Computer Animation
Natural Phenomena SS 18

Prof. Dr. Charles A. Wüthrich,
Fakultät Medien, Medieninformatik
Bauhaus-Universität Weimar
caw AT medien.uni-weimar.de

Introduction

- One of the most challenging parts of animation systems is trying to model nature
- Many techniques and special mathematics is needed to do so
- Since nature is complex, it is often very time consuming to simulate nature
- Typical simulations include plants, water, clouds

Plants

- Plants possess an extraordinary complexity
- Lots of work was done on modeling the static representation of plants (Prusinkiewicz & Lindenmayer)
- Their observation was that plants develop according to a recursive branching structure
- If one understands how recursive branching works, one can model its growing process
- On the book there is one page explaining the underlying botanical concepts

L-systems

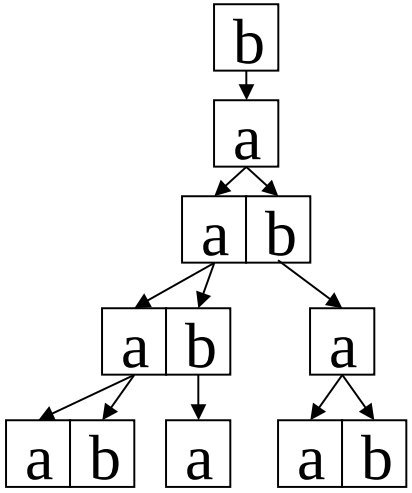
- Plants are simulated through L-systems
- L-Systems are parallel rewriting systems
- Simplest class of L-systems: D0L-system
  - D: deterministic
  - 0: productions are context free
- A D0L-system is a set of production rules $\alpha_i \longrightarrow \beta_i$, where
  - α_i: predecessor symbol
  - β_i: sequence of symbols
- In deterministic L-systems, α_i occur only once on the left hand side of the rules

- An initial string, the *axiom*, is given
- All symbols in the string that have production rules are applied to the current string at each step
  - This means replacing all symbols with a production rule
  - If there is no production rule for a symbol α_i, the production $\alpha_i \longrightarrow \alpha_i$ is applied
- Applying all production rules generates a new string
- This is done recursively until no production rules can be applied

Example

- Let the alphabet consist of the letters *a,b*
- Suppose we have two production rules:
  - $A \longrightarrow ab$
  - $B \longrightarrow a$
- And suppose that the axiom is *b*
- Then we obtain that we can generate the following strings

- *b*
  a
  ab
  aba
  abaab
  ....
- Or, more figuratively:

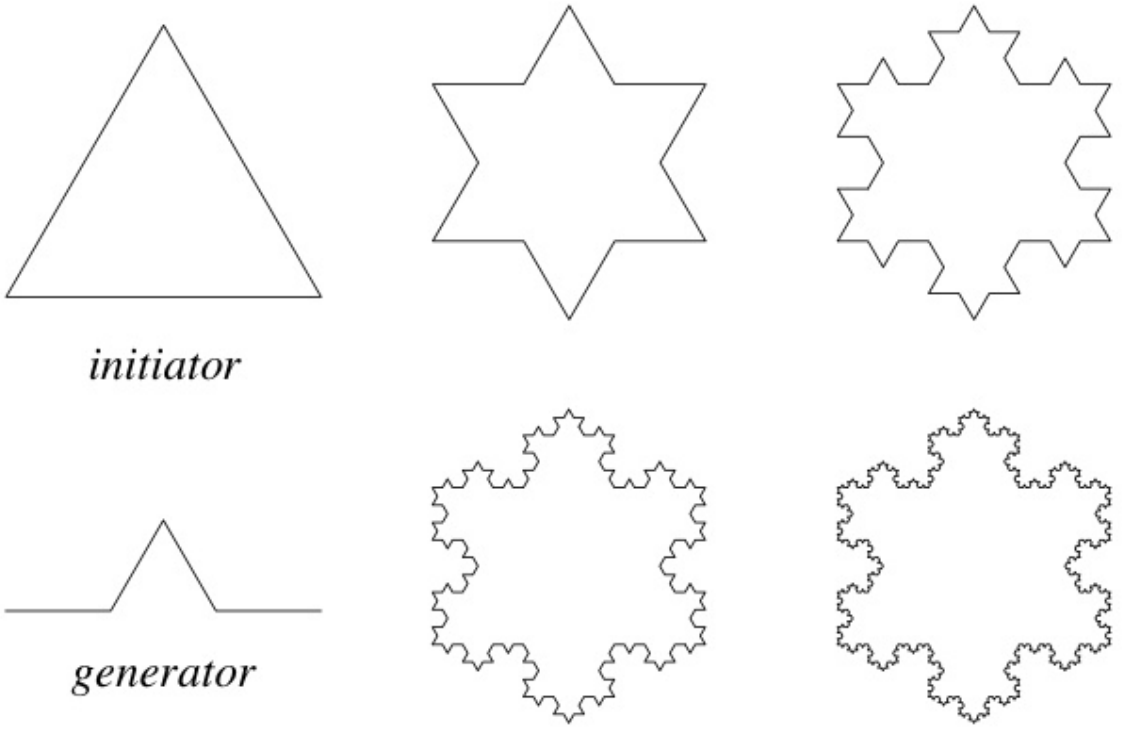


Interpreting L-systems

- The strings produced by L-systems are just strings
- To produce images from them one must interpret those strings geometrically
- There are two common ways of doing this
- Geometric replacement: each symbol of a string is replaced by a geometric element
  - Example: replace symbol X with a straight line and symbol Y with a V shape so that the top of the V sligns with the end of the straight line
  - Example: XXYYXX

The Koch curve

- The Koch curve is one of the most famous Lindenmayer generated systems.
- Fractal curve!

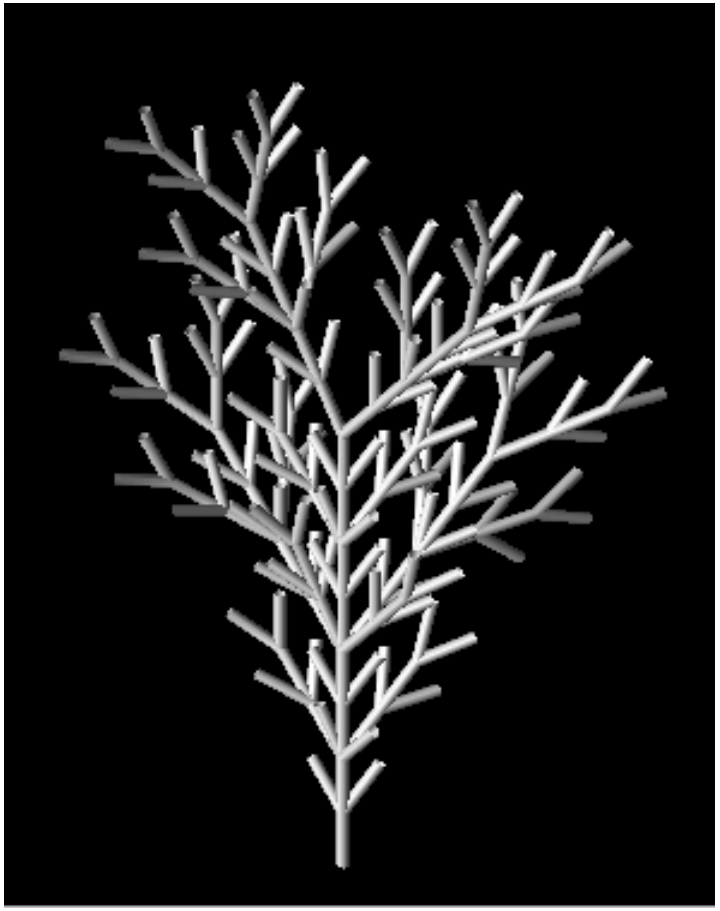


Interpreting L-systems

- Use turtle graphics: the symbols of the string are interpreted as drawing commands given to a simple cursor called *turtle*
- The state of a turtle at a given time is expressed as a triple (x,y,α) where x,y give the coordinate of the turtle in the plane, and α gives the direction of it is pointing to with respect to a given reference direction
- Two more parameters defined by the user are also used:
  - d: linear step size
  - δ: rotational step size
- Given the reference direction, the initial state of the turtle (x_0,y_0,α_0), and the parameters d and δ the user can generate the turtle interpretation of the string containing some symbols of the alphabet

L-systems

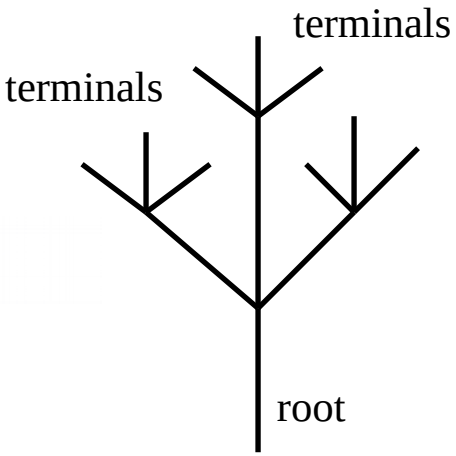
- Even more useful: if the symbols are interpreted as cells, or parts of a plant, the generation process of an L-system can simulate the growing of a plant
- The interpretation would be: substitute last year‘s leaf buds with a small piece of branch
- Or,, a branch will be replaced by three branches centered in the direction of the previous branch and having an angle between them of 22 degrees“
- Through this, the growing process of a plant can be simulated



Courtesy *Hung-Wen Chen*, Cornell University

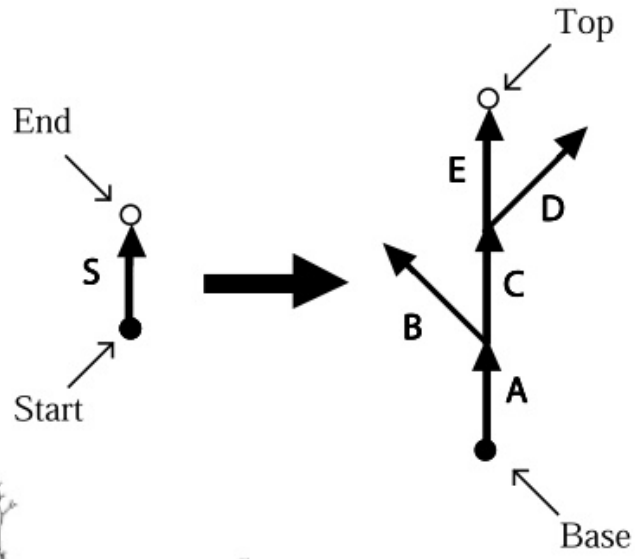
Rooted trees

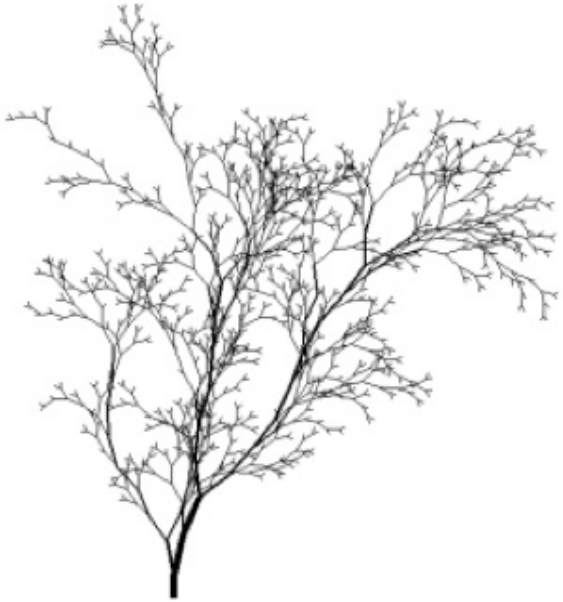
- Lindenmayer systems (or L-systems) such as the ones seen before are great to model branching structures, such as trees
- Among branching structures, *rooted trees* are used to model trees
- A rooted tree is a collection of edges which are labeled and directed
- A rooted tree has one particular edge: the *root*.
- All edges are similar to branch segments on a tree:
  - they connect to a father segments, and
    - either have themselves children segments, in which case they are called *intermediate* segments
    - Or have no children and are called *terminal* segments or *apex*



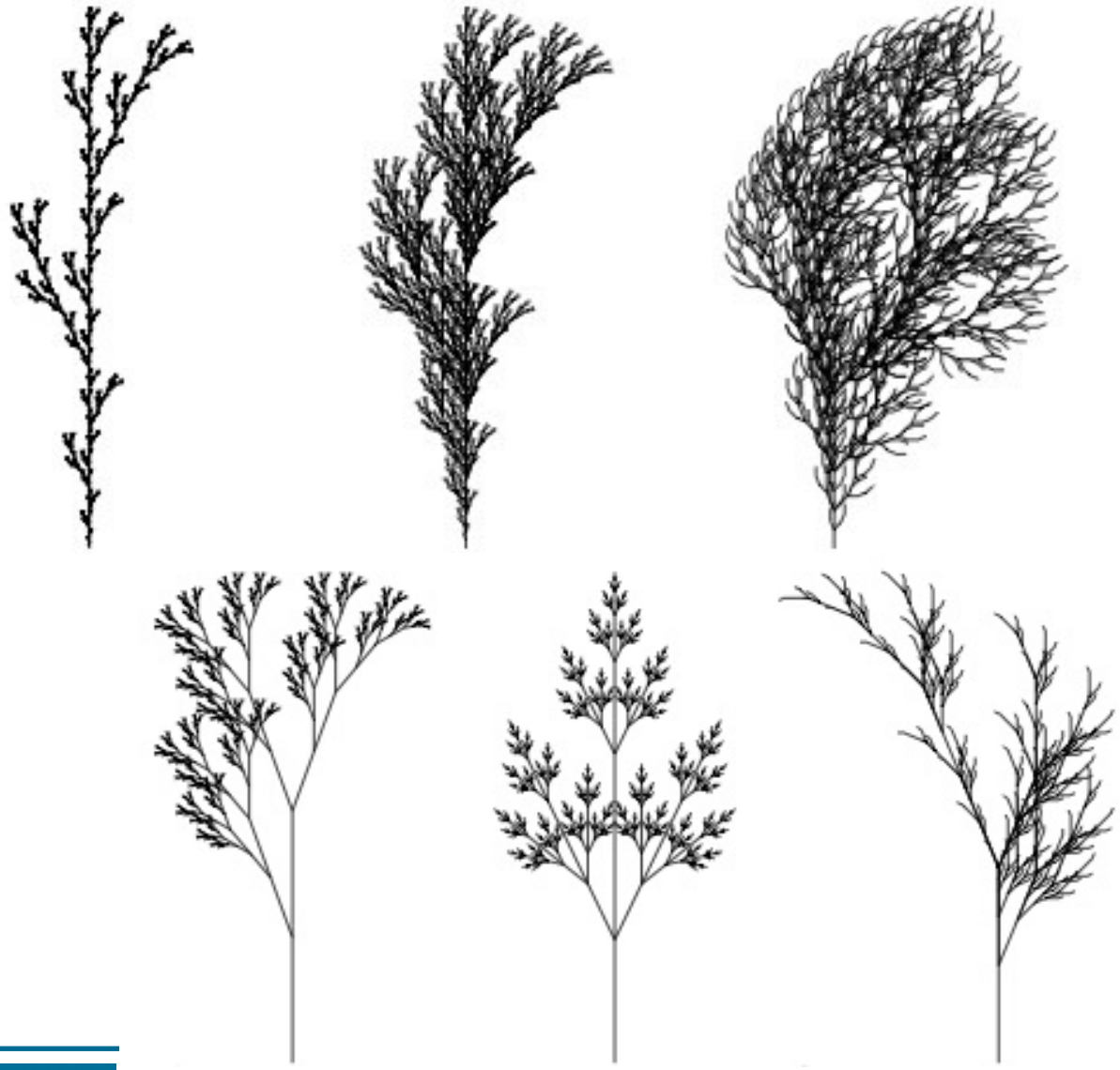
Axial trees

- Among rooted trees there are axial trees:
  - For each intermediate node, at most one of its children is distinguished
  - The other children are called *side* or *lateral segments*
- An *axis* is a sequence of incident segments such that
  - It originates at the root or at a lateral branch
  - It ends with an apex
  - It is made of consecutive segments
- An axis, with its descendants is called a *branch*.
- On trees, productions “look” much more complicated, but they are not: simple substitutions
- If productions are context free, we speak of a tree D0L-system



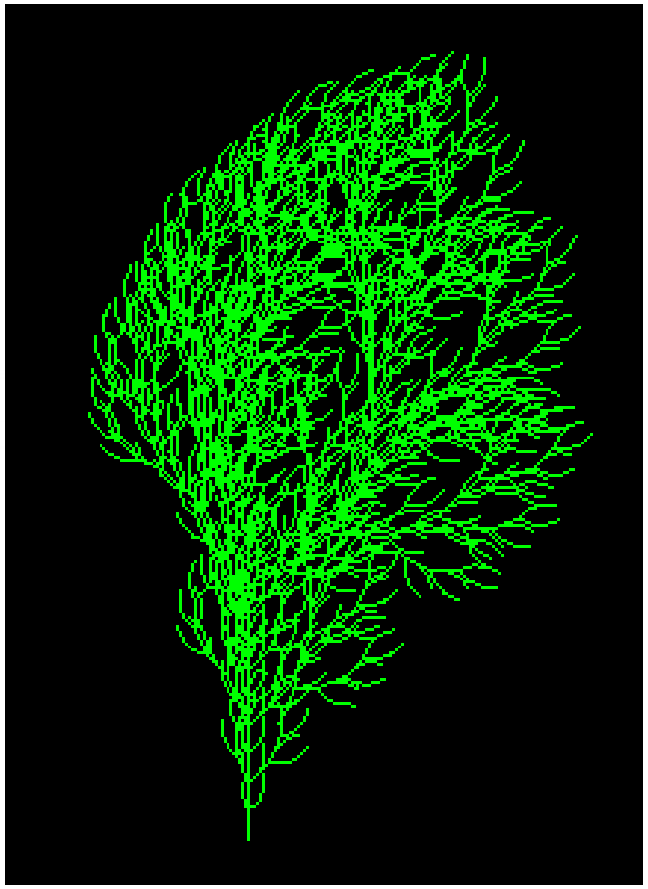


Some examples



Bracketed L-systems

- In bracketed L-systems, brackets are used to mark the beginning and end of additional offshoots of the main branch
- Production rules are context free but non deterministic, i.e. there are more than one production rule per symbol
- Which one is chosen? It can either be chosen at random or follow certain rules, which can be derived for example by „simulated temperature of that year“
- Let’s get into more detail



Courtesy *Hung-Wen Chen*, Cornell University

Stochastic L-systems

- Context free L-systems are are deterministic:
  - given the same seed, they reproduce always the same tree
- Is this realistic?
  - Not much

Stochastic L-systems

- Context free L-systems are are deterministic:
  - given the same seed, they reproduce always the same tree
- Is this realistic?
  - Not much
  - If nature would handle like this, we would have all trees looking the same.
  - This is why scientists have added a small but relevant modification to L-systems…..

Stochastic L-systems

- Context free L-systems are are deterministic:
  - given the same seed, they reproduce always the same tree
- Is this realistic?
  - Not much
  - If nature would handle like this, we would have all trees looking the same.
  - This is why scientists have added a small but relevant modification to L-systems…..
  - …by adding randomness to the transition function!

Stochastic L-systems

- And how do I add randomness to the transition function?
- Remember, an L-system $L=<\Sigma,\alpha,\Pi>$ was a triplet of
  - An alphabet Σ.
  - An axiom $\alpha \in \Sigma^*$.
  - A set of production (or *rewriting*) rules $\Pi: \Sigma \rightarrow \Sigma^*$.
- Instead of fixed rewriting rules, we need to add a probability function

  $$P: \Pi \rightarrow [0,1]$$

  that associates to the production rules a probability of being chosen.

- For a given symbol A of the alphabet we will have therefore a list of production rules

  $A \rightarrow \alpha_0$ (prob. p_0)
  $A \rightarrow \alpha_1$ (prob. p_1)
  ....
  $A \rightarrow \alpha_h$ (prob. p_h),

  whereby the sum of these probabilities equal 1.
- Example:

  $A \rightarrow ABA$ (prob. 25%)
  $A \rightarrow ABAA$ (prob. 75%)
- Any time we have to apply a production rule to the letter of the alphabet, we draw a random number in [0,1].
- If it is smaller than 0.25 we choose the 1st production, otherwise the 2nd.
- The new productions are called *stochastic productions*

Stochastic L-systems

- This way from the same axiom one can derive different branching shapes..
- …and more realistic pictures

Stochastic L-systems

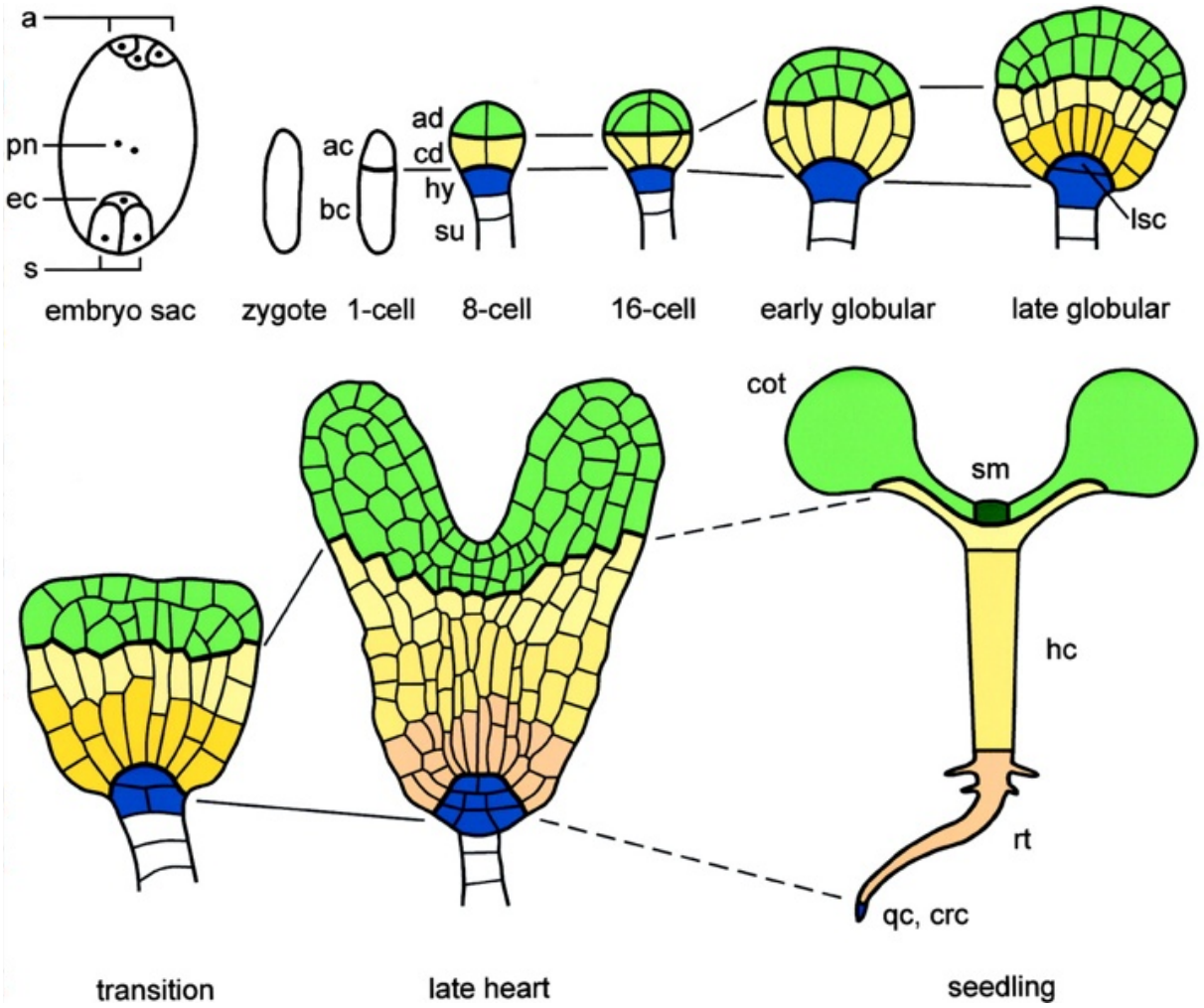
- This way from the same axiom one can derive different branching shapes..
- …and more realistic pictures

- It looks good, but does not model what nature does!

A further example

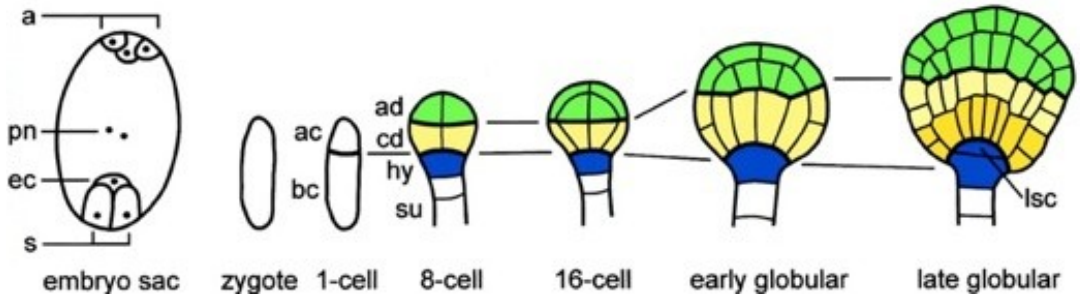
- Let us get back to the example of the last time….
- For a long time, botanists have been trying to describe and understand how plants develop
- For a long time, the plant called *arabidopsis thaliana* has been used as a good observation object for plant growth
- There are plenty of observations of how this plant develops from a cell

A further example



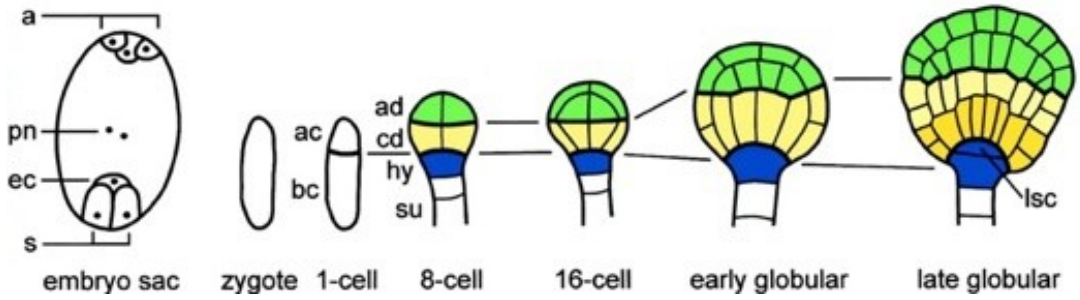
First phase of growth *arabidopsis thaliana*

Reasoning on the second example



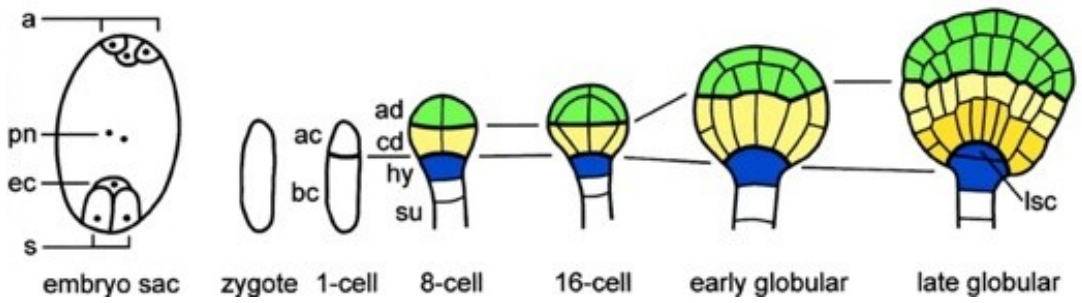
- What can one learn from this picture?

Reasoning on the second example



- What can one learn from this picture?
  - Cells differentiate in time
  - Start growing in different ways
    - depending on their position
    - and on their surroundings
  - Unfortunately our cell development model is too rigid: a cell develops always in the same way.
  - If the production rules in Π are such that the left hand side is a single element, the Lindenmayer system is called a *D0L-system*, or *context free Lindenmayer* system
  - BUT one can learn from this!

Reasoning on the second example

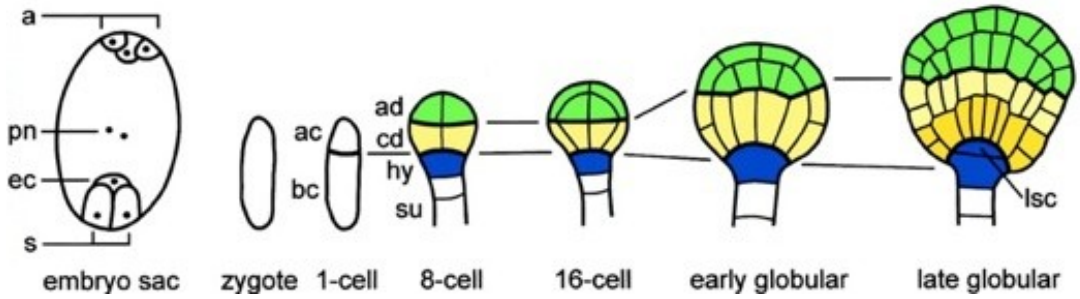


- Remember: we had for D0L-systems that the productions were

  $$\Pi: \Sigma \rightarrow \Sigma^*$$

  associating $a_i \rightarrow \alpha_j$
- Who says that at the left hand side of the productions we need to have a single element?
- Can't it be that depending on what the symbol to which we apply the production has around we can have two different production rules applied?
- If a_i is preceded by a certain string, and followed by a certain other string, we apply one production rule.
- If instead it is preceded and followed by other strings we apply another rule.

Reasoning on the second example



- In this case we have that the productions are still defined similarly

  $\Pi: \Sigma \rightarrow \Sigma^*$

  but they associate to a_i

  $\alpha_r a_i \alpha_s \rightarrow \alpha_j$

  in case that a_i is preceded by α_r and followed by α_s, while if it is preceded and followed by two different strings α_p and α_q, with

  $\alpha_p \neq \alpha_r$ and $\alpha_q \neq \alpha_s$, then a different production rule is applied:

  $\alpha_p a_i \alpha_q \rightarrow \alpha_m$

  with $a_m \neq a_j$.
- This means that AABC gets transformed into AADC, but CCBA into CCEA
- L-systems such that they use such production rules are called *D2L-systems*, whereby the 2L indicates that both sides of the letter to be substituted influence the production to be applied.
- If instead the production rule is influenced only by one side, they are called *D1L-systems.*
- D1L- and D2L-sytems are said to be *context sensitive systems*.

Contex-sensitive L-systems

- Let us recap the definition:
- A *D2L-system* is a triplet $L=<\Sigma,\alpha,\Pi>$, where
  - Σ is an alphabet, Σ^* is the set of the juxtaposition of letters of Σ.
  - $\alpha \in \Sigma^*$ is an axiom
  - The context sensitive set of production rules applied to letters of the alphabet is

$$\Pi: \quad \Sigma \rightarrow \Sigma^*$$

$$\alpha_r a_i \alpha_s \rightarrow \alpha_j$$

$$\alpha_p a_i \alpha_q \rightarrow \alpha_m$$

Contex-sensitive L-systems

Context-sensitive L-systems

- Context sensitive L-systems model better nature, since you can make the production rules dependent on the context you immerse your L-system into.

- For example, trees in a forest develop new branches only at the tip, so as to maximize light captured…

Context-sensitive L-systems

- Context sensitive L-systems model better nature, since you can make the production rules dependent on the context you immerse your L-system into.

- …or external forces, such as wind, force trees to develop opposite of the direction of prevailing wind…

Context-sensitive L-systems

- Context sensitive L-systems model better nature, since you can make the production rules dependent on the context you immerse your L-system into.

- ..or trees moving away from the façade of a house to collect more light.

Context-sensitive L-systems

- Context sensitive L-systems model better nature, since you can make the production rules dependent on the context you immerse your L-system into.
- ..or trees moving away from the façade of a house to collect more light.
- All these factors can be incorporated into the production rules to model nature.

Parametric and timed L-systems

- In parametric L-systems, symbols can have one or more parameters associated to them
- These parameters can be set and modified by the productions of the L-system
- Additionally, optional conditional terms can be associated with the productions
- All this to simulate differences in the change through time in a plant

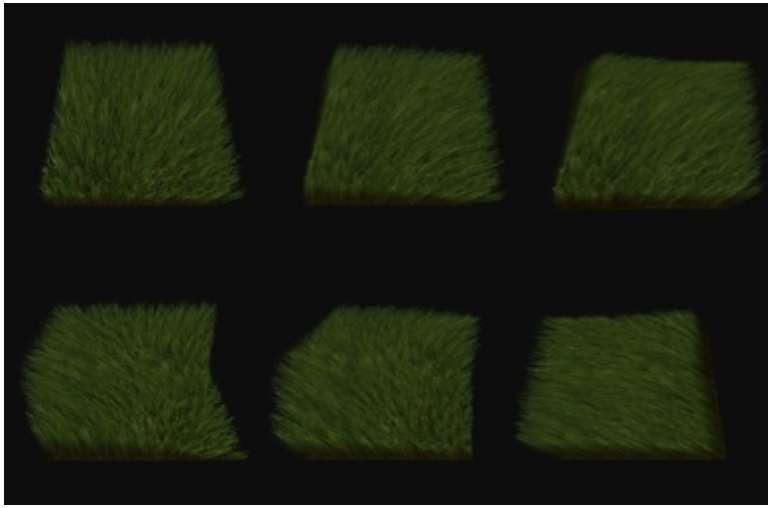
Parametric and timed L-systems

- Timed L-systems add two things
  - A global time variable helping control the evolution of a string And a local age value τ_i assoc. with each letter μ_i.
  - The production
    $(\mu_0,\beta_0) \rightarrow ((\mu_1,\alpha_1),...,(\mu_n,\alpha_n))$
    indicates that μ_0 has a terminal age of β_0.
  - Each symbol has one and only one terminal age
  - When a new symbol is generated, it is initialized at age 0 and exist until it reaches
  - After its lifespan ends, the symbol will become something else and „mutate“

Let’s look at where we are!

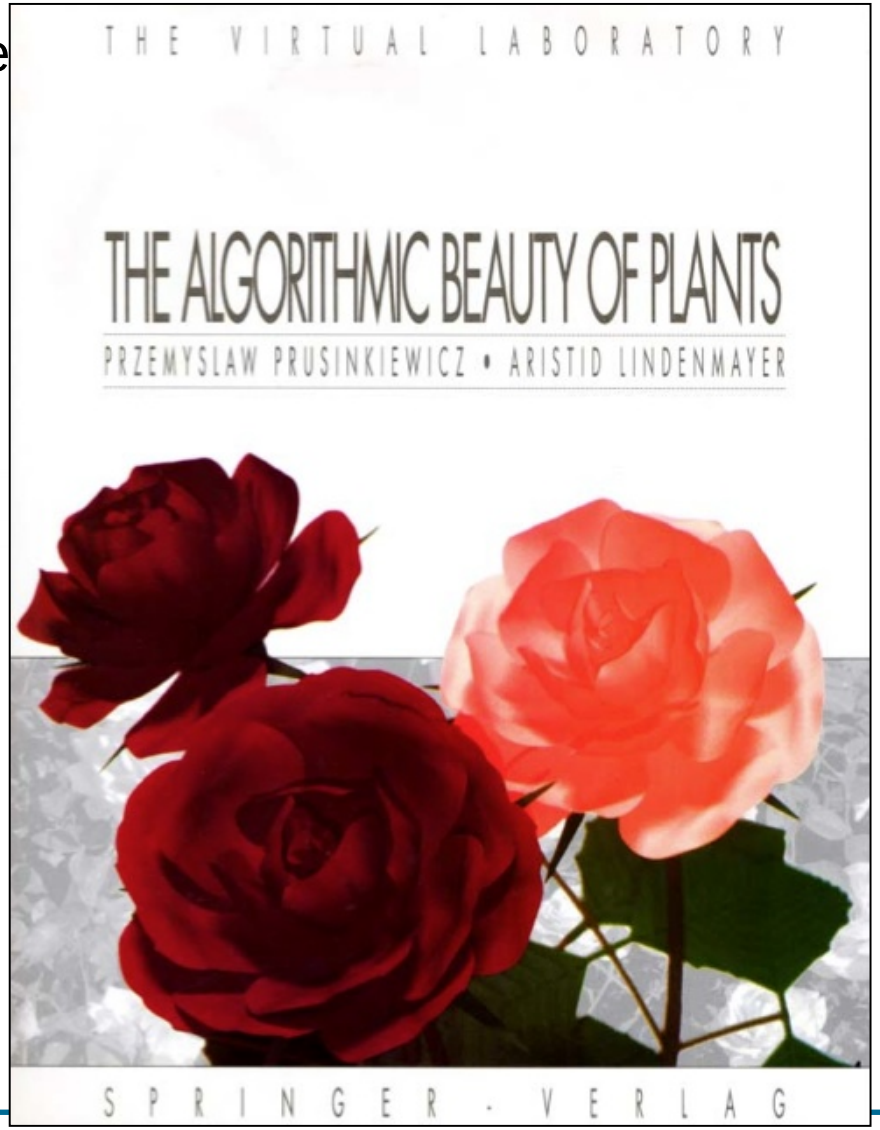
Forests

- Do the trees of a forest have to look similar to each other?
- What about forest borders?
- How do I render the forest?
- Close view?
- Far view?
- Wind?



More information

- Read the literature, if you are interested!



Water

- Water is challenging: its appearance and motion take various forms
- Modeling water can be done by adding a bump map on a plane surface
- Alternatively, one can use a rolling height field, to which ripples are added later in a postprocessing step
- When doing ocean waves, water is assumed not to get transported, although waves do travel either like sinus or cicloidally
- If water has to be transported (=flow) this adds a lot of computational complexity

Small waves

- Simple way: big blue polygon
- Add normal perturbation with sinuisoidal function and you have small waves
- Usually you would start sinuisoidal perturbation from a single point called source point
- Sinus perturbation has, however crests of the same amplitude. This is not so realistic, and waves can be perturbated through smaller radial waves to achieve non self-similarity
- Similarly, one can superimpose more different sinuisoidal waves to achieve an interesting complex surface
- All these methods give a first decent approximation, but not always very realistic

Wave functioning

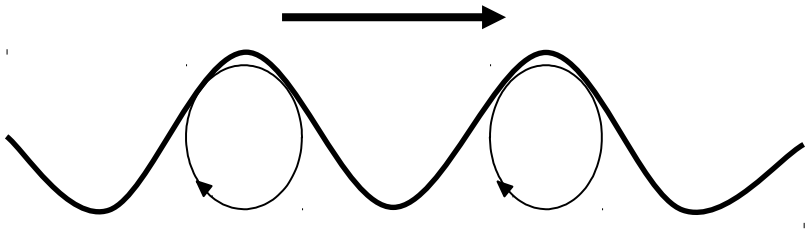
- A better way of doing water is to incorporate physical laws
- There is a variety of types of waves:
  - Tidal waves
  - Waves created by the wind
- In general, at a distance *s* of the sourcepoint we have that

$$f(s,t) = A\cos\left(\frac{2\pi(s - (Ct)}{L}\right)$$

- Where
  - *A* maximum amplitude
  - *C* speed of propagation
  - *L* wavelength (it holds *C=L/T*, with *T* time for one wave cycle to pass a given point (freq.))
  - *t* time
- Waves move differently from the water itself. A water particle would almost move circularly:
  - Follow wave crest, sink down and move backwards, then come up again

Wave functioning

- Small waves (with little steepmness) work almost like sinus curves
- The bigger they get, the more they look like a sharply crested peak, i.e. They approach the shape of a cycloid (point on wheel)
- When a wave approaches the shoreline, at an angle, the nearest part to the coastline slows down
- While its speed C and wavelength L reduce near the coast, its period stays the same and amplitude remains the same or increases.
- But because the speed of the water particles remains the same, the wave tends to break as it approaches the shore
- Litterally, particles are „thrown forward“ beyond the front of the wave



Gaseous Phenomena

- Gas is quite complicated to do
- But occurs often (smoke, fire, clouds)
- Fluid dynamics long studied, and applies to both gas and liquids
  - Uncompressible --> Liquid
  - Compressible --> Gas
- There are different types of movement in fluids
  - Steady state flow: velocity and acceleration at any point in space are constant
  - Vortices: circular swirls of material,
    - depend on space and not on time in steady state flow
    - In time varying flow, particles carrying non zerovortex strength travel through the environment and „push“ other particles. This can be simulated by using a distance-based force

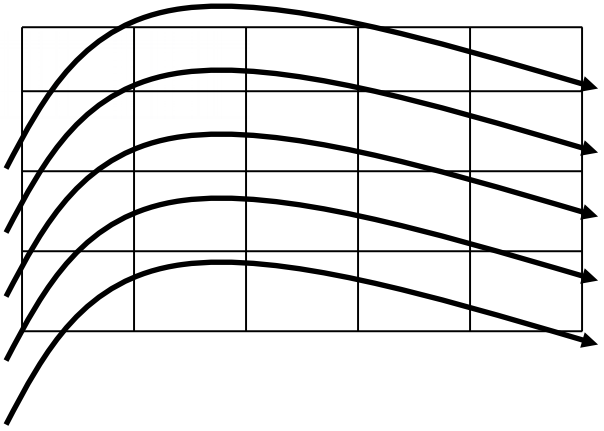
Gaseous phenomena

- There are 3 main approaches to modeling gas:
  - Grid-based methods (Eulerian formulation)
  - Particle-based methods (Lagrangian formulation)
  - Hybrid methods

Grid-based method

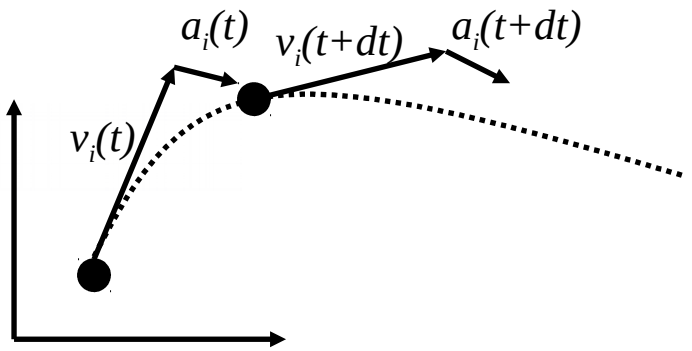
- Decomposes space into grid cells
- Density of gas in a cell is updated from time to time step
- The density of gas in a cell is used to determine the visibility and illumination for rendering
- Attributes of gas in a cell can be used to track the gas travelling across the cells
- Flow out of a cell is computed based on cell velocity, size and density
- External forces (wind or obstacles) are used to accelerate particles in a cell

- Major disadvantage: grid is fixed, so you have to know before what grid to lay over the whole simulated environment



Particle-based method

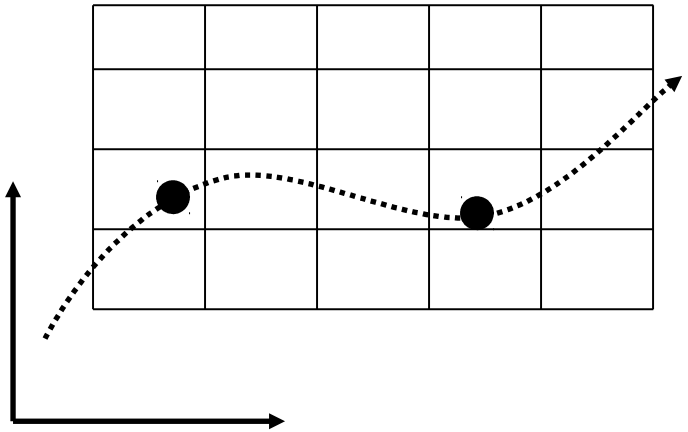
- Here, particles (or globs of gas) are tracked in space
- Often this is done like a particle system
- One can render either invividual particles, or as spheres of gas of a given density
- Technique similar to rigid body dynamics
- Disadvantage: loads of particles are needed to simulate a dense expansive gas
- Particles have masses, and external forces are easy to incorporate by updating the particle acceleration



Hybrid method

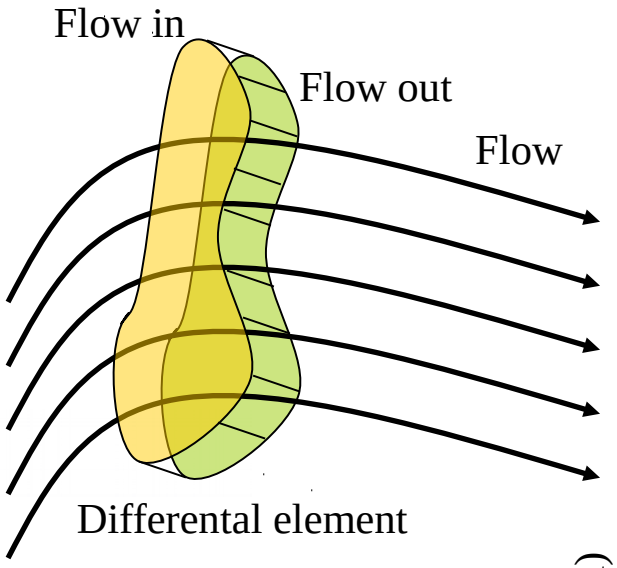
- In hybrid methods, particles are tracked in a spacial grid
- They are passed from cell to cell as they traverse the space
- Rendering parameters of the cells are determined by counting the particles in a cell at a certain time point and looking at the particle type

- Particles are used to carry and distribute attributes through the grid, and the grid is used for computing the rendering



Computational fluid dynamics

- CFD solves the physical equations directly
- Equations are derived from the Navier-Stokes equations
- Standard approach is based in a grid: set up differential equations based on conservation of momentum, mass and energy in and out of differential elements
- Quite complicated



Courtesy Japan Aerospace Exploration Agency (JAXA)

Clouds

- The biggest problem with clouds is that we are so familiar with them, i.e. we know well realistic looking ones
- Made of ice crystals or water droplets suspended on air (depending on temperature).
- Formed when air rises, and humidity condensates at lower temperatures
- Many many shapes: cirrus, stratocumulus, cumulus

Courtesy Daniel Bramer, UIUC

Clouds

- Clouds have differet detail at different scales
- Clouds form in a turbulent chaotic way and this shows in their structure
- Illumination charateristics are not easy, and vary because the ice and water droplets absorb, scatter and reflect light
- There are two illumination model types for clouds:
  - low albedo
  - High albedo

Courtesy Daniel Bramer, UIUC

Cloud illumination

- Low albedo: assumes that secondary scattering effects are neglegible
- High albedo: computes secondary order and high order scattering effects
- Optically thick clouds like cumuli need high albedo models
- Self shadowing and cloud shadowing on landscape have also to be considered

Courtesy Daniel Bramer, UIUC

Cloud illumination: surface methods

- Early models used either by using Fourier synthesis to control the transparency of large hollow ellypsoids
- Others used randomized overlapping spheres to genrate the shape
- A solid cloud texture is combined with transparency to control the transparency of the spheres
- Transparency near the edges is increased to avoid seeing the shape of the spheres
- Such surface models are not so realistic, because the surfaces are hollow

Cloud illumination: volume methods

- More accurate models have to be used in order to capture the 3D structure of a cloud [Kajiya, Stam and Fiume, Foster and Metaxas, Neyret]
- Meyret did a model based of a convective cloud model using bubbling and convection preocesses
- However, it uses large particles (surfaces) to model the cloud structure
- One can use particle systems, but a very large number of particles is needed
- Other approaches use volume-rendered implicit functions, sometimes combining them with particle systems approaches
- Implicit functions rendering can be used on the large scale, to define the global structure of a cloud, and combined with simpler procedural techniques to produce the detail
- To add a „bit“ to complexity, clouds also need to be animated since they change in time

Fire

- Fire is even more difficult:
  - it has the same complexity of gas and clouds
  - but has very violent internal processes producing light and motion
- Recently, good advances were made
- At the „exactness“ limit of the models, CFD can be used to produce fire and simulate its internal development, but it is difficult to control
- Studies on simulating the development and spreading of fire began to appear, but are usually not concerned with the internal processes within fire.

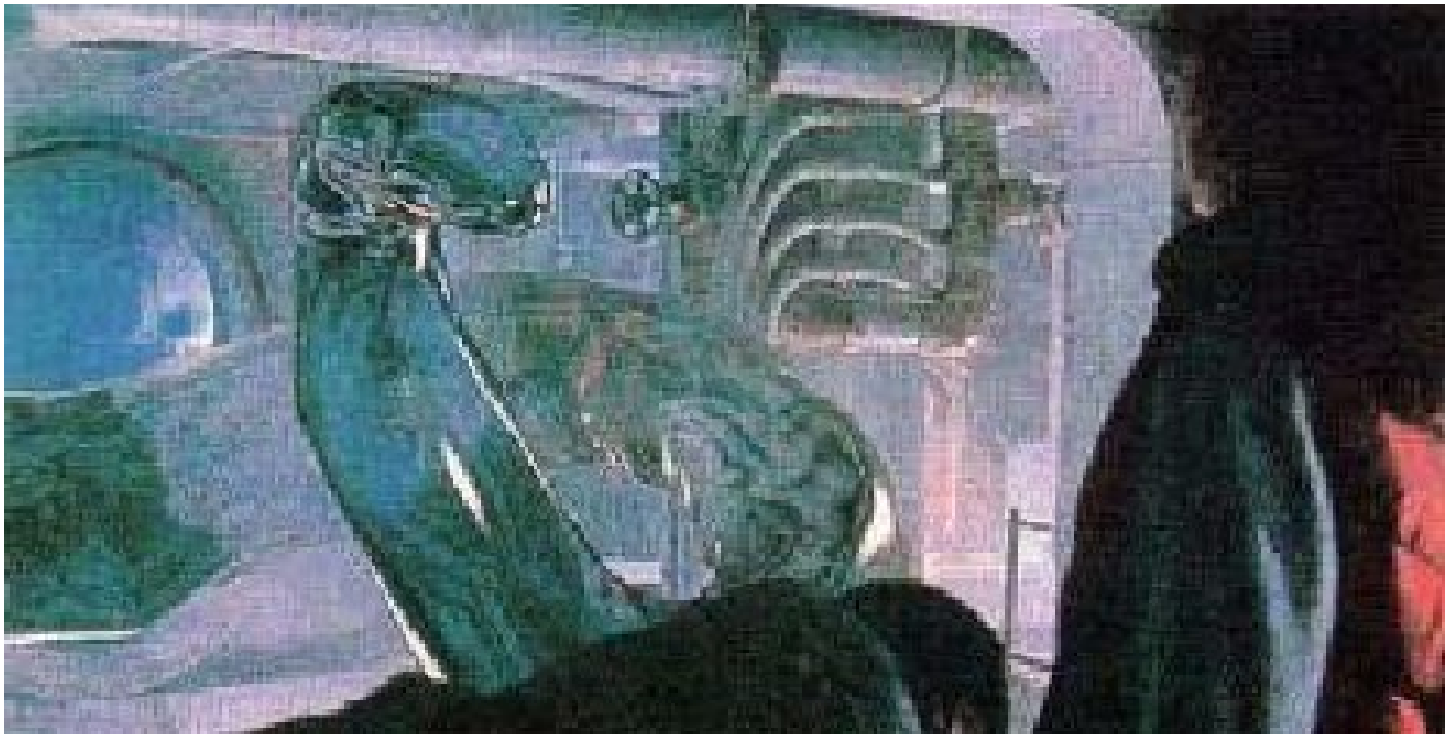
Fire: particle systems

- Computer generated fire has been used in movies since a long time, exactly since Star Wars II
- In this film, an expanding wall of fire spread out from a single impact point
- The model uses a two-level hierarchy of particles
  - First level at impact point to simulate initial ignition
  - Second level: concentric rings of particles, timed to progress concentrically to form a wall of fire and of explosions
- Each of these rings is made of a number of particle systems positioned on the ring and overlapping with neighbors so as to form a continuous ring.
- The individual particle systems are modelled to look like explosions
- Particles are oriented to fly up and away from the planet surface
- The initial position of a particle is randomly chosen from the circular base of the particle systems
- Initial ejection direction is forced into a certain cone

Fire: other approaches

- Two dimensional animated texture maps have been used to simulate a gas flame
- This works however only in one direction
- Others (Stam and Fiume) presented advection-diffusion equations to evolve both density and temperature fields
- The users control the simulation by specifying the wind field

End



Copyright (c) 1988 ILM

+++ Ende - The end - Finis - Fin - Fine +++ Ende - The end - Finis - Fin - Fine +++