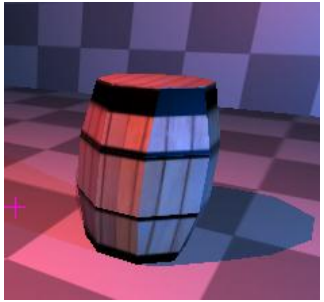
Interaction With 3D Geometry

Stan Melax
Graphics Software Engineer, Intel

GDC 13
GAME DEVELOPERS CONFERENCE
SAN FRANCISCO, CA
MARCH 25-29, 2013
EXPO DATES: MARCH 27-29
2013

3D Interaction Happens with Geometric Objects

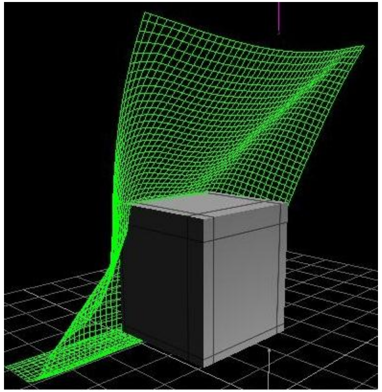


Rigid Body

```
{
  Mesh geometry;
  Vec3 position;
  Quat orientation;
};
```

Skinned Characters

```
{
  Mesh geometry;
  Vec3 position[];
  Quat orientation[];
};
```



Soft Body

```
{
  Mesh geometry;
  Springs connectivity;
};
```

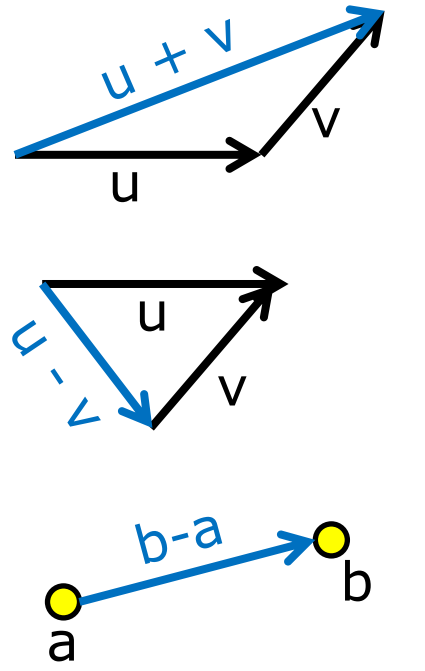
Agenda – Interacting with 3D Geometry

Practical topics among:

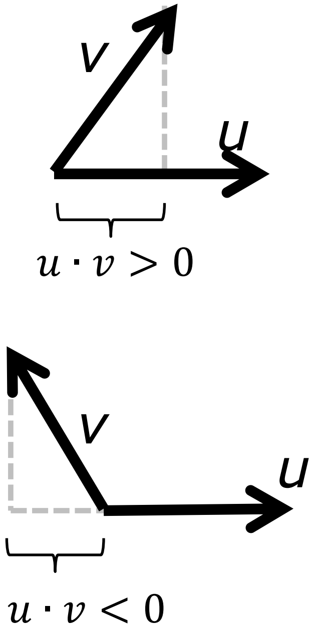
- Core Geometry Concepts
- Convex Polyhedra
- Spatial and Mass Properties
- Soft Geometry and Springs

With examples and implementation issues.

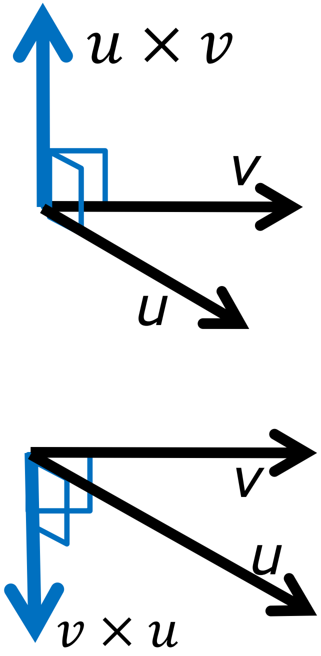
Warning: Some Math Ahead



Vector Arithmetic



Dot Product



Cross Product

$$Mv = \begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

$$uv^T = \begin{bmatrix} u_x v_x & u_x v_y & u_x v_z \\ u_y v_x & u_y v_y & u_y v_z \\ u_z v_x & u_z v_y & u_z v_z \end{bmatrix}$$

$$\frac{df}{dv} = \begin{bmatrix} \frac{\partial f_x}{\partial v_x} & \frac{\partial f_x}{\partial v_y} & \frac{\partial f_x}{\partial v_z} \\ \frac{\partial f_y}{\partial v_x} & \frac{\partial f_y}{\partial v_y} & \frac{\partial f_y}{\partial v_z} \\ \frac{\partial f_z}{\partial v_x} & \frac{\partial f_z}{\partial v_y} & \frac{\partial f_z}{\partial v_z} \end{bmatrix}$$

Matrix Stuff

What's an "outer product"?

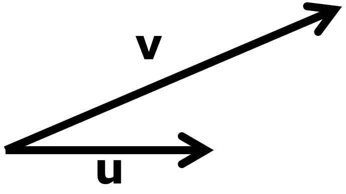
Familiar dot or inner product:

$$u \cdot v = u^T v = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = u_x v_x + u_y v_y + u_z v_z$$

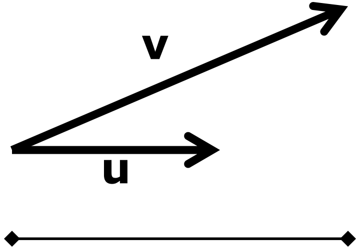
Outer product:

$$u \otimes v = u v^T = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \begin{bmatrix} v_x & v_y & v_z \end{bmatrix} = \begin{bmatrix} u_x v_x & u_x v_y & u_x v_z \\ u_y v_x & u_y v_y & u_y v_z \\ u_z v_x & u_z v_y & u_z v_z \end{bmatrix}$$

Outer Product - Geometric View

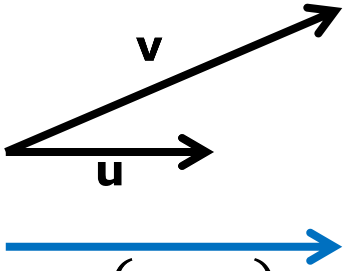


$\|u\| = 1$



$u \cdot v$

Distance of *v* along *u* (scalar)



$u(u \cdot v)$

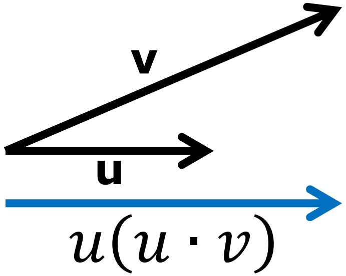
Projection of *v* along *u* (vector)

Outer product $\boldsymbol{u} \otimes \boldsymbol{u}$ of a unit vector $\boldsymbol{u}$ projects any vector along that direction.

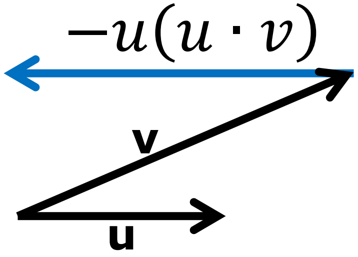
$$u(u \cdot v) \neq (u \cdot u)v$$

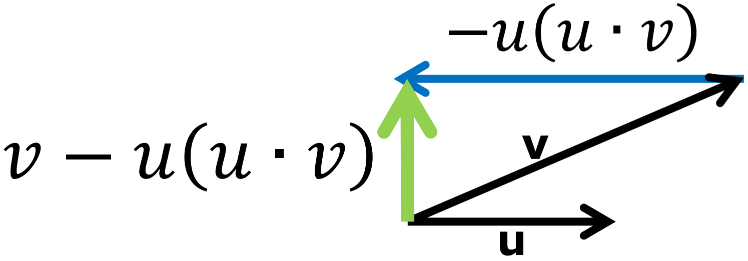
$$u(u \cdot v) = (u \otimes u)v$$

Outer Product for Plane Projection



or $(u \otimes u)v$



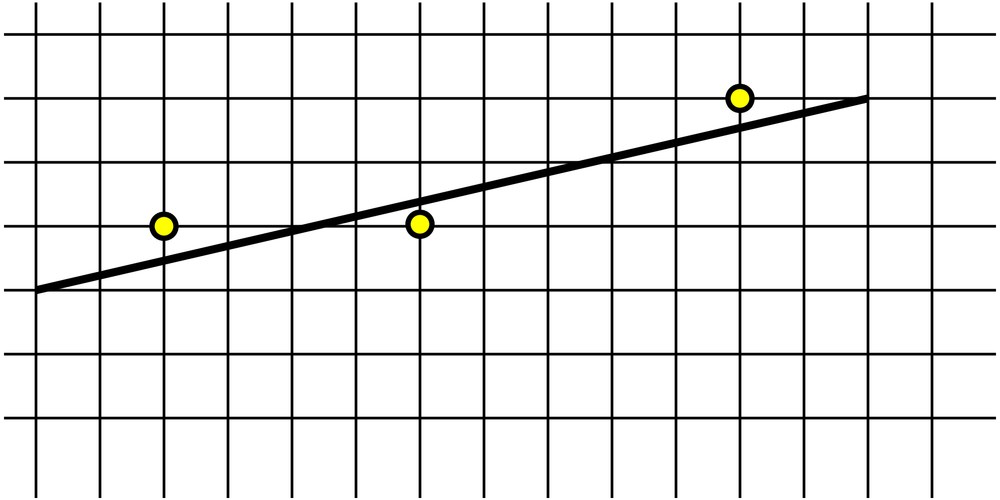


Remove portion of v that runs along u

$$v - u(u \cdot v) = Iv - (u \otimes u)v = (I - (u \otimes u))v$$

$\boldsymbol{I} - \boldsymbol{u} \otimes \boldsymbol{u}$ projects any vector onto plane with normal $\boldsymbol{u}$.

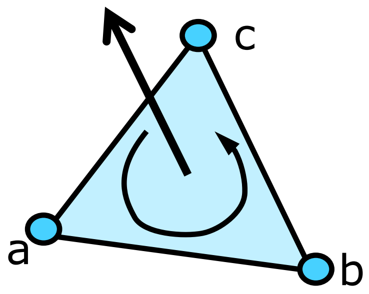
Numerical Precision Issues



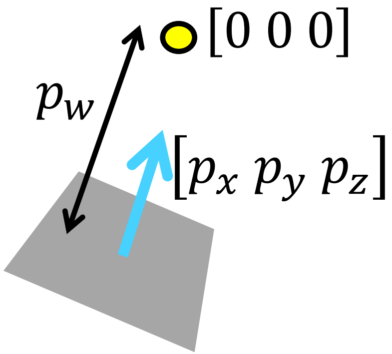
3 “collinear” points

Triangles and Planes

- Specify Front and Back



Triangle $(a\ b\ c)$
CCW winding



Plane $p = [p_x\ p_y\ p_z\ p_w]$

Could use:

$$Ax + By + Cz == D$$

Prefer convention:

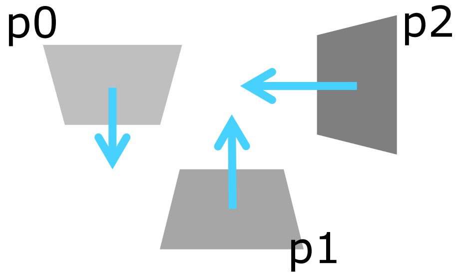
$$Ax + By + Cz + D == 0$$

Given a point $[xyz]$ its distance above plane is:

$$p_x\, x + p_y\, y + p_z\, z + p_w$$

- <0 below
- =0 on plane
- >0 above

Intersection of 3 planes



$$p0_x\, v_x + p0_y\, v_y + p0_z\, v_z + p0_w == 0$$

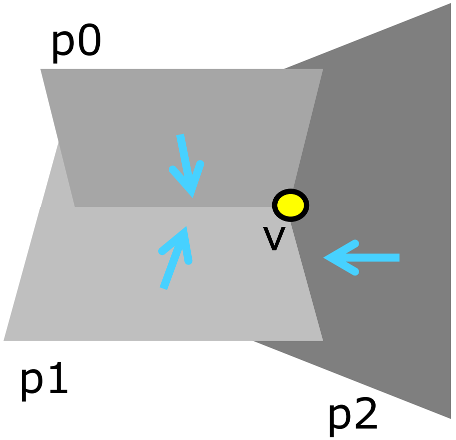
$$p1_x\, v_x + p1_y\, v_y + p1_z\, v_z + p1_w == 0$$

$$p2_x\, v_x + p2_y\, v_y + p2_z\, v_z + p2_w == 0$$

$$P = \begin{bmatrix} p0_x & p0_y & p0_z \\ p1_x & p1_y & p1_z \\ p2_x & p2_y & p2_z \end{bmatrix}, \qquad b = \begin{bmatrix} p0_w \\ p1_w \\ p2_w \end{bmatrix}, \qquad v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

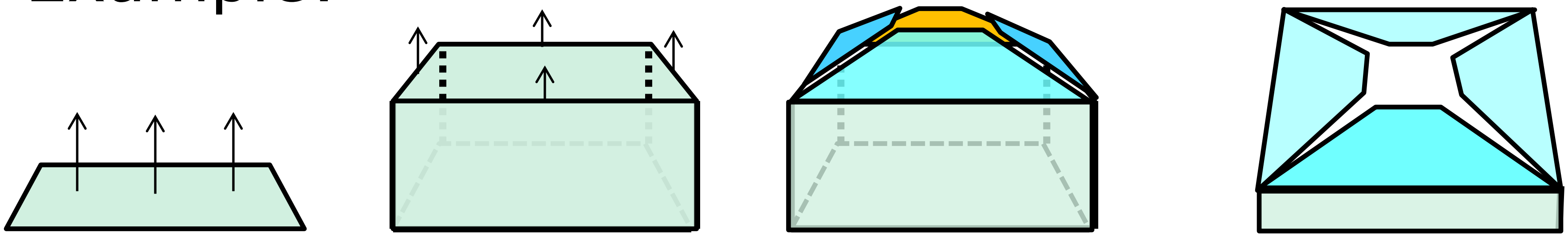
$$Pv = -b$$

$$\boxed{v = -P^{-1}b}$$



What if we have 4 planes?

Example:



Floor Only

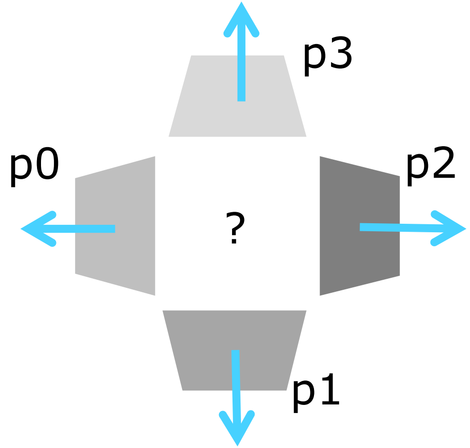
Walls

Roof Planes

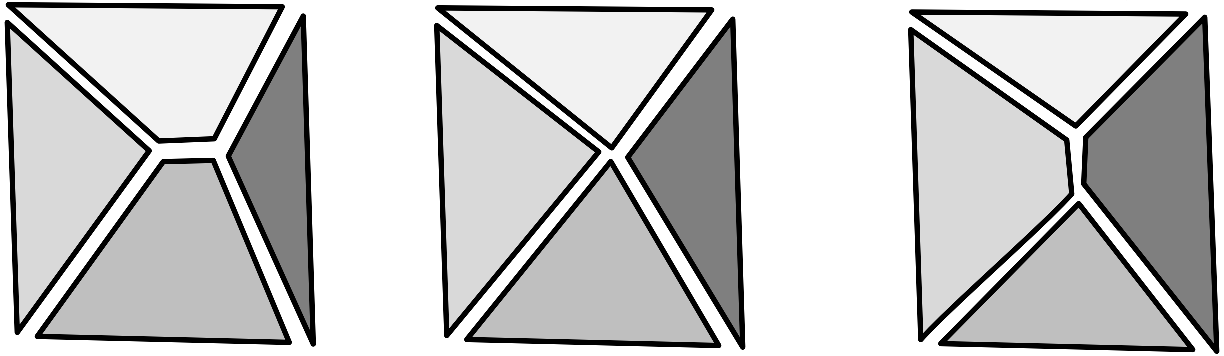
Roof Planes overhead view

As house grows bottom up, how will the 4 roof planes come together at the top of this house?

Determining How 4 Planes Meet



Note: these are top down views of a convex region

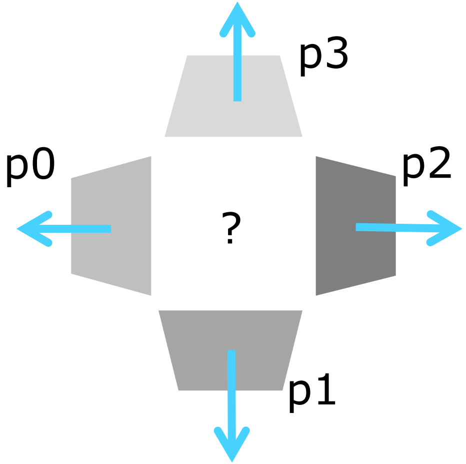


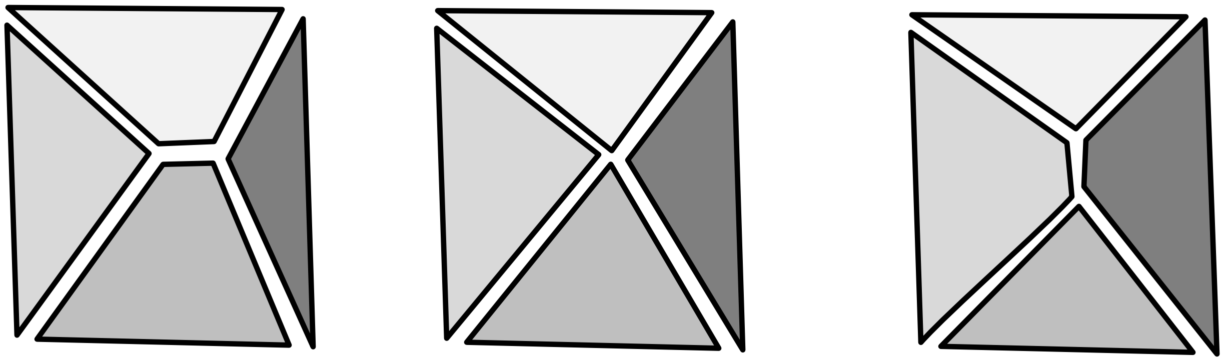
If 4 planes meet at a point xyz then we've found a solution to:

$$\begin{bmatrix} p0_x & p0_y & p0_z & p0_w \\ p1_x & p1_y & p1_z & p1_w \\ p2_x & p2_y & p2_z & p2_w \\ p3_x & p3_y & p3_z & p3_w \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Only possible if matrix singular.

Determin~~ing~~ant How 4 Planes Meet



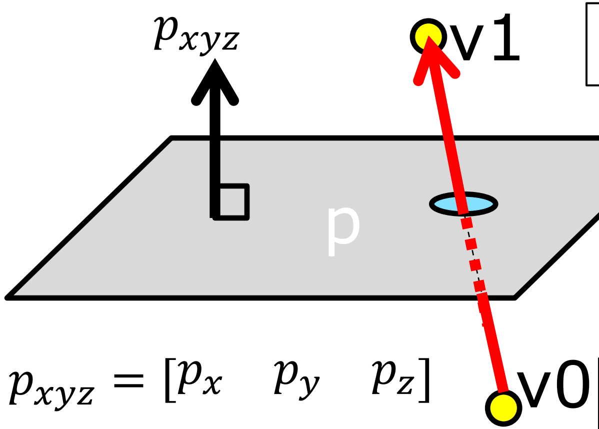


d<0 d==0 d>0

$$d = |p0 \quad p1 \quad p2 \quad p3| = \begin{vmatrix} p0_x & p1_x & p2_x & p3_x \\ p0_y & p1_y & p2_y & p3_y \\ p0_z & p1_z & p2_z & p3_z \\ p0_w & p1_w & p2_w & p3_w \end{vmatrix}$$

Notes: these are top down views of a convex region.
Planes well "behaved" all point same way.
Planes in CCW order.
Det of any 3x3 subblock from first 3 xyz rows is >0.

Intersect Line Plane

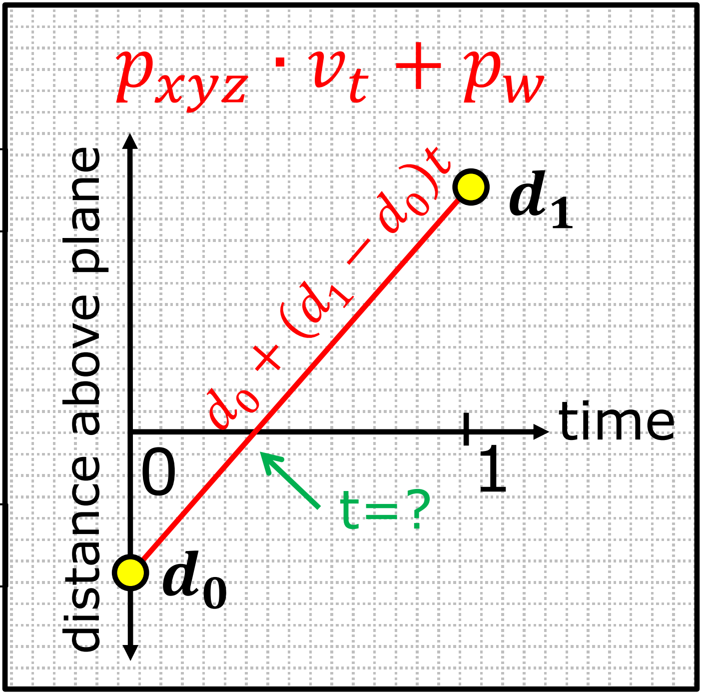


$$p_{xyz} = [p_x \quad p_y \quad p_z]$$

$$dist = p_w$$

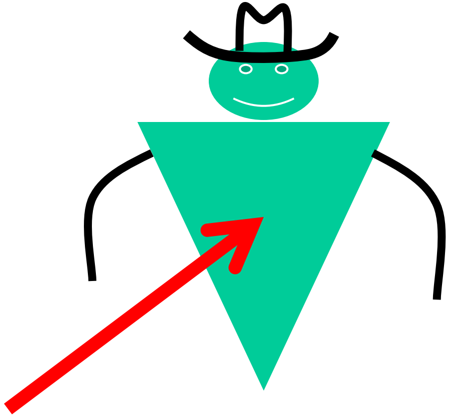
$$d_1 = p_{xyz} \cdot v_1 + p_w$$

$$d_0 = p_{xyz} \cdot v_0 + p_w$$

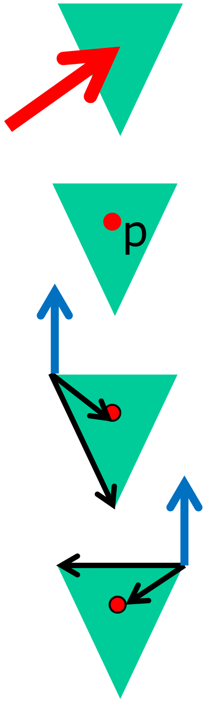


$$impact = v_0 + (v_1 - v_0)\, t \quad \Leftarrow \quad t = \frac{-d_0}{d_1 - d_0} = -\frac{p_{xyz} \cdot v_0 + p_w}{(p_{xyz} \cdot v_1 + p_w) - (p_{xyz} \cdot v_0 + p_w)}$$

Simple Ray Triangle Intersection Test

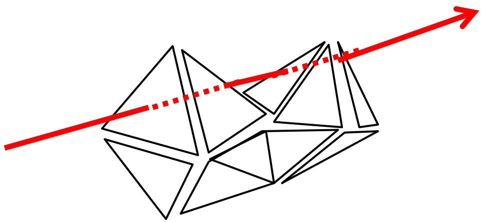


- Intersect ray with plane and get a point p.
- For each edge va to vb
  - Cross product with p-va
  - Dot result with tri normal n
  - <0 means outside
- Backside hit if dot(n,ray)>0

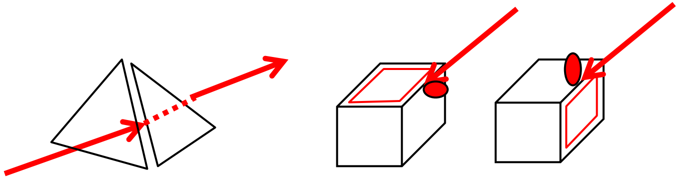


Ray Mesh Intersection

- Check Against Every Polygon
  - (spatial structures can rule out many quickly)
- Detecting a “hit” - might not be closest
- Numerical Robustness – don’t slip between two adjacent triangles.
- Mesh “intact” – t-intersections, holes caused by missing triangles.



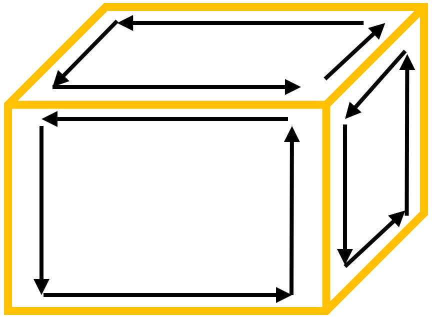
Multiple impact points – take closest.

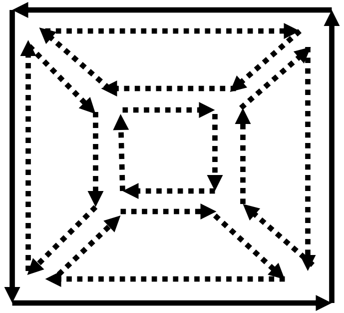


Numeric precision can result in plane intersection point landing just outside each time, thus missing both neighbors.

Solid Geometry

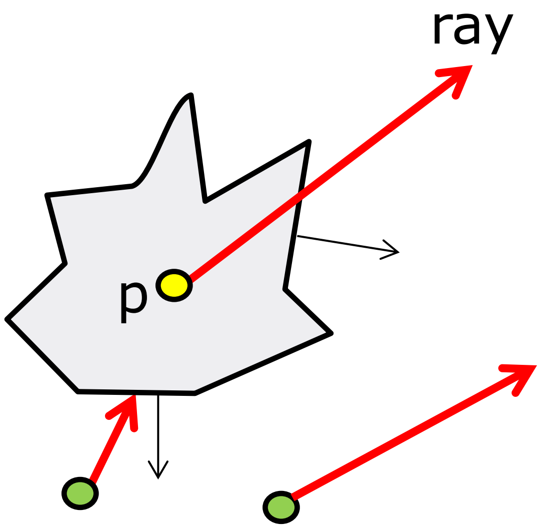
- "Water-Tight" <u>borderless</u> manifold mesh:
  - Every edge has 1 adjacency edge going other way
  - Consistent winding. Polygon normals all face to exterior.
- The mesh is the boundary representation - the infinitely thin surface that separates solid matter from empty space.





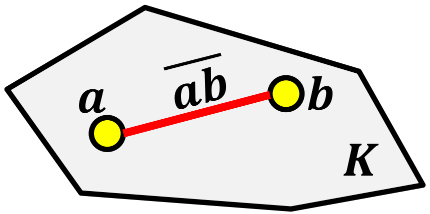
Inside/Outside Test of a point

- Cast a long ray from point
  - point is inside if it first hits the backside of a polygon.
  - point is outside the object if it first hits a front side or nothing at all.



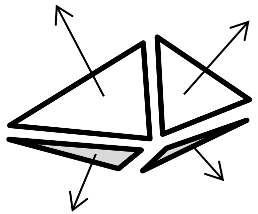
Convex Polyhedra

Convex Mesh

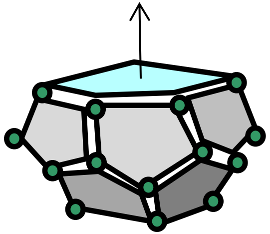


Math textbook:

$$K\ convex \leftrightarrow \forall a,b \in K \rightarrow \overline{ab} \subseteq K$$



- Neighboring face normals tilt away.
- Volume bounded by a number of planes.
- Every vertex lies at or below every face.

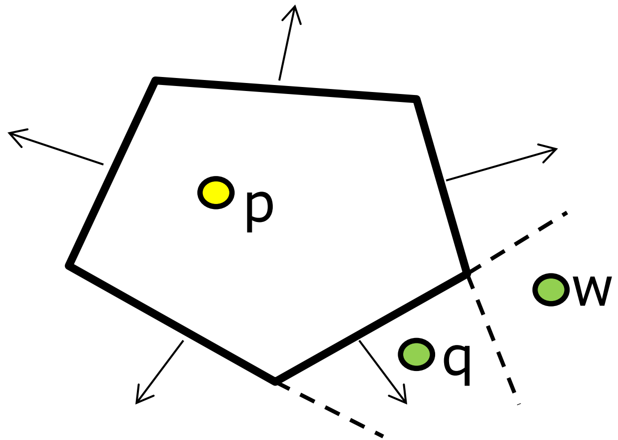


Many algorithms much easier when using convex shapes instead of general meshes.

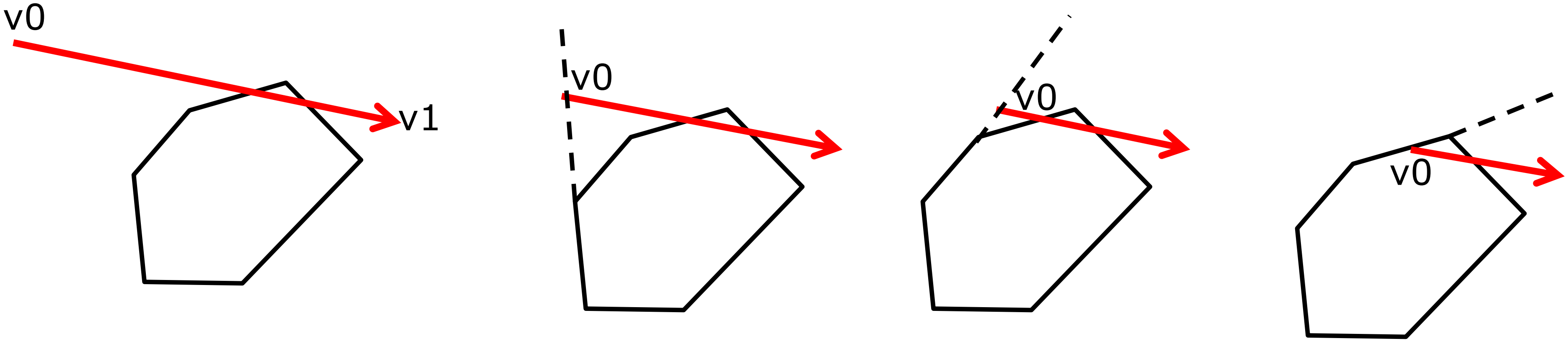
Convex In/Out test

A point is inside a convex volume if it lies under <u>every</u> plane boundary.

No testing with vertices or edges.

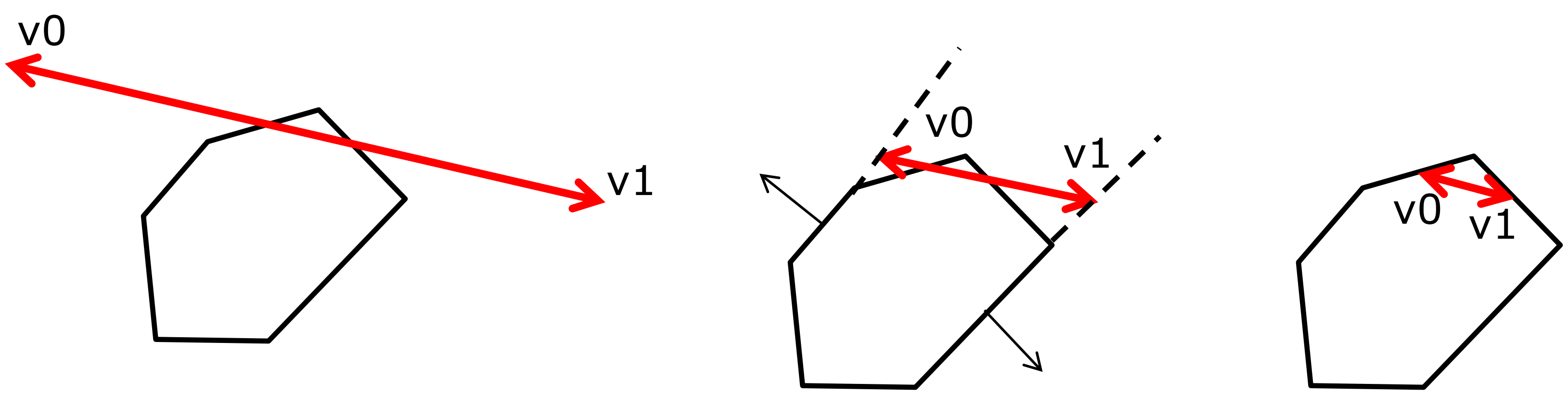


Convex Ray Intersection



- Crop Ray with all front facing planes: $n \cdot (v1 - v0) < 0$
- Impact at v0 if under all backside planes

Convex Line-Segment Intersection



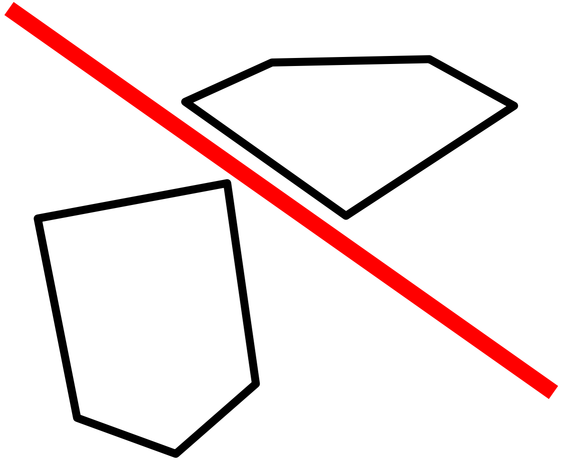
- Trim **v0** for **Front** facing planes $n \cdot (v1 - v0) < 0$
- Trim **v1** for **Back** facing planes $n \cdot (v1 - v0) > 0$

Convex-Convex Contact

- Separating Axis Theorem –

  Two non touching convex polyhedra are separated by a plane.

- The contact between two touching convex polyhedra will be either
  - A point
  - A line segment
  - A convex polygon



Physics engines like convex objects

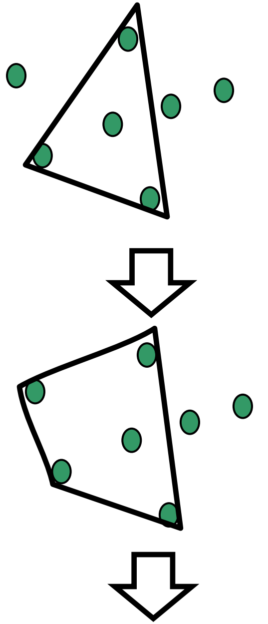
Convex Hull from points

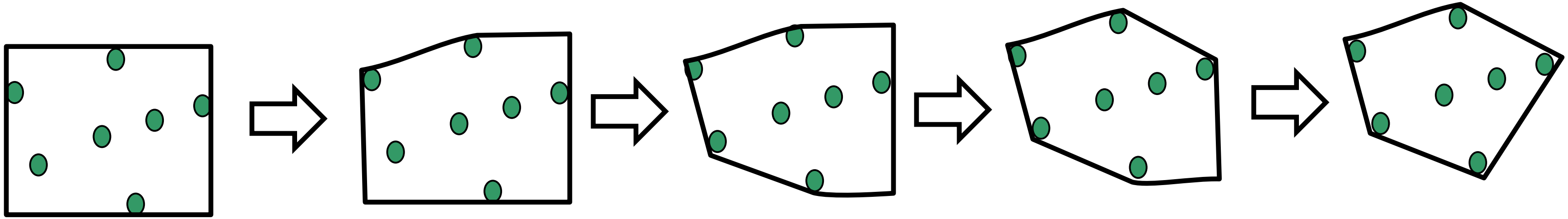
Convex Hull - smallest convex polyhedron that contains all the points.

- Convex hull of a mesh will contain the mesh.
- Often a sufficient proxy for interaction and collision

Techniques

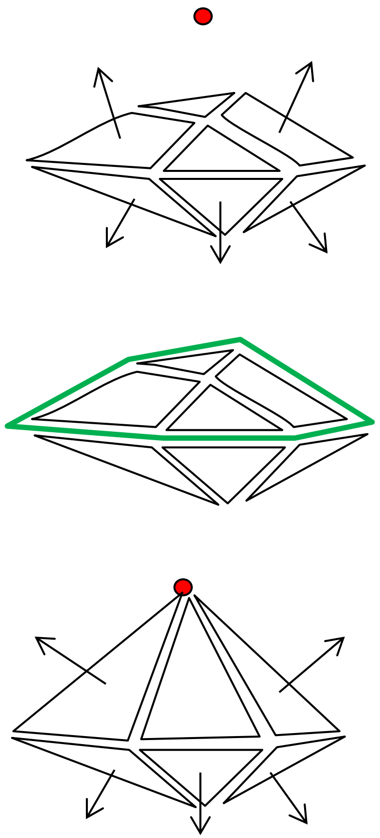
- Expanding outward: QuickHull [Eddy '77]
- Reducing inward: Gift Wrapping [Chand '70]



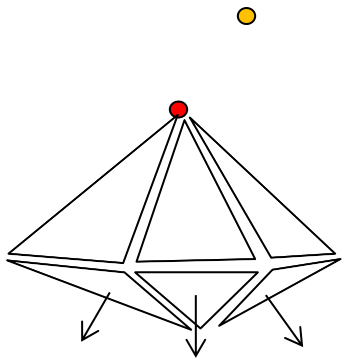


Compute 3D Convex Hull

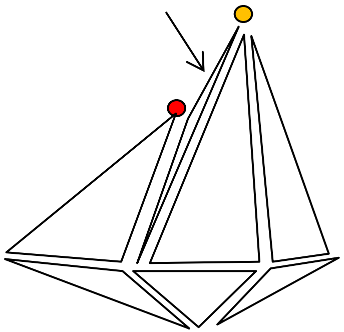
- Start with 2 back to back triangles. (or a tetrahedron)
- Find a vertex outside current volume (above faces).
  - Find edge loop silhouette (around all faces below point)
  - Replace with new fan of polys
  - Remove Folds (if any)
- Rinse and Repeat



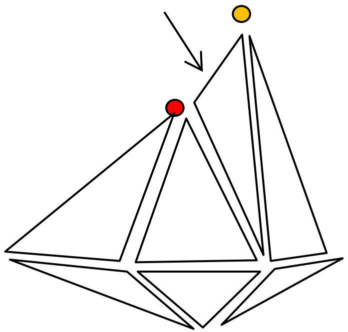
Hull Numerical Robustness



Grow Hull



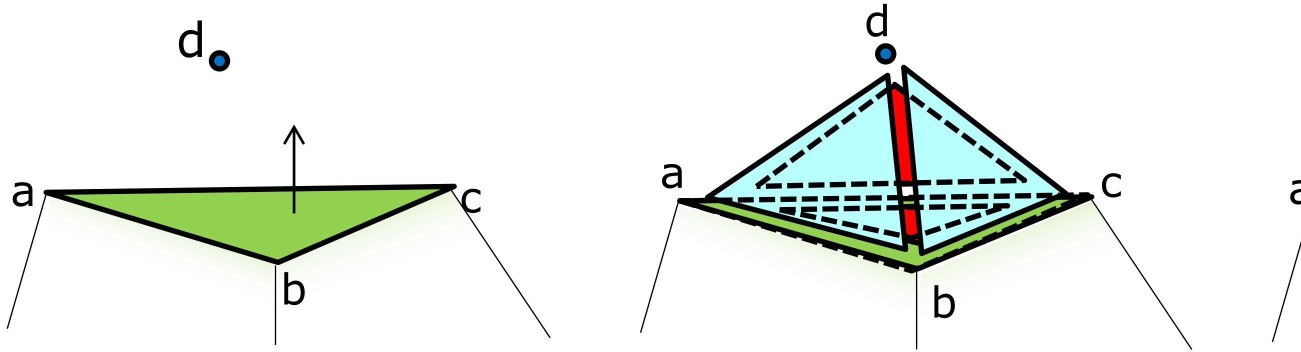
Generated skinny triangle with bad normal.



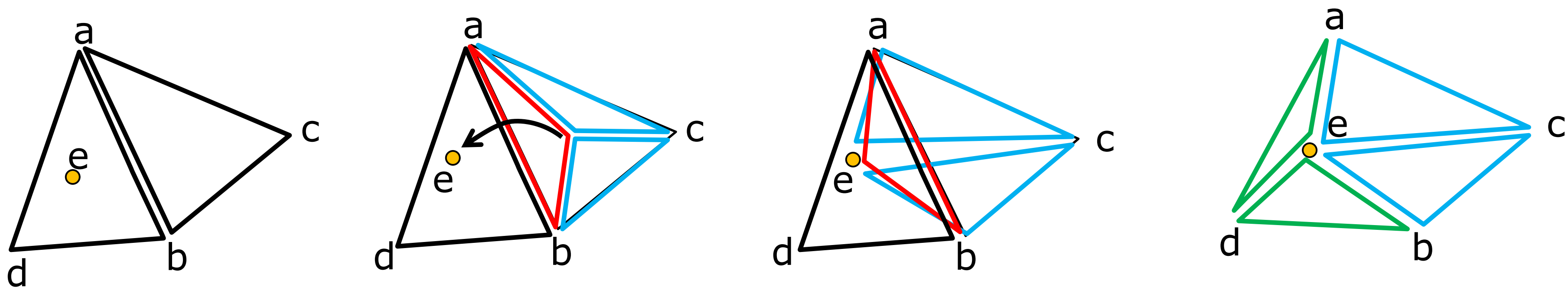
Flip edge to fix

Hull Tri-Tet Implementation

- Simple Triangle-mesh based approach
- When point *d* above any [*abc*] add tetrahedron [*abcd*] (triangles [*acb*][*dab*][*dbc*][*dca*]) to mesh.
- Prune any back-to-back tris such as [*abc*] and [*bac*]



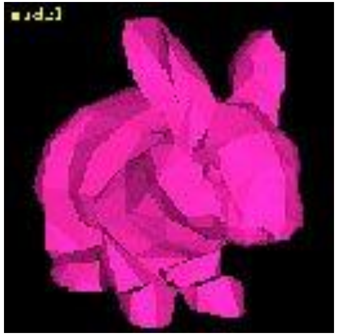
Hull Tri-Tet Numeric Robustness



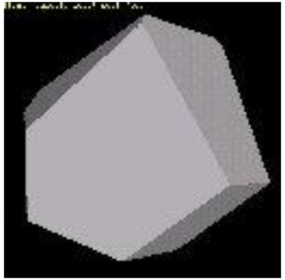
- 5 points {a,b,c,d,e} are coplanar-ish at one end of the point cloud
- But next point e tests <u>above</u> triangle [abc] but <u>below</u> [adb].
- Silhouette is {abc} for which we extrude a new tetrahedron up to e
- This produces triangles [eca],[ebc] (blue) facing the right way and [eab] (red) facing the wrong way – based on known interior point.
- This provides the hindsight to see that [adb] should also be extruded

Simplified Convex Hull

- Off-the-shelf solutions typically generate the complete hull.
- All hull vertices may not be needed and may be inefficient for usage.
- Instead use greedy incremental algorithm picking next vertex that increases hull size the most.
- Stop when vertex count or error threshold reached



Full Hull

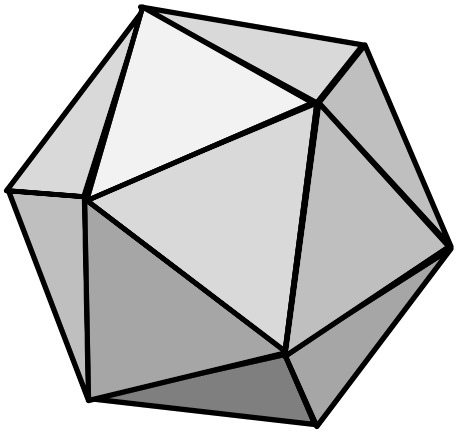


Simplified

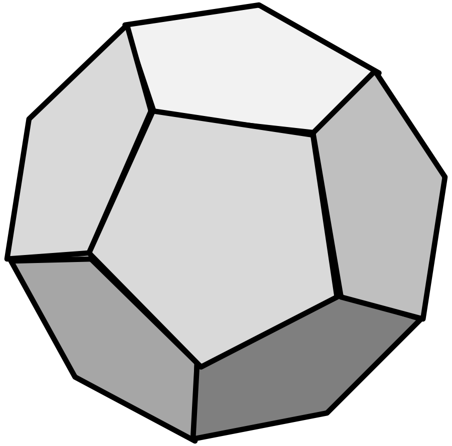
Minimize Number of Planes vs Points

Minimize Planes:

- Compute full hull
- Dual points $\left[\frac{p_x}{p_w}\ \frac{p_y}{p_w}\ \frac{p_z}{p_w}\right]$
- Compute simplified hull
- Take Dual

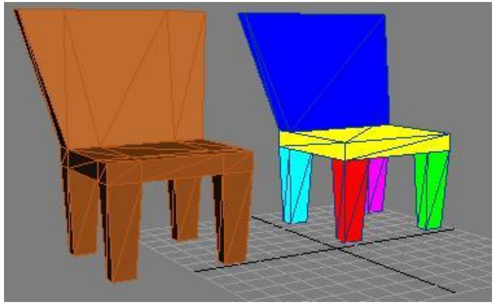


12 Vertices
20 Planes



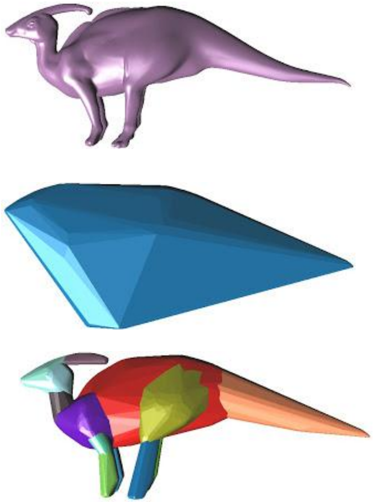
20 Vertices
12 Planes

Convex Decomposition



3DS Max Screenshot

Manual Creation of Collision Proxy



Automatic Mesh Decomposition [Ratcliff], [Mamou]

Use skinned mesh bone weights to create collision hulls

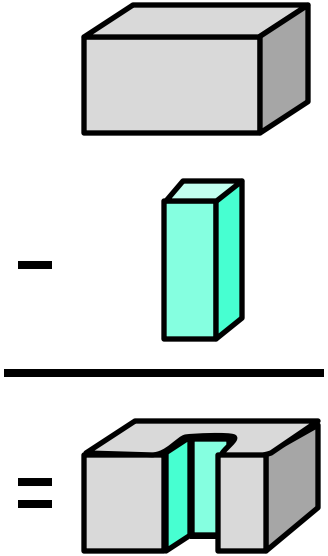
Example: Convex Bones For Collisions

Ragdolls on Stairs

Hand catching coins

Constructive Solid Geometry Boolean Operations

Solid Geometry Boolean Operations



Applications for Booleans

Art tools (already have CSG)

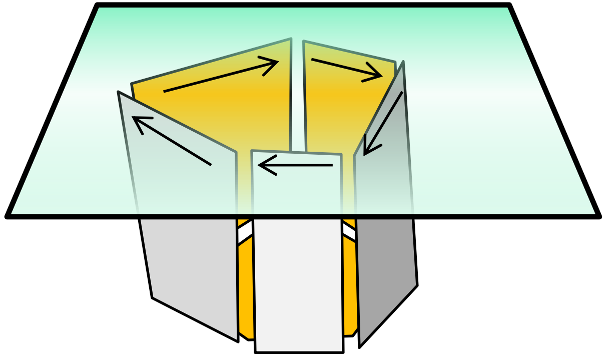
- Modeling and level design

Game Usages (less fidelity required):

- Destruction
- Geomod (tunneling)

A convex based approach may suffice.

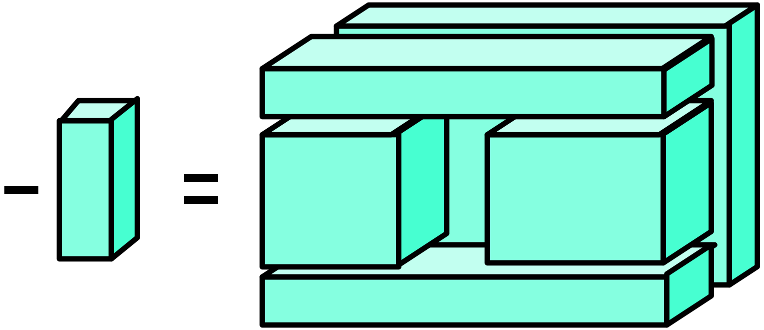
Convex Cropping and Intersection



- Like general mesh cropping, but only 1 open loop that is itself a convex polygon.

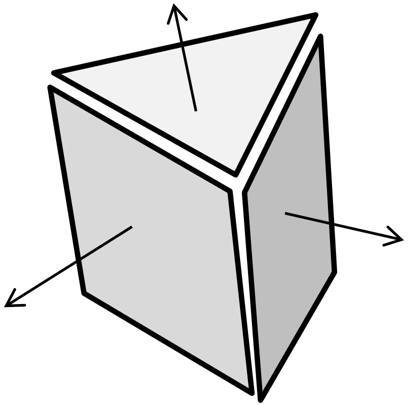
- Intersecting two convexes can be done by cropping one with all the planes of the other

- Non-convex operands can be implemented as a union

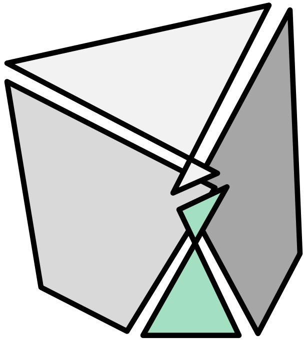


Convex Cropping Robustness

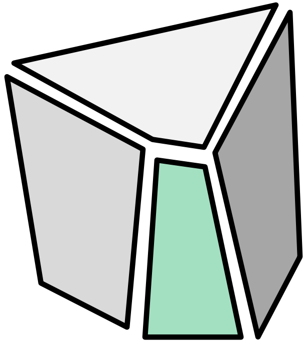
- Floating point issues can occur even in this most basic of mesh operations.
- One solution is to utilize plane equations to determine face/plane connectivity.



Pre Slice



Bad Slice



Correct Slice

Robustness with Quantum Planes

Let all planes p of CSG Boolean operands satisfy:

$$\forall p \exists s : sp = [s\,p_x \quad s\,p_y \quad s\,p_z \quad s\,p_w] \;, \qquad s\,p_{\{x,y,z,w\}} \in \mathbb{Z}, \qquad \left|sp_{\{x,y,z\}}\right| \le k$$

$$M = \begin{bmatrix} s_0 p0_x & s_0 p0_y & s_0 p0_z \\ s_1 p1_x & s_1 p1_y & s_1 p1_z \\ s_2 p2_x & s_2 p2_y & s_2 p2_z \end{bmatrix}$$

$$\boxed{v = -M^{-1} \begin{bmatrix} s0p0_w \\ s1p1_w \\ s2p2_w \end{bmatrix}}$$

Recall how vertices are the intersection of 3 or more planes

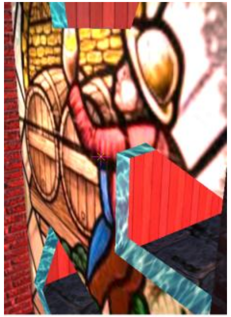
$$\boxed{\exists a, b:\; v_{\{x,y,z\}} = \frac{a}{b} \;, \qquad a, b \in \mathbb{Z} \;, \qquad |b| \le \det(M) \le 6k^3}$$

Generated points have Finite denominator

$$\boxed{\therefore\; \exists \varepsilon \forall u, v:\; \|u - v\| < \varepsilon \;\; \leftrightarrow \;\; u = v}$$

A fixed epsilon can now be used to indicate if two points are the same.

Destruction – geometry modification



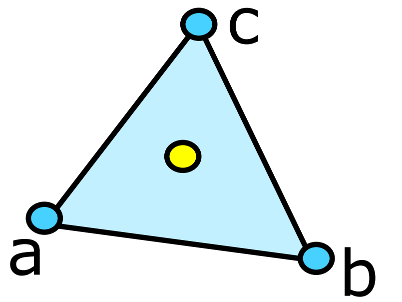
Texturing break

Objects in Motion
Spatial Properties

Spatial and Mass Properties

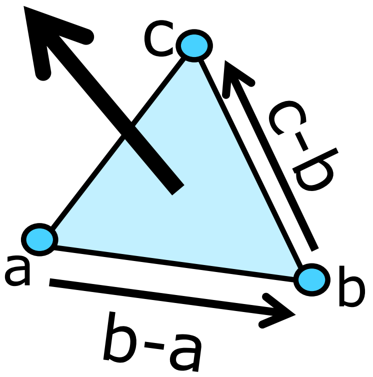
- With a solid mesh we can correctly derive properties that affect motion:
  - Volume
  - Center of mass
  - Covariance (and 3x3 Inertia Tensor)

Triangle Center and Area



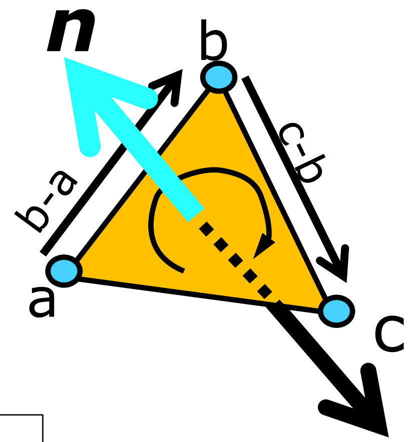
$$\text{center} = \frac{a + b + c}{3}$$

i.e. Center of mass.
There are other ways to define “center”.



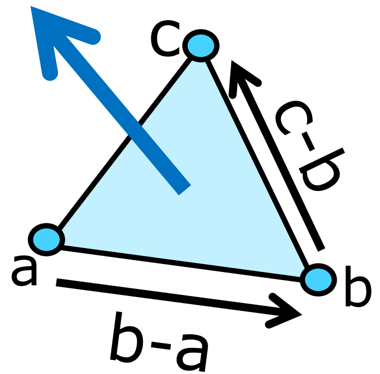
$$\text{abs area} = \frac{\|(b - a) \times (c - b)\|}{2}$$

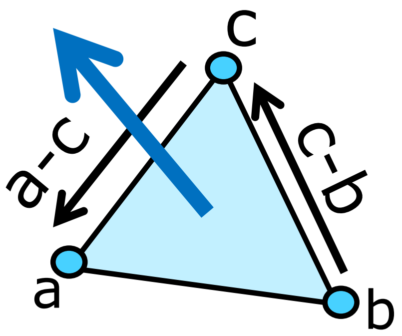
$$\text{area} = \frac{\left((b - a) \times (c - b)\right) \cdot n}{2}$$

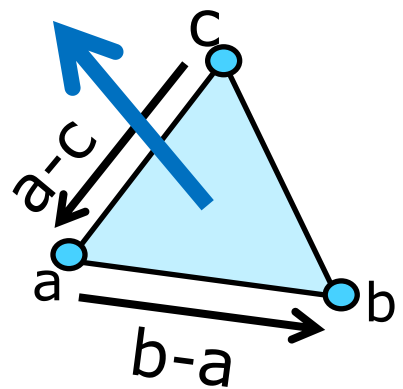


Given a pre set normal ***n***, result is signed.
So area could be negative.

Cross Product - Edge Choice Irrelevant







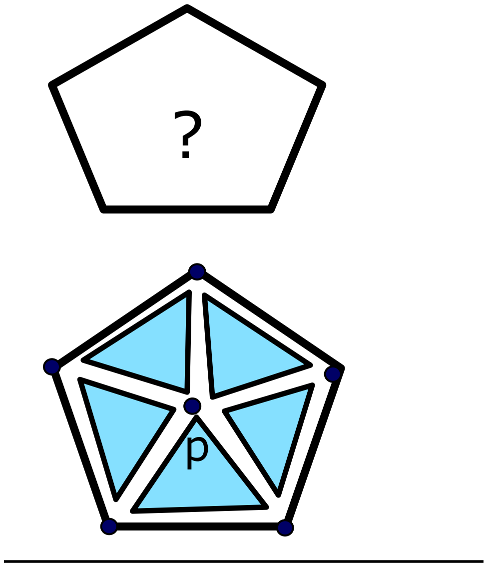
$$(b-a)\times(c-b) = (c-b)\times(a-c) = (a-c)\times(b-a)$$

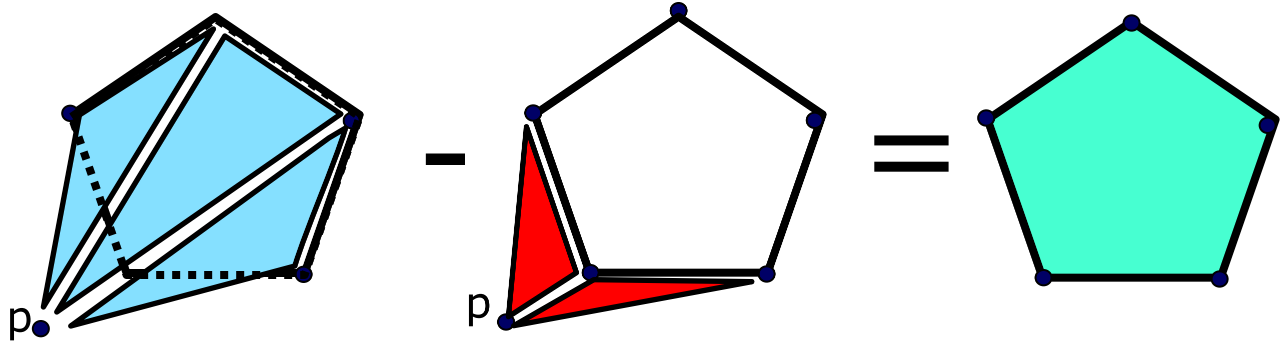
All the same

Area of Polygon (2D)

Triangle Summation:

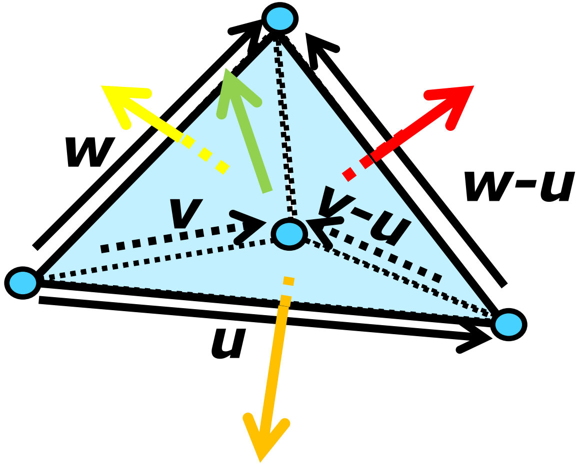
- Pick a reference point p. Normal *n* known.
- For each edge (a,b) sum area of triangle (p,a,b): $\sum \frac{1}{2}(a-p) \times (b-a) \cdot n$
- Signed Area - upside down triangles cancel out extra area if p outside.



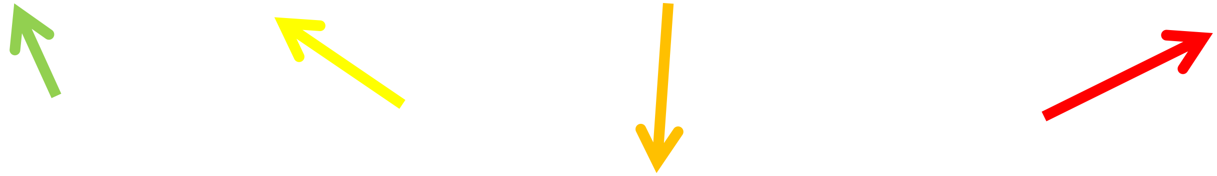


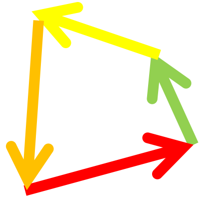
Solid's Area Weighted Normals Cancel

Sum of cross products for each tetrahedron face is zero.



$$u \times w \ + \ w \times v \ + \ v \times u \ + (v - u) \times (w - u) = 0$$

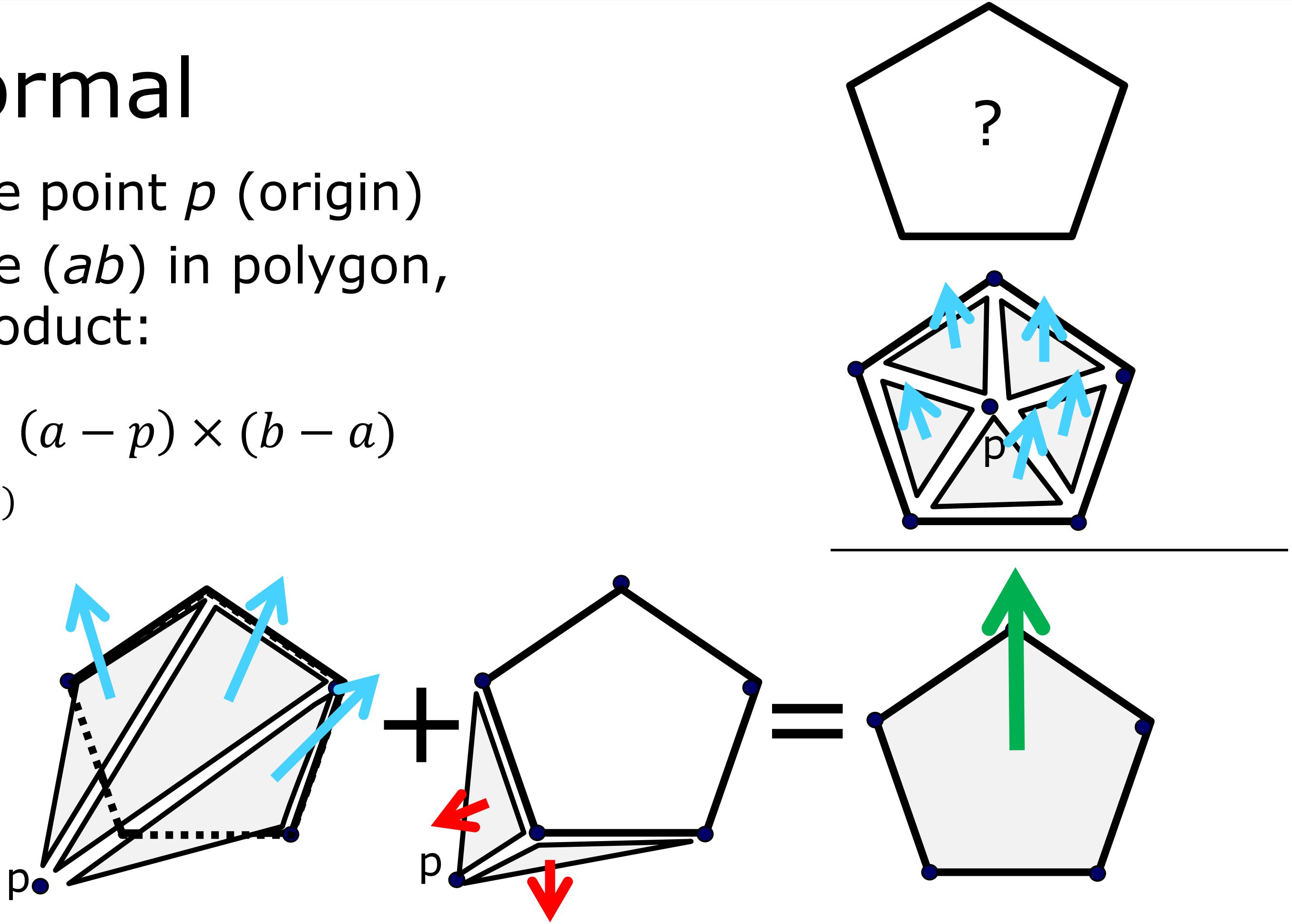




Polygon Normal

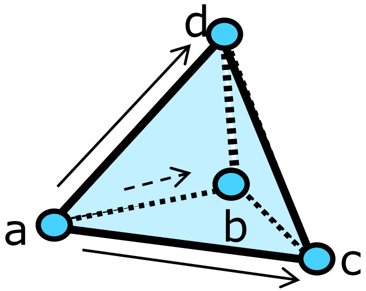
- Pick reference point *p* (origin)
- For each edge (*ab*) in polygon, sum cross product:

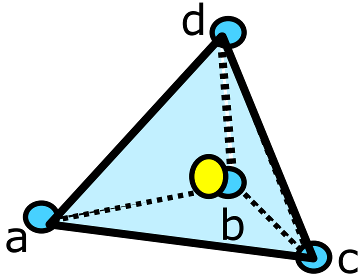
$$\sum_{Edge\ (a,b)} (a-p)\times(b-a)$$



For a trapezoid explanation search for "Newell Normal"

Tetrahedron Volume and Center





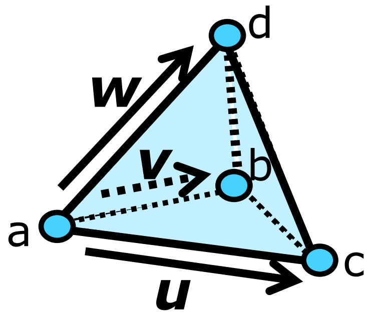
$$\text{volume} = \frac{(c-a)\times(b-a)\cdot(d-a)}{6}$$

$$\begin{matrix}\text{center} \\ \text{of mass}\end{matrix} = \frac{a+b+c+d}{4}$$

$$= \left|c-a \quad b-a \quad d-a\right| \div 6$$

Triple product in determinant form

Tetrahedron Integration



Edges: *u,v,w*
a=[0 0 0] (origin)

General Form:

$$\int_0^u \int_0^{v(1-\frac{\alpha}{u})} \int_0^{w(1-\frac{\alpha}{u}-\frac{\beta}{v})} f(\alpha+\beta+\gamma)\ \ d\gamma\, d\beta\, d\alpha$$

Letting f(X)=1, we get our volume:

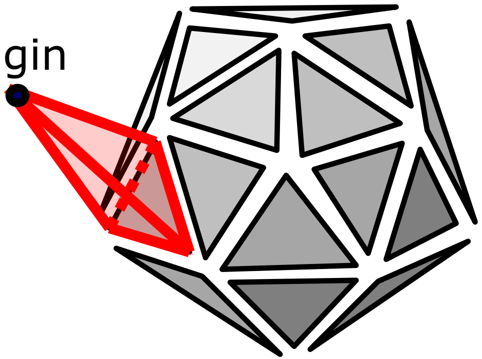
$$\int_0^u \int_0^{v(1-\frac{\alpha}{u})} \int_0^{w(1-\frac{\alpha}{u}-\frac{\beta}{v})} (1)\ \ d\gamma\, d\beta\, d\alpha \ = \ \frac{uvw}{6}$$

Letting f(X)=X/vol for center of mass (relative to vertex *a*):

$$\int_0^u \int_0^{v(1-\frac{\alpha}{u})} \int_0^{w(1-\frac{\alpha}{u}-\frac{\beta}{v})} \left(\frac{\alpha+\beta+\gamma}{uvw/6}\right)\ \ d\gamma\, d\beta\, d\alpha \ = \ \frac{u+v+w}{4}$$

Tetrahedral Summation (3D)

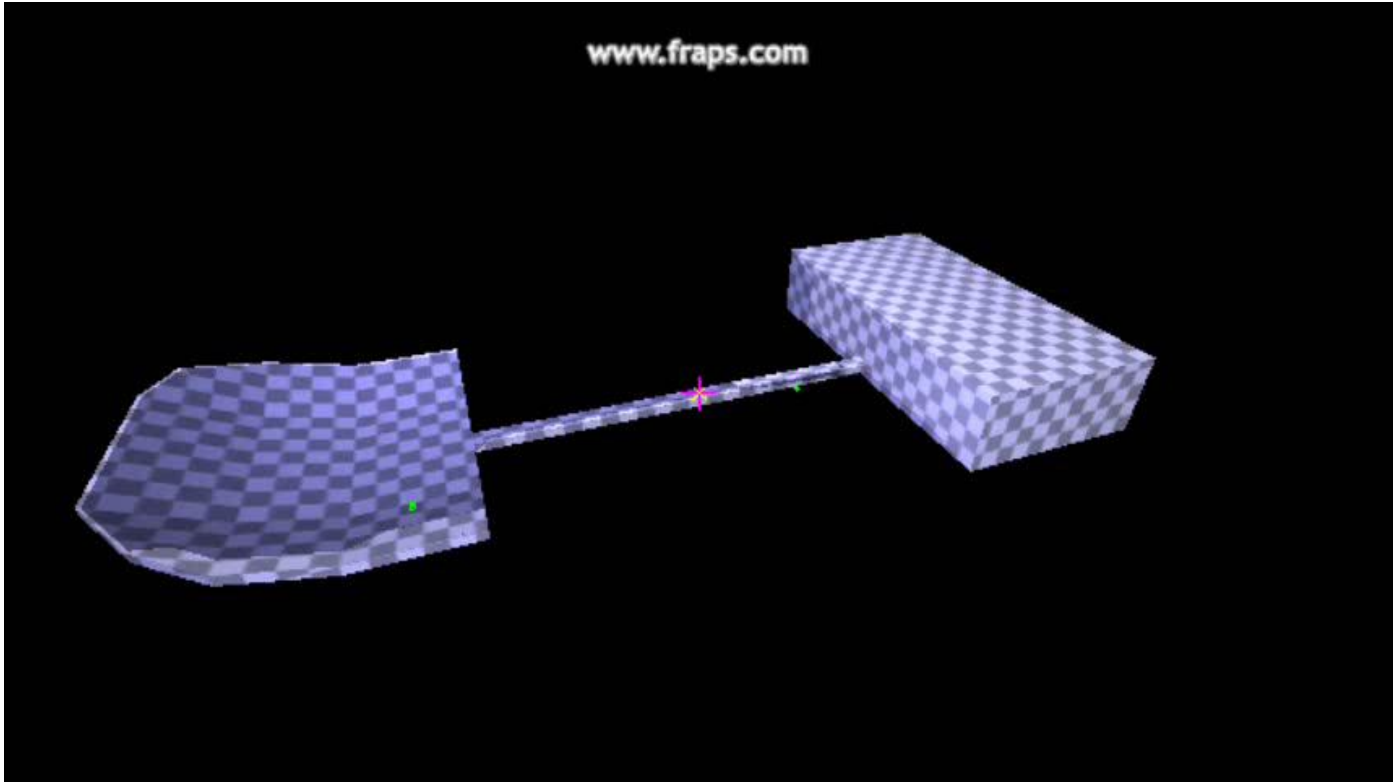
$$volume = \sum_{(u,v,w)\in mesh} |u\ v\ w|/6$$

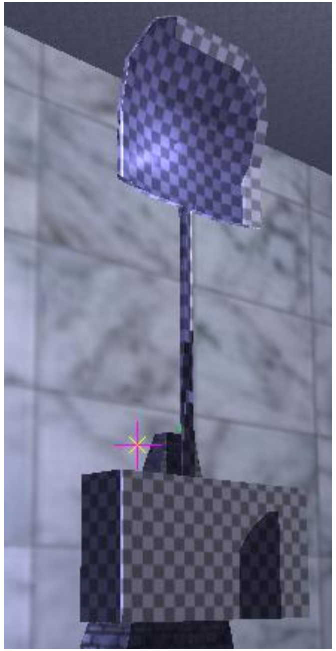


$$\begin{matrix} center \\ of\ mass \end{matrix} = \sum_{(u,v,w)\in mesh} \frac{(u+v+w)}{4}\frac{|u\ v\ w|}{6} / vol(mesh)$$

Center of Mass Affects Gameplay

Catapult geometry:

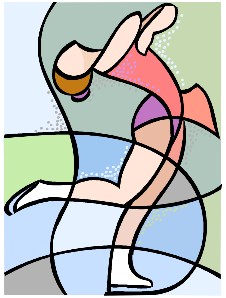




Inertia Calculation

2D:

$$\underset{inertia}{moment\ of} = \int r^2 = \int x^2 + y^2 = \underbrace{\int x^2}_{\text{Variance in } x} + \underbrace{\int y^2}_{\text{Variance in } y}$$



Fixed Axis

3D: 3x3 Inertia Tensor

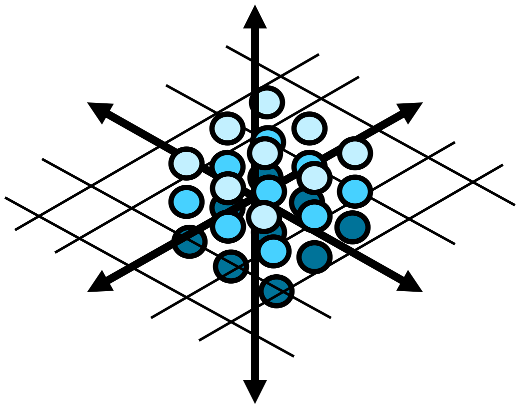
Related to covariance

$$\begin{bmatrix} yy + zz & -xy & -xz \\ -xy & xx + zz & -yz \\ -xz & -yz & xx + yy \end{bmatrix}$$

Covariance (origin = center of mass)

- If object was a collection of point masses *v*

$$\sum vv^T = \sum \begin{bmatrix} {v_x}^2 & v_x v_y & v_x v_z \\ v_y v_x & {v_y}^2 & v_y v_z \\ v_z v_x & v_z v_y & {v_z}^2 \end{bmatrix}$$



- Single Tetrahedron (0,u,v,w)…

$$\int_0^1 \int_0^{1-a} \int_0^{1-a-b} (au + bv + cw) \otimes (au + bv + cw)\ dc\ db\ da * vol\_adj$$

Tetrahedron (0,u,v,w) Covariance

u,v,w are vectors from center of mass to triangle on mesh

$$\int_0^1 \int_0^{1-a} \int_0^{1-a-b} (a\,u + b\,v + c\,w) \otimes (a\,u + b\,v + c\,w) \; dc \; db \; da$$

$$\int_0^1 \int_0^{1-a} \int_0^{1-a-b} \left(u^2\,a^2 + v^2\,b^2 + w^2\,c^2\right) + (\,a\,b\,u\,v + b\,c\,v\,w + a\,c\,w\,u\,) + (\,a\,b\,v\,u + a\,c\,u\,w + b\,c\,w\,v\,) \; dc \; db \; da$$

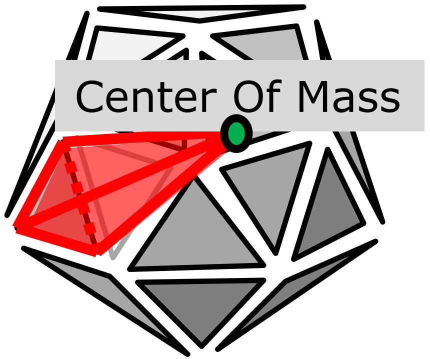
$$\int_0^1 \int_0^{1-a} \frac{1}{3}\,(a+b-1)\left(-3\,a^2\,u^2 - w^2\,(a+b-1)^2 - 3\,b^2\,v^2\right) + (a+b-1)\,(w\,(a+b-1)\,(a\,u+b\,v) - 2\,a\,b\,u\,v) \; db \; da$$

$$\int_0^1 \frac{1}{12}\,(a-1)^2\left(a^2\left(6\,u^2+v^2+w^2\right) - 2\,a\left(v^2+w^2\right) + v^2 + w^2\right) + -\frac{1}{12}\,(a-1)^3\,(a\,(4\,u\,(v+w) - v\,w) + v\,w) \; da$$

$$\boxed{\frac{1}{60}\left(uu + vv + ww\right) + \frac{1}{120}\,(\,uv + vw + wu\,) + \frac{1}{120}\,(\,vu + uw + wv\,)}$$

Inertia Tetrahedral Summation

$$intertia\!:T = \begin{bmatrix} yy+zz & -xy & -xz \\ -xy & xx+zz & -yz \\ -xz & -yz & xx+yy \end{bmatrix}$$

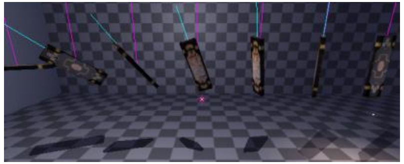


Sum for each triangle uvw,
Covariance matrix C:

$$c_{i,j} = \sum_{(u,v,w)\in mesh-com} \frac{2u_iu_j + 2v_iv_j + 2w_iw_j + u_iv_j + v_iw_j + w_iu_j + u_iw_j + v_iu_j + w_iv_j}{120} \frac{|u\ v\ w|}{6}$$

$$inertia\!:T = Identity\,(c_{0,0} + c_{1,1} + c_{2,2}) - C$$

Inertia Tensor and Object Motion



$$\omega = q\,T^{-1}\,q^{-1}\,h$$

$\omega \rightarrow spin$

$h \rightarrow momenum$

$q \rightarrow orientation$

$T \rightarrow Inertia$

Time Integration

Proof it's the same:

$$\lim_{(\|\omega\|\,dt)\to 0} s \to \left[\frac{\omega}{2}dt, 1\right]$$

$$s * q_t = [0001] * q_t + \left[\frac{\omega}{2}dt, 0\right] * q_t$$

$$s * q_t = q_t + \left[\frac{\omega}{2}, 0\right] * q_t\,dt$$

Updating state to the next time step.

- Position: $p_{t+dt} = p_t + velocity * dt$
- Orientation:

Build a Quat for Multiplication

$$s = \left[\frac{\omega}{\|\omega\|}\sin\left(\frac{\|\omega\|\,dt}{2}\right), \cos\left(\frac{\|\omega\|\,dt}{2}\right)\right]$$

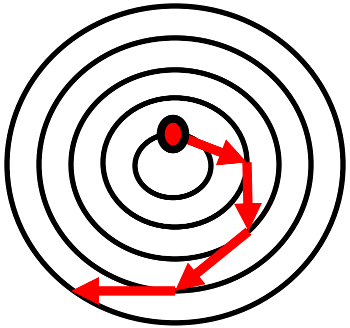
$$q_{t+dt} = s * q_t$$

Add Derivative

$$q_{t+dt} = q_t + \frac{dq}{dt}\,dt$$

$$q_{t+dt} = q_t + \frac{\omega}{2}q_t\,dt$$

Time Integration without Numerical Drift



$$k_1 = \frac{\omega(q_t)}{2} * q_t$$

$$k_2 = \frac{\omega(q_t + k_1 * dt/2)}{2} * (q_t + k_1 * \frac{dt}{2})$$

$$k_3 = \frac{\omega(q_t + k_2 * dt/2)}{2} * (q_t + k_2 * \frac{dt}{2})$$

$$k_4 = \frac{\omega(q_t + k_3 * dt)}{2} * (q_t + k_3 * dt)$$

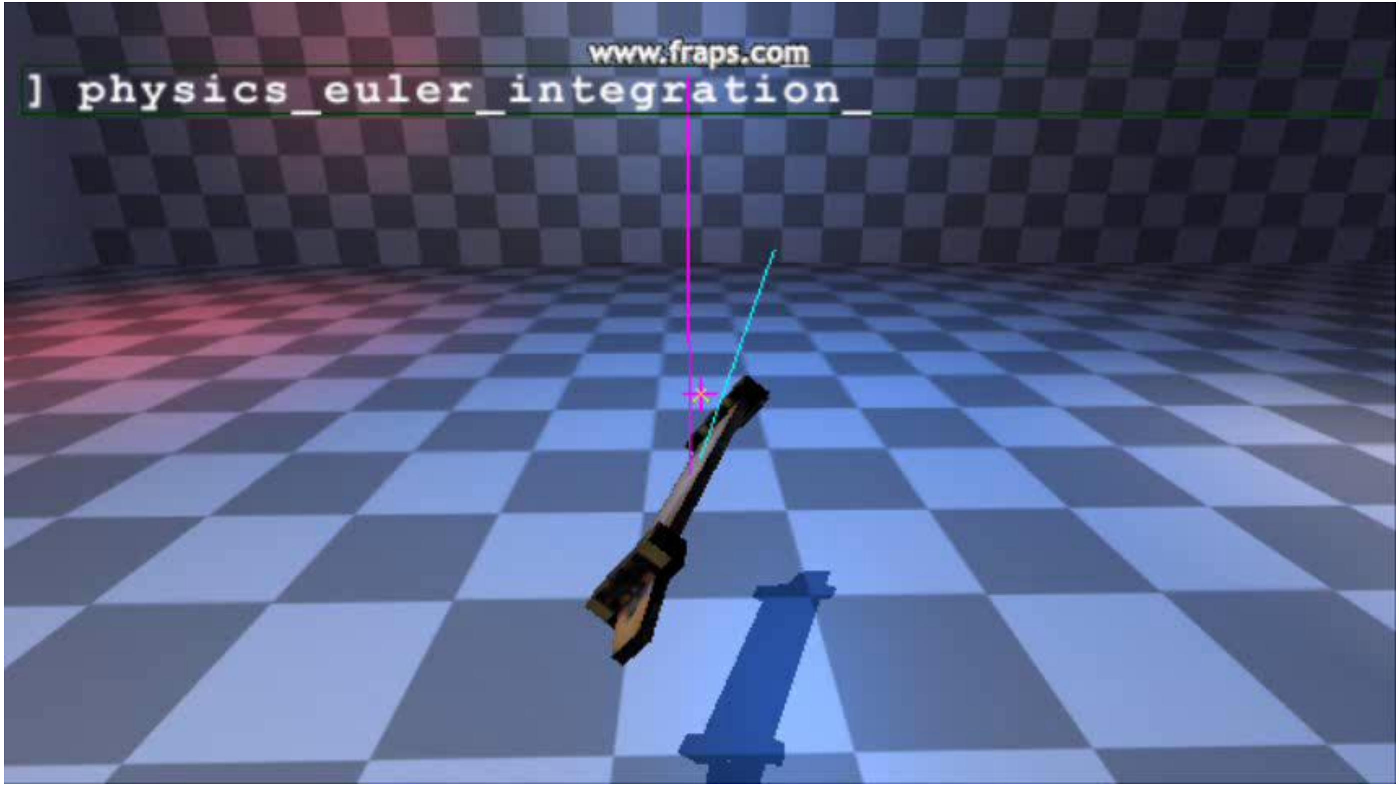
$$q_{t+dt} = q_t + \frac{\omega(q_t)}{2} * q_t * dt$$

Forward Euler

$$q_{t+dt} = q_t + k_1 * \frac{dt}{6} + k_2 * \frac{dt}{3} + k_3 * \frac{dt}{3} + k_4 * \frac{dt}{6}$$

Runge Kutta

Time Integration Euler vs RK4



Forward Euler:

- Spin drifts toward principle axis
- Energy gained

Runge Kutta

- Spin orbits as expected
- Energy stays constant

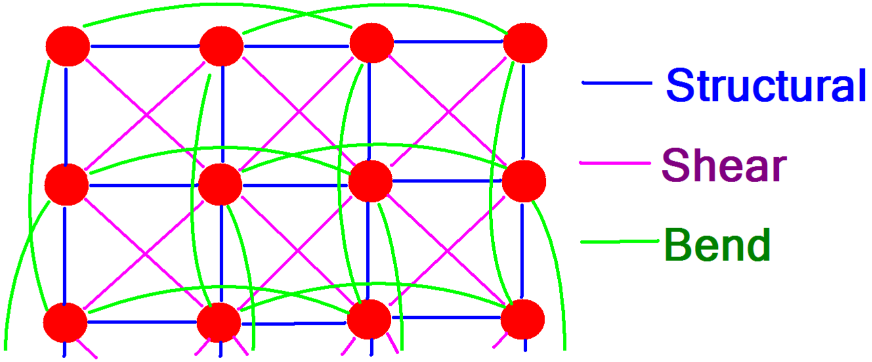
Soft Body Objects

Soft Body Meshes

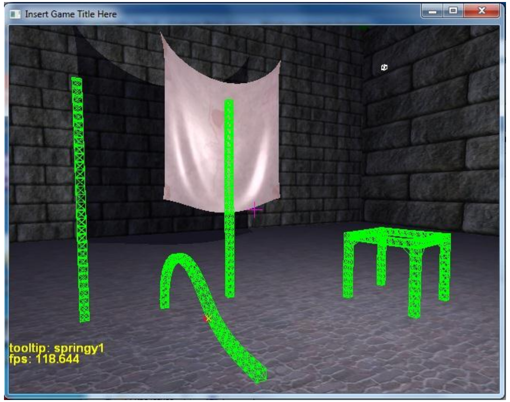
- Every vertex has its own position and velocity
- Vertices are connected via springs or constraints.

Object Construction

- Connection topology can differ from visual mesh
- Stiffness can vary to simulate different objects and materials



Cloth



Cube Lattice Table

Time Integration – Simulating Soft Body

Kinematic

- Connections as constraints
- Iterative position projection
- Easy to Implement
- Numerically Robust (will never explode)
- Most common system used
- May not converge under stress (compression or stretch)

Dynamic

- Connections as springs
- Forward Euler can only handle weak "springy" forces
- Implicit Integration required for stiff springs
- Damped behavior
- Converges to force-correct state
- Harder to Implement

Kinematic Solver

Realistic Behavior

Easy To Implement

Algorithm:

```
repeat a few times
  for each constraint
    move endpoints
    toward rest-length
```

Implicit Integration Spring Network

- Forward Euler

$$v_{t+dt} = v_t + m^{-1} force_t \, dt$$

- Implicit Euler

$$v_{t+dt} = v_t + m^{-1} force_{t+dt} \, dt$$

$$force_{t+dt} = f_{t+dt} = f_t + \frac{\partial f}{\partial p} \Delta p + \frac{\partial f}{\partial v} \Delta v \,, \qquad position \; p \quad velocity \; v$$

Derivatives of Force

Force at spring endpoint

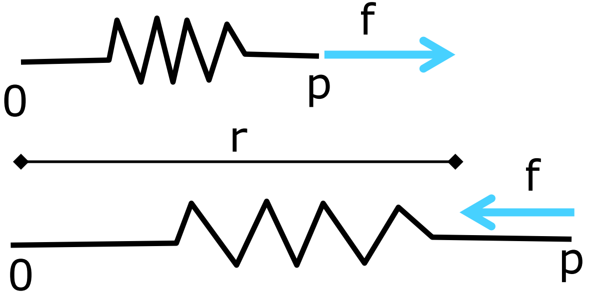
$$f = -k\left(p - r\frac{p}{\|p\|}\right)$$

$$\frac{df}{dv} = \begin{bmatrix} \frac{\partial f_x}{\partial v_x} & \frac{\partial f_x}{\partial v_y} & \frac{\partial f_x}{\partial v_z} \\ \frac{\partial f_y}{\partial v_x} & \frac{\partial f_y}{\partial v_y} & \frac{\partial f_y}{\partial v_z} \\ \frac{\partial f_z}{\partial v_x} & \frac{\partial f_z}{\partial v_y} & \frac{\partial f_z}{\partial v_z} \end{bmatrix}$$

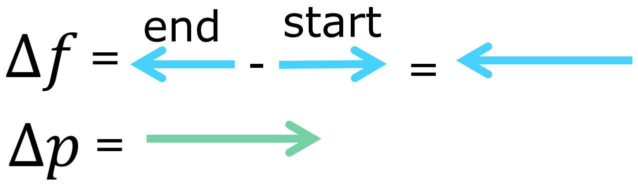
Jacobian

Compressed Spring:

Stretched Spring:



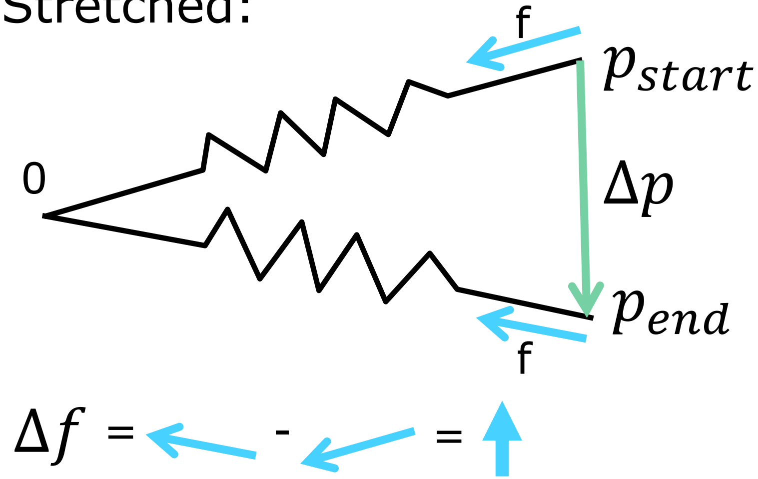
How force changes as endpoint moves **along** spring direction:



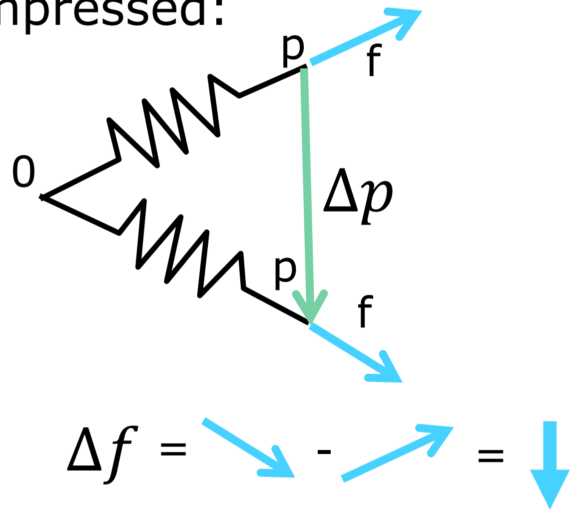
$$\frac{\Delta f}{\Delta p} = -k$$

Derivatives of Force – Endpoint moves orthogonal

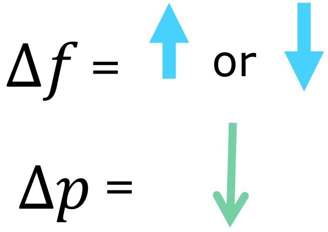
Stretched:



Compressed:



How force changes as endpoint moves **lateral** to spring dir:



$$\frac{\Delta f}{\Delta p} = -k\left(1-\frac{r}{\|p\|}\right)$$

Derivatives of Force – 3x3 Jacobian

General change in Force for a given change in position:

$$\boxed{\frac{\partial f}{\partial p} = -k\left(\frac{p \otimes p}{p \cdot p} + \left(I - \frac{p \otimes p}{p \cdot p}\right)\left(1 - \frac{r}{\|p\|}\right)\right)}$$

Avoid compression singularity by clamping r/||p|| .

Solve for Δv in linear system:

$$A \;\Delta v = b \;, \qquad A = I - \frac{\partial f}{\partial v} dt - \frac{\partial f}{\partial p} dt^2, \qquad b = f\;dt + \frac{\partial f}{\partial p} v\;dt^2$$

(note: slightly oversimplified)

Implicit Integration

Force Based

Stiff Springs

Realistic Behavior

Responsive

Convergence (Jitter Free)

Interacting with 3D Geometry

Summary

$$\int \mu m m \alpha r \gamma$$

Variety of topics:

Moving beyond triangles, dot and cross product:

- Objects as 3D Solids
  - Convex parts
  - Spatial props

Volume Integration

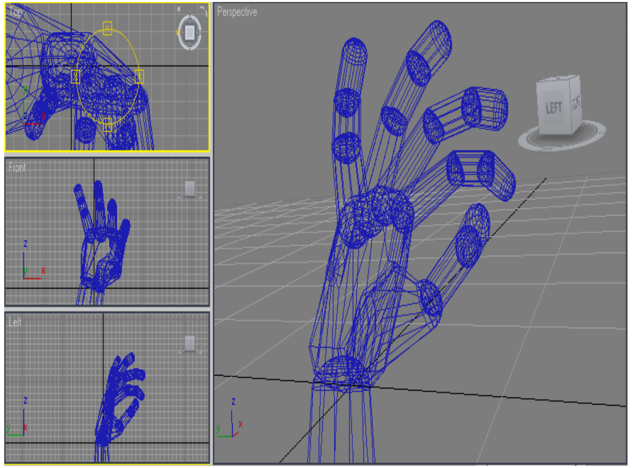
- "2nd Year" Maths
  - Object Motion
  - Stiff systems

Time Integration

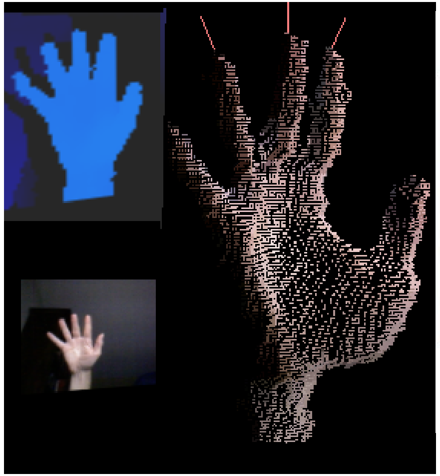
Q&A

Stan Melax
stan@melax.com

Hand Tracking – if someone asks….

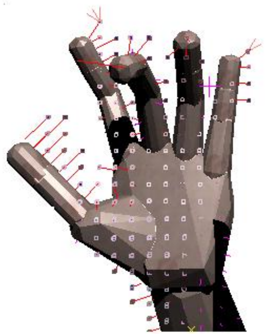


Tracking model

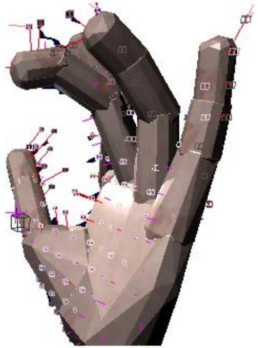


Depth data

RGB



Front



Side

Fit model