Problem: Generate high-quality Delaunay mesh using the GPU.
Input: Point set in \( \mathbb{R}^2 \) or \( \mathbb{R}^3 \) with [constraints] and [point weights].

Framework

1. **Input**
   - Points
   - [Constraints]
   - [Weights]

2. **Point Insertion**
   - Delaunay triangulator

3. **Flipping**
   - 2D flip
   - 3D flip

4. **Constraint Enforcement**

5. **Refinement**

6. **Output**

2D Delaunay Triangulation \(^1\)

A direct application of our framework with any flipping sequence efficiently leads to the 2D Delaunay triangulation.

2D Regular Triangulation \(^2\)

- **Key techniques**:
  - Identify redundant points through unflippable edges.
  - Combine flipping of regular and non-regular edges to remove redundant points.

3D Delaunay Triangulation \(^3\)

- **Key techniques**:
  - After flipping, only a few regions contain locally non-Delaunay facets.
  - Adaptive star splaying locally fix these regions.

Performance

- **2D Delaunay triangulation**
- **2D constrained Delaunay triangulation**
- **2D regular triangulation**
- **3D Delaunay triangulation**

Ongoing work

- Constrained Delaunay triangulation in \( \mathbb{R}^3 \).
- Delaunay refinement in \( \mathbb{R}^2 \) and \( \mathbb{R}^3 \).
- Local transformation in higher dimensions.

References

\(^1\) Computing 2D constrained Delaunay triangulation using the GPU \( \text{[TVCG '12]} \)
\(^2\) Flip-Flop: Convex hull construction via star-shaped polyhedron in 3D \( \text{[i3D '13]} \)
\(^3\) A GPU accelerated algorithm for 3D Delaunay triangulation \( \text{[i3D '14]} \)