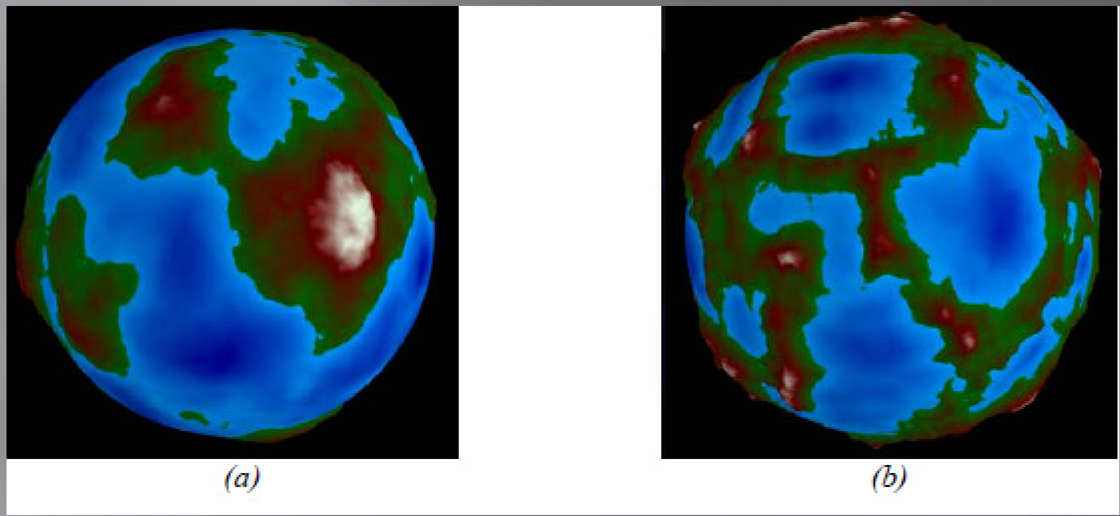
Ex.: Offset = random (-Amp,+Amp);
Offset = Offset/2↑L;
Offset = Offset + Offset*average (parents) *k;

k – height scale change

Planets: Perlin noise

- Input: a sphere arbitrarily defined (usually a mesh)
- 1 iteration is enough
- The height (x,y,z) on the sphere is changed by the value of 3D Perlin noise in this point
- Version 1: a multifractal – computation of Perlin function considers the terrain height of already computed Perlin function
- Version 2: so-called ridge P. noise – slightly modified function produces longer, thinner islands, peninsulas, mountain ridges

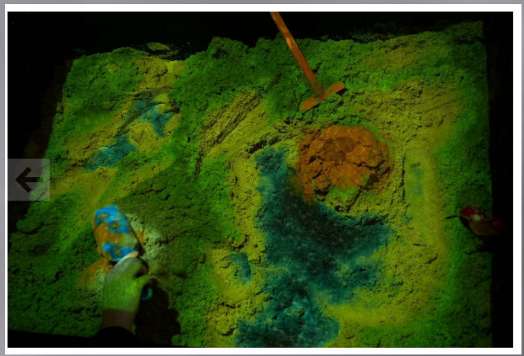
Planets: Perlin noise



a) Normal Perlin noise b) Ridge Perlin noise [Lin07]

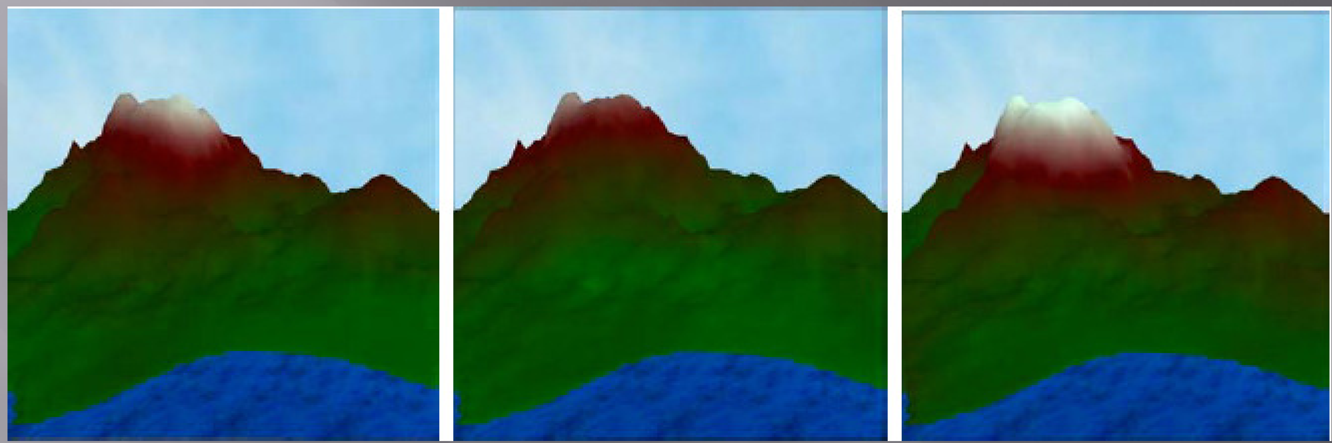
Planets colouring

- Basical approach: to color by height, see geodetic scale or Kinect sandbox



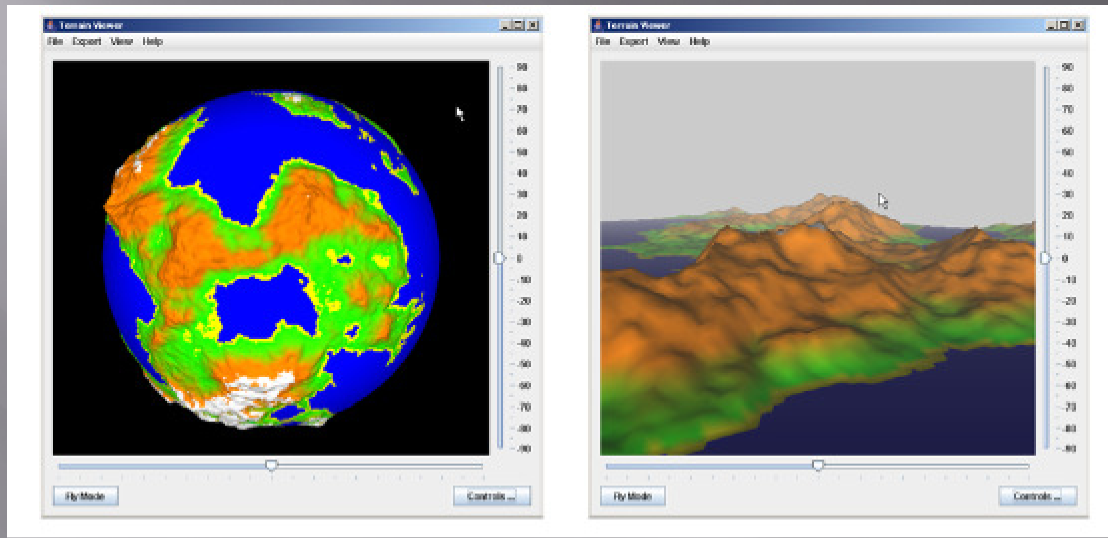
- Colours are interpolated linearly or by a spline + mild randomness, e.g., using Perlin noise or a random deviation, see particle systems

Planets colouring

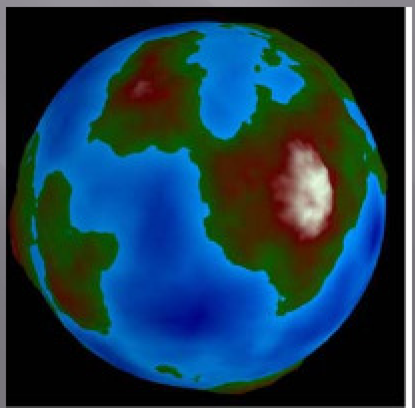


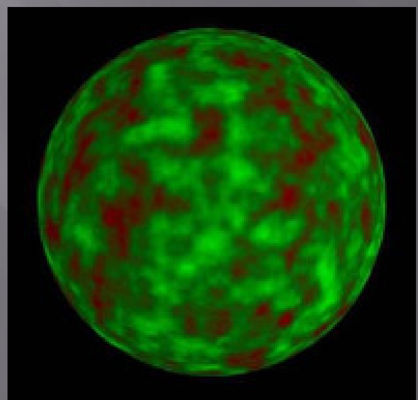
Colouring by height with various random perturbations [Lin07]

Planets



Result from TerraJ [TerJ]

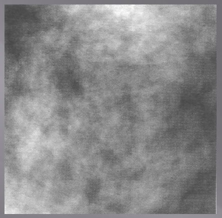




Results using Perlin noise[Lin07]

Clouds: random (mid)point relocation

- 2D clouds: perpendicular projection of a fractal surface, height represented by colours/grey intensities



- 3D: generalize by 1 dimension, the 4th dimension – density
- Projection from 3D to 2D: ray tracing or animation for moving clouds
- Clouds usually serve as a background, thus often billboards and half-transparent layers are used
- The most general approach: 3D noise functions represented as a 3D volume

Clouds by 3D cell automaton

- [Dob00]
- Simple, fast, inaccurate, produces only cumuli

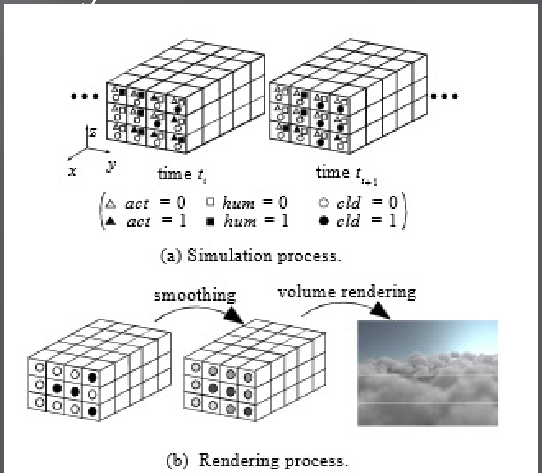
Fig.: Wikipedia

- Clouds – air bubbles, they thin due to the heat from the Earth, rise to the areas of lower pressure where the bubble expands, thus it is cooled and its humidity increases, a phase transition into water drops appears and so a cloud comes into being

Clouds by 3D cell automaton

- In each cell 3 logical variables: humidity hum, cloud cld, phase transition (activation) act
- Rules: formation, extinguishment, shift by wind
- Result: cell status – cloud or not (cld)
- Visualization: smoothing by density computation,

  then voxel visualization



Clouds by 3D cell automaton

- Initialization: cld=0, hum and act have random values 0 or 1
- hum=1 - vapour enough to form a cloud
- act=1 - phase transition from vapour to water should be done
- cld=1 - cloud will be formed

Clouds by 3D cell automaton

- Basic rules:

$$hum(i, j, k, t_{i+1}) = hum(i, j, k, t_i) \wedge \neg act(i, j, k, t_i)$$

$$cld(i, j, k, t_{i+1}) = cld(i, j, k, t_i) \vee act(i, j, k, t_i)$$

$$act(i, j, k, t_{i+1}) = \neg act(i, j, k, t_i) \wedge hum(i, j, k, t_i) \wedge f_{act}(i, j, k)$$

$$\begin{aligned} f_{act}(i, j, k) = {} & act(i+1, j, k, t_i) \vee act(i, j+1, k, t_i) \vee act(i, j, k+1, t_i) \vee act(i-1, j, k, t_i) \\ & \vee act(i, j-1, k, t_i) \vee act(i, j, k-1, t_i) \vee act(i-2, j, k, t_i) \vee act(i, j-2, k, t_i) \\ & \vee act(i, j, k-2, t_i) \vee act(i+2, j, k, t_i) \vee act(i, j+2, k, t_i) \end{aligned}$$

Clouds by 3D cell automaton – cloud extinguishing

- Add probability of extinguishing p_{ext}
- **If** cld=1

  Generate r , 0<=r<= 1, randomly;

  if $r < p_{ext}$ cld := 0 **endif**

  endif
- To enable cloud revival in the cell, change randomly also act and hum to 1 (prob. p_{act}, p_{hum})
- Rules:

$$cld(i,j,k,t_{i+1}) = cld(i,j,k,t_i) \wedge IS(rnd > p_{ext}(i,j,k,t_i))$$

$$hum(i,j,k,t_{i+1}) = hum(i,j,k,t_i) \vee IS(rnd < p_{hum}(i,j,k,t_i))$$

$$act(i,j,k,t_{i+1}) = act(i,j,k,t_i) \vee IS(rnd < p_{act}(i,j,k,t_i))$$

where IS returns T/F of a logical expression

Clouds by 3D cell automaton – wind influence

- Clouds are moved in the direction of wind – variables in cells are moved adequately, wind velocity $v(z_k)$ can be modified according to height, integer values
- Rules:

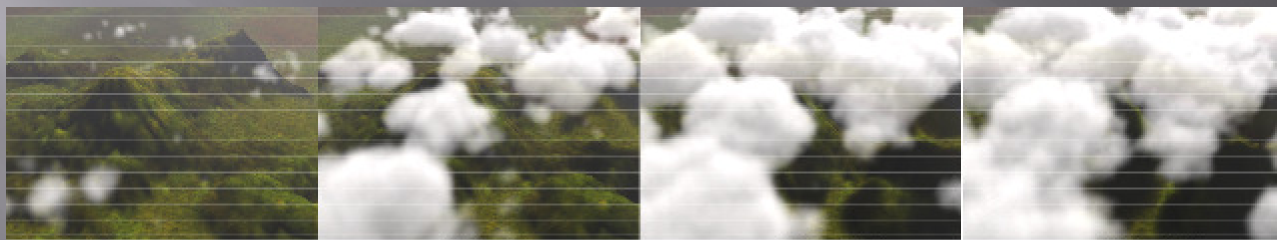
$$hum(i,j,k,t_{i+1}) = \begin{cases} hum(i - v(z_k), j, k, t_i) & pro \quad i - v(z_k) > 0 \\ 0 & jinak \end{cases}$$

$$cld(i,j,k,t_{i+1}) = \begin{cases} cld(i - v(z_k), j, k, t_i) & pro \quad i - v(z_k) > 0 \\ 0 & jinak \end{cases}$$

$$act(i,j,k,t_{i+1}) = \begin{cases} act(i - v(z_k), j, k, t_i) & pro \quad i - v(z_k) > 0 \\ 0 & jinak \end{cases}$$

Clouds by 3D cell automaton – wind influence

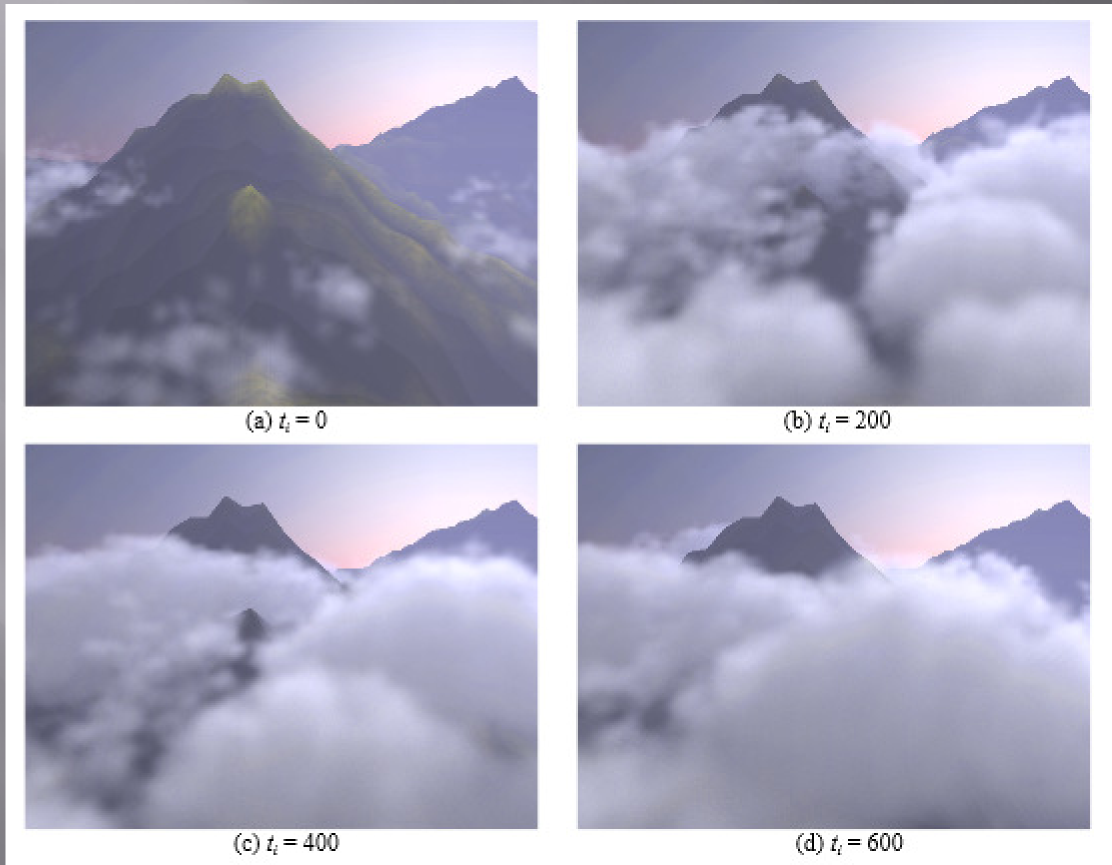
- Movement for animation can be controlled by ellipsoids - p_{act}, p_{hum} bigger near their centres than near edges, p_{ext} vice versa
- The whole ellipsoid is moved
- Position and shape of ellipsoids – random or given by the operator



(a) $t_i = 0$ (b) $t_i = 50$ (c) $t_i = 100$ (d) $t_i = 150$

Results from [Dob00]
256x128x20 cells, random generation of ellipsoids,
p_{ext}, = 0.1, p_{act}= 0.001, p_{hum}=0.1

Clouds by 3D cell automaton

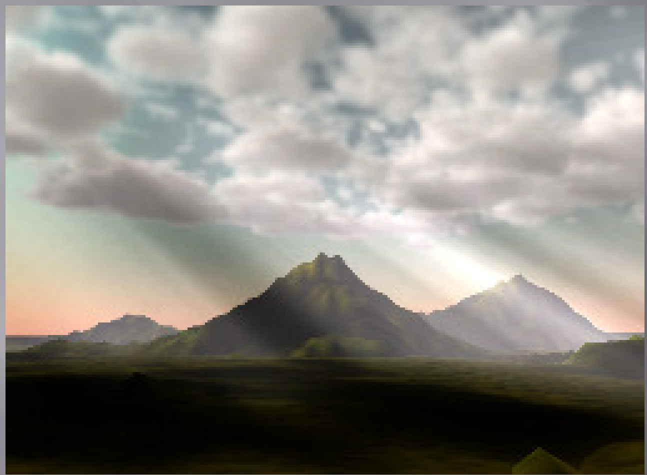


(a) $t_i = 0$ (b) $t_i = 200$ (c) $t_i = 400$ (d) $t_i = 600$

Simulation results from [Dob00]
256x128x20 cells, ellipsoids placed in hand around the mountains,
p_{ext}, = 0.1, p_{act}= 0.001, p_{hum}=0.1, zero inside the mountains

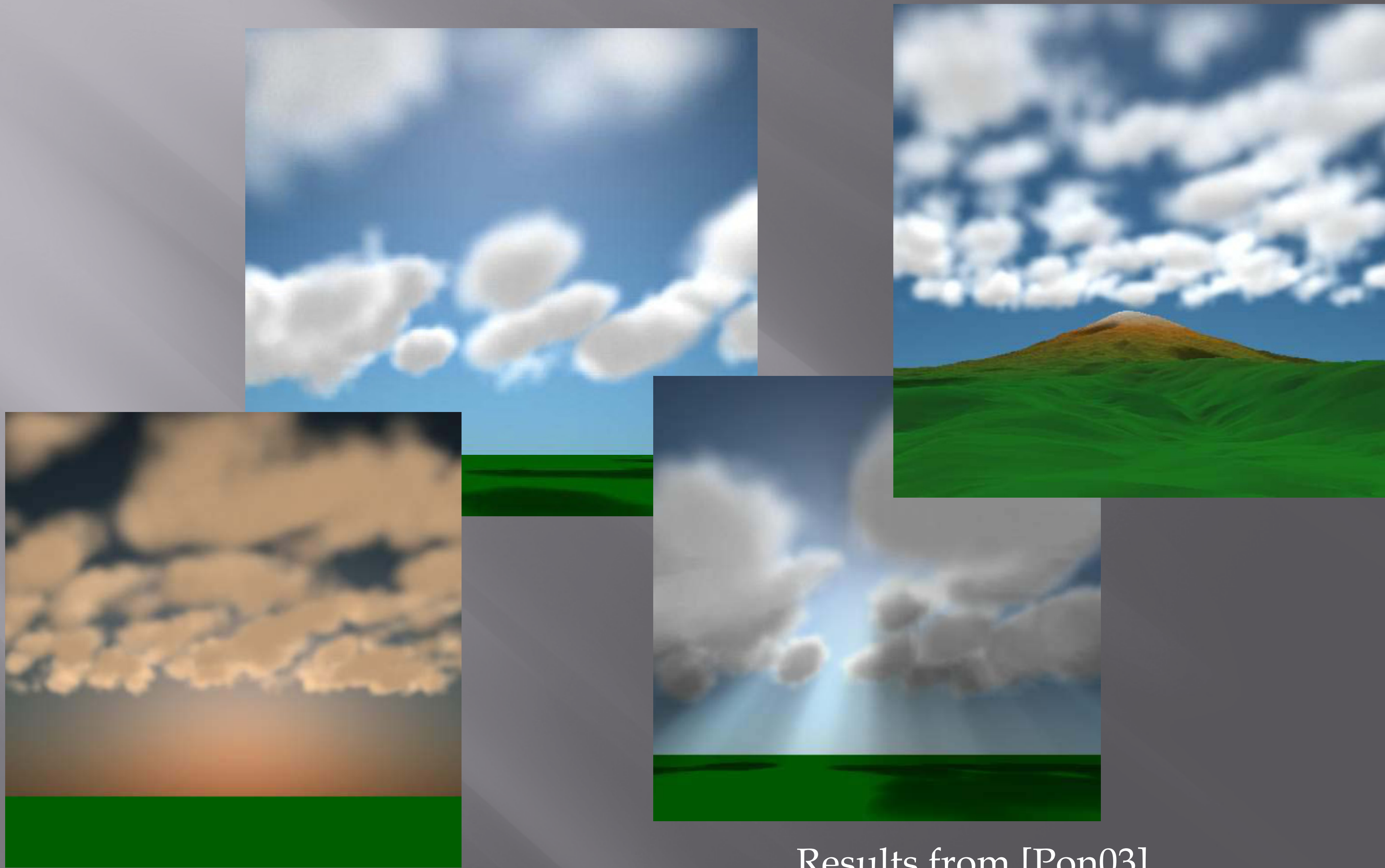
Clouds by 3D cell automaton

- Possibility to combine with light rays



Results of simulation from [Dob00] ,
256x256x20 cells, combination with sun rays,
daily and nightly light

Clouds by 3D cell automaton



Results from [Pon03]

Plants ecosystems [Deu98]

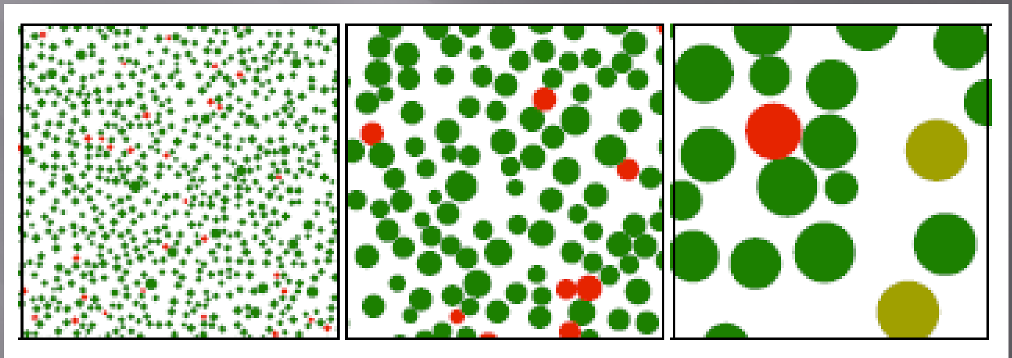
- First a terrain is modelled
- Then a plant population specified
  - How to distribute the plants on the surface:
    - By measurements in countryside
    - Or the simulation of plants' interactions – often cell automata
    - Or the user sets interactively (e.g., by bitmap edit)
    - Or an artificial generation on the base of some "good looking" distribution function

Plants ecosystems

- Simulation example:
  - Each plant grows and exists in sc. ecologic neighbourhood – a circle got by the projection of the plant on the ground
  - At first, circles placed randomly in a grid, with random initial starting radii from a given interval
  - As the plant grows, the neighbourhood grows
  - When two plants collide, the stronger wins, the weaker dies
  - When the plant achieves its limit size, it is considered old and dies

Plants ecosystems

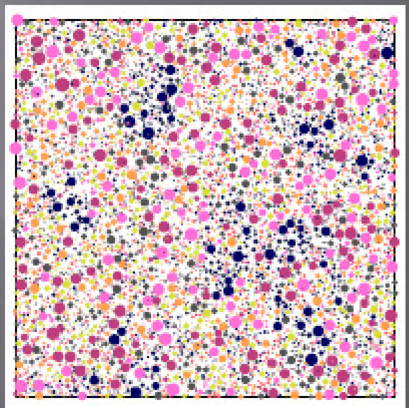
- After several iterations we get visually authentic plants distribution



99, 134 and 164th simulation step; green – common plants, red – dominant, yellow – old plants [Deu98]

Plants ecosystems

- More complex system: more kinds of plants
  - Each kind described by parameters – max. size, average growth, xerophily, average increase of population size in one simulation step, ability to survive in comparison to other plant kinds, etc.
  - If the circles of different plant intersect, the stronger dominates, the weaker may die



Ex.: 8 kinds, blue prefer humidity [Deu98]

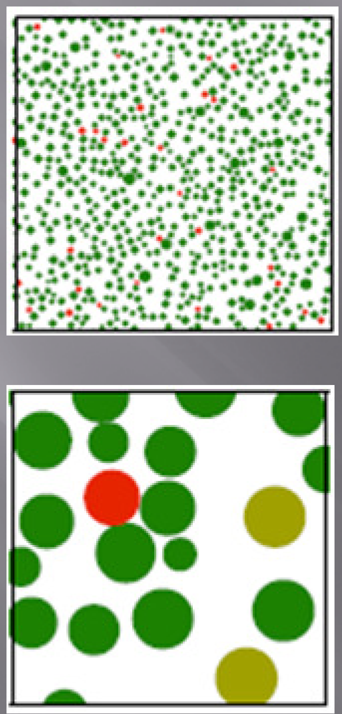
Plants ecosystems

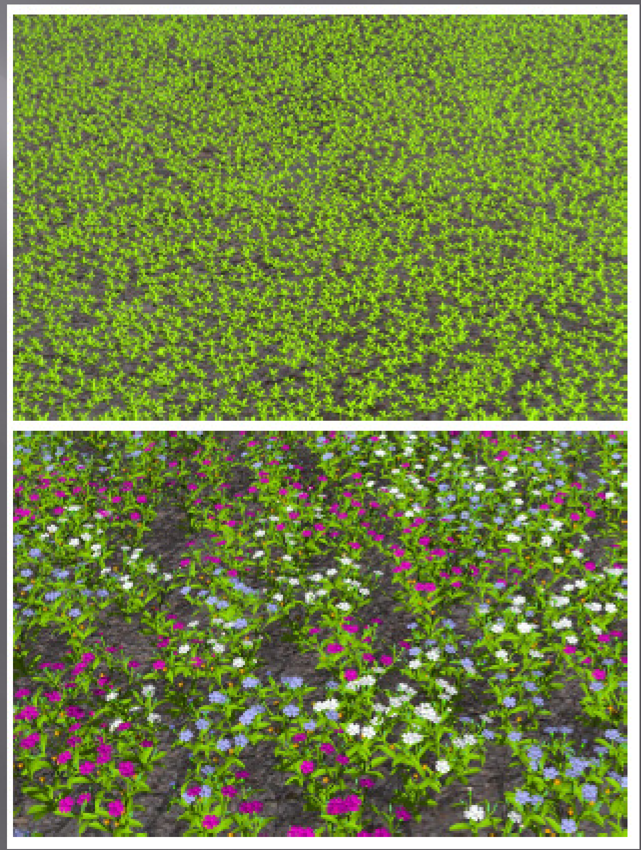
- A plant – L-system or particle system
- Plants generated procedurally– memory savings in comparison to polygons
- To save more memory, instances are used (more plants derived from one) and hierarchy (groups of plants, a plant, branches, leaves, blossoms…)

- Sometimes an agent model is used: agents enter and bring discomfort, they, e.g., try to remove some kind of plants at some place

Plants ecosystems

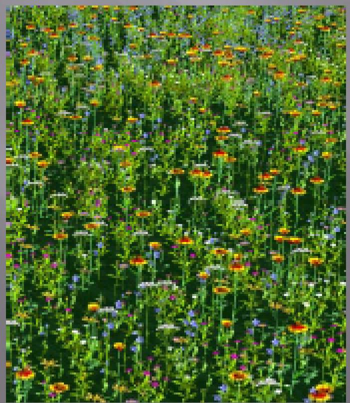
Ex. Results of a simulation after 99 and 164 iterations,
7 different plants for each kind, changed according to their age, 16, 000 plants at a total, due to instancing only 6.7 MB [Deu98]

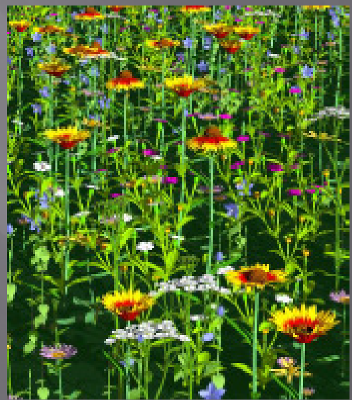


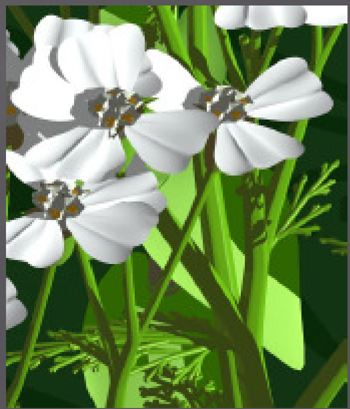


Plants ecosystems

Ex. Zoom on a mountain meadow – 8 kinds of plants,
100,000 plants in the scene, only 151 MB
(polygons would have about 200GB) [Deu98]







Plants ecosystems

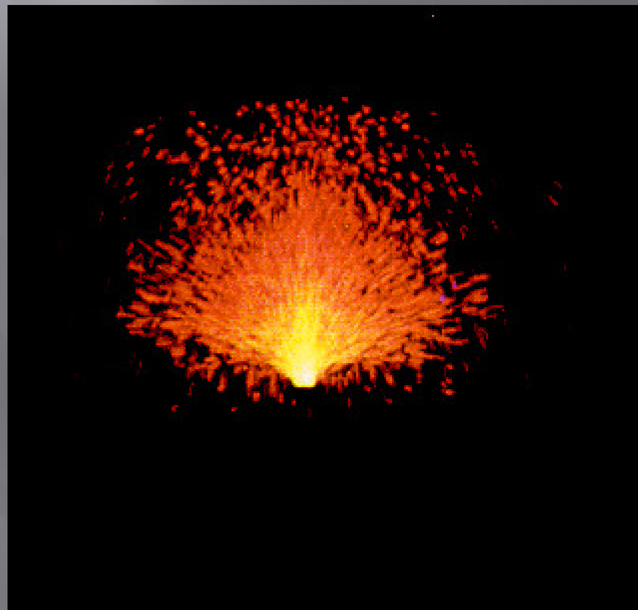
Ex. Lawn from 10 various instances of grass clusters, daisies concentration controlled by a parameter [Deu98]

Plants ecosystems

Scene with a basic distribution of plant systems done interactively [Deu98]

Fire, clouds, water: particle systems

- See a previous lecture



Literature

[Dob00] Dobashi Y., Kanoda K., Yamashita H., Okita T., Nishita T.: A Simple, Efficient Method for Realistic Animation of Clouds, SIGGRAPH 2000, pp.19-28

[Deu98] Deussen O., Hanrahan P., Lintermann B., Měch R.: Realistic Modeling and Rendering of Plant Ecosystems, s.275-286, SIGGRAPH 1998

[Lin07] O. Linda: Generation of Planetary Models by Means of Fractal Algorithms, bakalářská práce, vedoucí ing.J.Sloup, ČVUT, 2007

[Pon03] M. Poneš: Modleování a renderování mraků, bakalářská práce, vedoucí ing.J.Sloup, ČVUT, 2007

[TerG] Terragen project home page, URL: http://www.planetside.co.uk/terragen/

[TerJ] TerraJ project home page, URL: http://terraj.sourceforge.net/

[Zar04] J.Žára, B. Beneš, J. Sochor, P.Felkel: Moderní počítačová grafika, Computer Press, Praha, 2004