1. What is a Voronoi Diagram?

A Voronoi Diagram subdivides a space into regions.
- These regions are called Voronoi Cells.
- The objects inside these regions are called Voronoi sites.
Each cell consists of points that are closer to one site than to all other sites.

2. How is a Generalized Voronoi Diagram Different?

Unlike standard Voronoi Diagrams, Generalized Voronoi Diagrams separate entire objects.
- Using GVDs, we can easily determine:
  - what object is closest to a particular point
  - which objects are neighbors to other objects
  - what path to follow to avoid collisions between objects

3. Our Focus

Our focus is to compute Generalized Voronoi Diagrams:
- on everyday consumer hardware
- in real time, on animated datasets
- in both 2D and 3D
- on datasets that contain:
  - disconnected geometry
  - non-manifold geometry
  - self-intersections
  - inter-object intersections

4. Difficulties Computing the Generalized Voronoi Diagram

The edges of GVD cells are curved, and require more CPU time and memory. Current GVD algorithms use lines on a grid to approximate curves, but most techniques either fail on dense datasets or are too slow to be useful.

5. Algorithm Overview

- To improve performance, we break up GVD construction into manageable work items.
- These work items are solved in parallel on readily available graphics cards.
- In parallel, we:
  1) build a vertex octree
  2) use the vertex octree to create a facet octree
  3) perform a color wavefront on the facet octree.
  4) use the color wavefront to surface the GVD.

6. Building the Vertex Octree

An octree is a memory efficient tree structure where each node has eight children.
- To improve construction performance, we implement Tero Karras' algorithm:

   1) We convert cartesian coordinates into Morton order by interleaving the bits.

   2) We sort the morton codes using a parallel radix sort

   3) We use the sorted morton codes to construct a binary radix tree.

   4) We traverse the leaves of the radix tree up to the root, creating the octree along the way.

7. Using the Vertex Octree to Create a Facet Octree

Karras' Algorithm produces a vertex octree, which if surfaced would give us a standard voronoi diagram. However, we need entire facets to be separated to get a generalized voronoi diagram.

8. Generating the Color Wavefront

- After creating the facet octree, we assign colors to corners of the octnodes that intersect objects.
- We continually propagate colors to neighbor vertices until all vertices have been colored.

9. Surfacing the Generalized Voronoi Diagram

In 3D, we compute the 2D GVD for each face of a given node, and then triangulate the 2D GVDs with the cell's GVD centers.

10. Results

11. References