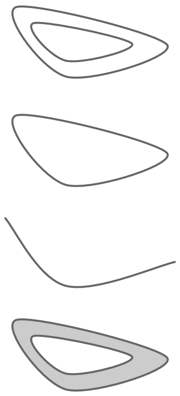
Tesselated GPU Path Rendering

Stephen White, GPU-MON

What’s a path?

- path: unordered set of contours
- contour: ordered set of segments
- segment: line, quadratic, cubic, conic
- may have holes, may be self-intersecting
- winding rule to specify how holes are filled
- Web: SVG, <canvas>, rounded rect, fonts



Why do we even need to do this?

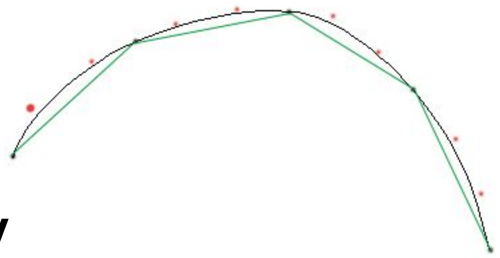
- GPUs render very fast, but only triangles
- must convert from path to triangles

Several approaches

- rasterize in software, upload to texture
- stencil-and-cover
- distance fields
- tessellation
- Loop-Blinn (plus tessellation)

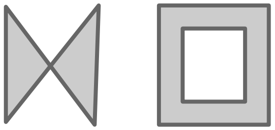
How to handle curved segments?

- linearize them (subdivision)
- contours now piecewise-linear
- use screenspace threshold
- problem now: tessellate arbitrary polygonal contours into triangles

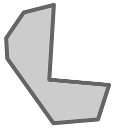


Animals in the polygon zoo

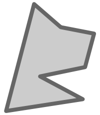
- Complex
  - self-intersections, holes



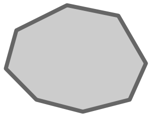
- Simple
  - no self-intersections, no holes, concave



- Monotone
  - monotonically incr; all pts one side, concave

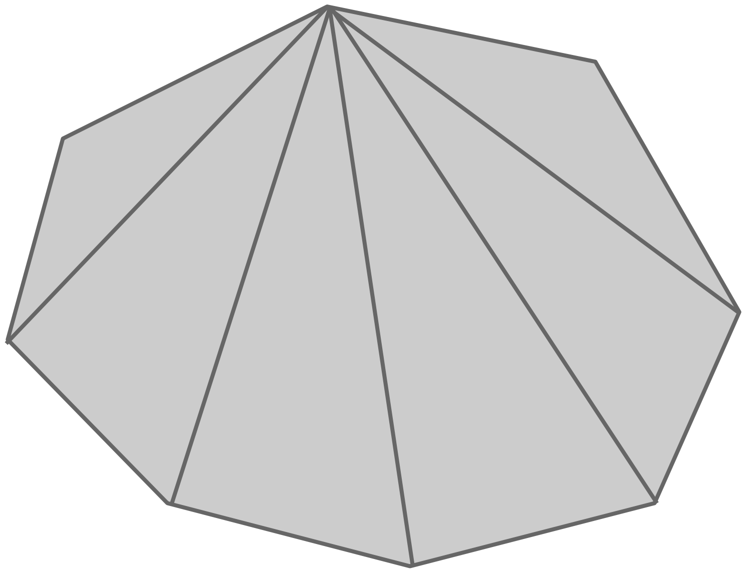


- Convex
  - all angles <= 180°



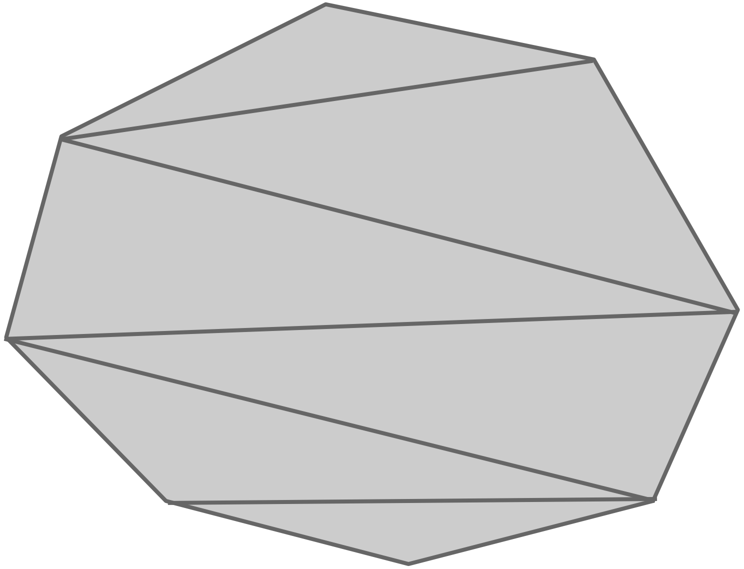
Convex polygons

First approach: fan



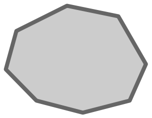
Convex polygons

Second approach: Alternate sides



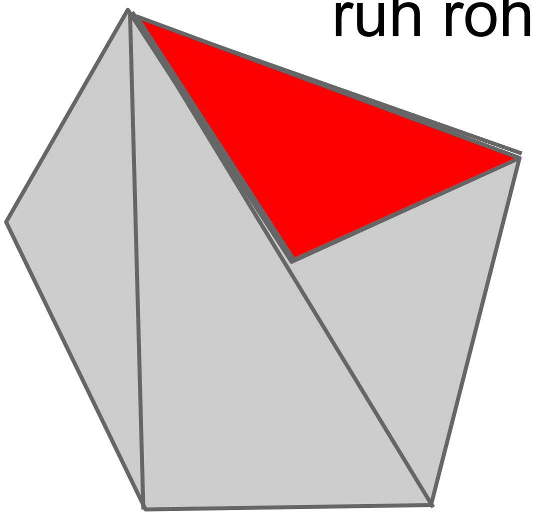
Animals in the polygon zoo

- Complex
  - self-intersections, holes
- Simple
  - no self-intersections, no holes, concave
- Monotone
  - monotonically incr; all pts one side, concave
- Convex
  - all angles <= 180°



✓

Concave: doesn't work



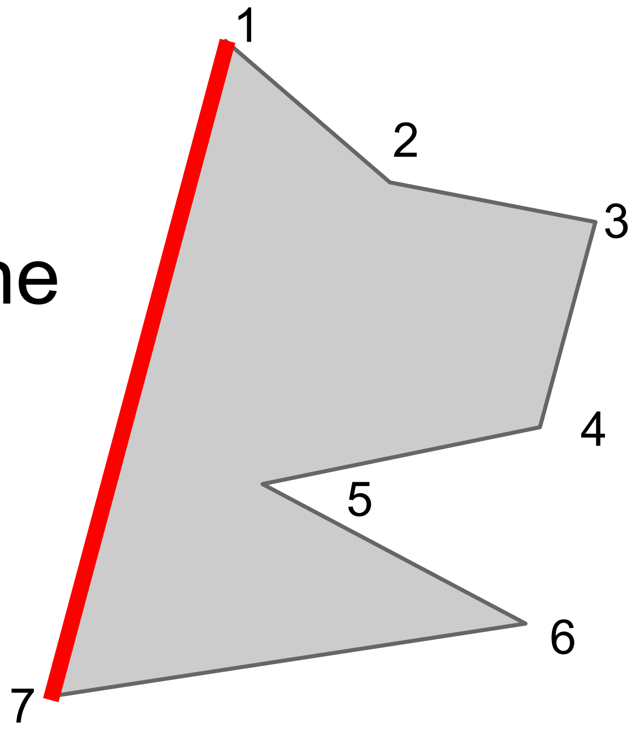
Concave: doesn’t work

“If at first you don’t succeed ... try something easier.”

- Alfred E. Neumann

Monotone polygon

All points lie to one side of the minY->maxY line



Points are monotonically increasing in Y

Triangulate monotones: ear clipping

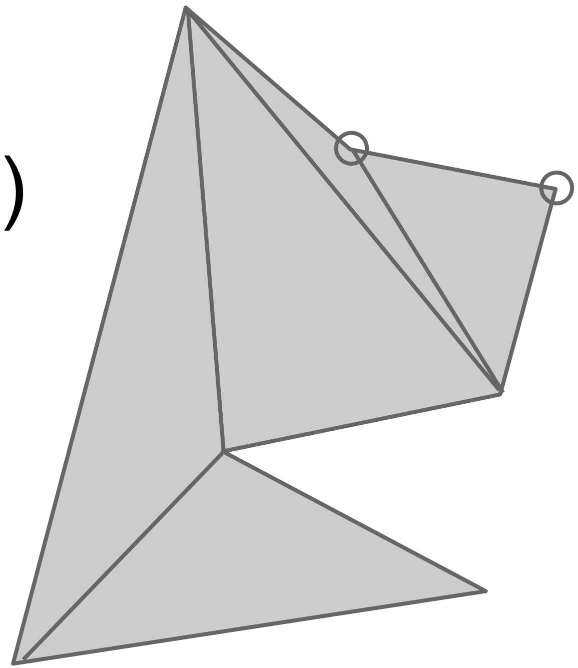
1) find the first convex vertex (O(n))

2) clip its ear (O(1))

3) find convex neighbour

4) clip its ear

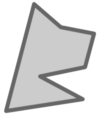
5) repeat



backtrack at most one step => O(n)

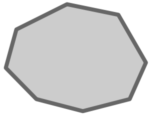
Animals in the polygon zoo

- Complex
  - self-intersections, holes
- Simple
  - no self-intersections, no holes, concave
- Monotone
  - monotonically incr; all pts one side, concave



✓

- Convex
  - all angles <= 180°

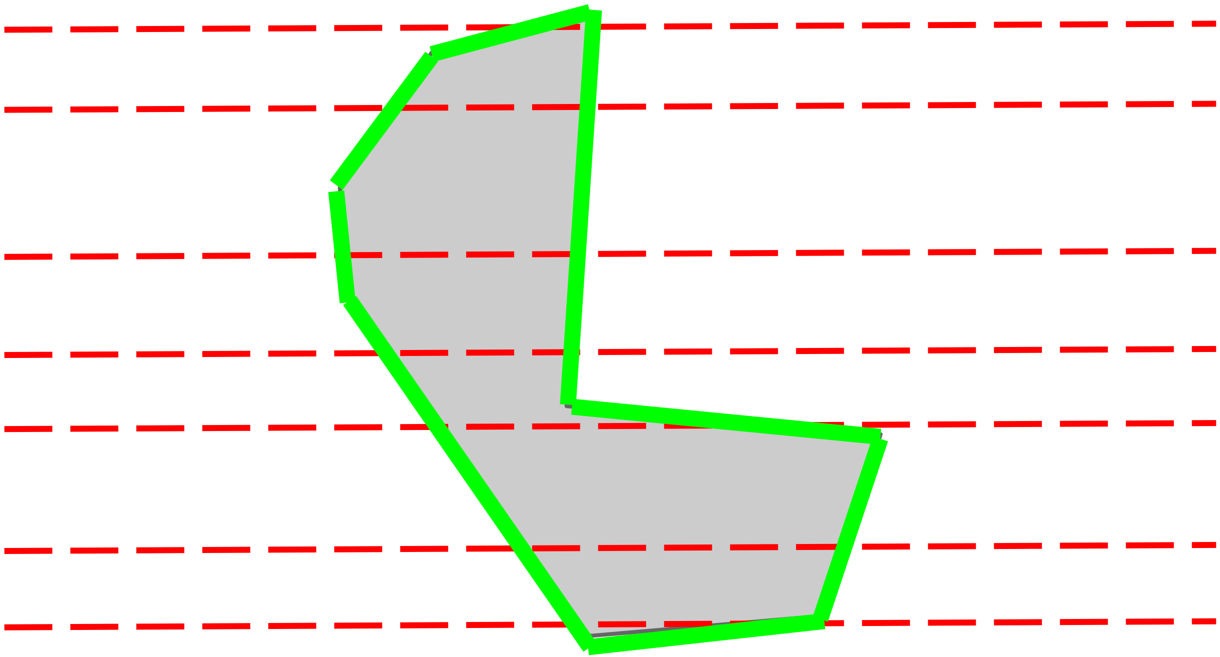


✓

Sweep Line Algorithms

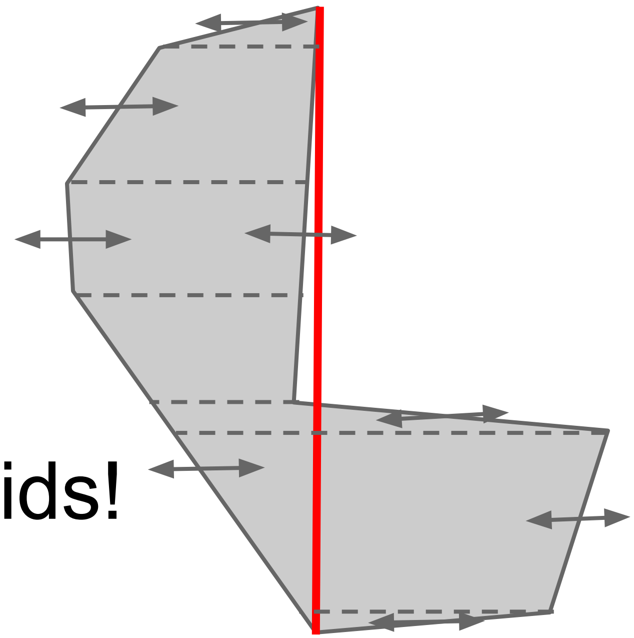
- Sort vertices in Y (O(N lg N))
- Sweep line passes from top to bottom
- Active Edge List: top vertex seen, bottom not
- Two vertices on the same sweep line?
  - WLOG: rotate slightly in Z
- $y_1 == y_2$ => Secondary compare in X
- “enclosing edges” of a vertex

Active edge list: example



Lil’ bit harder now: simple polygon

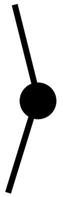
- Obviously, not monotonic
- Trapezoidal decomposition: turn simple into monotones
- I say, forget about the trapezoids!
- Which monotone lies left/right of each edge?



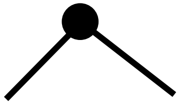
Lil’ bit harder now: simple polygon

Categorize verts in the zoo

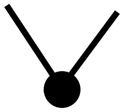
All degree 2, so three species:



type 1:
one edge above,
one edge below



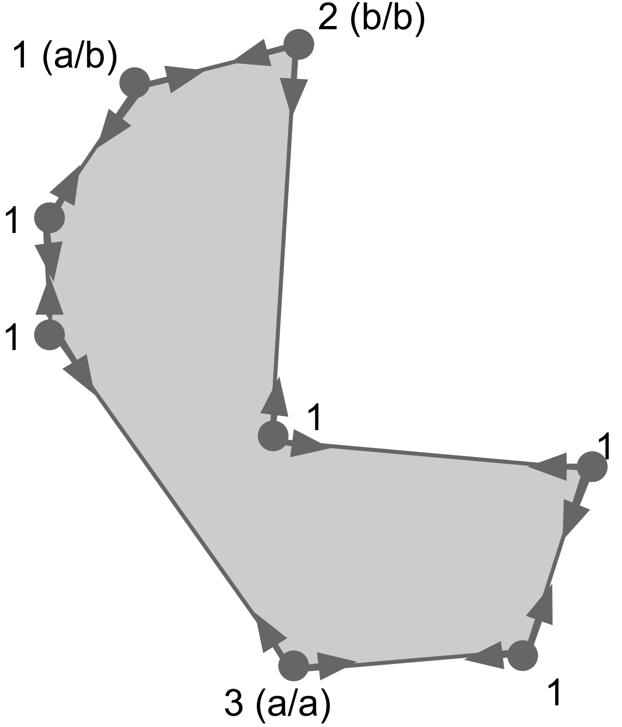
type 2:
both edges below



type 3:
both edges above

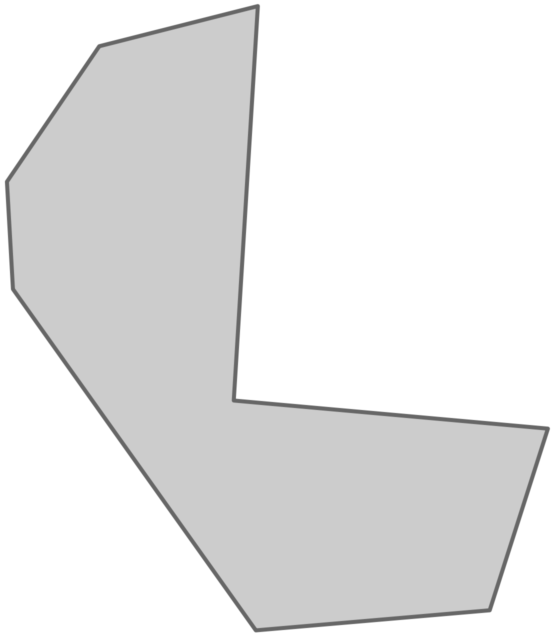
Lil' bit harder now: simple polygon

- vertex types



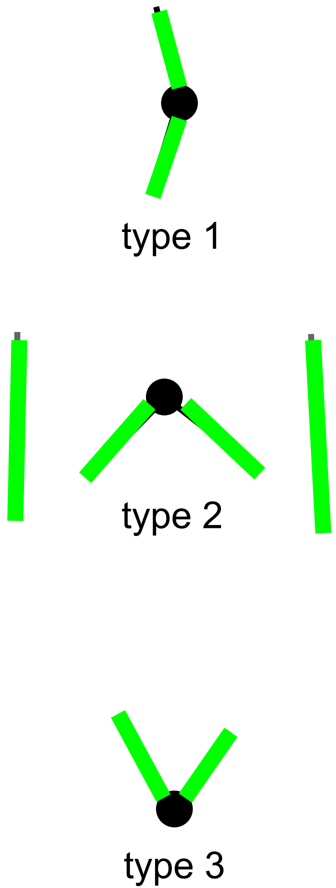
Lil' bit harder now: simple polygon

- Sort vertices in Y (secondarily in X)
- Each edge stores left mono, right mono



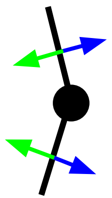
Simple to monotone: da rules

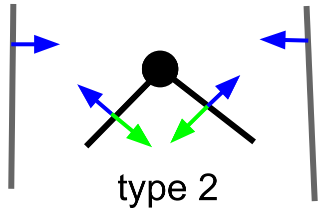
- type 1:
  - AEL: remove incoming, insert outgoing
- type 2:
  - AEL: find enclosing edges (if any), and insert new edges in between
- type 3:
  - AEL: remove incoming edges

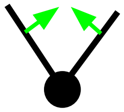


Simple to monotone: da rules

- type 1:
  - copy L/R monotones from incoming
- type 2:
  - copy enclosing monotone (if any) to exterior edges
  - open new monotone below
- type 3:
  - close monotone above



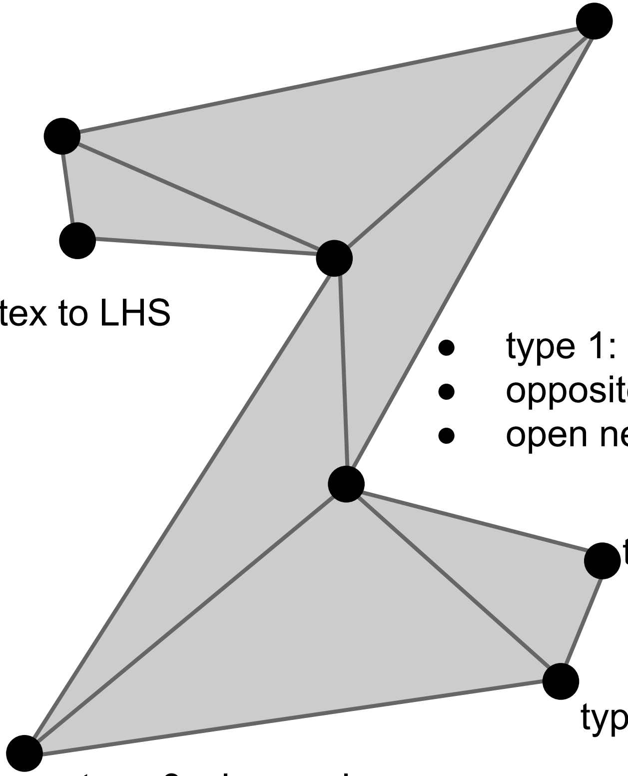




Simple to monotone: da rules

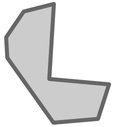
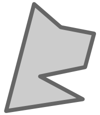
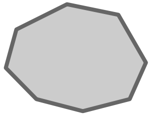
- if incoming edge has left monotone
  - add current vertex to monotone’s RHS
- if incoming edge has right monotone,
  - add current vertex to monotone’s LHS
- if added on opposite side to handedness:
  - close monotone & start a new one
  - use previous & current vertex as the first two verts

Simple to monotone: example



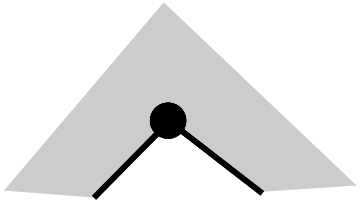
Fine print: left-handed monotones are triangulated bottom-up (simplifies code)

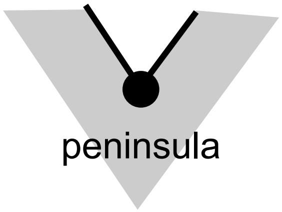
Animals in the polygon zoo

- Complex
  - self-intersections, holes
- Simple
  - no self-intersections, no holes, concave  ✓
- Monotone
  - monotonically incr; all pts one side, concave  ✓
- Convex
  - all angles <= 180°  ✓

What about islands / peninsulas?

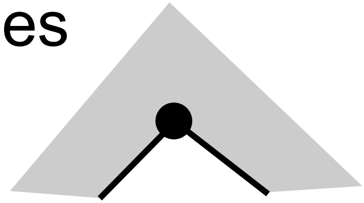
- Island: a lone type-2 enclosed inside a larger poly (e.g., hole)
- Peninsula: type-3 that joins two previously-unrelated polys


island

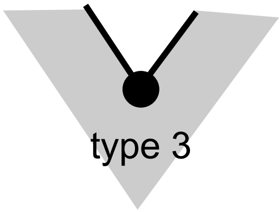


Moar rules: splitting and merging

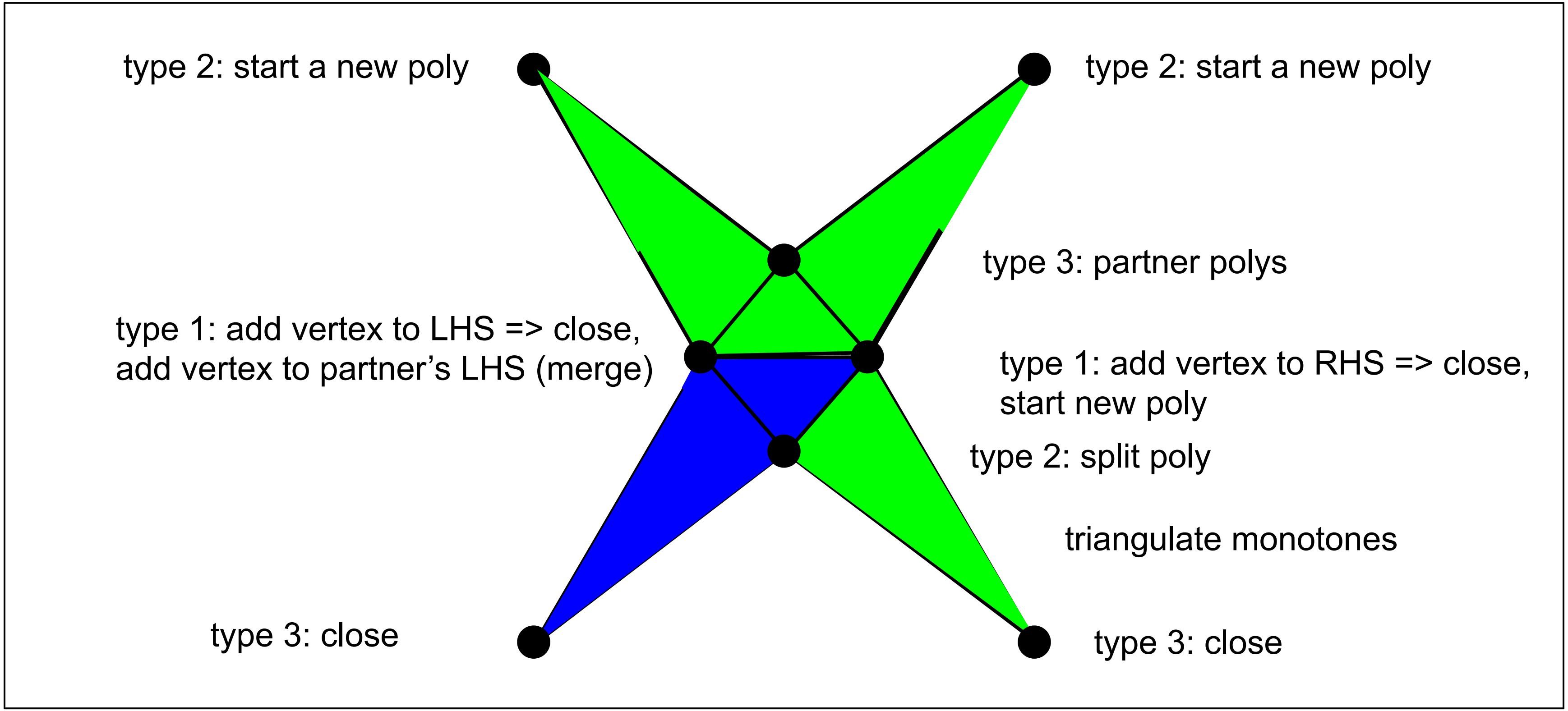
- Island:
  - if monotone above, split into two monotones
- Peninsula
  - merge left & right shapes via partnership
  - partnership ends at subsequent vertex
  - instead of starting a new monotone, add vertex to partner monotone



type 2



Splitting and merging: example

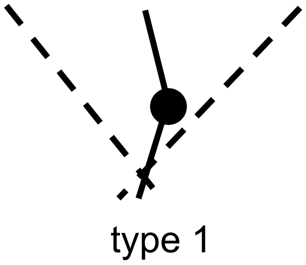


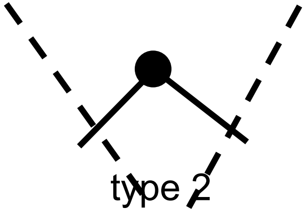
Intersections

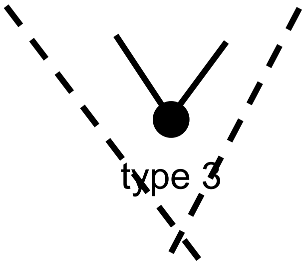
Seems O(N^2) in edges?

Bentley-Ottman (sweep line):

- key observation: all intersecting edges are neighbours in AEL
- check for intersections:
  - type 1 & 2: outgoing vs. enclosing edges
  - type 3: left enclosing vs right enclosing
  - O((N + k) lg N) for k intersections





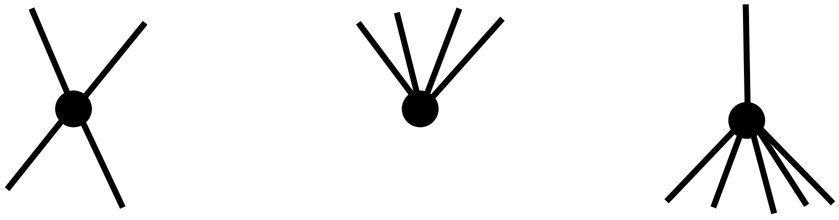


Intersections, cont'd

- insert new vertex at intersection point
  - always below sweep line
- split edges
- check for coincident vertices; merge

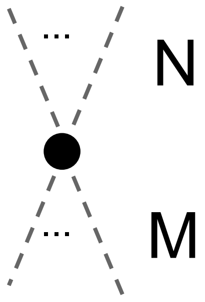
Intersections

The vertex zoo just got bigger...



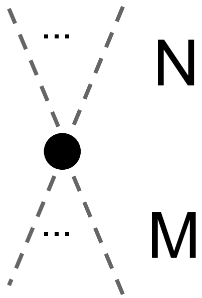
Vertex zoo, generalized

- N edges above, M edges below
- For AEL:
  - N == 0 ⇒ type 2
  - M == 0 ⇒ type 3
  - else ⇒ type 1



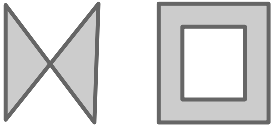
Vertex zoo, generalized

- When N > 2, close multiple polys above
- When M > 2, open multiple polys below
- poly membership is copied along leftmost & rightmost edges



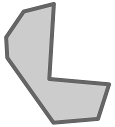
Animals in the polygon zoo

- Complex
  - self-intersections, holes



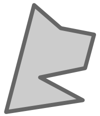
✓

- Simple
  - no self-intersections, no holes, concave



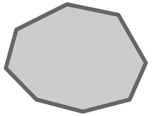
✓

- Monotone
  - monotonically incr; all pts one side, concave



✓

- Convex
  - all angles <= 180°



✓

Pretty easy so far

- Nice computational geometry algorithms
- Fairly easy to implement
- O(N lg N)

Excrement, meet rotary ventillator

(fun with floating point)

- Bentley-Ottman requires exact math
- intersection is 5th order polynomial
- 24 bits mantissa => 160 bits :(

Float sucks: what do we do?

Two schools:

- These algorithms don’t work; give up
- Use arbitrary precision arithmetic

Float sucks: what do we do?

Drop out of school:

- assume intersections are random;
  adjust mesh geometry to match
- inaccuracies are in ULPs, not pixels

Excrement, meet rotary ventillator

Intersections can:

- change outgoing edge order within vertices
  - solution: intersection => check & reorder top vertex
- change edge order in active edge list
  - solution: intersection => check & reorder w/neighbours in AEL

Excrement, meet rotary oscillator

Merging coincident vertices can:

- add edge above type 2 => type 1
- remove last edge above type 1 => type 2
- both can change enclosing edges and polys

Solution: on type change,

- redo find enclosing edges
- restart intersection checks

Future work

- shipping in M44, Android, MSAA
- caching (threshold scale), shipping in M46
- enable in <canvas>
- alpha ramp AA (no MSAA required)
  - Desktop: Intel MSAA perf is terrible
  - 1-pixel-wide alpha ramp around geometry
  - batching

References

- Fournier & Montuno, "Triangulating Simple Polygons and Equivalent Problems"
- Bentley, J. L.; Ottmann, T. A. (1979), "Algorithms for reporting and counting geometric intersections"
- Jean-Daniel Boissonnat and Franco P. Preparata. Robust plane sweep for intersecting segments. *SIAM Journal on Computing*, 29:1401-1421, 2000.

Questions?

Performance results

https://docs.google.com/spreadsheets/d/1S1hlb5hHCG04fP-9XeFgc73Zo07N8HIa5lYC07ju-Gw/edit#gid=0

https://docs.google.com/spreadsheets/d/1mMU39qbUt7LmQRUnggqRJ0oL6sE7zwI8nElx3j__bnek/edit#gid=0