Computer Animation
Natural Phenomena SS 13

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

Groups of objects

- Multiple objects can form groups
- Among them, we will introduce
  - Particles
  - Flocks
  - Autonomous agents

Controlling groups of objects

- A *particle system* is a large collection of individual elements which taken together represent a conglomerate object
- The „global“ behaviour of the particles is called *emergent behaviour*
- This can be used both for particle systems (which usually have more individuals) and for *flocking*

- Flock members have a more sophisticated behaviour than a simple element of particle system
- While particle systems behave according to physics, flocking particles add some intelligence to the behaviour of the individuals
- The more intelligence is added, the more the element moves in a more interesting way, and the more it shows *autonomous behaviour*

Particle systems

- In a particle system, due to the no of its elements, simplified assumptions are made
- Typical assumptions are
  - Particles do not collide among themselves
  - Particles do not cast indiv. shadows, but the aggregate may do
  - Particles only cast shadows on the rest of the environment, not among themselves
  - Particles do not reflect light, each is modeled as a point light source
- Often particles are modeled as having a finite life span
- To avoid dull behaviour, often randomness is added
- When a particle system is computed, the following steps are taken:
  - Generate new particles born this frame
  - Initialize attributes of new particle
  - Remove dying particles
  - Animate active particles
  - Render them

Particle generation

- Particles are usually generated according to a stochastic process
  - At each frame, a random number r_P of particles is generated
  - Generation has a user specified distribution centered at the desired number of particles per frame
  - $r_P = ave + Rand(seed) \cdot range$ where ave is the desired average and range is the desired variation range
- Sometimes it may be convenient to have this random function as a function of time, i.e. to make the number of desired particles increase in time
- If the particles are used to model a fuzzy object, then the area of the screen covered by the object A_s is used to control the number of particles

$$r_P = ave + Rand(seed) \cdot range \cdot A_s$$

Particle attributes

- Attributes of the particles are typically
  - Position
  - Velocity
  - Shape parameters
  - Color
  - Transparency
  - Lifetime
- At each frame, the lifetime of each particle is decremented by one until it reaches zero
- During lifetime, particles are animated (position, velocity, shape, color, transparency)
- At each frame, forces on the particles are computed
- These result in an acceleration, which determines a velocity
- Also other attributes may be a function of time
- Rendering is often done modeling them as a point light source adding color to the pixel
- This to avoid particles to contribute to lighting computations

Flocks

- Here the number of members is smaller
- But each member has some intelligence and simple physics (avoid collision, gravity, drag)
- Aggregate behavior emerges from the members (emergent behavior)
- Each member is called a boid

- Two forces govern flock behavior:
  - collision avoidance: both with other boids and with obstacles
  - Motion has some random parameter to keep it from looking regular
  - flock centering: the boid tries to be a flock member
  - Flock centering keeps together the flock but does not have to be absolute, otherwise flocks cannot split around objects

Flocks: local behavior

- Controlling locally the behavior is what one aims at
- Three processes may be modeled:
  - Physics: similar to particle with gravity, collision detection and response
  - Perception of the environment: each boid views its direct neighbors and obstacles directly in front
  - Reasoning and reaction to determine the behavior
  - Additionally velocity matching is added (each boid tryies to match the speed of its neighbours)
- Global control is either applied to all boids or to a group leader
  - In this case the boids follow the leader
- The leader role can be rotated among boids in time
- Usually all this is implemented as three controllers which are priorized in the following order: collision avoidance, velocity matching and flock centering

Flock complexity

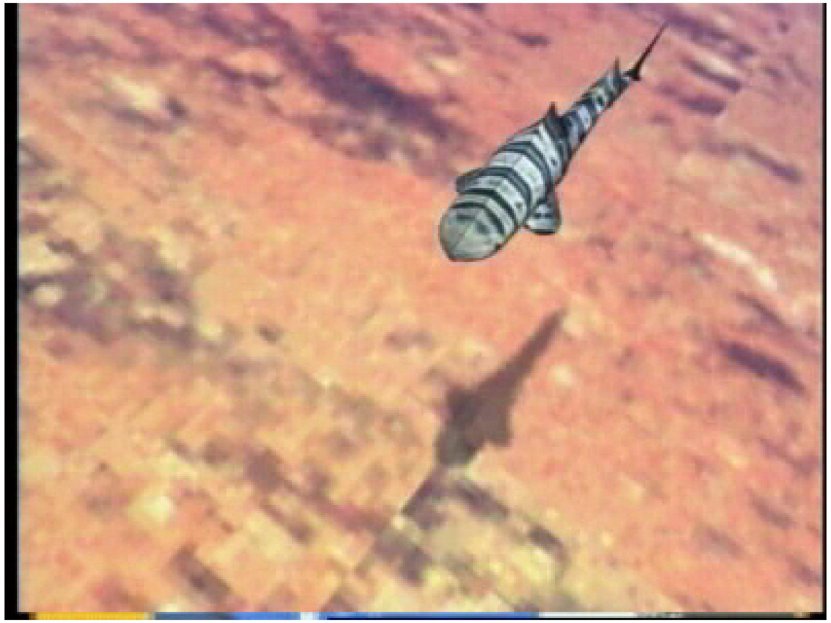
- The major problem with flocks is the fact that processing complexity is N^2.
- Even if interactions are allowed only with k nearest neighbors, those have to be found
- One way to find efficiently is to perform a 3d bucket sort and then check adjacent buckets for neighbors
- Of course, efficiency depends on the bucket size:
  - The more buckets, the less boids per bucket
- Another way of doing it is through message passing, where each boid informs the flock of its whereabouts

Collision avoidance

- There are several ways to avoid collisions
  - The simplest way is adding a repelling force around an object
  - However, this looks weird as the boid keeps attempting to aim at the repelling surface and contantly gets blown away
  - Another method computes if the boid trajectory hits the surface and starts a steering behavior
  - Quite complicated is the simulation of a splitting flock around an obstacle, since a balance has to be found between collision avoidance and flock cohesion

Autonomous behaviour

- Recently, authors concentrated in learning and simulating how “intelligent” behaviour can be implemented.
- Needs-goals priorization
- Some concentrated on “learning simulations” and life games



Autonomous behaviour

- Modeling intelligent behaviour is a complex task
- Autonomous behaviour models an object knowing about its environment
- This can become as complicated as one wants
- Usually applied to animals, but also to people, cars on a road, planes, or soldiers in a battle
- Knowledge of the environment is provided by providing access to the environment geometry
- Subjective vision can be achieved by rendering the environment from the point of view of the object
- Internal state is modeled by **intentions** = the urge to satisfy a need
- High level goals can be decomposed in single low level tasks (levels of behaviour)
- Internal state and knowledge of the environment are input to the reasoning unit, which produces a strategy (=what needs to be done)
- Such strategy is turned into a sequence of actions by the planner, and actions are turned into movement
- If intentions are competing, they must be prioritized
- Look at this link: http://www.youtube.com/watch?v=pqBSNAOsMDc&feature=related

Natural Phenomena

- 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

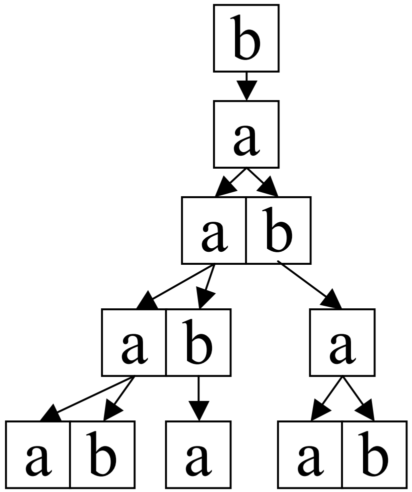
L-systems

- 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 $\alpha_i \rightarrow \beta_i$, 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 $\alpha_i \rightarrow \alpha_i$ is applied
- Applying all production rules generates a new string
- This is done recursively until no production rules can be applied

Example

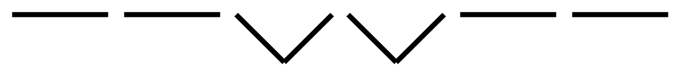
- Let the alphabet consist of the letters *a,b*
- Suppose we have two production rules:
  - *a* → *ab*
  - *b* → *a*
- And suppose that the axiom is *b*
- Then we obtain that we can generate the following strings

- *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 signs with the end of the straight line
  - Example: XXYYXX

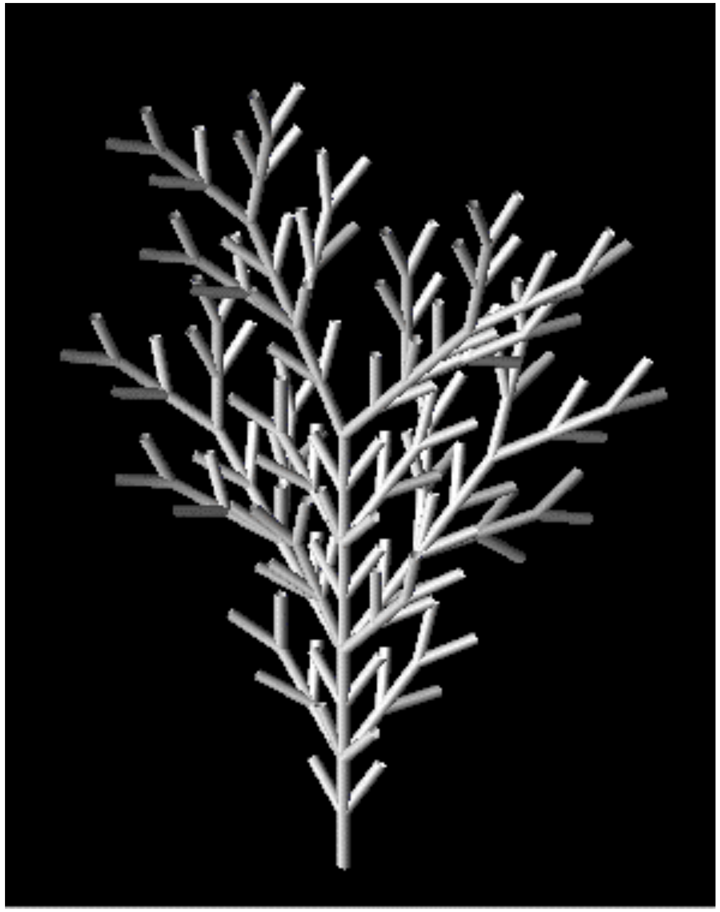


Interpreting L-systems

- 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

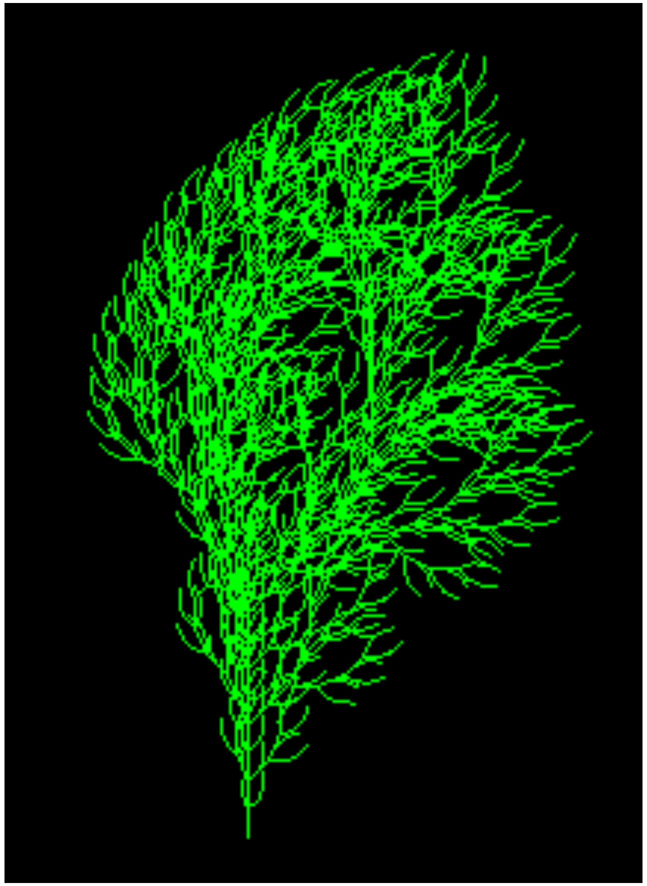
- Even more useful: if the symbols are interpreted as cells, or parts of a plant, the generation process of an L-system can simulate the growing of a plant
- The interpretation would be: substitute last year‘s leaf buds with a small piece of branch
- Or,, a branch will be replaced by three branches centered 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

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"



Courtesy *Hung-Wen Chen*, Cornell University

Stochastic and Context sensitive L-systems

- Stochastic L-systems assign a user-specified probability to each production so that the left hand side symbol probabilities add to 1
- These productions will control how likely a production will form a branch at a branching point

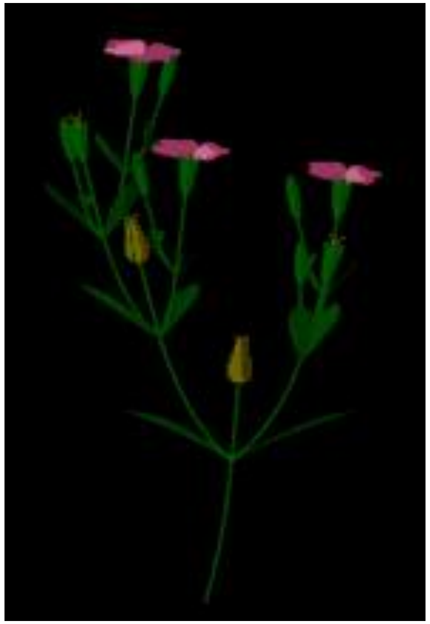
- In context sensitive L-systems, the productions are sensitive to a sequence of symbols rather than a single symbol
- If n left symbols are considered in the production, and m right symbols are produced, we have a (n,m)L-system

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
- 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 $(\mu_0,\beta_0)\rightarrow((\mu_1,\alpha_1),\ldots,(\mu_n,\alpha_n))$ 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 exists until it reaches β_0
  - After its lifespan ends, the symbol will become something else and „mutate“
- The environment can influence plant growth in many ways, which can influence the production rules

L-systems

- Adding all these factors allow the generation of very complex objects
- They look pretty realistic too



Courtesy Przemyslaw Prusinkiewicz, Mark Hammel, Radomir Mech Univ. Of Calgary

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

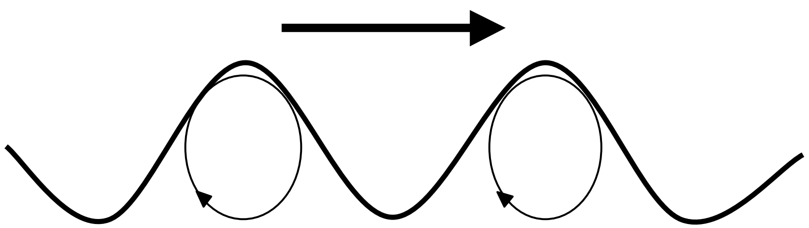
- A better way of doing water is to incorporate physical laws
- There is a variety of types of waves:
  - Tidal waves
  - Waves created by the wind
- In general, at a distance *s* of the sourcepoint we have that

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

- 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 steepness) 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

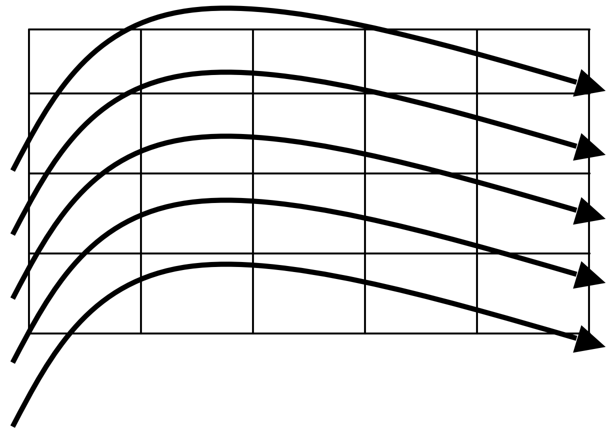
Gaseous phenomena

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

Grid-based method

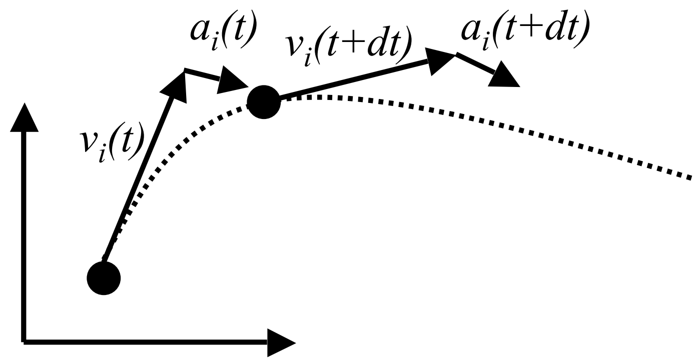
- Decomposes space into grid cells
- Density of gas in a cell is updated from time to time step
- The density of gas in a cell is used to determine the visibility and illumination for rendering
- Attributes of gas in a cell can be used to track the gas travelling across the cells
- Flow out of a cell 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

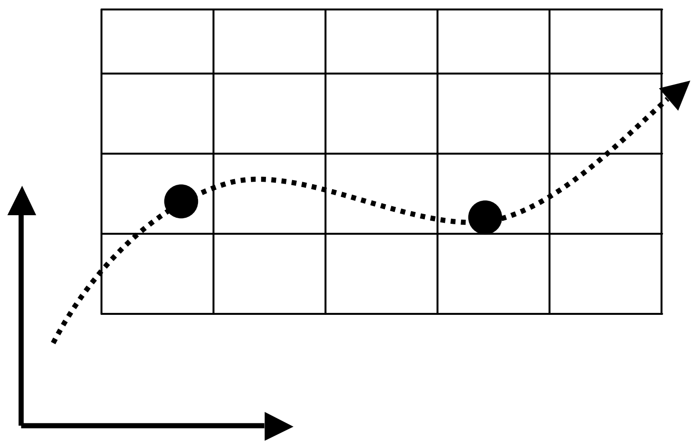
- Here, particles (or globs of gas) are tracked in space
- Often this is done like a particle system
- One can render either invividual particles, or as spheres of gas of a given density
- Technique similar to rigid body dynamics
- Disadvantage: loads of particles are needed to simulate a dense expansive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



Hybrid method

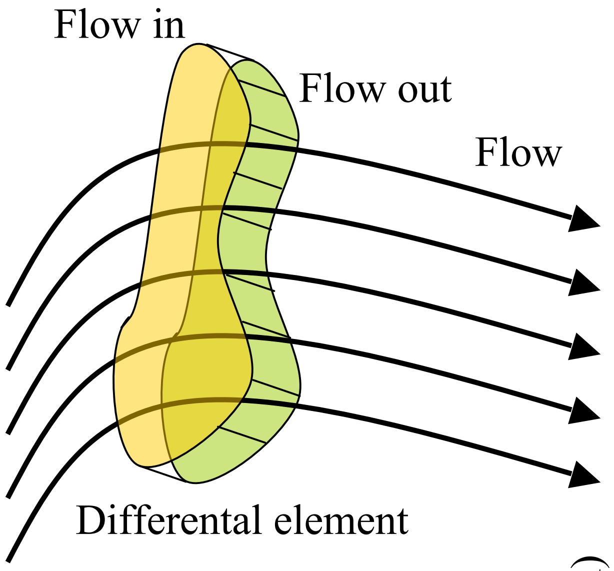
- In hybrid methods, particles are tracked in a spacial grid
- They are passed from cell to cell as they traverse the space
- Rendering parameters of the cells are determined by counting the particles in a cell at a certain time point and looking at the particle type

- 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 clous 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.

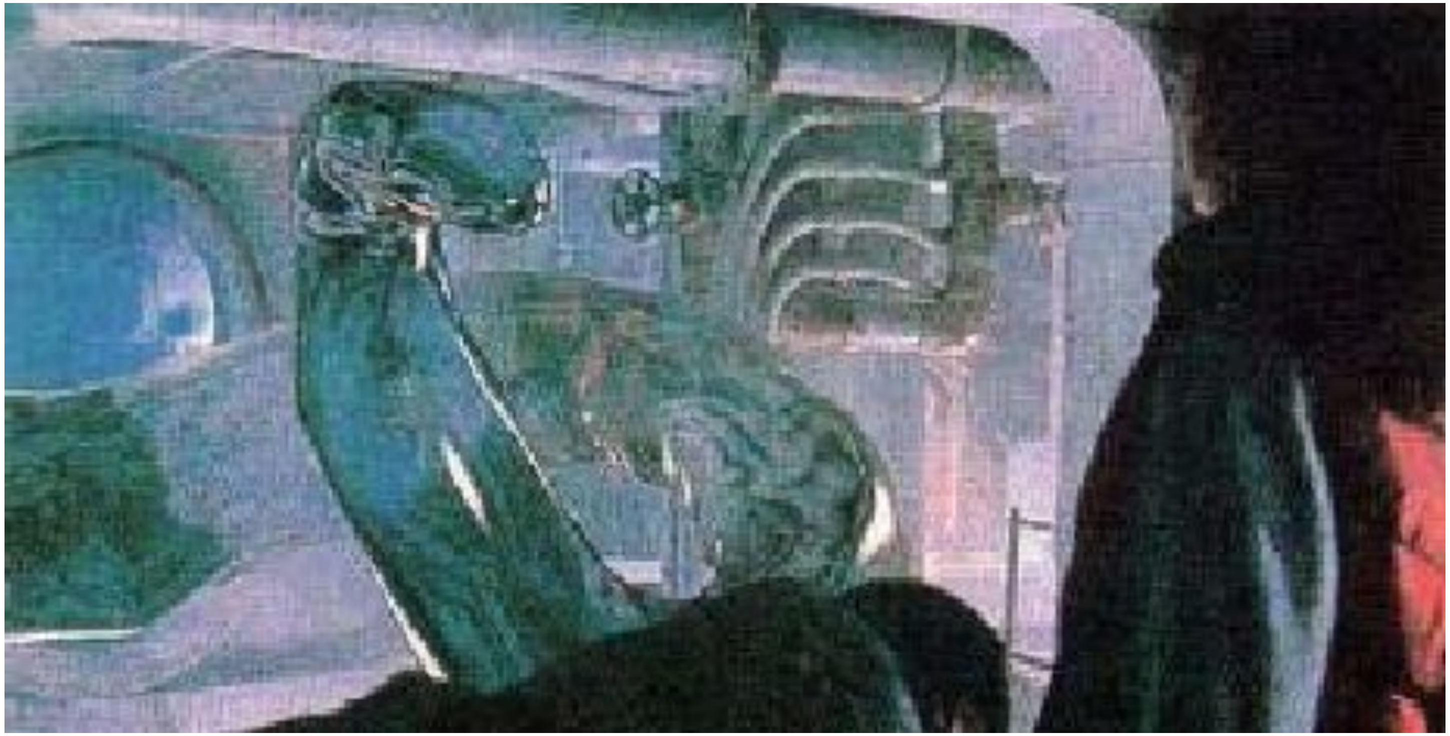
Fire: particle systems

- 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



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