Overview

- Virtual reality
- Interactive computer animation
- Surgical simulation; preoperative planning (preoperative planning is more patient-specific) (Anatoscope offers software solutions to reconstruct and simulate a digital twin from various patient's data)
- Computational robotics; manipulation
- Video games
- Assembly planning
- Scientific visualization
- Education
- E-commerce (try cloth before purchase, Ming Lin)

Topics:

- 1. Overview of computer animation and simulation
- 2. Primer on numerical simulation and linear algebra for graphics
- 3. Cloth
- 4. FEM
- 5. Rigid body dynamics
- 6. Keyframe animation
- 7. Motion capture
- 8. Quaternions
- 9. Inverse kinematics
- 10. Character rigging
- 11. Facial animation
- 12. Maya
- 13. Constraints and contact
- 14. Collision detection
- 15. Haptics
- 16. Fluids
- 17. Sound simulation
- 18. Neural fields
- 19. CUDA
- 20. Computer animation middleware software - Havok

2. Bilinear and Trilinear Interpolation

Bilinear interpolation (2D)

- **What it is**: linear-blend first along the x-axis, then along the y-axis inside one rectangular cell.
- $F(u,v) = (1-v)\big[(1-u)F_{00} + uF_{10}\big] + v\big[(1-u)F_{01} + uF_{11}\big]$, where $u, v \in [0,1]$ are the fractional distances from the cell's left/bottom edges and F_{ij} are the four corner samples.

Trilinear interpolation (3D)

- Trilinear interpolation is a direct extension of the bilinear interpolation technique. It can be seen as the linear interpolation of two bilinear interpolations: one for the front face of the cell and one for the back face.
- $F(u,v,w) = (1-w)\,F(u,v)_{\text{front}} + w\,F(u,v)_{\text{back}}$, where each $F(u,v)$ on the right-hand side is a bilinear blend of the front/back face.

Application

- Volume rendering: Interpolate to compute RGBA away from grid, nearest neighbor yields blocky images, use trilinear interpolation
- GPU 3D texture sampling: games sample density/lighting stored in 3D textures
- Medical imaging: reslicing CT/MRI voxel data.
- 3D fluid solvers: look up velocity or density between MAC grid cell centers
- Procedural noise: Perlin or Simplex noise use trilinear blends of gradient vectors

Worked example (bilinear, vector field)

- Grid: one unit square $[0,1] \times [0,1]$ with velocities $w_{00} = (0,1),\ w_{10} = (2,0),\ w_{01} = (0,2),\ w_{11} = (2,0)$
- Query point: $P = (x,y) = (0.3, 0.8)$. Fractions: $u = 0.3,\ v = 0.8$
- Blend along x:
  $a = (1-u)w_{00} + uw_{10} = 0.7(0,1) + 0.3(2,0) = (0.6,\ 0.7)$
  $b = (1-u)w_{01} + uw_{11} = 0.7(0,2) + 0.3(2,0) = (0.6,\ 1.4)$
- Blend along y:
  $w(P) = (1-v)\,a + v\,b = 0.2(0.6, 0.7) + 0.8(0.6, 1.4) = (0.6,\ 1.26)$
- The velocity used in the semi-Lagrangian back-trace is $w(P) = (0.6,\ 1.26)$

Newton Method

Solving System of Equations

$Ax = b$

- A: $n \times n$ matrix, x: $n \times 1$ vector, b: $n \times 1$ vector
- $f(x) = Ax - b$
- $f(x) = o^n, f : R^n \to R^n$, general non-linear equation in n-dimension
- Pardiso(Intel MKL), Intel ONEAP 1, Intel TBB: parallel programming API

Newton-raphson Method

- $x_0, x_1, x_2, \ldots \to x$
- $o^n = f(x_k + \Delta x_k) \doteq f(x_k) + \frac{df}{dx}|_{x=x_k} \cdot \Delta x_k + O(||\Delta x_k||^2)$
- x_k: known, Δx_k: unknow solve for it, $\frac{df}{dx}|_{x=x_k}$ Jacobian, $O(||\Delta x_k||^2)$ omit
- 1D: $f(x_k) + f'(x_k)\Delta x_k + O(\Delta x_k^2)$
- $\frac{df}{dx}|_{x=x_k} \cdot x_k = -f(x_k)$
- $x_{k+1} = x_k + \Delta x_k$

Numerical Timestepping

Explicit Euler

- $y \in R^n, \dot{y} = G(y), G : R^n \to R^n$
- $y_{k+1} = y_k + h \cdot \dot{y}_k + O(h^2)$

Sample Question

- $y' = 4y^2, y_0 = 1, h = 1$, calculate forward and backward at t = 1
- Forward/Explicit: $y_1 = y_0 + 1 \cdot f(t_0, y_0) = 1 + 1 \cdot 4 = 5$
- $f(t_0, y_0) = y_0' = 4y_0^2 = 4$
- Backwar/Implicit: $y_1 = y_0 + 1 \cdot f(t_1, y_1) = 1 + 1 \cdot 4y_1^2$
- $f(t_1, y_1) = y_1' = 4y_1^2$
- $4y_1^2 - y_0 + 1 = 0$ solve $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- We have $b^2 - 4ac = 1 - 16 = 15 \to$ no real root

3. Cloth

In general, cloth simulation is the process of making a sheet-like material deform under gravity, collisions, and internal stretch, shear, and bend forces. Other methods: mass-spring systems, position-based dynamics/extended PBD, projective dynamics, thin-shell FEM, yarn-level models.

Baraff-Witkin Cloth

Large Steps in Cloth Simulation [David Baraff and Andrew Witkin 1998]

Yarn-level cloth simulation: recent year cloth simulation technique

Marvelous Designer: film industry, not commercial successful

CLO3D: fashion industry

- $< a, b >= a^T b = a \cdot b$
- $E = \frac{k}{2} < C(x), C(x) >= \frac{k}{2}||C(x)||^2$
- Force: $f = -\frac{\partial E}{\partial x} = -k\frac{\partial C}{\partial x} \cdot C$
- Energy: $E = E_{stretch} + E_{shear} + E_{bend}$
- Hessian: $H = \frac{\partial F}{\partial x}$
- Cloth simulation in the Large Steps in Cloth Simulation (Baraff & Witkin, 1998) framework treats a piece of fabric as a triangle mesh whose physical state is described by three energy terms—stretch, shear, and bend—plus an implicit time integrator that lets us take very large time steps without blowing up.

Cloth Forces

Stretch:

- $(u, v) \in R^2 \mapsto w(u, v) \in R^3$
- $||\frac{dw}{du}|| = ||\frac{dw}{dv}|| = 1$ when not streched where $\frac{dw}{du} \in R^3$, $\frac{dw}{dv} \in R^3$
- $C_{stretch}(x) = a \cdot \begin{bmatrix} ||\frac{dw}{du}|| - 1 \\ ||\frac{dw}{dv}|| - 1 \end{bmatrix}$ can model both stetch and compress
- a: triangle area of every triangle
- $C_{stretch}(x)$: how strongly the stretch is
- Energy $E_{stretch} = \frac{k_{stretch}}{2} < C_{stretch}, C_{stretch} >$

Shear:

- more shear, $\frac{dw}{dv}$ and $\frac{dw}{du}$ are more close to each other
- $C(x) = a \cdot < \frac{dw}{du}, \frac{dw}{dv} > /(||\frac{dw}{du}|| \cdot ||\frac{dw}{dv}||)$
- $< \frac{dw}{du}, \frac{dw}{dv} >$ when no shear, dot product = 0
- $/(||\frac{dw}{du}|| \cdot ||\frac{dw}{dv}||)$ omit, because it is close to 1

Bend:

- Suppose flat, check the dihedral angle θ between two adjacent triangles
- 4 vertices, $3 \times 4 = 12$ DOF
- $cos(\theta) = N_1 \cdot N_2$
- $sin(\theta) = (N_1 \times N_2) \cdot e$ where e is a unit vector along the common edge
- $C(x) = \theta = arctan\frac{sin\theta}{cos\theta}$
- $E = \frac{k_{bend}}{2}\theta^2$
- limitation: material property
- Kawabata evaluation system: used to measure the mechanical properties of fabrics (1982). Test to check qualitative match, quantitative match is impossible

4. Mass Spring System

- Several mass points
- Connected to each other by springs
- Springs expand and stretch, exerting force on the mass points
- Very often used to simulate cloth

Newton's Laws

- Newton's 2nd law: $\vec{F} = m\vec{a}$ tells you how to compute acceleration, given the force and mass
- Newton's 3rd law: If object A exerts a force F on object B, then object B is at the same time exerting force -F on A.

Single spring

- Obeys the Hook's law: $F = k(x - x_0)$
- x_0 = rest length
- k = spring elasticity (aka stiffness)
- For $x < x_0$, spring wants to extend
- For $x > x_0$, spring wants to contract

Hook's law in 3D

- Assume A and B two mass points connected with a spring.
- Let L be the vector pointing from B to A: $L = A - B$
- Let R be the spring rest length
- Then, the elastic force exerted on A is: $\vec{F} = -k_{Hook}(|\vec{L}| - R)\frac{\vec{L}}{|\vec{L}|}$ where $\frac{\vec{L}}{|\vec{L}|}$ is unit vector

Damping

- Springs are not completely elastic
- They absorb some of the energy and tend to decrease the velocity of the mass points attached to them
- Damping force depends on the velocity: $\vec{F} = -k_d\vec{v}$
- k_d = damping coefficient
- k_d different than k_{Hook}

Damping in 3D

- Assume A and B two mass points connected with a spring
- Let L be the vector pointing from B to A: $L = A - B$
- Then, the damping force exerted on A is: $\vec{F} = -k_d\frac{(\vec{v_A} - \vec{v_B})\cdot\vec{L}}{|\vec{L}|}\frac{\vec{L}}{|\vec{L}|}$
- Here v_A and v_B are veclocities of points A and B
- Damping force always opposes the motion

A network of springs

- Every mass point connected to some other points by springs
- Springs exert forces on mass points: Hook's force, Damping force
- Other forces: External force field (Gravity, Electical or magnetic force field), Collision force

Structural springs

- Connect very node to its 6 direct neighbors
- Node (i,j,k) connected to (i+1,j,k), (i-1,j,k), (i,j+1,k), (i,j-1,k), (i,j,k+1), (i,j,k-1), for surface nodes, some of these neighbors might not exists
- Structural springs establish the basic structure of the jello cube

Shear springs

- Disallow excessive shearing
- Prevent the cube from distorting
- Every node (i,j,k) connected to its diagonal neighbors
- Shear spring resists stretching and thus prevents shearing

Bend springs

- Prevent the cube from folding over
- Every node connected to its second neighbor in every direction (6 connections per node, unless surface node)
- Bend spring resists contracting and thus prevents bending

External force field

- If there is an external force field, add that force to the sum of all the forces on a mass point $\vec{F}_{total} = \vec{F}_{Hook} + \vec{F}_{damping} + \vec{F}_{forcefield}$
- There is one such equation for every mass point and for every moment in time

Collision detection

- The movement of the jello cube is limited to a bounding box
- Collision detection easy: Check all the vertices if any of them is ouside the box
- Inclined plane: $F(x, y, z) = ax + by + cz + d = 0$ Initially, all points on the same side of the plane, $F(x, y, z) > 0$ on one side of the plane and $F(x, y, z) < 0$ on the other. Can check all the vertices for this condition

Collision response

- When collision happens, must perform some action to prevent the object penetrating even deeper
- Object should bounce away from the colliding object
- Some energy is usually lost during the collision
- Several ways to handle collision response
- We will use the penalty method

The penalty method

- When collision happens, put an artificial collision spring at the point of collision, which will push from the colliding object
- Collision springs have elasticity and damping, just like ordinary springs
- Direction is normal to the contact surface
- Magnitude is proportional to the amount of penetration
- Collision spring rest length is zero

Integrators

- Network of mass points and springs
- Hook's law, damping law and Newton's 2nd law give acceleration of every mass point at any given time
- $F = ma$: Hook's law and damping provide F, m is point mass, the value for a follows from $F = ma$
- Now, we know acceleration at any given time for any point
- Want to compute the actuall motion
- The equations of motion: $\frac{d\vec{x}}{dt} = \vec{v}$, $\frac{d^2\vec{x}}{dt^2} = \frac{d\vec{v}}{dt} = \vec{a}(t) = \frac{1}{m}(\vec{F}_{Hook} + \vec{F}_{damping} + \vec{F}_{forcefield})$
- x = point position, v = point velocity, a = point acceleration
- They describe the movement of any single mass point

- F_{hook} = sum of all Hook forces on a mass point
- $F_{damping}$ = sum of all damping forces on a mass point
- When we put these equations together for all the mass points, we obtain a system of ordinary differential equations
- In general, impossible to solve analytically
- Must solve numerically
- Methods to solve such systems numerically are called integrators
- Most widely used: Euler, Runge-Kutta 2nd order (aka the midpoint method) (RK2), Runge-Kutta 4th order (RK4)
- RK4 is often the method od choice
- RK4 very popular for engineering applications
- The time step should be inversely proportional to the square root of the elasticity k [Courant condition]

Integrator design issues

- Numerical stability: If time step too big, method "explodes"; t = 0.001 is a good starting choice for the assignment; Euler much more unstable than RK2 or RK4 (Requires smaller time-step, but is simple and hence fast); Euler rarely used in practice
- Numerical accuracy: Smaller time steps means more stability and accuracy, but also means more computation
- Computational cost: Tradeoff: accuray vs computation time

The Finite Element Method

Mass Spirng system vs FEM:

- mass spring system cannot capture volumetric effects
- mass spring system's behavior depends on the structure of the mesh

Hooke's law: $P = \frac{f_n}{A} = E\frac{\Delta l}{l}$

- P: pressure, f: force, A: area, E: Young's modulus
- the stronger $\frac{f_n}{A}$, the larger the relative elongation $\frac{\Delta l}{l}$
- the magnitude of $\frac{f_n}{A}$ increases when E increases
- $\sigma = E\varepsilon$ where $\sigma = \frac{f_n}{A}$ is the applied stress, $\varepsilon = \frac{\Delta l}{l}$ is the resulting strain

Graphics Interface (GI): a top conference in Canada
Matthias Muller:
https://matthias-research.github.io/pages/publications/GI2004.pdf
Vega FEM:
https://viterbi-web.usc.edu/ jbarbic/vega/
Inversion Recovery

Young's modulus: E

- describe the stiffness 刚度越低，柔度越高
- 受到的力（应力）与所产生形变量（应变）之间的比值
- Jello: $10Pa$
- Brain: $100Pa$
- Fat: $1000Pa$
- Muscle: $10000Pa$
- Human bone: 10^6Pa
- steel: $10^{11}N/m^2 = 10^{11}Pa$
- rubber: $10^7 - 10^8N/m^2$

Young's modulus of steel
link: https://www.sciencedirect.com/science/article/pii/S0
Young's modulus of human bone
https://onlinelibrary.wiley.com/doi/epdf/10.1002/jor.1100

material vs linear material:

- co-rotational linear material: simplest possible choice for non-linear material
- 2nd popular material: St.Venant-Kirchhoff (StVK) material

Possion's ratio (nu): $\frac{\Delta v}{v} = (1 - 2v) \cdot \frac{\Delta l}{l}$

- measure volume preservation when deform
- when $v = \frac{1}{2}$ volume preserved
- when $v = 0$ no shrink in width
- $-1 <= v <= \frac{1}{2}$
- in general, volume preserved in real life
- human body preserve volume becuase 70% of human is made of water and water preserve volume

Linear FEM vs. FEM

- Linear FEM: deform more, inprecise

Given a paragraph, fill out the terms, put in Young's modu

5. Rigid Body Dynamics

Reference vs. World Frame

- $[E_x, E_y, E_z]$ reference axes, origin O at the body's center of mass.
- $[e_x, e_y, e_z]$ world axes, origin $x_c(t)$ is the translated center of mass.
- Any material point p in the reference volume V_{ref} moves to $x(p,t) = x_c(t) + R(t)\,p$ with rotation matrix $R(t) \in SO(3)$.
- Distance preservation: $\|x(p,t) - x_c(t)\| = \|p\|$ for all p.

Kinematics

- Orthogonality: $R\,R^T = I_3$ and $\dot{R}\,R^T + [\,\dot{R}\,R^T]^T = 0$ where $\dot{R}\,R^T$ not always symmetric.
- Define angular velocity by $\dot{R}\,R^T = [\omega]_\times$, hence $\dot{R} = [\omega]_\times R$.
- Point velocity: $\dot{x}(p,t) = v_c(t) + \omega(t) \times \big(x(p,t) - x_c(t)\big)$

Kinetic Energy

- Total $T = \frac{1}{2}\,m\,\|v_c\|^2 + \frac{1}{2}\,\omega^T I(t)\,\omega$.
- Inertia tensor in world frame: $I(t) = R(t)\,I_{\text{ref}}\,R^T(t)$, where $I_{\text{ref}} = \int_{V_{\text{ref}}} \mu\,\big(\|p\|^2 I_3 - p\,p^T\big)\,dV$.

Linear & Angular Momentum

- Linear: $P = m\,v_c, \quad \dot{P} = F = \sum_i F_i$.
- Linear momentum changes only if a net external force acts on the body.
- Angular: $L = I\,\omega, \quad \dot{L} = \tau = \sum_i (x_i - x_c) \times F_i$.

Rotation Parameterizations

- Euler angles (3 DOF) — prone to gimbal lock.
- Quaternion (4 DOF) — used in Bullet 3, avoid gimbal lock.
- Axis-angle (3 DOF) — used in Rigid IPC, avoid gimbal lock
- Gibbs / Rodrigues vector $g = u\,\tan(\theta/2)$ (3 DOF).
- Full 3×3 matrix (9 DOF) — used in Affine Body IPC.

vector cross product For two 3D vectors $a = (a_x, a_y, a_z)$ and $b = (b_x, b_y, b_z)$,
$a \times b = \big(a_y b_z - a_z b_y,\ a_z b_x - a_x b_z,\ a_x b_y - a_y b_x\big)$

6. Keyframe Animation

Computer Animation

- Models have parameters: polygon positions, normals, spline control points, joint angles, camera parameters, lights, color, etc.
- n parameters define and n-dimensional state space
- Values of n parameters = point in state space
- Animation designed by path through state space
- To produce animation:
  1. start at beginning of state space path
  2. set the parameters of your model
  3. render the image
  4. move to next point along state space path
  5. Goto 2
- Path usually defined by a set of motion curves (one for each parameter)
- Animation = specifying state space trajectory

Animation vs Rigging and Modeling

- Modeling, rigging and animation are tightly coupled.
  Modeling: What is the neutral shape of the object?
  Rigging: What are the control knobs and what do they do? (embed skeleton, prepare for animation)
  Animation: how to vary the knobs to generate desired motions?
- Building models that are easy to control is a very important part of doing animation: Hierarchical modeling can help
- Where does rigging end and animation begin? Sometimes a fuzzy distinction

Basic Animation Techniques

- Traditional (frame by frame)
- Keyframing
- Procedural techniques: animate 1 object then procedural 2nd object based on math equations $w_1R_1 = w_2R_2 \rightarrow w_2 = \frac{w_1R_1}{R_2}$
- Behavioral techniques (e.g. flocking)
- Performance-based (motion capture): recording a performance
- Physically-based (dynamics)

Traditional Animation

- Film runs at 24 frames per second (fps)
  That's 1440 pictures to draw per minite
  1800 fpm for video (30fps)
- Production issues:
  Need to stay organized for efficiency and cost reasons
  Need to render the frams systematically
- Artistic issues:
  How to create the desired look and mood while conveying stroy?
  Artistic vision has to be converted into a sequence of still frames
  Not enough to get the stills right-must look right at full speed: Hard to "see" the motion given the stills, hard to "see" the motion at the wrong frame rate
- Rookie animators issues: They generate too many keyframes; They keyframe the limbs to penetrate throught the body. They cause foot skate. They exceed the natural joint limits.

Traditional Animation Process

- Story board sequence of sketches with story
- Key frames: Important frames; Motion-based description; Example: beginning of stride, end of stride
- Inbetweens: draw remaining frames: Traditionally done by (low-paid) human animators

Layered Motion (in 2D)

- It's often useful to have multiple layers of animation: How to make an object move in front of a background? Use one layer for background, one for object; Can have multiple animators working simultaneously on different layers, avoid re-drawing and flickering
- Transparent acetate allows multiple layers: Draw each separately; Stack them on a copy stand; Transfer onto film by taking a photograph of the stack

Principles of Traditional Animation [Lasseter, SIGGRAPH

- "Animator Guide" textbook
- Stylistic conventions followed by Disney's animators and others (but this is not the only intersting style, of course)
- From experience built up over many years
- Squash and Stetch: convey rigidity and mass of an object by distoring its shape
- Timing: speed conveys mass, personality
- Anticipation: the preparation for an action
- Followthrough and overlapping action: the termination of an action and establishing its relationship to the next action
- Slow in and out: the spacing of the in-between frames to achieve subtlety of timing and movement
- Arcs: motion is usually curved
- Exaggeration: emphasize emotional content
- Secondary Action: action that results from another action
- Appeal: audience must enjoy watching it

Computer Animation

Computer-Assisted Animation

- Computerized Cel painting: Digitize the line drawing, color it using seed fill; Eliminates cel painters; Widely used in production (little hand painting any more); e.g. Lion King
- Cartoon Inbetweening: Automatically interpolate between two drawings to produce inbetweens (similar to morphing); Hard to get right: inbetweens often don't look natural. what are the parameters to interpolate? Not clear. not used very often

True Computer Animations

- Generate images by rendering a 3D model
- Vary parameters to produce animation

- Brute force: Manually set the parameters for every frame; 1440n values per minute for n parameters; Maintenance problem
- Computer keyframing: Lead animators create important frames; Computers draw inbetweens from 3D; Dominant production method

Interpolation

- Hard to interpolate hand-drawn keyframes: Computers don't help much
- The situation is different in 3D computer animation: Each keyframe is a defined by a bunch of parameters (state); Sequence of keyframes = points in high-dimensional state space
- Computer inbetweening interpolates these points using splines

Keyframe Animation

- Despite the name, there aren't really keyframes, per se
- For each variable, specify its value at the "important" frames. Not all variables need agree about which frames are important
- Hence, key values rather than key frames
- Greate path for each parameter by interpolating key values

Keyframing: Issues

- What should the key values be?
- When should the key values occur?
- How can the key values be specified?
- How are the key values interpolated?
- What kinds of bad things can occur from interpolation? Invalid configurations (pass through objects); Unnatural motions (painful twists/bends); Jerky motion

Keyframing: Production Issues

- How to learn the craft: apprentice to an animator; practice
- Pixar starts with animators, teaches them computers and starts with computer folks and teachs them some art

Interpolation

- Splines: non-uniform, C1 is pretty good
- Velocity control is needed at the keyframes
- Classic example: a ball bouncing under gravity: zero vertical velocity at start; high downward velocity just before impact; lower upward velocity after; lower upward velocity after; motion produced by fitting a smooth spline looks unnatural
- What kind of spline might we want to use?
- Hermite is good

Problems with Interpolation

- Splines don't always do the right thing
- Classic problems:
  Important constraints may break between keyframes
  feet sink through the floor
  hands pass through walls
  3D rotations: Euler angles don't always interpolate in a natural way
- Classic solutions:
  More keyframes
  Quaternions help fix rotation problems

Example: Toy Story (1995) & Tory Story 2

Some Research Issues

- Inverse kinematics: How to plot a path through state space; Multiple degrees of freedom; Also important in robotics

Baking: Baking refers to calculating and saving out the result of some computer animation process, such as mesh vertex positions at every frame, or joint angles at every frame. The results of baking are then used as input to subsequent computer animation stages.

7. Motion Capture

What is Motion Capture?

- performance animation
- Motion capture is the process of tracking real-life motion in 3D and recording it for use in any number of applications.

Why Motion Capture?

- Keyframes are generated by instruments measuring a human performer - they do not need to be set manually
- The details of human motion such as style, mood, and shifts of weight are reproduced with little effort
- Keyframes are slow

Why not?

- Difficult for non-human characters: Can you move like a hamster/duck/eagle? Can you capture a hamster's motion?
- Actors needed: Which is more economical: Paying an animator to place keys or hiring a martial arts expert

When to use Motion Capture?

- Complicated character moation: Where "uncomplicated" ends and "complicated" begins is up to question; A walk cycle is often more easily done by hand; A Flying Monkey Kick might be worth the overhead of mocap
- Can an actor better express character personality than the animator?

1. Mocap Technologies: Optical (Camera)

- Multiple high-resolution, high-speed cameras
- Light bounced from camera off of reflective markers
- High quality data
- Markers placeable anywhere
- Lost of work to extract joint angles
- Occlusion
- Which marker is which? (correspondence problem)
- 120-240 Hz @ 1Megapixel
- equipments are expensive
- markers are on the surface
- need manual cleanup to get good data
- highest resulting
- most common in film
- CMU Motion Capture Dataset
- Facial Motion Capture

2. Mocap Technologies: Electromagnetic

- Sensors give both position and orientation
- No occlusion or correspondence problem
- Little post-processing
- Limited accuracy
- No camera
- Con: very heavy backpack
- NOTIOM: chinese company, design a network s.t. can transit data to a computer that does not need to be in the backpack.

3. Mocap Technoligies: Exoskeleton

- Really Fast (500Hz)
- No occlusion or correspondence problem
- Little error
- Movement restricted
- Fixed sensors
- Con: alter the motion due to equipment

Question: How can we determine if 2 pose are similar?

- $P_1, P_2 \in \mathbb{R}^{60}$ position
- $\dot{P}_1, \dot{P}_2 \in \mathbb{R}^{60}$ velocity
- Loss $= \sum_{i=1}^{60} w^i (P_1^i - P_2^i)^2 + \lambda \cdot \sum_{i=1}^{60} \hat{w^i} (\dot{P_1}^i - \dot{P_2}^i)^2$, $w^i \in \mathbb{R}$
- Edge $\Leftrightarrow$ Loss $< \varepsilon$
- J. Lee, J. Chai, P. Reitsma, J. Hodgins, N. Pollard: Interactive Control of Avatars Animated with Human Motion Data, SIGGRAPH 2002

8. Quaternion

Rotations

- Very important in computer animation
- Joint angles, rigid body orientations, camera parameters
- 2D or 3D

Rotation in 3D:

- Orthogonal matrices: $RR^T = R^T R = 1$, $det(R) = 1$
- $R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$

Euler Angles

- Roll: rotate around z-axis
- Pitch: rotate around x-axis
- Yaw: rotate around y-axis
- no redundancy (good)
- gimbal lock singularities

Gimbal Lock:

When all three gimbals are lined up (in the same plane), the system can only move in two dimensions from this configuration, not three, and is in gimbal lock.

Euler Angles → Rotation matrix:

- First rotate around X by angle θ_1, then around Y by θ_2, then around Z by angle θ_3
- $R = \begin{bmatrix} cos(\theta_3) & -sin(\theta_3) & 0 \\ sin(\theta_3) & cos(\theta_3) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta_2) & 0 & sin(\theta_2) \\ 0 & 1 & 0 \\ -sin(\theta_2) & 0 & cos(\theta_2) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_1) & -sin(\theta_1) \\ 0 & sin(\theta_1) & cos(\theta_1) \end{bmatrix}$
- simplified and Rotation Matrix to Euler Angles

Quaternions

- generally considered the "best" representation
- redundant (4 values), but only by one DOF (not severe)
- stable interpolations of rotations possible
- Generalization of complex number
- Three imaginary numbers: i, j, k
- $i^2 = -1$, $j^2 = -1$, $k^2 = -1$, $ij = k$, $jk = i$, $ki = k$, $ji = -k$, $kj = -i$, $ik = -j$
- $q = s + xi + yj + zk$, s, x, y, z are scalars
- Quaternions are not commutative: $q_1 q_2 \neq q_2 q_1$
- all usual amnipulation are valid, except cannot reverse multiplication order

Quaternion Properties

- $q = s + xi + yj + zk$
- Norm: $|q|^2 = s^2 + x^2 + y^2 + z^2$
- Conjugate quaternion: $\bar{q} = s - xi - yj - zk$
- Inverse quaternion: $q^{-1} = \bar{q}/|q|^2$
- Unit quaternion: $|q| = 1$
- Inverse of unit quaternion: $q^{-1} = \bar{q}$

Rotation to Unit Quaternions

- Let (unit) rotation axis to be $[u_x, u_y, u_z]$ and angle θ
- Corresponding quaternions is $q = cos(\frac{\theta}{2}) + sin(\frac{\theta}{2})u_x i + sin(\frac{\theta}{2})u_y j + sin(\frac{\theta}{2})u_z k$
- Composition of rotations q_1 and q_2 equals $q = q_2\, q_1$
- 3D rotations do not commute
- $-q = -cos(\frac{\theta}{2}) - sin(\frac{\theta}{2})u_x i - sin(\frac{\theta}{2})u_y j - sin(\frac{\theta}{2})u_z k$
  $-q = -cos(\frac{\theta}{2}) - sin(\frac{\theta}{2})(u_x i + u_y j + u_z k)$
  $-q = cos(180 - \frac{\theta}{2}) + sin(\frac{\theta}{2})(-(u_x i + u_y j + u_z k))$
  $-q = cos(\frac{360-\theta}{2}) + sin(180 - \frac{\theta}{2})(-(u_x i + u_y j + u_z k))$
  $-q = cos(\frac{360-\theta}{2}) + sin(360 - \frac{\theta}{2})(-(u_x i + u_y j + u_z k))$
  $-q = cos(\frac{-\theta}{2}) + sin(-\frac{\theta}{2})(-(u_x i + u_y j + u_z k))$
  $-q = cos(\frac{\theta}{2}) + sin(\frac{\theta}{2})(-(u_x i + u_y j + u_z k)) = q$

SLERP: $Slerp(q_1, q_2, u) = \frac{sin((1-u)\theta)}{sin(\theta)} q_1 + \frac{sin(u\theta)}{sin(\theta)} q_2$

> Notes: if $q1$ and $q2$ are unit quaternions, then the slerp produces a unit quaternion

- $cos(\theta) = q_1 \cdot q_2 = s_1 s_2 + x_1 x_2 + y_1 y_2 + z_1 z_2$
- If the cosine is positive, then the path from q_1 to q_2 is shorter, otherwise the path from q_1 to $-q_2$ is shorter

9. Forward Kinematics

- Forward kinematics: Given angles, compute positions, starting from the joint angles, find where the end-effector is in space.
- Forward Dynamics: starting from applied joint torques, predict how the joint angles (and therefore the end-effector) will accelerate and move over time $M\theta + D\dot{\theta} + f_{int}(\theta) = t$
- Inverse kinematics: refers to calculating the joint angles from end-effector positions. This is a satatic calculation; no dynamics is involved. $(x, y) \mapsto \theta$, no solution if $x^2 + y^2 > (l_1 + l_2 + l_3)^2$, exactly one solution if $=$, more than one solution if $<$
- Inverse dynamics: we calculate joint torques, so that the character's end effectors undergo a prescribed trajectory over time. $\theta = \theta(t) \mapsto t = t(t)$

Inverse Kinematics

Steps

- $\Delta p \mapsto \Delta\theta$, $FK(\theta) = p + \Delta p$
- 1. User specifies Δp
- 2. Run IK $\rightarrow \Delta\theta$
- 3. $\theta \leftarrow \theta + \Delta\theta$
- 4. Compute $\hat{p} = FK(\theta)$
- 5. $\Delta p \mapsto p + \Delta p - \hat{p}$
- 6. Repeat (be careful with user's new input)
- Cannot reach Δp directly because we cancel out $O(\Delta\theta^2)$ and numerical error

Pseudoinverse Method "Least squares"

- $\Delta\theta = J^{\dagger} \cdot \Delta p$
- $J^{\dagger}$ logic $= J^{-1}$, we cannot write -1 directly because J is not a square matrix
- $J^{\dagger} = J^T(JJ^T)^{-1} \in R^{3\times 2}$
- proof: $J \cdot \Delta\theta = J \cdot J^{\dagger}\Delta p = JJ^T(JJ^T)^{-1}\Delta p = \Delta p$

Tikhonove Regularization "Damped least squares"

- $E(\Delta\theta) = \frac{1}{2}||J\Delta\theta - \Delta p||^2 + \frac{\lambda}{2}||\Delta\theta||^2$
- $\lambda \geq 0, \lambda = 10^{-3}$, when $\lambda = 0$ this equation is useless because of infinite number of solution
- $\Delta\theta = argmin_{\Delta\theta} E(\Delta\theta)$
- $E : R^3 \mapsto E$
- $0 = \frac{dE}{d\Delta\theta} = (J\Delta\theta - \Delta p)^T J + \lambda(\Delta\theta)^T$
- $0 = J^T(J\Delta\theta - \Delta p) + \lambda\Delta\theta \rightarrow (J^TJ + \lambda I)\Delta\theta = J^T\Delta p$

Skinning

Linear Blend Skinning

- pro: very fast, con: no physics
- skinning transformation: $X \mapsto R_jX + t_j$, $j = 1, ..., J$
- O_j, E_j^1, E_j^2, E_j^3, o_j, e_j^1, e_j^2, e_j^3 are known, chosen by rigger
- R_j: 3×3 rotation matrix
- t: 3×1 translation vector
- x: 3×3 non-bind position
- X: 3×3 neutral/bind/undeformed position
- J: number of joints
- $\mapsto$: automorphism

Solve unknown R_j:

- $R_j \cdot E_j^k = e_j^k \rightarrow R_j = [e_j^1\, e_j^2\, e_j^3][E_j^1\, E_j^2\, E_j^3]^{-1}$ where $k = 1, 2, 3$
- proof: $R_j \cdot E_j^2 = [e_j^1\, e_j^2\, e_j^3] \cdot [0\,1\,0]^T = e_j^2$
- $E_j^k = [E_j^1\, E_j^2\, E_j^3]$ world $\rightarrow$ undeformed position
- $e_j^k = [e_j^1\, e_j^2\, e_j^3]$ world $\rightarrow$ deformed position

Solve unknown t_j:

- $O_j \mapsto o_j$
- $t_j + R_j \cdot O_j = o_j \rightarrow t_j = o_j - R_j \cdot O_j$

Skinning Weights (pre determined):

- $\sum_{j=1}^{J} w_i^j = 1$
- $x_i = skin(X_i) = \sum_{j=1}^{J} w_i^j \cdot X_i^j = \sum_{j=1}^{J} w_i^j(t_j + R_j \cdot X_i)$
- $x_i = \sum_{j=1}^{J} w_j M_j \bar{X}_i$

Dual Quaternions

- pro: easy & fast, con: no physics
- $t_j, R_j, R_j \mapsto q_j$
- Form $(q_j, \frac{1}{2}t_j \cdot q_j) := \hat{q}_j$ dual quaternion
- Blend: $(\hat{q_0}, \hat{q_1}) = \sum_{j=1}^{J} w_{ij}(q_j, \frac{1}{2}t_j \cdot q_j)$
- Normalize: $(\frac{\hat{q_0}}{||\hat{q_0}||}, \frac{\hat{q_1}}{||\hat{q_0}||} - \frac{\hat{q_1}\cdot\hat{q_0}}{||\hat{q_0}||^3}\hat{q_0}) =: (\tilde{\hat{q}_0}, \tilde{\hat{q}_1})$
- Normalize dual quaternion to unit quaternion
- $\tilde{\hat{q}_0} \rightarrow R$: rotation
- $\tilde{\hat{q}_1} = \frac{1}{2}t\tilde{\hat{q}_0} \rightarrow t = 2\tilde{\hat{q}_1}\tilde{\hat{q}_0}^{-1}$: translation

Character Rigging

- Match skeleton to the Character
- skinning
- Check Maya and Keyframe animation

11. Physically-based Facial Modeling

Motivation

- Why a talking head?
  Enhanced comminication for pople with disabilities
  Training scenario software
  Entertainment: Games and Movies
- Why physically based?
  Unburdens animators
  Provides more realistic looking simulations

Anatomy of the face

- There are 268 voluntary muscles that contribute to your expression
- Linear/Parallel muscles (share a common anchor): contract longitudinally towards their origin, e.g. levator labii sup., zygomaticus minor/major
- Sheet muscles (run parallel, activated together): composed of several linear muscles side-by-side, e.g. frontalis
- Sphincter muscles (contract to a center point): contract radially towards a center point, e.g. orbicularis oris, orbicularis oculi

Muscles

- Bundles of thousands of individual fibers: Thankfully, can be modeled as these bundles; When activated, all of the fibers contract
- Contraction only: Most parts of the body use opposing pairs of muscles, but the face relies on the skin
- Bulging: Occurs due to volume preservation; Thicker on contraction, thinner on elongation; Important for realistic faces (e.g. pouting lips)

Skin

- Epidermis: Thin, stiff layer of dead skin (no real simulation)
- Dermis: Primary mechanical layer; Collagen and Elastin fibers
- Subcutaneous or Fatty tissue (Fat): Allows skin to slide over muscle bundles; Varies in thickness

Modeling viscoelastic skin

- Collagen fibers - low strain for low extensions
- Near maximum expansion, stress rises quickly
- When allowed to, elastin fibers return system to rest state quickly
- Biphasic model: Two picewise linear models; Threshold extension to pock spring constant

The skull

- Unlike most of the body, the face only has a sigle joint and 2 bones: jaw bone and skull
- All other expression is due to computer unfriendly soft tissues
- Can be treated as a rigid body

Facial Action Coding System (FACS) Encode the expression using "atomic" expressions (as defined by the FACS standard). Specify intensity A-E, and permit combinations of expressions. Example: 26E + 11F + 5B.

- Proposed by Ekman and Friesan in 1978
- Describes facial movement in terms of the muscles involved
- Purposely ignores invisible and non movement changes (such as blushing)
- Defines 46 action units pertaining to expression-related muscles
- Additional 20 action units for gross head movement and eye gaze
- qualitative

Blendshapes: linear blendshape facial rig

- input FACS coordinate, output facial position
- all the mesh must have the same amount of vertices, only position changes
- model: $x = n + \sum_{i=1}^{N} \alpha_i (b_i - n)$
- $1 \geq \alpha_i \geq 0$: precreated when design the game, "sliders", "extra parameters"
- $b_i - n$: difference between blendshape and neutral
- ICP iterative closet point [USC 1991 Gerard Medioni]

Combination shapes: improve realism

- $c_{ij}, i = 1, ..., N,\ j = 1, ..., N$
- $\alpha_4 = 1,\ \alpha_7 = 1 \Rightarrow \beta_{47} = 1$
- $x = n + \sum_{i=1}^{N} \alpha_i (b_1 - n) + \sum_{i<j} \beta_{ij} (c_{ij} - n)$

MPEG-4 Facial Animation Specify the expression by giving the displacement vector for a set of facial landmarks. The landmarks are specified by the standard.

- Defines 84 feature points with position and zone of influence on a few basis keyframes of a standard 3D mesh
- Defines animation independently of the visual rep.
- 68 facial action parameters (FAPS), defined in terms of face independedn FAP units (FAPUs)
- Most define a rotation or translation of one or more feature points, with a few selecting entirely new key frames (e.g. an emotion basis)
- Same animation can be used on different model, provided the model is properly annotated
- quantitative
- specify physical displacement of landmark
- pro: can mock up the point, render easily
- con: hard to simulate

12. Maya

1. Modeling

- models, created by modeler
- will create triangle mesh + texture mapping (texture artist / UV artist)
- hero character (main character): a good number no more than 100k in computer games (in real time), 500k if in film
- Z-BRUSH: create the model
- MAYA: can create but most poeple import model from Z-BRUSH [Economic moat](An economic moat refers to a company with a long-term, sustainable competitive advantage, which protects its profits from competitors and external threats. If a business is said to have an economic moat, or "moat," for short, then it has a differentiating factor enabling the company to hold a competitive edge)
- SUBSTANCE DESIGNER: can virtually print the color, UV in texture mapping
- 3D S MAX: similar to MAYA
- Blender: open source

Steps:

- Block-out the shape with primitives (polyCube, polyCylinder, polySphere, . . .)
- Refine: Booleans (union / intersect / difference) or Combine (Merge Vertices for seamless parts)
- Parent rigid sub-parts so transforms propagate (e.g. parent lampShade stand;). Rule: child inherits translation/rotation/scale of the parent.
- Freeze transforms, delete history, name all meshes (e.g. Lamp_Base_GEO).

2. Rigging

- rigger: who create the skeleton
- why rigging? we convert model into "animation ready model"
- famous rigging algorithm: linear blend skinning (LBS), dual quaternions, anatomy based

Steps:

- Place joints in logical order such as Base, LowerArm, UpperArm, and Head, then parent them so rotations flow down the chain.
- Decide whether the chain will move by Forward Kinematics (rotate each joint by hand) or by an Inverse Kinematics handle that lets you drag the end joint while Maya solves the intermediate angles automatically.
- Create simple controller curves, snap them to the joints or the IK handle, and freeze their transforms so they start at zero.
- Bind the lamp mesh to the joint chain so the geometry follows the motion; paint weights so each part of the mesh is influenced by the correct joint.

What is an IK handle?

- An IK handle is a Maya tool that connects the start and end joints of a chain. When you move the handle, Maya calculates the joint rotations needed to keep the end joint on the handle, making tasks such as foot placement or lamp-head positioning much easier.

Why bind the skin? Rigging vs. Skinning

- Rigging sets up the skeleton and control system. Skinning binds the visible mesh to that skeleton so vertices follow the joints. Without skinning the skeleton would move but the mesh would stay behind, so binding is essential for the model to deform correctly.
- Charater skinning refers to calculating the mesh vertex positions as a function of the character's joint angles.
- Character rigging is a broader concept. It involves any method to calculate mesh vertex positions, even if it does not employ the joint angles. For example, one may use inverse kinematics, blendshap deformers, free-form deformation or any other suitable method.

3. Animation

- animator provide angles
- plot the angles in a certian time
- In game industry: technical artist (TA)
- In film: technical director (TD) → in between computer science & art.

Steps:

- Set the playback range, for example 0–120 frames at 24 fps.
- Pose the controls at important moments such as crouch, lift-off, apex, landing, and settle, and press S to set a keyframe at each pose.
- Add breakdown keys between the main poses to smooth the motion and adjust spacing.
- Switch the curves in the Graph Editor from stepped to spline tangents and edit the handles to add ease-in and ease-out.
- Playblast the scene to check timing and arcs, tweak keys until the jump looks natural, and render the final animation.

13. Constraints and contact The idea of constrained particle dynamics is that our description of the system includes not only particles and forces, but restrictions on the way the particles are permitted to move. For example, we might constrain a particle to move along a specified curve, or require two particles to remain a specified distance apart. The problem of constrained dynamics is to make the particles obey Newton's laws, and at the same time obey the geometric constraints.

14. Collision Detection

Erwin Coumans: creater of Tiny Differentiable Simulator which is a header-only C++ and CUDA physics library for reinforcement learning and robotics with zero dependencies.

Classes of Objects & Problems

- 2D vs. 3D
- Convex (random 2 point connect will not go out of boundary) vs. Non-Convex
- Polygonal (triangle, quads) vs. Non-Polygonal
- Open surfaces vs. Closed volumes
- Geometric (thin-shell) vs. Volumetric
- Rigid vs. Non-rigid (deformable/flexible)
- Pariwise vs. Multiple (N-Body)
- CSG (Constructive solid geometry: intersection, union, complement) vs. B-Rep (polygonal mesh: list of vertices and triangles)
- Static vs. Dynamic
- And so on ... This may include other geometric representation schemata, etc.

Bounding Volumes (BVs)

- Sphere, AABB, OBB, 6-dop, Convex Hull
- $\rightarrow$ incresing complexity & tightness of fit
- $\leftarrow$ decresing cost of (overlap tests + BV update)

Bounding Volumes Hierarchies

Model Hierarchy:

- each node has a simple volume that bounds a set of triangles
- children contain volumes that each bound a different portion of the parent's triangles
- The leaves of the hierarchy ususally contain indicidual triangles (or the number of triangles in the leaf below a threshold)

Bounding Volume Test Tree (BVTT):

- recurse on both tree
- recurse on higher level tree (better but why?)

Principal Component Analysis (PCA):

- Algorithm to split
- Use to find the normal vector of an Objects
- data: $x_i, i = 1, ..., N$
- center: $c = \frac{1}{N}\sum_{i=1}^{N} x_i$
- $y_i = x_i - c, i = 1, ..., N$
- Covariance Matrix: $A = \frac{1}{N}\sum_{i=1}^{N} y_i y_i^T$
- $A^T = A$ is symmetric matrix because $(y_i y_i^T)T = (y_i^T)^T \cdot y_i^T = y_i \cdot y_i^T$
- eigendecomposition $A = U\Lambda U^T$ where A is 3*3 symmetric, U is 3*3 Orthogonal, Λ is 3*3 diagonal
- $\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3 \end{bmatrix}$ where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$
- $U = [D * *]$ where D is desired direction

Self Collision Detection:

- Copy the BVH then compare
- Another way is to brute force (compare every pair of triangles except neighboring ones) and parallel computing

Spatial Data Structures & Subdivision

- Uniform Spatial Sub: uniformly split among the space
- Quadtree (2D) / Octree (3D): split evenly into 4, and concentrate into one grid
- kd-tree: similar to quad tree but split unevenly, and only split in 1D, common in AI
- BSP-tree: kd-tree but accept tilted split

Uniform Spatial Subdivision

- Decompose the objects (the entire simulated environment) into identical cells arranged in a fixed, regular grids (equal size boxes or voxels)
- To represnet an object, only need to decide which cells are occupied. To perform collision detection, check if any cell is occupied by two objects
- Storage: to represent an object at resolution of n voxels per dimension requires upto n^3 cells
- Accuracy: solids can only be "approximated"
- step 0 : clear the list
- step 1 : go over the first object, add element to grid's list
- step 2 : do the samething for the second object
- step 3 : for every cell, if two lists are non-empty, pair-wise collision checking
- self collision detection: for every cell, if the list has 2+ non-neighboring entries, pair-wise collision checking
- issue: space memory (need to store empty cell) and speed (need to go over all cells)
- solution: hash table with psudo-randomness
- step 1 : convert every grid to a very long table (e.g. index = row $*$ num horizontal cell + collumn)
- step 2 : condense it to a small table using a hash function
- step 3 : erase the long table
- for each

signed distance field

- Six-DoF Haptic Rendering of Contact between Geometrically Complex Reduced Deformable Models
- for each grid, store closest distance A to the object
- + outside, - inside
- for one point, if no signed distance field, check if $A \cdot N > 0 \rightarrow$ outside, or A and N has the same direction
- if signed distance field, read 8 points from memory
- bilinear in 2D, trilinear in 3D

Contact force: $F_C = -k_C dN$

- $-k_C$ penalty stiffness, d signed distance field value ($d < 0$ when in contact)
- Summ over all points
- Torque $= r \times F_C$
- Also compute gradients

Distributed Contact

- difficult contact - multiply contact point
- one is point shell (deformable), one is signed distance (rigid)

15. Haptics

What is haptics?

- relating to or based on the sense of touch
- characterized by a predilection for the sense of touch ¡a haptic person¿

Current force feedback devices

- 1 DOF (SAMSUNG / IMMERSION TECHNOLOGIES)
- 2 DOF (IMMERSION TECHNOLOGIES)
- 3/6 DOF (PHANTOM - Sensable Technologies) (OMEGA - Force Dimension)
- medical training (XITACT)
- bimanual interfaces (JPL)
- grasping capabilities (VIRTUAL TECHNOLOGIES)
- ultimate workstations (IMMERSION)
- golf trainer (Gradener And Sasch Orlic, PGA Golf Professionaal)

God Object Algorithm

- prevent tunneling
- god object position 1.never inside object, 2.local minimum distance to you relevant to previous location
- $F = k(god - handle)$
- god object decrease distance to the local minimum

16. Fluids

What Are Fluids?

- Liquids: water, pigment
- Gases: wind, smoke
- Granular: sand, snow
- Others: fire

The Navier-Stokes Equations

- The math foundation of fluid simulation, the newton second law for fluid
- $\frac{D\vec{u}}{Dt} = \frac{1}{\rho}\nabla \cdot \sigma + \vec{f}$
- Or after expansion, $\frac{\partial \vec{u}}{\partial t} = -(\vec{u} \cdot \nabla)\vec{u} - \frac{1}{\rho}\nabla p + v\Delta\vec{u} + \vec{f}$

Different Approaches

Eulerian:

- Tracks properties at fixed grid points
- Grid-based
- Good for capturing flow
- Examples: MAC(Marker And Cell), Stable Fluids

Lagrangian:

- Tracks individual particles
- Particles or mesh
- Good for shape deformation
- Examples: SPH (Smoothed Particle Hydrodynamics), Vortex sheets method

Hybird Approaches:

- FLIP (Fluid Implicit Particle)
- APIC (Affine Particle in Cell)
- MPM (Material Point Method)

1D Example: Cars on a Freeway Eulerian

- Tracks a fix point on the road
- Velocity field: $u(x,t)$
- Like a speed trap
- Density: $\rho(x,t)$, number of cars per unit length near x at time t
- Flux: ρ
-

Divergence Theorem

- For close interval [a,b], the net flux out equals the total source inside: ρ
- Assumes constant density
- Local form: $s(x,t) = \rho$

Stable Fluids

- Paper: Jos Stam. Stable Fluids. SIGGRAPH 1999
- Cited by 2677
- Key Features: Grid-based discretization; Projection operator based on Helmholz-Hodge decomposition; Operator splitting scheme; Semi-Lagrangian advection step (unconditionally stable)

Gird Based Discretization

- A 3D grid consists of $N \times N \times N$ grid points with uniform space Δx
- A scalar field $\phi(\vec{x},t)$ at a specific time t is discretized as $N \times N \times N$ scalars
- A vector field $\vec{F}(\vec{x},t)$ at a specific time t is discretized as $N \times N \times N$ vectors
- Use trilinear interpolation if one needs to access a quantity off the grid point

Operator Splitting

- Instead of solveing: $\frac{d\vec{u}}{dt} = A(\vec{u}) + B(\vec{u})$
- We solve: First $\frac{d\vec{u}}{dt} = A(\vec{u})$, then: $\frac{d\vec{u}}{dt} = B(\vec{u})$
- Justification: $e^{\Delta t(A+B)} = e^{\Delta tA}e^{\Delta tB} + O(\Delta t^2)$
- Applied to Navier-Stokes equations: $\frac{\partial \vec{u}}{\partial t} = P(-(\vec{u}\cdot\nabla)\vec{u} + v\Delta\vec{u} + \vec{f})$
- $w_0(x) = \vec{u}(\vec{x},t) \rightarrow$ add force $\vec{f}$
- $w_1(x) \rightarrow$ advect $-(\vec{u}\cdot\nabla)\vec{u}$
- $w_2(x) \rightarrow$ diffuse $v\Delta\vec{u}$
- $w_3(x) \rightarrow$ project P
- $w_4(x) = \vec{u}(\vec{x}, t+\Delta t)$

Step 1: Add Force $\frac{\partial \vec{u}}{\partial t} = \vec{f}$

- Do a simple forward Euler step $\vec{w_1}(\vec{x}) = \vec{w_0}(\vec{x}) + \vec{f}(\vec{x},t)\Delta t$
- In grid discretization $\vec{w_1}^{i,j,k} = \vec{w_1}^{i,j,k} + \vec{f}^{i,j,k}\Delta t$

Step 2: Advection $\vec{w_2} = \vec{w_1}(\vec{p}(\vec{x}, -\Delta t))$

- $\vec{p}(\vec{x}, -\Delta t) := \vec{x} - \vec{w_1}(\vec{x})\Delta t$
- Trilinear interpolate $\vec{w_1}$ to get $\vec{w_1}(\vec{p}(\vec{x}, -\Delta t))$
- $\vec{w_2}(\vec{x}) \leftarrow \vec{w_1}(\vec{p}(\vec{x}, -\Delta t))$

Step 3: Diffusion $\frac{\partial \vec{u}}{\partial t} = v\Delta\vec{u}$

- Solve using backward Euler: $\frac{\vec{w_3}\vec{w_2}}{\Delta t} = vL\vec{w_3}$, $\vec{w_2}, \vec{w_3} \in \mathbb{R} \Leftrightarrow (I - v\Delta tL)\vec{w_3} = \vec{w_2} \Leftrightarrow \vec{w_3} = (I - v\Delta tL)^{-1}\vec{w_2}$

Step 4: Projection $\vec{w_4} = P(\vec{w_3})$

- Helmholz-Hodge decomposition $\vec{w_3} = \nabla \times \vec{A} + \nabla\phi$

17. Sound simulation make virtual objects vibrate and turn those vibrations into audio you can hear.

Synthesizing Sounds from Physically Based Motion - James F. O'Brien

Synthesizing sounds from rigid-body simulations - James F. O'Brien

how sound travels:

- wave equation: $\nabla^2 p = \frac{1}{c^2}\frac{\partial^2 pt}{\partial t^2}$ (c = speed of sound)
- BEM (Boundary element method): solve the wave on the object's surface, no volume grid
- High frequency waves act like straight rays → can't bend around corners.

Stereo sound (2 channel, left ear & right ear, can determine location) vs mono (1 channel)

HRTF (head related transfer function): is defined as the ratio between the Fourier transform of the sound pressure at the entrance of the ear canal and the one in the middle of the head in the absence of the listener, filters quantifying the effect of the shape of the head, body, and pinnae on the sound arriving at the entrance of the ear canal.

human audible range: 25-20000Hz

how it vibrates:

- $M\ddot{u} + D\dot{u} + f_{int}(\mu) = f_{ext}(t)$ where $f_{int}(u) = K\mu + \Theta(||u||^2)$
- $M\ddot{u} + D\dot{u} + K(\mu) = f_{ext}(t)$
- K: stiffness matrix
- D: damping matrix $D = \alpha M + \beta K$ Rayleigh Damping to model energy loss, $\alpha, \beta \geq 0$ is constant

Modal analysis:

- Solve generalized eigenvalue problem $Kx = \lambda Mx$
- get modes (ψ) and natural frequencies (λ)
- n vertices; $K, M : 3n \times 3n$ matrix; $x : 3n \times 1$ vector
- $\lambda \geq 0$
- $\lambda_1, \psi_1, \lambda_2, \psi_2, ...$
- $U = [\psi_1, ..., \psi_k]$ columns are natural way to vibrate
- k = 20: Keep 20 lowest modes, they cover the loud, low frequency band humans hear; fewer modes = faster and still sounds right.
- $\mu(t) = U \cdot q(t)$: track modal coordinates q(t); each mode behaves like a damped spring

Turn vibration into sound:

- $(U^T MU)\ddot{q} + (U^T DU)\dot{q} + (U^T KU)q = U^T f_{ext}$
- solution: $q_1 = q_1(t), q_2 = q_2(t), ..., q_{20} = q_{20}(t)$
- combine together to get 1 signal: $s(t) = \sum_{i=1}^{20} q_i(t)$
- add with weights: $s(t) = \sum_{i=1}^{20} c_i q_i(t)$ (many method to compute weights)

Why frequency $f = \sqrt{\lambda}/(2\pi)$ (and $\omega = \sqrt{\lambda}$)

- Start from the undamped, free vibration equation $M\ddot{u} + Ku = 0$.
- Assume a harmonic solution (object vibrates like a sine wave) $u(t) = \phi\, e^{i\omega t} \quad \Rightarrow \quad \ddot{u} = -\omega^2 u$.
- Plug in: $-\omega^2 M\phi + K\phi = 0 \implies K\phi = \omega^2 M\phi$.
- Compare with the modal eigenproblem $Kx = \lambda Mx \quad \Rightarrow \lambda = \omega^2$.
- Therefore the angular frequency is $\omega = \sqrt{\lambda}$ (rad/s) .
- Convert to ordinary frequency (cycles per second): $f = \frac{\omega}{2\pi} = \frac{\sqrt{\lambda}}{2\pi}$ (Hz) .

Linear modes & modal projection:

- **Linear mode**: eigen-vector ψ of the generalized problem $K\psi = \lambda M\psi$ (small-deformation assumption).
- Each mode oscillates independently at angular frequency $\omega = \sqrt{\lambda}$.
- **Why project**: convert $M\ddot{u} + D\dot{u} + Ku = f_{\text{ext}}$ into k decoupled damped springs in modal coordinates $q(t)$, keeping only the first $k \approx 20$ low-frequency modes for real-time speed.
- Saves computation, keeps audible quality, and avoids numerical coupling.
- Precomputed Acoustic Transfer: Output-Sensitive, Accurate Sound Generation for Geometrically Complex Vibration Sources – James & Barbič & Pai, SIGGRAPH 2006

18. Neural Fields

- Definition: A field is a quantity defined for all spatial and / or temporal coordinates.
- Examples of Fields: Audio, 3D Signed Distance Fields (Implicit Surface), 3D Parabola (Explicit Surface), Image, Vector Field
- Implicit Surfaces: $f(x,y,z) = d = \sqrt{x^2+y^2+z^2} - 1$
- Explicit Surfaces: Polygon Mesh with triangles
- Boolean Operations: min(union), max(intersection)

Representation:

- Continuous: smooth, compact, complex
- Discrete: Simple, bulky, fast
- Neural: learnable $f_\theta(x,y,z) \to d$; compact & continuous, evaluate at any point.

Neural Network - Multi-layer Perceptrons:

- $f_\theta(x,y,z) = d$
- Parametric map $f_\theta : \mathbb{R}^3 \to \mathbb{R}^m$ (scalar field $m=1$, vector field $m>1$)
- for each edge, it has its own weights
- number of layers are uncertian → hidden layer
- number of output depends on vector field or scalar field
- Trained with Adam Optimizer.

Why Neural Fields?

- 1. Compactness: weights $\ll$ dense voxel grid.
- 2. Regularization: smoothness is built in through network continuity.
- 3. Domain Agnostic: same architecture fits audio, images, SDFs, etc.

Example system

- Neural Geometric Level of Detail (CVPR 2021): Sparse Grids + CUDA + NNs = Real-Time Neural Field Rendering
- Query point → Octree feature volume → Voxel feature retrieval → Trilinear interpolation → Summed features → Surface extractor → Predicted distance

Volume Rendering

Core idea

- Cast a viewing ray; sample density σ and color c along it.
- Accumulate color with *opacity* α (front to back or back to front).
- Result: a 2D pixel that shows the interior of the 3D data set.

Three volume rendering techniques

- Volume ray casting (CPU/GPU ray march)
- Splatting (project each voxel as a footprint onto the image)
- 3D texture mapping (slice the volume and let GPU blend)

Ray Casting

- Integrate emitted color and absorbed light through the volume.
- Use regular (x,y,z) grid when possible; use finite elements for irregular data (e.g. ultrasound).
- Can 3D rasterize geometric primitives to accelerate empty space skipping.

Accumulating Opacity

- $\alpha = 1$ is opaque, $\alpha = 0$ is fully transparent
- Composite multiple layers according to opacity
- Use local gradient of opacity for enhanced display of boundaries
- $C(i)_{out} = C(i)_{in} * (1 - \alpha(i)) + C(i) * \alpha(i)$
- Early ray termination when accumulated α nears 1

Transfer Functions

- Transform scalar data values to RGBA values
- Apply to every voxel in volume
- Highly application dependent
- Start from data histogram
- Opacity for emphasis

Neural volume rendering (NeRF style)

- MLP $f_\theta(x,v) \to (\sigma, c)$ gives density σ and view dependent color c, where x is spatial point and v is view(ray) direction
- Plug σ and c into the volume rendering integral (differentiable).
- Train with Adam from multi view photos; network becomes a compact, continuous 3 D scene.
- Example: **NeRF**, **NGLOD**, etc.

19. CUDA

What is GPGPU?

- General-Purpose computing on a Graphics Processing Unit
- Using graphic hardware for non-graphic computations

What is CUDA?

- Compute Unified Device Architecture
- Parallel computing platform and API by Nvidia
- Compute Unified Device Architecture
- Software architecture for managing data-parallel programming
- Introduced in 2007; still actively updated
- Con: collision detection is slow because of branching

Motivation

- GPU explode when image size increase

CPU vs. GPU

- CPU: Fast caches, Branching adaptability, High performance
- GPU: Multiple ALUs, Fast onboard memory, High throughput on parallel tasks (Executes program on each fragment/vertex)
- CPUs are great for task parallelism
- GPUs are great for data parallelism
- Hardware - GPU has more transistors devoted to data processing

GPU Memory Architecture

- Uncached:
- Registers (fastest)
- Shared Memory
- Local Memory
- Global Memory (fourth fastest)
- Cached: Constant Memory; Texture Memory

Software Requirements / Tools

- CUDA device driver (download required)
- CUDA Toolkit (compiler, CUBLAS, CUFFT)
- nvcc (NVIDIA's CUDA Compiler) compiles .cu and g++ compile .c .cpp files and connect
- CUDA Software Development Kit (Emulator)

To compute, we need to:

- Allocate memory for the computation on the GPU (incl. variables)
- Provide input data
- Specify the computation to be performed
- Read the results from the GPU(output)
- While CUDA computing on GPU, CPU will not wait by default, eg. running user interface, mouse clicking, or any other program

Process

- Allocate Memory in the GPU card
- Copy content from the host's memory to the GPU card memory
- Execute code on the GPU
- Copy results back to the host memory

The Kernel

Grid and Block Size block = 3 warp warp = 32 threads multiple-processor has limited threads can perform

20. Computer Animation Middleware Software

Game Engines

- Unreal Engines (Epic Games)
- Unity (Unity Technologies)
- Source, Source2 (Value)
- CryEngine (Crytek)
- AnvilNext (Ubisoft)
- Frostbite (Electronic Arts)
- (not an exhaustive list)

Character Animation Middleware

- NaturalMotion (real-time motion control using biomechanics) (acquired by Zynga for $527M in 2024)
- IKinema (full-body IK solver)

Physics in games

- Custom, in-house software
- Off-the shelf libraries
- Physics middleware

Physics Engines

- Real-time: video games
- High precision: slow, file, scientific computing

Real-time physics engines: open source

- Open Dynamics Engine (ODE)
- Bullet
- SOFA
- Vega FEM
- and several others

Real-time physics engines: commercial

- Havok (Ireland) (Intel $\rightarrow$ now Microsoft)
- Physx (formerly NovodeX, now nVidia)
- Vortex (Montreal)
- Rubikon (Valve)

Components of physics engine

- Collision detection
- Dynamics: rigid objects, cloth, fluids
- Fracture

Rigid object contact

- Penalty-based: popular with soft/deformable objects
- Impulse-based
- Constraint-based: expensive, suitable for continuous contact

Real-time simulation

- Speed more important than accuracy
- Objects have two representations: Complex geometry (rendering); Simplified geometry (collision detection, dynamics)

Other Topics

Digital Twin

- Digital twin: a digital copy of a human/animal/buiding...
- Digital humans: input an image and output a digital human
- Apic games - Metahuman: have a library of data and artist create a digital human based on those
- pro: easier to communicate in long distance
- con: security, law

Metaverse 元宇宙

- differ from Multiverse 多元宇宙
- Interact with each other in VR
- "2D Social Media" → Metaverse: the next generation Facebook, Zoom, Apple Facetime, Microsoft Teams, Google Meet etc.
- Cybersecurity, Global Warming, Digital Twins
- Pro:
- Go on dates with long distance partners/spouses in VR
- Meet friends and family in VR
- Business meeting in VR → less airplane travel -¿ less CO2 emitted -¿ slow down global warming
- Doctors meeting patients in far away places in VR
- Shop in VR, attend concerts, school in VR, etc.
- Kardashev Type 1 civilization "world communication system": 0: Nothing, 0.5: We are here, 1: Control the planet, 2: Control the solar system, 3: Control the galaxy

License

- MIT, BSD: similar to PDL, except they cannot use you to advertices (prefered)
- LGPL: library
- GPL: if take code from GPL, then have to license under GPL. (company do not want this)
- Public Domain Licence: everyone can take it and I don't care (abuse)

Important People

Lisa Su

- CEO of AMD
- Age: 55
- Country of birth: Taiwan
- Undergraduate, MS and PhD degrees at MIT
- Oct 2014 - Feb 2025: 76x AMD stock growth (57% per year!)
- "Run toward the hardest problems. This approach has helped me to learn a tremendous amount from both success and failure."

Edwin Catmull

- Co-Founder, Pixar
- Preident, Walt Disney Animation Studios
- Turning Awars, 2019
- Age: 78
- Country of birth: United States
- Undergrauate, PhD at Univ. of Utah (Physics and CS)
- Invented texture mapping, (co-invented) z-Buffering, Catmull-Rom spline
- "Failure isn't a necessary evil. In fact, it isn't evil at all. It is a necessary consequence of doing something new."
- "If you give a good idea to a mediocre team, they will screw it up. If you give a mediocre idea to a brilliant team, they will either fix it or throw it away and come up with something better."
- "You are not your idea, and if you identify too closely with your ideas, you will take offense when ehty are challenged."

Hayao Miyazaki 宫崎骏

- Co-Founder, Director, Studio Ghibli: My Neighbor Totoro, Proco Posso, Spirited Away, ...
- Animator, film-maker, manga artist
- "Godfather of animation in Japan"
- Super famous. Numerous awards for his films
- Country of birth: Japan. Age: 84
- "I strongly feel that AI is an insult to life itself."
- "Do everything by hand, even when using the computer."
- "I would like to make a film to tell children: 'It's good to be alive.' "
- "Violence is innate in humans. The real issue, is how to control it."
- "No cuts." Famously objected to Harvey Weinstein cutting his films.
- "I think 2D animation disappeared from Disney because they made so many uninteresting films. It's too bad. I thought 2D and 3D could coexist happily."

Crowd Animation

Boids: an artificial life simulation developed by Craig Reynolds

Constrained Animation of Flocks: Matt Andersion

History

- Particle systems
- Flocking systems
- Behavioral systems

Particle Systems

- Tens of Thousands of members
- No intelligence
- Respond to global forces, including collision reaction

Flocking Systems

- Thousands of memebrs
- Limited intelligence
- Some physical basis
- Collision avoidance
- Local control system

Flocks, Herds, and Schools: A Distributed Behavioral Model

- "Boids" given simple behavioral constraints: Separation; Alignment; Cohesion
- Often cited as one of the first examples of flocking and emergent behavior
- Craig Reynolds 1987

Behavioral Systems

- Tens to thousands of members
- Each member is highly intelligent
- Collision avoidance
- Behavior based on rules

Applications of Crowd Models

- Computer animation; Games, interactive graphics, and virtual reality; Robotics; Aerospace; Education; Artificial life; Art; Biology; Physics...

Boids Algorithm

Simulated Flocks

- Create a boid model that supports geometric flight
- Add individual behaviors that oppose each other: Separation; Alignment; Cohesion; Obstacle Avoidance
- The boids must be able to arbitrate conflicting behaviors as well.
- r: weights (how far away)
- c: constant
- a: acceleration
- v: relative velocity of i

Simulated Flocks: Separation $a_{sep} = c_{sep} \cdot \sum_i (1/r_i^2) n_i$

- Avoid crowding local boids

Simulated FLocks: Alignment $a_{align} = c_{align} \cdot (\sum_i (1/r_i^2) v_i) \cdot 1/(\sum_i (1/r_i^2))$

- Steer toward the general heading of the rest of the flock

Simulated Flocks: Cohesion $a_{coh} = c_{coh} \cdot (\sum_i (1/r_i^2) x_i) \cdot 1/(\sum_i (1/r_i^2))$

- Move toward the average position of local flockmates

Avoiding Obstacles

- Independently model the shape for rendering and collision avoidance
- Boids model uses a steer-to-avoid as opposed to force-field concept.

Putting it all together $a = a_{sep} + a_{coh} + a_{align} + a_{coll}$

- Simulated Flocking behavior that mimics real life flocks and herds: When combined with low priority goal seeking results in a scripted path of the flock

Conclusion

- Boids is a model of non-colliding motion of flocks based on simulating behavior of individual boids
- Boid model is an example of emergent behavior: Simple local rules lead to complex global behavior

Acceleration Quota

- priotize a_{coll} when nearly collide