Convex Hull Generation with Quick Hull

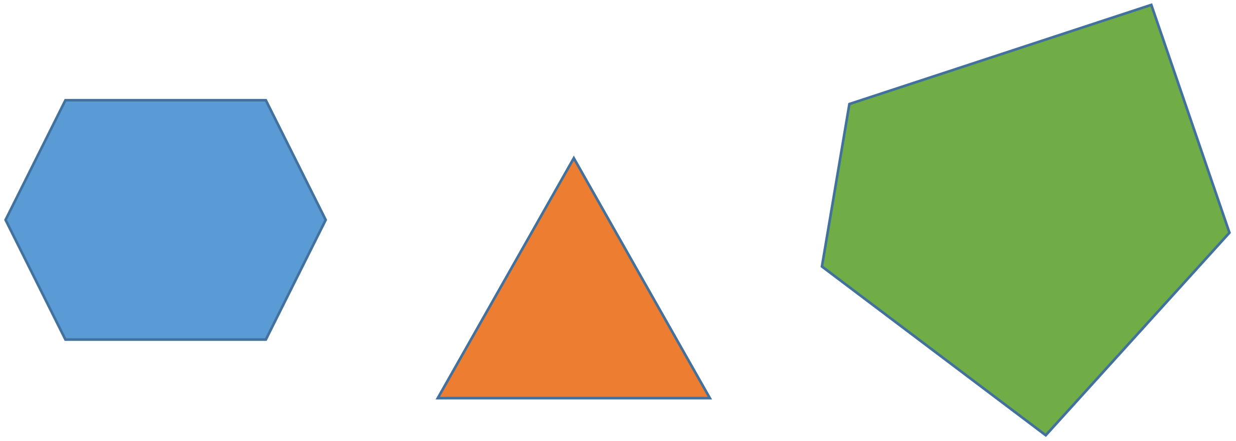
Randy Gaul

Special thanks to Stan Melax and Dirk Gregorius for contributions on this topic

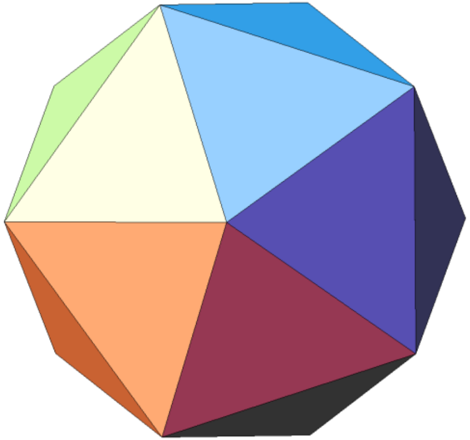
Overview – Quick Hull (QHull)

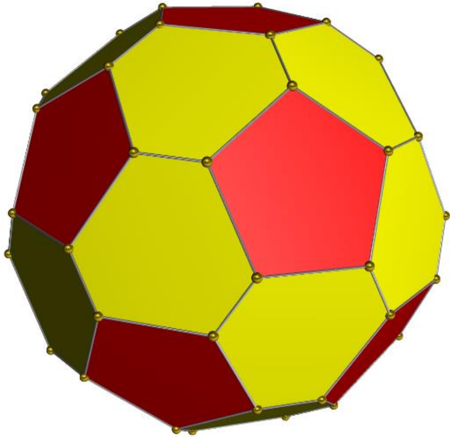
- Convex Hulls
- Why use them
- Computing a Convex Hull in 2D
- 3D Considerations
- Half Edge Mesh
- Simplified Convex Hulls
- Real time usage
- Optimization

Convex Hulls



Convex Hulls





Convex Hulls

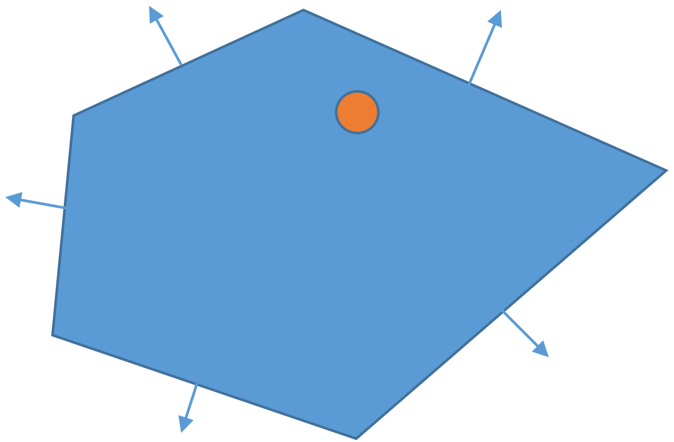
- Every vertex is on or behind every plane
- Cast a ray at the shape. Should penetrate and exit only once
    - If ray starts in shape should only hit inside of one face
- All normals of all faces point away from the center of mass
- Volume bounded by a number of planes
- Neighboring face normals point away from each other
- More…

Why use Convex Hulls?

- Convex hulls simplify collision detection
  - Collision detection is very difficult
- Efficiently represented

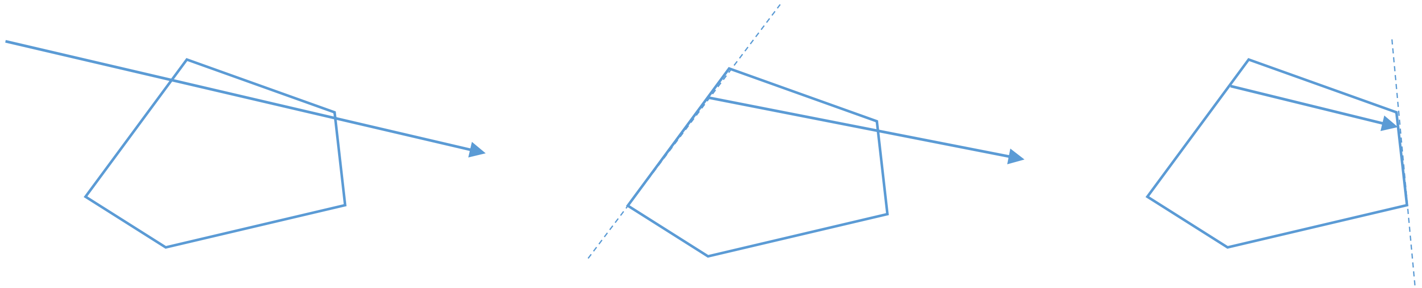
Why use Convex Hulls?

- Convex to point test
- Test against each plane



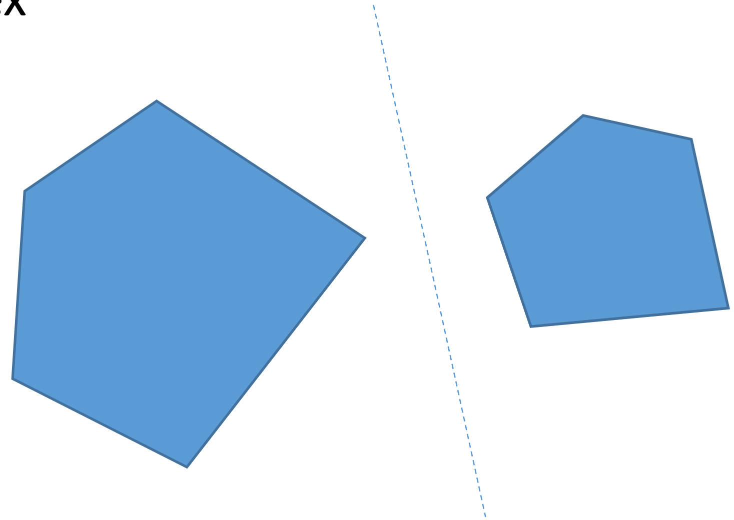
Why use Convex Hulls?

- Convex to ray test
- Trim ray against all planes



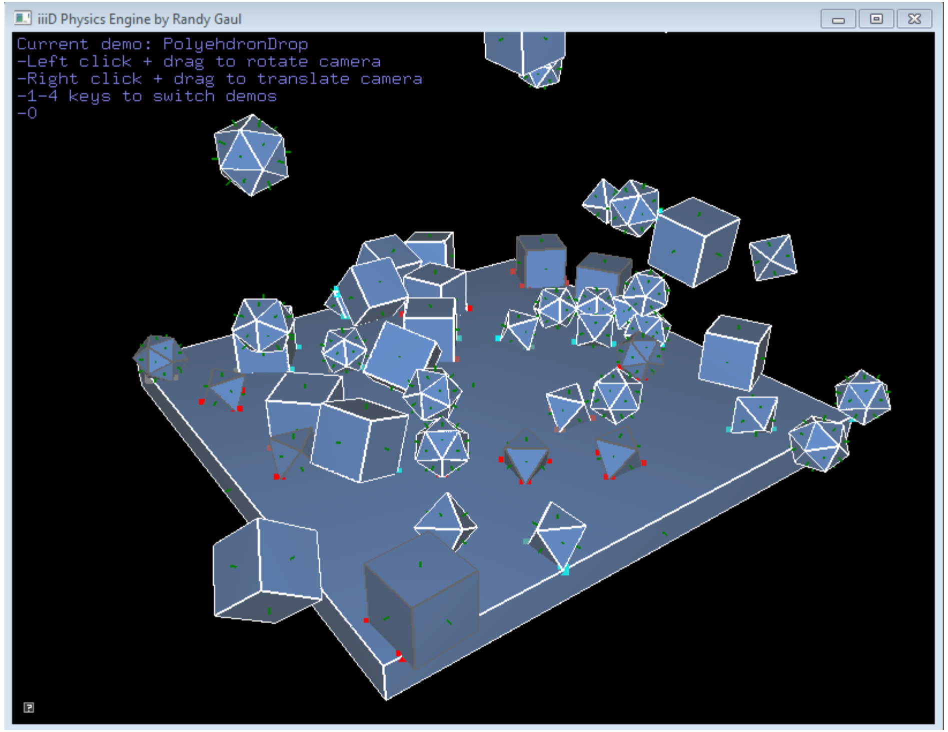
Why use Convex Hulls?

- Convex to convex



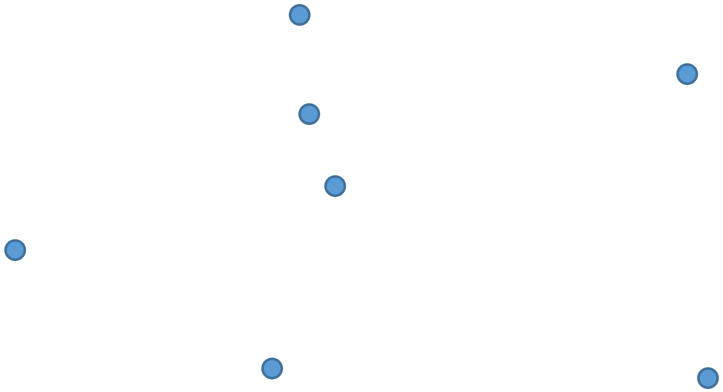
- People like convex hulls

Why use Convex Hulls?



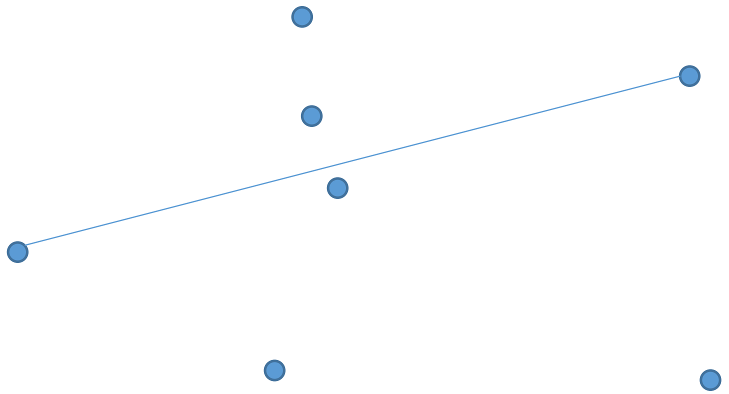
2D Quick Hull

- Initial triangle



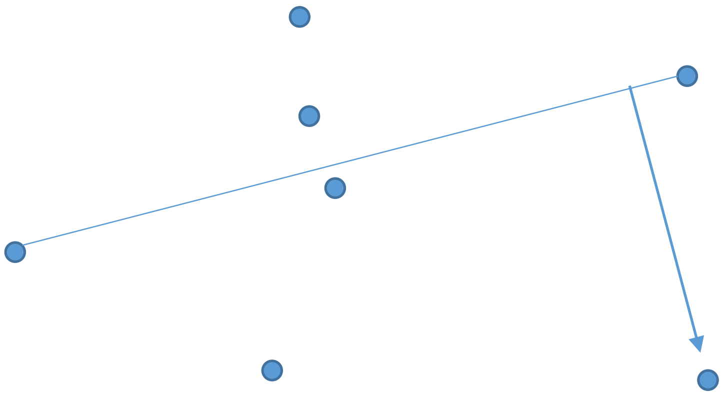
2D Quick Hull

- Farthest two points



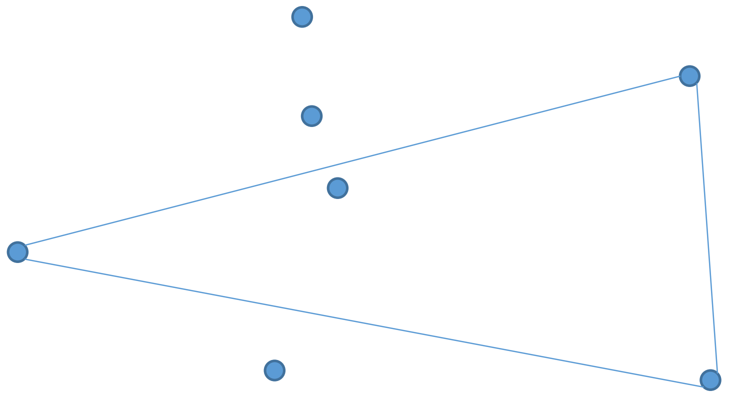
2D Quick Hull

- Farthest point to line



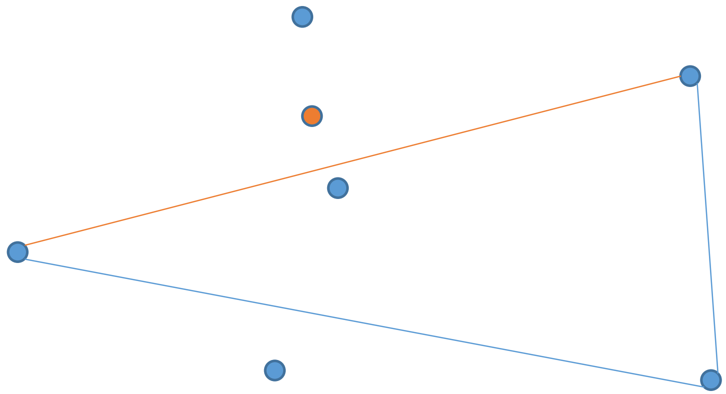
2D Quick Hull

- Now comes the main recursive loop



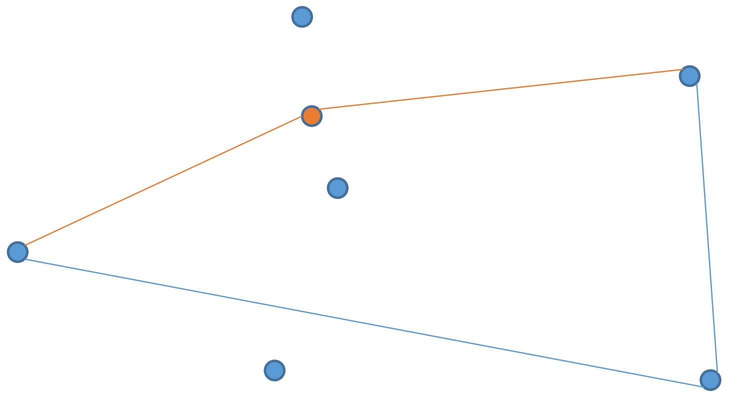
2D Quick Hull

- For a given outside point, find all visible faces



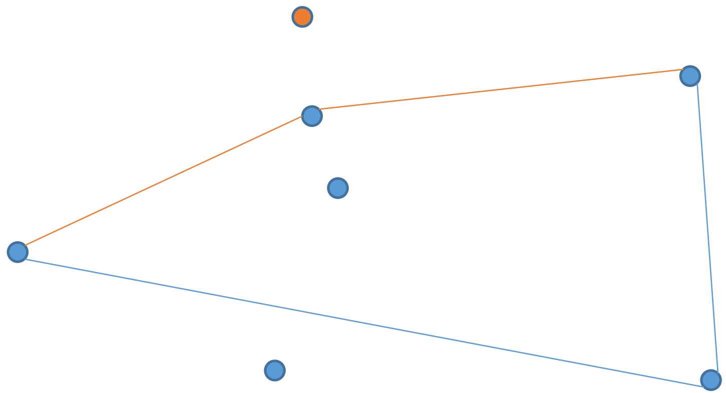
2D Quick Hull

- Delete visible faces, expand to new point



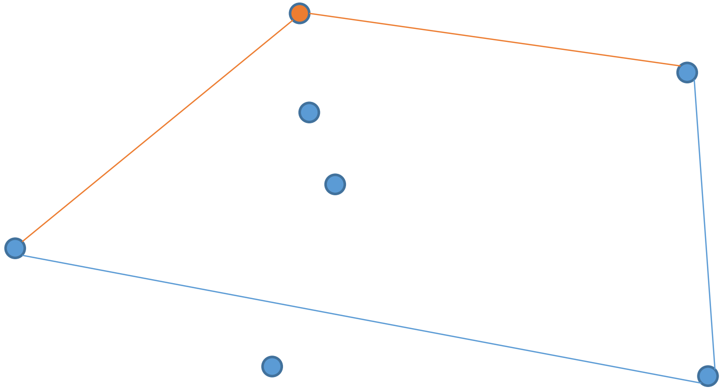
2D Quick Hull

- For a given outside point, find all visible faces



2D Quick Hull

- Delete visible faces, expand to new point

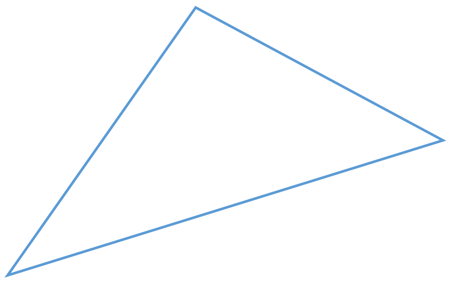


2D Quick Hull Outline

- Create initial triangle
- Assign exterior points to each face
  - If a point is above a face, it is assigned to that face
  - If a point is in front of multiple faces, just assign it to one
- For each face with a non-empty point set
  - Find furthest point from face, extreme point EP
  - Find all faces visible to EP
  - Delete all visible faces, expand to EP
  - Assign all remaining exterior points to the new expanded faces
  - Discard any interior points

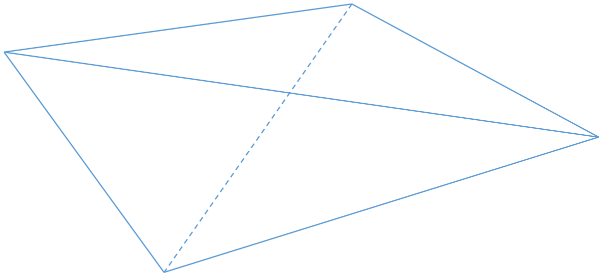
3D Quick Hull

- Initial tetrahedron
- Find initial triangle like in 2D



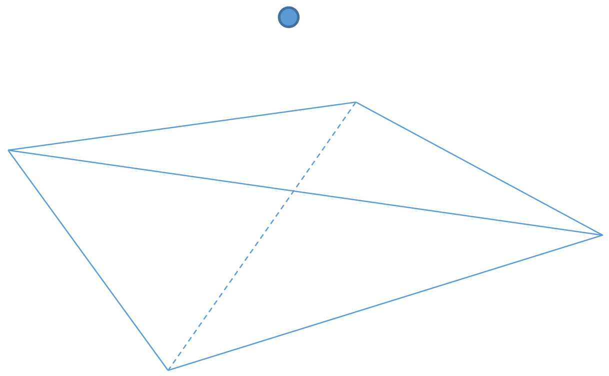
3D Quick Hull

- Furthest point to plane
- Hook up 3 side faces to furthest point



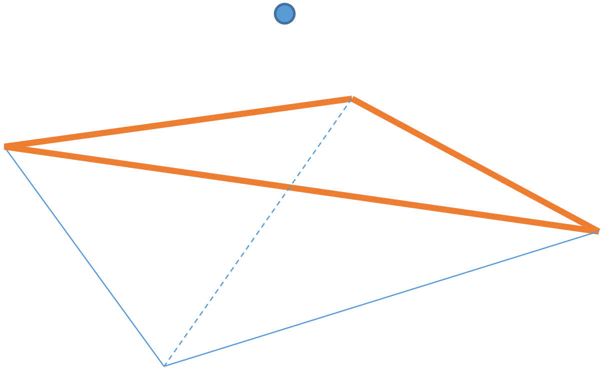
3D Quick Hull

- Expand faces recursively
- Given a face, find extreme point EP



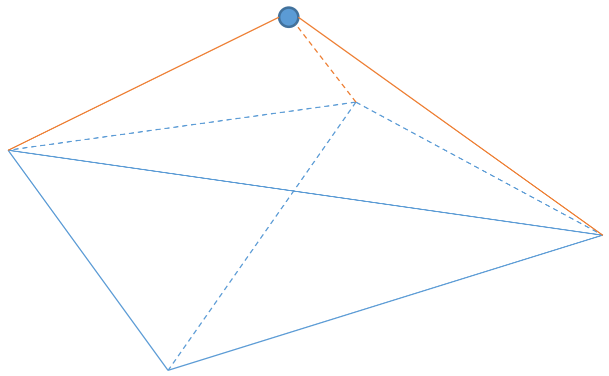
3D Quick Hull

- Find all faces visible to EP
- Delete all visible faces
- Record horizon line



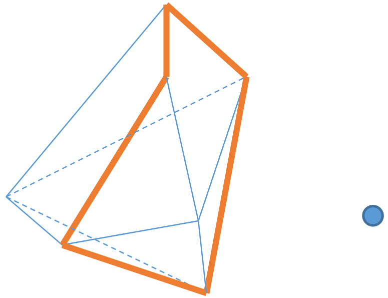
3D Quick Hull

- Expand horizon edges to EP by creating new faces



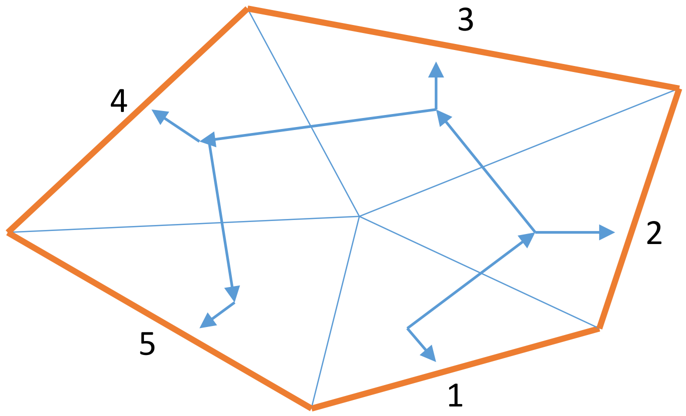
3D Quick Hull

- Horizon is the ring around all visible faces
- How do you find this horizon?



3D Quick Hull

- Depth first search upon the mesh
  - While at a visible face, if an adjacent face is not visible, shared edge is on the horizon
- Use the starting face as seed
- Mesh face winding is CCW
- Horizon is recorded CCW



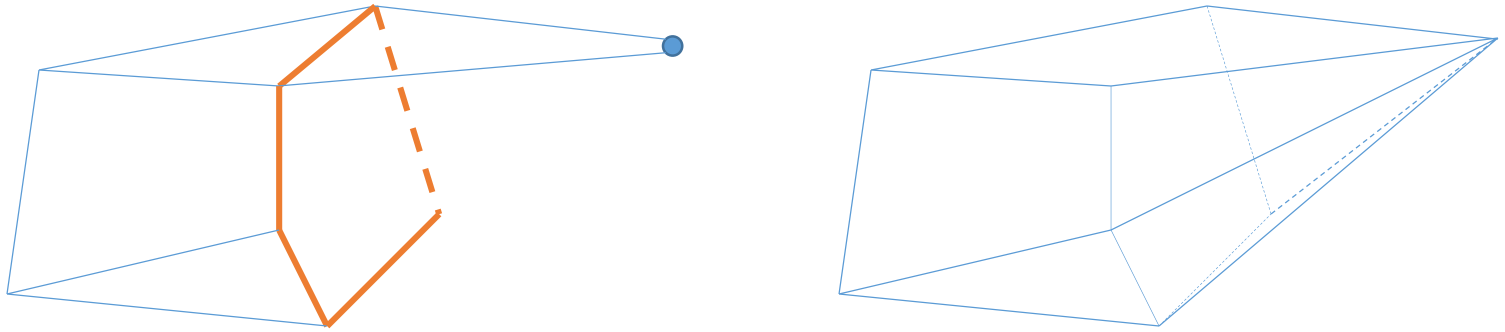
Numerical Inaccuracy – Bane of Our Existence

- Numerical robustness issues
  - Coplanar faces
  - Inverted faces

Coplanar Faces 3D

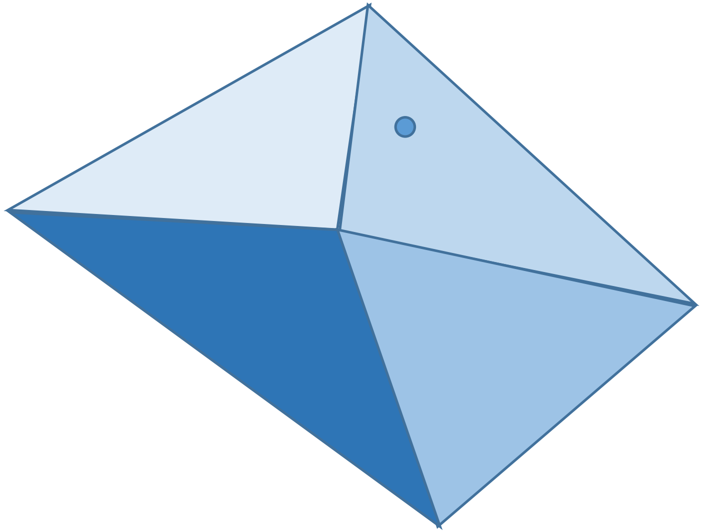
- When creating new faces to expand to EP
- Check face across horizon line
- Test for coplanar-ness
  - Merge two faces by deleting interior edges

Coplanar Faces 3D



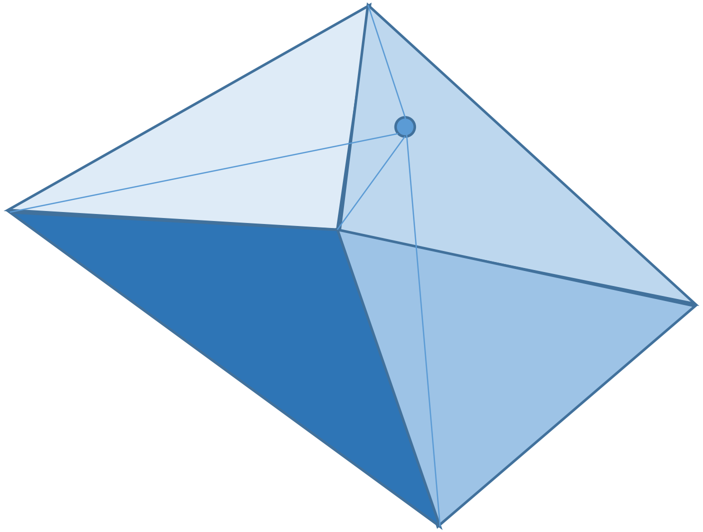
Inverted Faces

- Want to expand to EP



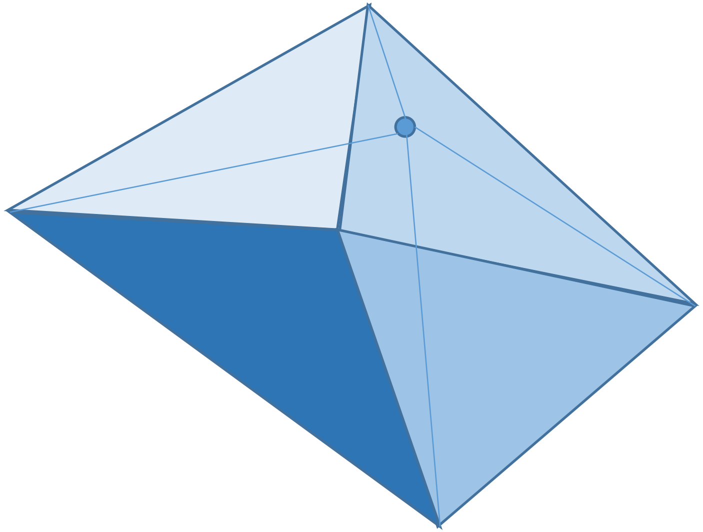
Inverted Faces

- Numerical inaccuracy introduce concavity



Inverted Faces

- Detect and correct the concavity

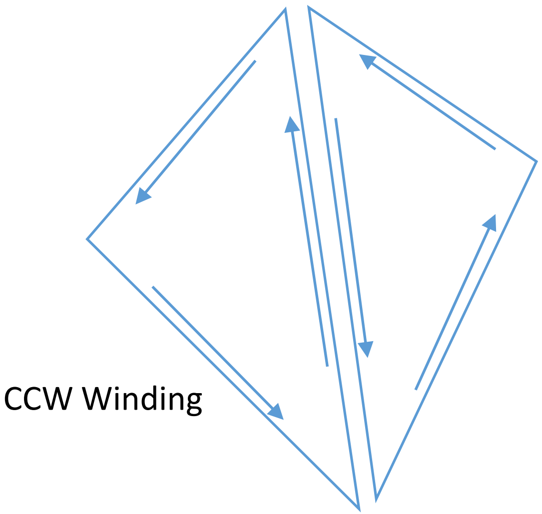


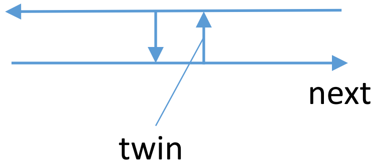
Inverted Faces

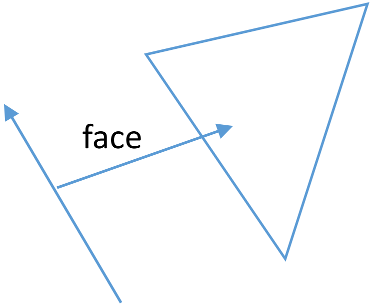
- Detect new concave faces by keeping vertex within initial simplex
  - Average vertices of initial simplex
  - This is your reference point RP
- While adding a new face to expand to EP
  - Test plane normal n with vector from RP to a point on new face
    - If RP dot n is negative you have a flipped face

How to Represent Meshes

- Use half edge data structure







Half Edge Mesh Format

```
struct HalfEdge
{
  char vert;
  char next;
  char twin;
  char face;
};
```

```
struct Face
{
  char edge;
};
```

```
struct Hull
{
  Vec3 centroid;
  int vertexCount;
  Vec3 *vertices;
  int faceCount;
  Face *faces;
  Plane *planes; // use faceCount
  int edgeCount;
  HalfEdge *edges;
};
```

What about this Simpler Mesh Format?

- If you’re making meshes it’s for physics
- If you’re making physics meshes you should be using SAT
  - EPA is getting outdated
- 3D SAT requires Gauss Map optimization to be fast
  - See Dirk Gregorius GDC 2013, SAT and the Gauss Map optimization
- Gauss Map optimization requires edge lookup
- Any mesh format works so long as you can easily do:
  - Face->Edges->Vertices

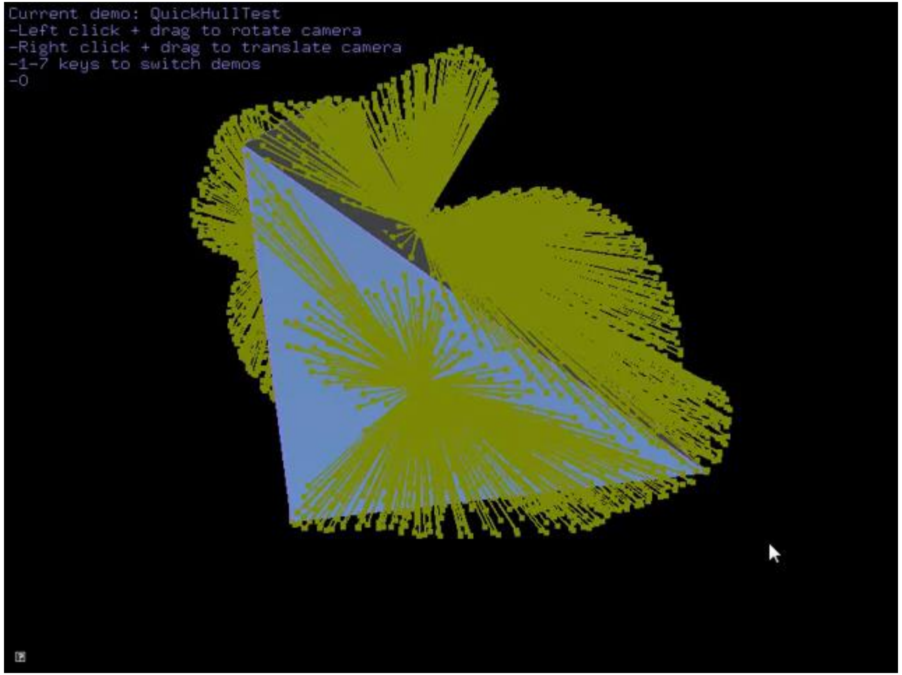
Simplified Convex Hull

- Too many vertices is not helpful

Greedy Hull

- Quick Hull is n log n
- We don’t care about that anymore, lets make it O(k * n)
  - k is specified number of output vertices
- Idea:
  - New recursive step
  - Loop over all faces with point set, find farthest EP
  - Expand to this EP
  - Repeat

Greedy Hull


Current demo: QuickHullTest
-Left click + drag to rotate camera
-Right click + drag to translate camera
-1-7 keys to switch demos
-0

Real-Time Quick Hull

- Quick Hull can be run in real-time
  - Usually it’s just a pre-processing step though
- If you can optimize Quick Hull, can treat as operation

Real-Time Quick Hull

- Brittle fracturing

Real-Time Quick Hull

- Simple modelling
  - Add/remove points from hull
- In-game construction (magic crayon flash game?)
- Merging volumes

Optimization

- Performance dominated by cache coherency
- Use pre-allocated arrays
- Use index references, not pointers
  - Lets you modify hull at run-time, copy it around, etc.
- For final hull use char for index references
  - Smaller memory footprint, easier to fit into cache
- Do greedy hull
  - Sure it’s O(n^2), but big O notation isn’t “real”
  - O(n * k) can be very fast and stable

Questions?

Resources

- Quick Hull
  - Dirk Gregorius – GDC 2014 Quick Hull Lecture
  - Original Quick Hull paper – “The Quick Hull Algorithm for Convex Hulls”
  - http://www.cs.ubc.ca/~lloyd/java/quickhull3d.html
- Greedy Hull (Stan Hull)
  - http://www.bulletphysics.org/Bullet/phpBB3/viewtopic.php?f=12&t=255
- Half edge mesh
  - Graham Rhodes – Computational Geometry (slides) GDC 2013