Tetrahedralization And Volume Rendering

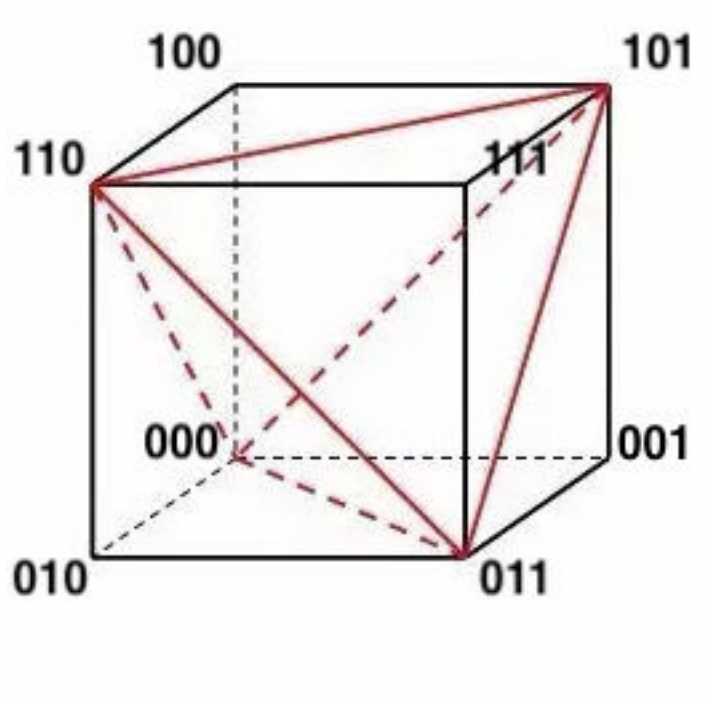
Ziyi Yu, Chuan Li, Yuhang Gong.

OVERVIEW

- 1. Tetrahedralization
- 2. Computing Screen Space Coordinates
- 3. Extracting Intersection Records
- 4. Calculating Intersection Effects
- 5. Sorting Intersection Effects List
- 6. Composition

1. Tetrahedralization

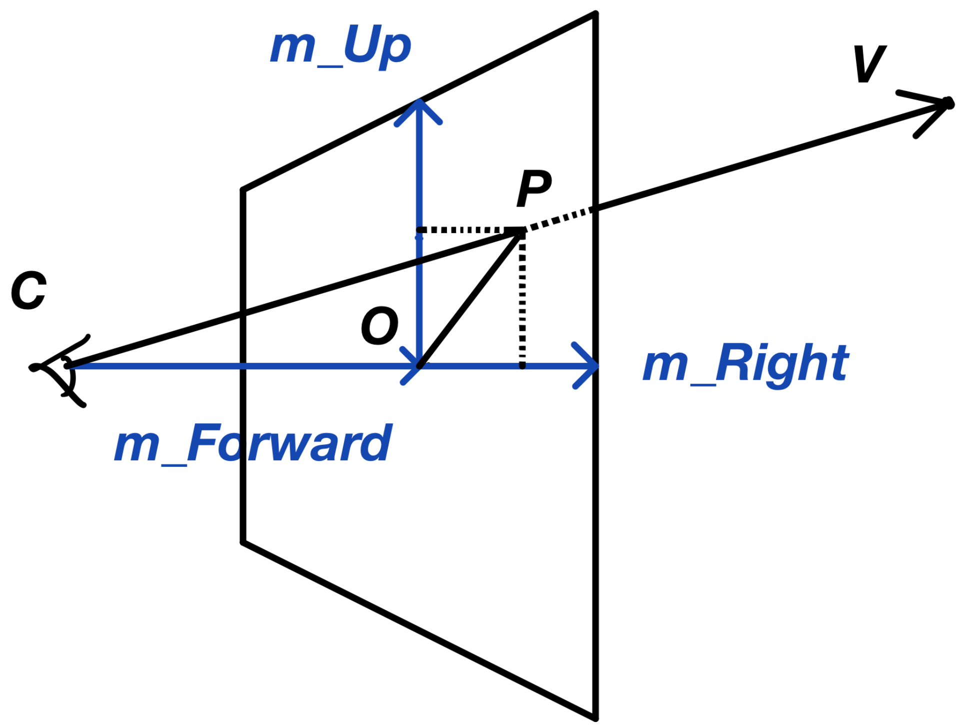
1. Tetrahedralization



- Tetrahedron Label Sets
  - {000, 001, 011, 101}
  - {000, 011, 010, 110}
  - {000, 100, 110, 101}
  - {000, 110, 011, 101}
  - {111, 101, 110, 011}

2. Computing Screen Space Coordinates

2. Computing Screen Space Coordinates

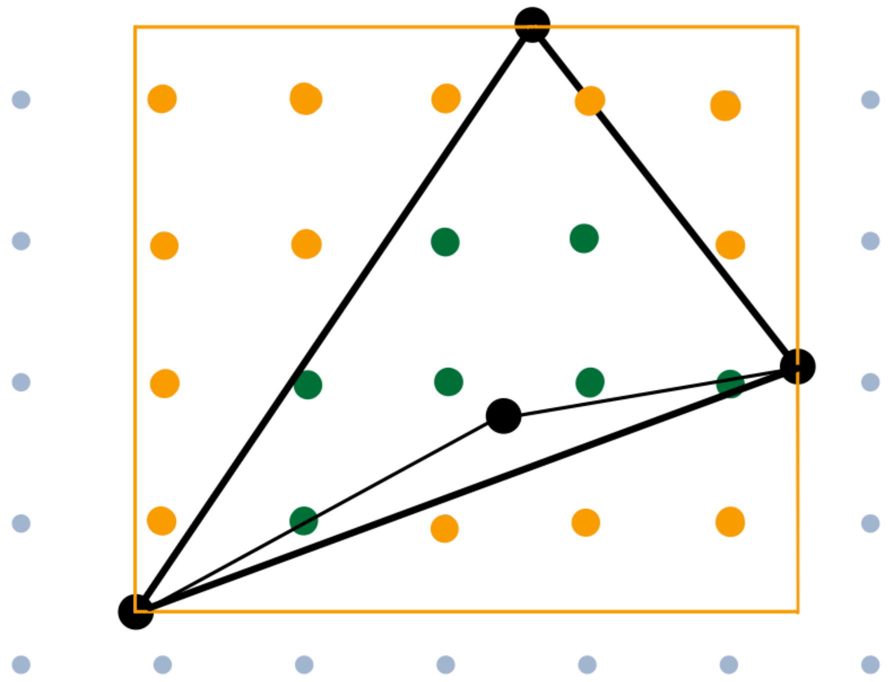


3. Extracting Intersection Records

Helper Functions

- Cross Product
- Line Side
- Is In Triangle

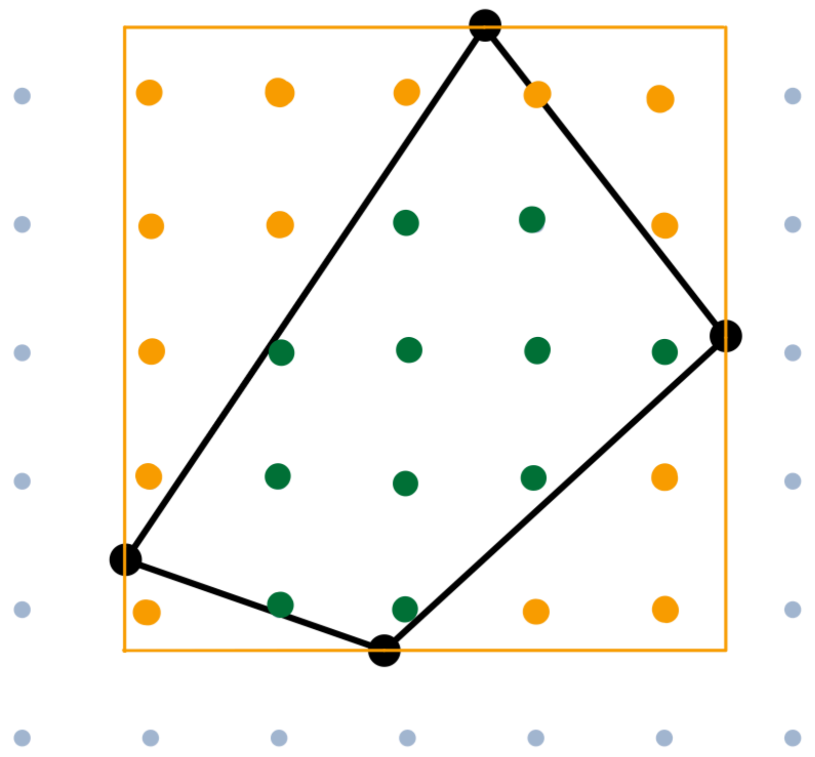
3. Extracting Intersection Records



● Tested & under the projection

● Tested & not under the projection

3. Extracting Intersection Records

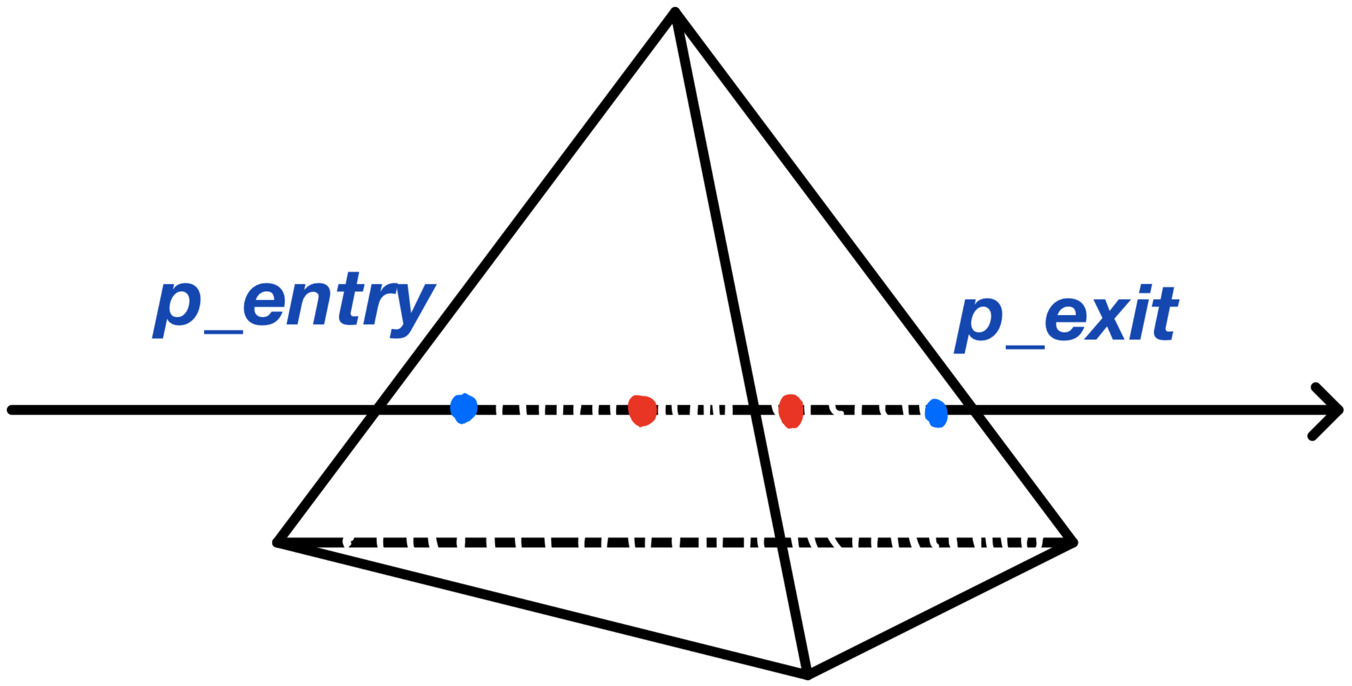


● Tested & under the projection

● Tested & not under projection

4. Calculating Intersection Effects

4. Calculating Intersection Effects



5. Sorting Intersection Effects List

5. Sort Intersection Effects Lists In Ascending Order By Distance

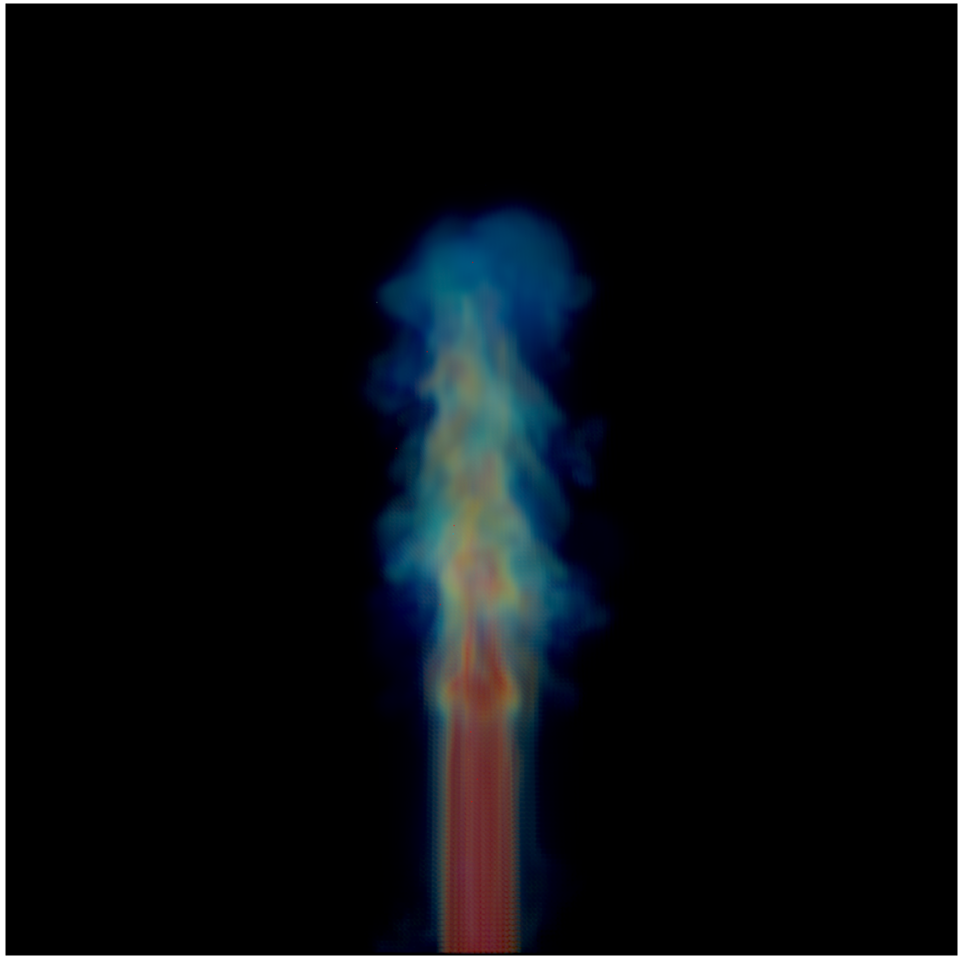
- bool SortFunc(Intersection_effect ef_a, Intersection_effect ef_b)
  - return ef_a.dist < ef_b.dist;

6. Composition

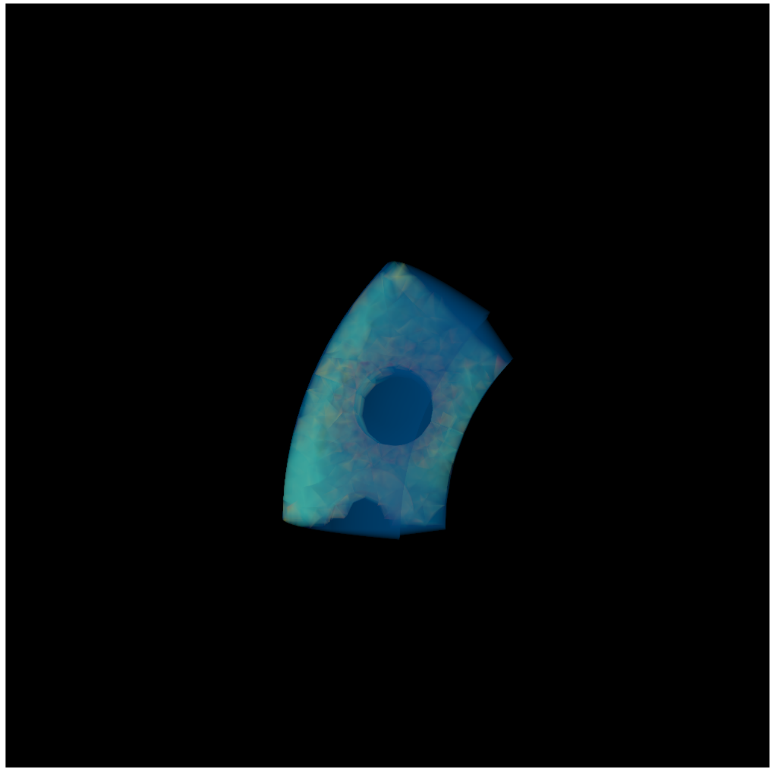
6. Composition

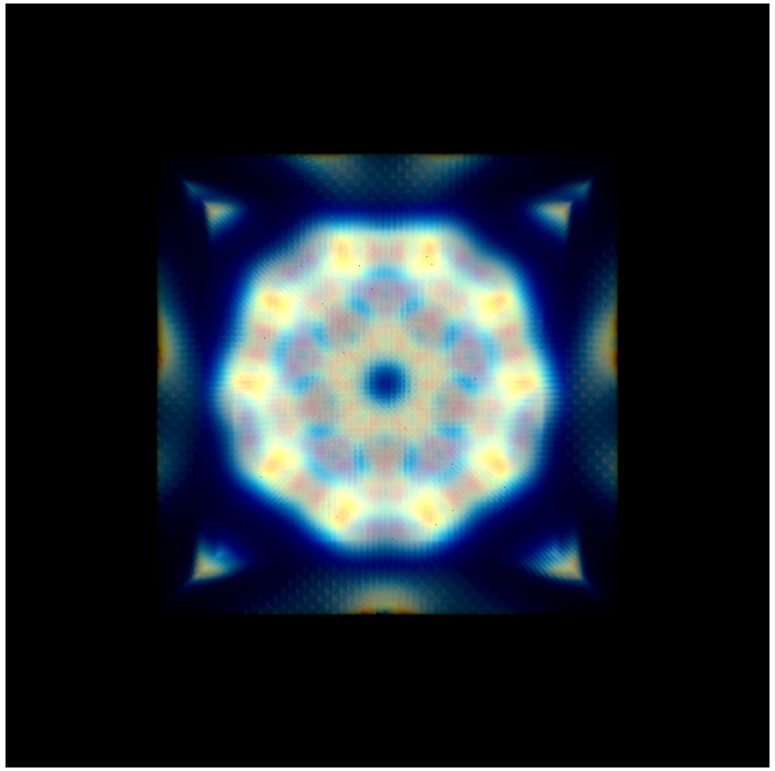
- For each intersection of the pixel
  - c_color += (1 - c_opacity) * r_color;
  - c_opacity += (1 - c_opacity) * r_opacity;
  - If the opacity is too high, composition is stopped.

RESULTS



RESULTS





RESULTS