EECS 487: Interactive Computer Graphics

Lecture 41:
Introduction to Procedural Modeling and Animation

- Fractals
- Dynamics
- Particle systems
- Behavioral animation

Procedural Modeling

Constructs 3D models using algorithms

- for models that are too complex (or tedious) to create manually, e.g.,
  - landscapes, mountains, clouds, planets
  - trees, plants, ecosystems
  - buildings, cities
- usually defined by a small set of data, or rules, that describes the overall properties of the model, e.g., trees defined by branching properties and leaf shapes
- model is then constructed by an algorithm
- to add variety and realism,
  - often involves fractal geometry
  - often includes randomness
  - e.g., a single tree pattern can be used to model an entire forest

Schulze

Self-Similarity

Self-similar:

- viewing scale not apparent from object appearance
- object features are statistically similar between object parts and the overall object
  - for example, we always see a jagged line no matter how close we look at a coastline
- infinite details: looks good and natural at every resolution
- achieved using random numbers and fractional dimension

Usually generated by recursively applying the same operation (or set of operations) to an object

Yu,Merrell

Randomness

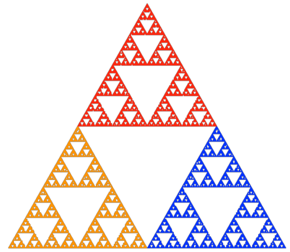
Makes models more interesting, natural, and less uniform and "clean"

Due to the discrete and finite nature of computers, we can only generate pseudo-random numbers, based on some initial seed value

- pseudo-random sequences are repeatable, simply by resetting the seed value
  - a different seed value generates a different sequence of pseudo-random numbers
- for repeatability, be careful of dynamic events effecting the use of the pseudo-random sequence

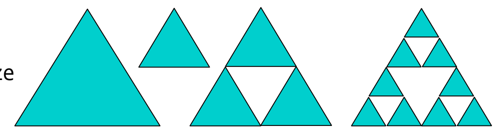
Schulze

Fractal Dimension



Measures the "roughness" of object
- more jagged objects have larger fractal dimension

Use Hausdorff variant to approximate fractal dimension:
- subdivide object into self-similar pieces with scaling factor s
- count number of pieces (n) covered by original object
- fractal dimension $d = \log(n)/\log(s)$
- e.g., Sierpinski Triangle
  - take a triangle
  - scale down to ½
  - make 3 copies and arrange into a triangle of original size
  - repeat forever
  - $d = \log(3)/\log(2) = 1.58$



Yu,Merrell

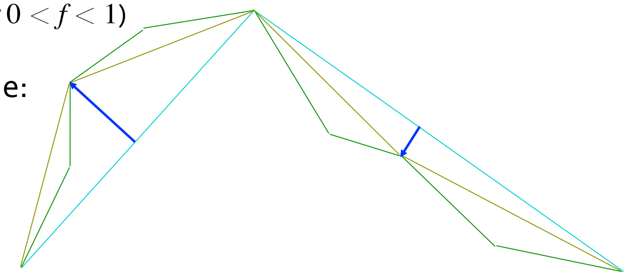
Procedural Terrain Modeling

Landscapes are often constructed as height fields

Want: a height function $y = h(x)$

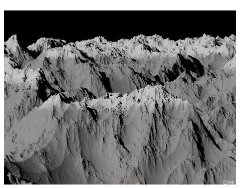
Random midpoint displacement
- start with some initial figure (e.g., line or triangle)
- split at midpoints and add random displacement
- recurse, decreasing the magnitude of displacements (by a factor $0 < f < 1$)

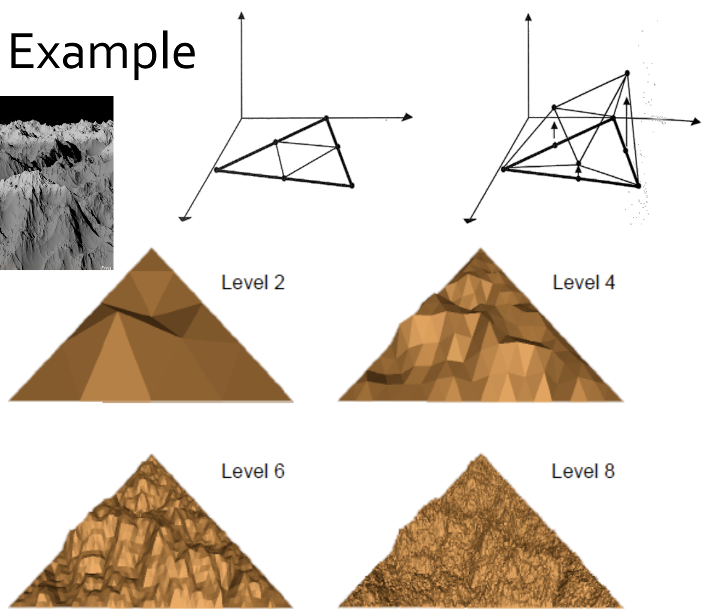
1D Example:



Yu,Merrell

2D Example





Yu

L-Systems

Developed by A. Lindenmayer, a biologist, in 1968 to model growth patterns of algae and plants

Grammar-based fractal-like models, a.k.a., "graftals"

Based on parallel string-rewriting rules
- describe an object by a string of symbols
- and a set of production/rewriting rules
- incorporate notions such as branching, pruning, …

Schulze,Yu

L-Systems Extensions

Bracketed: save/restore state (for branches)

Parametric: production governs by parameter, e.g., change color at certain level of recursion

Stochastic: randomly choose one of several production rules provided for a symbol

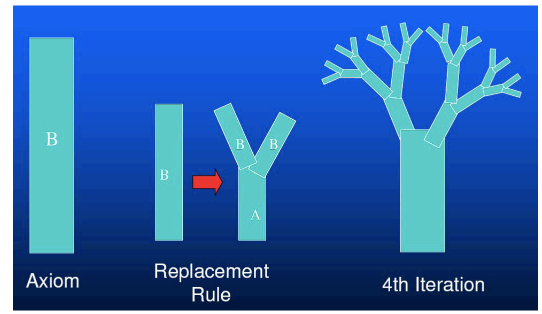
Context sensitive: production based on neighboring symbols

Hodgins

L-Systems

Define a grammar: $G = \{V, S, \omega, P\}$
- V: alphabet, set of symbols that can be replaced (variables)
- S: set of symbols that remain fixed (constants)
- ω: axiom, string of symbols defining initial state
- P: production/rewriting rules

Example:
- V: `A`, `B`
- S:
  - `+`: turn left
  - `−`: turn right
  - `[`: push state onto stack
  - `]`: pop state from stack
- ω: `B`
- P: `B→A[+B][−B]`



Schulze,Kuffner

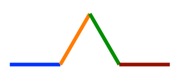
Parametric L-System

Turtle graphics:
- `F`: draw forward
- `f`: move forward
- `+`: turn left by **angle** // parameter
- `−`: turn right by **angle** // parameter
- `[`: push current state onto stack
- `]`: pop current state from the stack

Start: `F`

Generation 1:

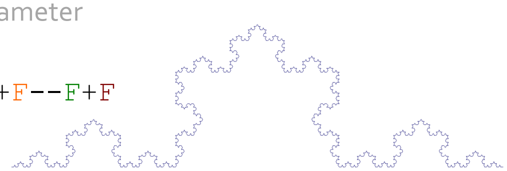
`F+F−−F+F`



Generation 2:
- `F+F−−F+F+F+F−−F+F−−F+F−−F+F−−F+F+F+F−−F+F`

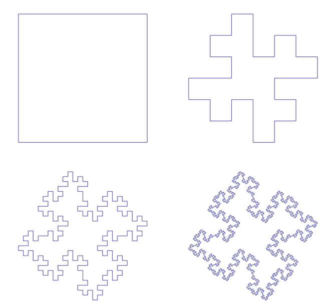
Koch's Snowflake
- **angle** `(2`π`)/6` // parameter
- axiom `F`
- production rule: `F→F+F−−F+F`

Generation n:

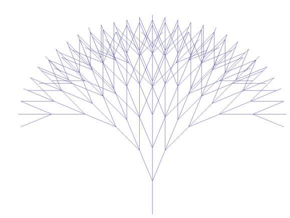


Turtle Graphics

Koch's Island:
- **angle** `(2`π`)/4`
- axiom `F+F+F+F`
- production rule: `F→F+F−F−FF+F+F−F`

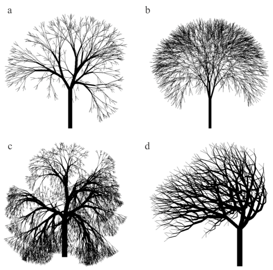


Tree:
- **angle** `(2`π`)/16`
- axiom `++++FS`
- production rule: `S→+[FS]−[FS]−[FS]`

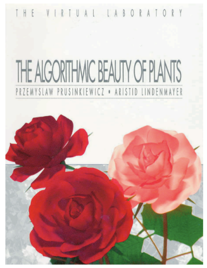


L-Systems for Plants

L-systems can capture a large number of plant species, though designing rules for specific species is not easy



See algorithmicbotany.org/papers/
- a free 200 pages ebook
- covers many variant of L-systems and different plant types



Hodgins

L-System for Cities


Parish01

real | generated

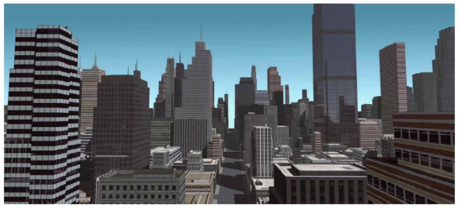
Street system:
- start with a single street
- branch and extend with parametric L-system
- parameters: goals and constraints
  - goals control street direction, spacing
  - constraints allow for parks, bridges, road loops

Hodgins

L-System for Cities


Müller et al. 06 | Wonka et al. 03

Buildings:
- building shapes are represented as CSG operations on simple shapes

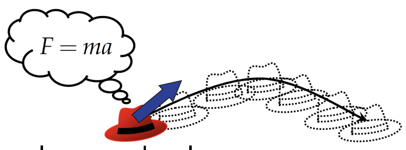

Parish&Müller01

Hodgins

Motion

Keyframing: interpolates motion from "key" positions
- \+ perfect control
- − can be tedious
- − no realism

Procedural: solves equations to compute motion
- \+ realistic motion
- \+ automatic generation
  - once you have the program, you can get lots of motion
- − difficult to control
- − hard to tell a story with purely procedural means
  - mostly used for supporting background/effects



McMillan

Procedural Animation

Using a process to control or animate some attribute, including shape (modeling), of the object

Steps:

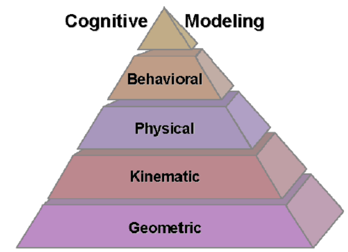
- program some rules for how the system will behave
- choose some initial conditions for the world
- run the program, maybe with user input to guide what happens
- program outputs the position/shape of the scene over time

Chenney

Procedural Animation

Physically-based animation

- dynamics: movements
- point mass particle systems
- spring-mass: animating the shape and reaction of cloth
- fluid flow: water waves



Behavioral animation

- bird or fish flocking
- crowd animation

Physically-based Animation

Kinematic (key framing): describes motion without considering causes leading to motion

- considers only poses

Dynamics (procedural): considers underlying forces

- inverse dynamics:
  - given prescribed motion, what are the forces and torques required?
- forward dynamics:
  - compute motion from initial conditions and physical laws:
    - given forces and torques, what is the motion?

Funkhouser09,Hodgins,Funge9

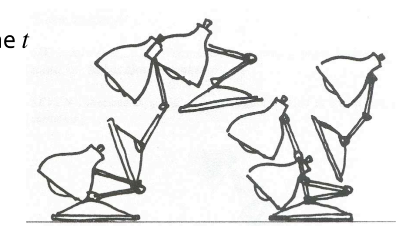
Inverse Dynamics

Given prescribed motion, what are the forces and torques required?

Active character: character has internal source of energy

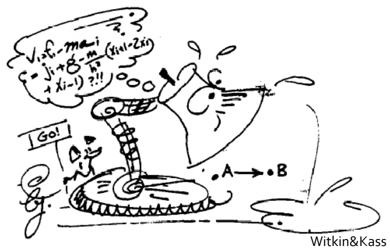
Animator specifies constraints:

- character's physical structure
  - e.g., articulated figure
- character's task
  - e.g., jump from here to there in time *t*
- other physical structures present
  - e.g., floor to push off and land
- motion requirements
  - e.g., minimize energy



Funkhouser, Lasseter

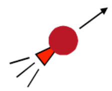
Inverse Dynamics



Witkin&Kass

Computer solves for the "best" physical motion satisfying constraints

Example: object with jet propulsion



- $x(t)$: position of object at time t
- $f(t)$: propulsion force at time t
- equation of motion: $mx'' - f - mg = 0$
- task: move from a to b from t_0 to t_1 with minimum jet fuel, minimize $\int_{t_0}^{t_1} |f(t)^2|\, dt$ with $x(t_0) = a$ and $x(t_1) = b$
- solve with iterative optimization method

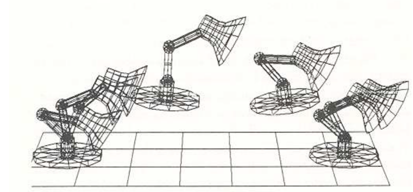
Funkhouser

Inverse Dynamics

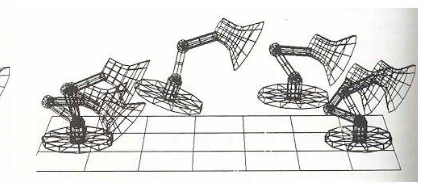
Advantages:

- animators don't have to specify details of physically realistic motion with spline curves
- easy to vary motions due to new parameters and/or new constraints

For example, adapting motion:



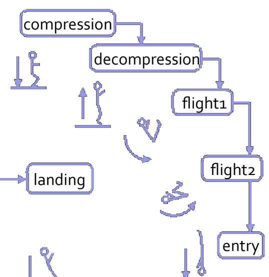
original jump



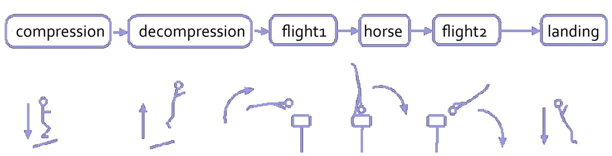
heavier base

Funkhouser, Witkin&Kass

Controlling Motion



E.g., with state machines

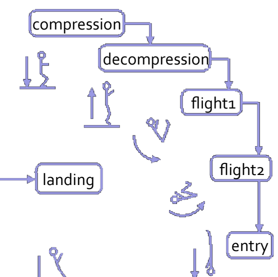


Challenges:

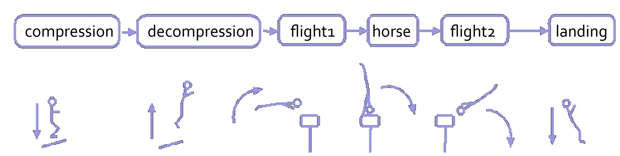
- specifying constraints and objective functions
- avoiding local minima during optimization
- retargeting motion to new characters
- need a larger variety of motions (not just well-defined sport motions)
- real-time performance

Funkhouser, Hodgins

Controlling Motion



E.g., with state machines



Challenges:

- specifying constraints and objective functions
- avoiding local minima during optimization
- retargeting motion to new characters
- need a larger variety of motions (not just well-defined sport motions)
- real-time performance

Funkhouser, Hodgins

Forward Dynamics

Given initial conditions and physical laws, what is the motion?

- initial conditions: mass, forces, torques
- apply physical laws, e.g., Newton's laws, Hook's law, etc.
- simulate physical phenomena:
  - gravity, momentum (inertia), friction, collisions, elasticity, solidity, flexibility, fracture, explosion, fluid flow, aerodynamics (drag/viscosity, turbulence)
- motion computed using numerical simulation methods
  - particle systems
  - rigid bodies
  - soft-objects (spring-mass)
  - fluid

Funkhouser, Hodgins, McAllister

How to Render Fire?

Texture mapping polygons is fast and acceptable for short-lived effects

- overlay a flame point with a series of textured polygons
- for enhanced realism, we could introduce secondary light sources at the fire

Problems:

- hard to sustain for long periods
- hard to spread or change shape
- no translucency

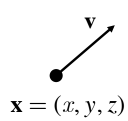
Gilles

Particle Systems

Approximate flame, and other amorphous objects, by a discrete set of small particles

Single particles are very simple, just point mass with attributes:

- mass, position
- velocity, forces
- color, temperature, shape
- lifetime



Large groups can produce interesting effects:

- rockets: fireworks
- clouds: water drops

Gilles, Funkhouser

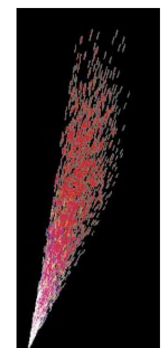
Particle Systems

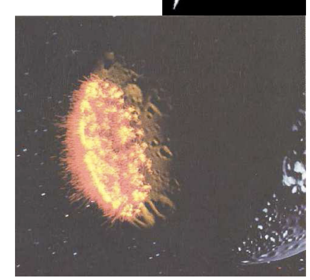
For each frame:

- create new particles according to a probability distribution and assign attributes
- delete any expired particles
- update particles based on attributes and physics
  - numerical solutions to ODE
- render particles: motion blur, compositing

Where to create particles?

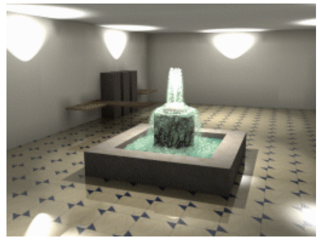
- predefined sources
- surface of shape
- where particle density is low, etc.





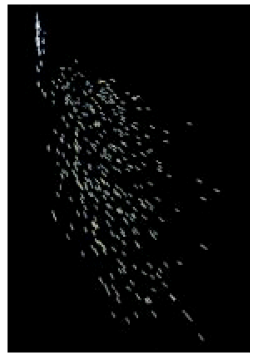
Gilles, Funkhouser, Reeves

Particle Systems



When to delete particles?
- predefined sink
- surface of shape
- where density is high
- lifetime
- random

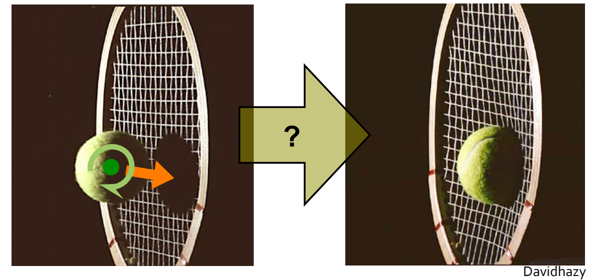
Example: water
- new particles created each frame
  - the number created per frame is normally distributed
- each particle has initial downward velocity
  - again, normally distributed
- on each successive frame, each particle is acted on by "wind" force to the right



Funkhouser, Gilles, McAllister

Impact

Dynamic:
- position ($\mathbf{x}$)
- velocity ($\mathbf{v}$)
- rotation?



Davidhazy

Static:
- radius
- mass (m)
- racquet info

Newtonian particles: $\mathbf{F} = m\,\mathbf{a}$
- $\mathbf{F}$ and $\mathbf{a}$ are vectors
- given $\mathbf{F}$ and position at time t, $\mathbf{x}(t)$, how does the position change to $\mathbf{x}(t+1)$?

Treuille

Differential Equations

Differential equations describe the relation between an unknown function and its derivatives

Solving a differential equation means finding a function that satisfies the relation, plus some additional constraints

Ordinary Differential Equation (ODE):
- "ordinary": function of one variable
- partial differential equation (PDE): more variables

Witkin&Baraff

Differential Equations

position, velocity: $\mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$

Discrete time: $\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), t)$

Continuous time: $\boxed{\dfrac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}(t), t)}$

notation $\dot{\mathbf{x}}$ for $d\mathbf{x}/dt$

Computing particle motion requires solving a 2^{nd}-order differential equation: $\ddot{\mathbf{x}} = \dfrac{\mathbf{F}(\mathbf{x}, \dot{\mathbf{x}}, t)}{m}$

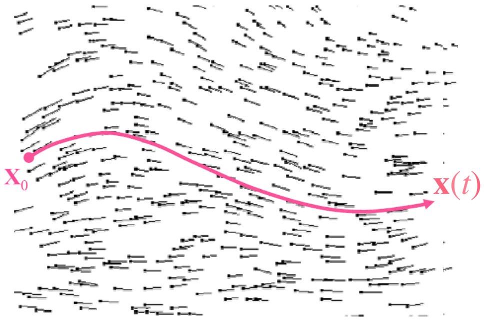
Instead, add variable $\mathbf{v}$ to form coupled 1^{st}-order differential equations: $\mathbf{v} = \dot{\mathbf{x}}$, $\dot{\mathbf{v}} = \dfrac{\mathbf{F}(\mathbf{x}, \mathbf{v}, t)}{m}$

For animation, want a series of values $\mathbf{x}(t_i)$, $i = 0, 1, 2, \ldots$
- samples of the continuous function $\mathbf{x}(t)$

Funkhouser, Durand

Path Through a Field

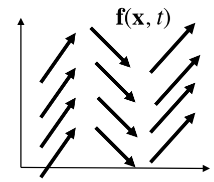
$\mathbf{f}(\mathbf{x}, t)$ is a vector field defined everywhere
- e.g., a velocity field



how to compute?

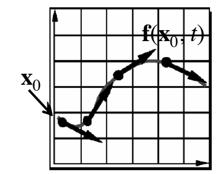
- it may change based on t
- $\mathbf{x}(t)$ is a path through the field: $\mathbf{x}(t) = \mathbf{x}_0 + \int_{t_0}^{t} \mathbf{f}(\mathbf{x}, t)\, dt$
- usually no analytical solution

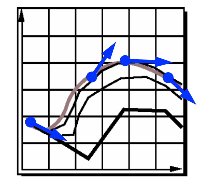
Solving the Integration



Since we're working in fixed-frame intervals, we can use a simple approximation (Euler's method):

- define step size h
- given $\mathbf{x}_0 = \mathbf{x}(t_0)$, take step:
  $t_1 = t_0 + h$
  $\mathbf{x}(t_1) = \mathbf{x}_0 + h\,\mathbf{f}(\mathbf{x}_0, t_0)$
- piecewise-linear approximation of the curve
  - step size controls accuracy: smaller, more accurate
  - may need to take many small steps per frame





Durand

Solving the Integration

Euler's method is the simplest numerical method

Consider the Taylor expansion of $\mathbf{x}(t)$:

$$\mathbf{x}(t+h) = \mathbf{x}(t) + h\frac{d\mathbf{x}}{dt} + \frac{h^2}{2}\frac{d^2\mathbf{x}}{dt^2} + \cdots$$

the equation for Euler's method simply disregards higher-order terms and replaces the first derivative with the flow field function
⇒ the error is on the order of $O(h^2)$

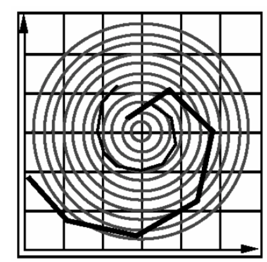
Consequences: Euler's method is inaccurate and unstable

Durand

Euler's Method: Inaccurate

To move along a circle:

$$\mathbf{x}(t) = \begin{pmatrix} r\cos(t+h) \\ r\sin(t+h) \end{pmatrix}$$



Moving along tangent, e.g.,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) = \begin{pmatrix} -y \\ x \end{pmatrix}$$

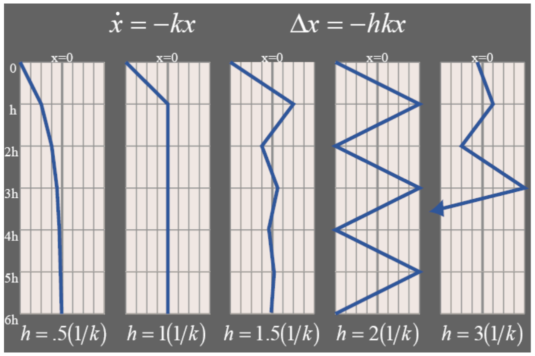
Euler's method $\mathbf{x}(t_i) = \mathbf{x}_{i-1} + h\,\mathbf{f}(\mathbf{x}_{i-1}, t_{i-1})$ spirals outward, can leave curve, no matter how small h is
- smaller h will just diverge more slowly

Durand

Euler's Method: Unstable

Consider the following system: $\dot{\mathbf{x}} = -k\mathbf{x},\ \mathbf{x}(0) = 1$
Exact solution is decaying exponential: $\mathbf{x} = \mathbf{x}_0 e^{-kt}$
Limited step size: $\mathbf{x}_i = \mathbf{x}_{i-1} - h\left(k\mathbf{x}_{i-1}\right) = (1 - hk)\mathbf{x}_{i-1}$
If k is big, h must be small

$$\begin{cases} h \leq 1/k, \text{ ok} \\ h > 1/k, \text{ oscillates } \pm \\ h > 2/k, \text{ explodes} \end{cases}$$



Durand

Solving the Integration

How can we improve upon Euler's method?

By adding another term from the Taylor's expansion of $\mathbf{x}(t)$:

$$\mathbf{x}\left(t+h\right) = \mathbf{x}\left(t\right) + h\frac{d\mathbf{x}}{dt} + \frac{h^2}{2}\frac{d^2\mathbf{x}}{dt^2} + \cdots$$

Consider two alternatives:
- the Trapezoid method
- the Midpoint method

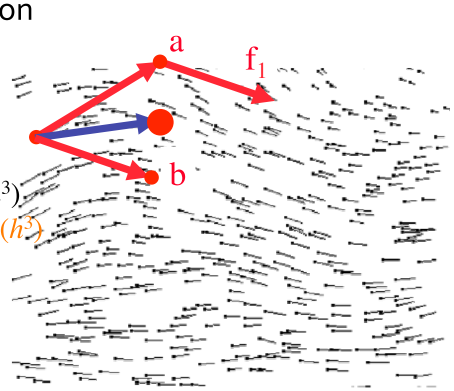
both have errors on the order of $O(h^3)$

Trapezoid Method

$$\mathbf{x}(t_i) = \mathbf{x}_{i-1} + h\,\mathbf{f}(\mathbf{x}_{i-1}, t_{i-1})$$

Problem: approximated $\mathbf{f}$ varies from actual $\mathbf{f}$
Idea: consider $\mathbf{f}$ at the arrival of the step and compensate for variation

Let $\mathbf{f}_0 = \mathbf{f}(\mathbf{x}_0, t_0)$
$\mathbf{f}_1 = \mathbf{f}(\mathbf{x}_0 + h\mathbf{f}_0, t_0 + h)$
Then $\mathbf{a} = h\mathbf{f}_0$, $\mathbf{b} = h\mathbf{f}_1$,
$\mathbf{x}(t_0 + h) = \mathbf{x}_0 + (\mathbf{a} + \mathbf{b})/2 + O(h^3)$
$\boxed{\mathbf{x}(t_0 + h) = \mathbf{x}_0 + h(\mathbf{f}_0 + \mathbf{f}_1)/2} + O(h^3)$



This is the trapeziod method,
a.k.a. improved Euler's method

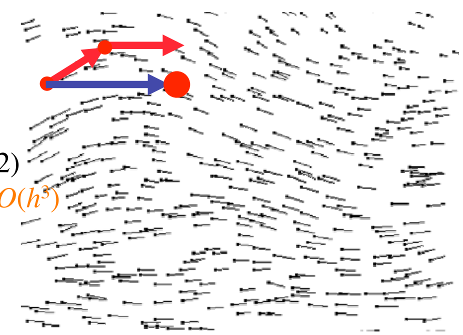
Durand

Midpoint Method

$$\mathbf{x}(t_i) = \mathbf{x}_{i-1} + h\,\mathbf{f}(\mathbf{x}_{i-1}, t_{i-1})$$

Problem: approximated $\mathbf{f}$ varies from actual $\mathbf{f}$
Idea: consider $\mathbf{f}$ at half step and compensate for variation

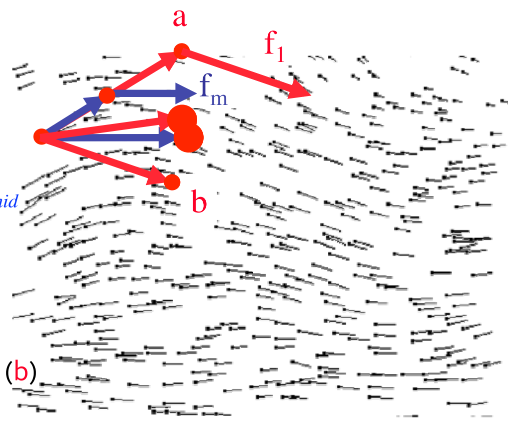
Choose
$\Delta\mathbf{x} = h/2\ \mathbf{f}(\mathbf{x}_0, t_0)$
then rearrange as before,
let $\mathbf{f}_0 = \mathbf{f}(\mathbf{x}_0, t_0)$
$\mathbf{f}_{mid} = \mathbf{f}(\mathbf{x}_0 + h/2\ \mathbf{f}_0, t_0 + h/2)$
then $\boxed{\mathbf{x}(t_0 + h) = \mathbf{x}_0 + h\mathbf{f}_{mid}} + O(h^3)$



This is the midpoint method,
a.k.a., 2nd-order Runge-Kutta

Durand

Comparison



Midpoint:
- ½ Euler step
- evaluate $\mathbf{f}_{mid}$
- compute step using $\mathbf{f}_{mid}$

Trapezoid:
- Euler step (a)
- evaluate $\mathbf{f}_1$
- compute step using $\mathbf{f}_1$ (b)
- average (a) and (b)

Not exactly the same result, but same order of accuracy

Durand

Higher-Order Runge-Kutta

4th order: $k_1 = h\mathbf{f}(\mathbf{x}_0, t_0)$

$$k_2 = h\mathbf{f}(\mathbf{x}_0 + \frac{k_1}{2}, t_0 + \frac{h}{2})$$

$$k_3 = h\mathbf{f}(\mathbf{x}_0 + \frac{k_2}{2}, t_0 + \frac{h}{2})$$

$$k_4 = h\mathbf{f}(\mathbf{x}_0 + k_3, t_0 + h)$$

$$\mathbf{x}(t_0 + h) = \mathbf{x}_0 + \frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4 + O(h^5)$$

popular because orders higher than 4th need more stages than orders to compute

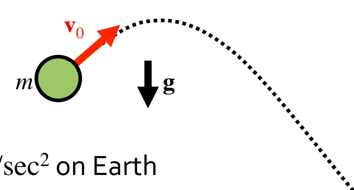
order	1	2	3	4	5	6	7	8
stages	1	2	3	4	6	7	9	11

Treuille

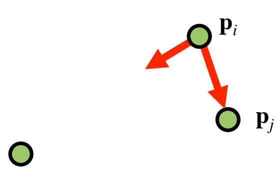
May the Force . . .

Forces acting on the particle system is a sum of a number of things

Force fields:
- gravity: constant downward force proportional to mass
  $\mathbf{f} = -m\mathbf{g}$,
  $\mathbf{g}$: gravitational constant, $9.78\ \mathrm{m/sec}^2$ on Earth



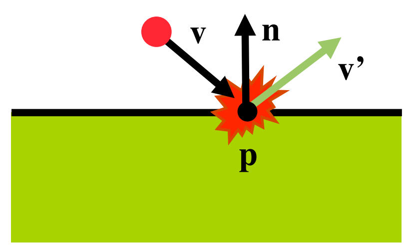
- other particles: particles mutually attract/repell
  - attractive force: $\mathbf{f} = Gm_1m_2\frac{\mathbf{d}}{\|\mathbf{d}\|^3}$
  - repulsive force: $\mathbf{f} = -k_r\frac{\mathbf{d}}{\|\mathbf{d}\|^3}$
  - beware $O(n^2)$ complexity!

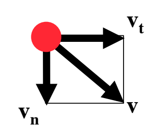


O'Brien,Durand

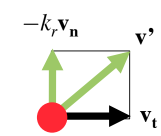
Collisions

Collisions with environment, other particles
- detection: potential large number needed, use hierarchical bounding volumes
- response: particle shape and size needed
- due to temporal aliasing, sub-frame calculation required so as not to miss collision time





$\mathbf{v_n} = (\mathbf{v}\cdot\mathbf{n})\mathbf{n}$
$\mathbf{v_t} = \mathbf{v} - \mathbf{v_n}$



$\mathbf{v'} = \mathbf{v_t} - k_r\mathbf{v_n}$
k_r: coeficient of restitution

Durand

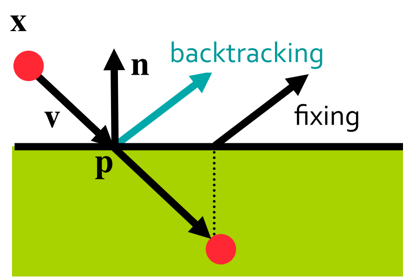
Missed Collisions

Want to detect collision when $(\mathbf{x}-\mathbf{p})\cdot\mathbf{n} < \varepsilon, \varepsilon \geq 0$

Often collision detected when $\mathbf{v}\cdot\mathbf{n} < 0$

Solution: backtrack
- compute intersection point
  - ray-object intersection!
- compute response there
- reconstruct for remaining fractional time step

Other solution/hack:
- project to surface point closest to object



Durand

Particle Systems Rendering

Simple rendering:
- project particles to view frame
- blend particle projection with framebuffer content (translucency)

Particle color:
- make color a function of age
- make color a function of temperature
- requires other ODEs to govern these properties

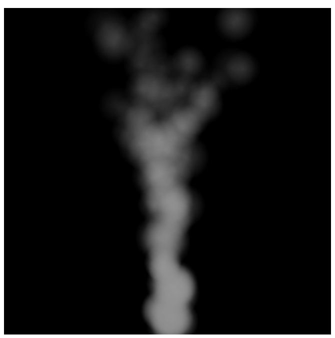
Gilles

Smoke Particle System

Constantly create particles

Particles move upwards, with Perlin turbulence added

Draw them as partially transparent circles that fade over time



Chenney

Quake

Pre-render bitmap of fireball

Real-time rendered glow

Animated glowing particles

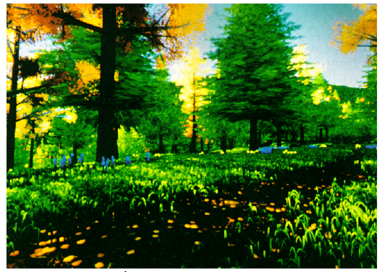
Gilles

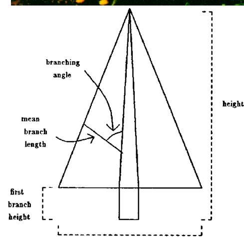
Trees in *Andre and Wally B.*

Lots of trees

Each tree branches recursively

Leaves as particles
- millions of particles
- need a simple model for shading each



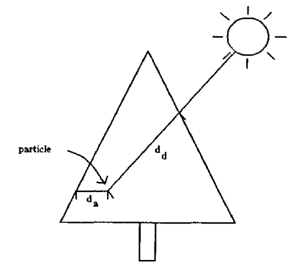


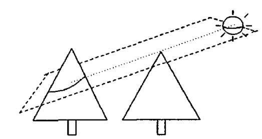
Shading Model for Trees

Ambient
- dependent on how deep in the tree d_a
- independent of light position

Diffuse
- dependent on distance d_d inside the tree, in the direction of light

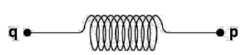
Shadowed
- if below plane, only ambient used





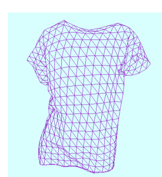
Spring-Mass Systems

Model objects as systems of springs and masses



The springs exert forces, controlled by changing their rest length

A reasonable, but simple, physical model for muscles

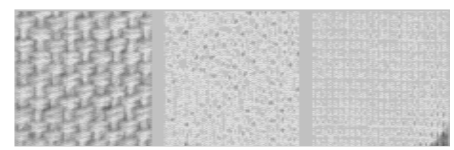


Advantage: good looking motion when it works

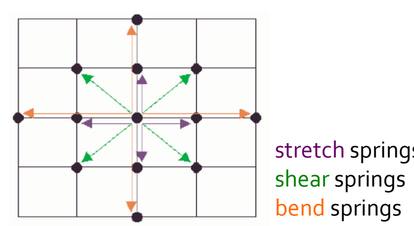
Disadvantage: expensive and hard to control

Chenney

Cloth

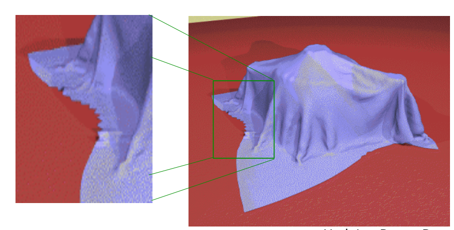


Many types of cloth
- very different properties
- not a simple elastic surface
- woven fabrics tend to be very stiff
- anisotropic



Resolution of mesh is critical

Computation of collisions is expensive



Hodgins, Breen, Popovi

Challenges for Physically-Based Animation

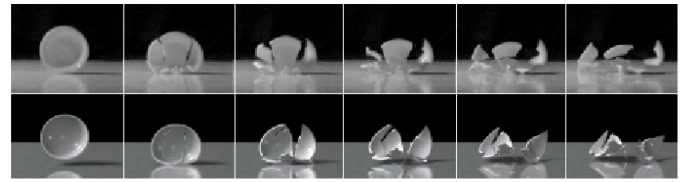
Expensive and not necessarily realistic

What pieces of physics are necessary for appearance?

How to give animator control?
- how to give artists and directors the results they want?

Perceptual Hacks

Viewers can be pretty oblivious to things like incorrect bounces and can't exactly predict how things break



Just make sure objects don't go through walls

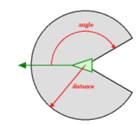
Shift emphasis from physical accuracy to fast-and-looks-good

Behavioral Animation

Define rules for the way an object behaves and interacts
- programs implement the rules
- objects respond to their changing environment

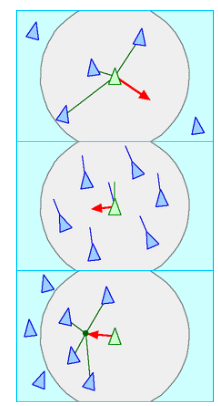
Classic example: "boids" (Craig Reynolds)
- emergent behavior: flocking
- really just a particle system with a bit of perception and a bit of brain power

Flocking



Each boid perceives neighbors in a neighborhood

Each boid's behavior is a simple function of nearby boids:
- separation force
  - steer to avoid crowding local team mates, keep minimum distance
- alignment force
  - align velocities
- cohesion force
  - move towards average position of neighbors



http://www.red3d.com/cwr/boids/
http://www.riversoftavg.com/downloads.htm

Animation Summary (brief)

Technique	Control	Time to Create	Computation Cost	Interactivity
Key-Framed	Excellent	Poor	Low	Low
Motion Capture	Good at time of creation, after that poor	Medium	Medium	Medium
Procedural	Poor	Poor to create program	High	High

Chenney

How Are Movies Animated?

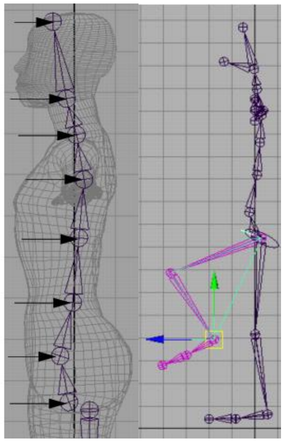
Keyframing mostly

Articulated figures, inverse kinematics

Skinning
- complex deformable skin, muscle, skin motion

Hierarchical controls
- smile control, eye blinking, etc.
- keyframes for these higher-level controls

A huge time is spent building the 3D models, its skeleton and its controls



Durand

Mixing Techniques

Apply physical simulation of secondary motion on top of key-framed primary motion
- particularly appropriate for cloth, hair, water
- use particle systems for "fuzzy" objects

Mix motion-capture and physics:
- motion-captured person kicks a ball which is then physically simulated for trajectory

Chenney