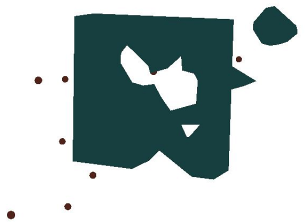
Shattering Glass

Haotian Ma, Haonan Peng, Zixin Xu, Tenny Yin

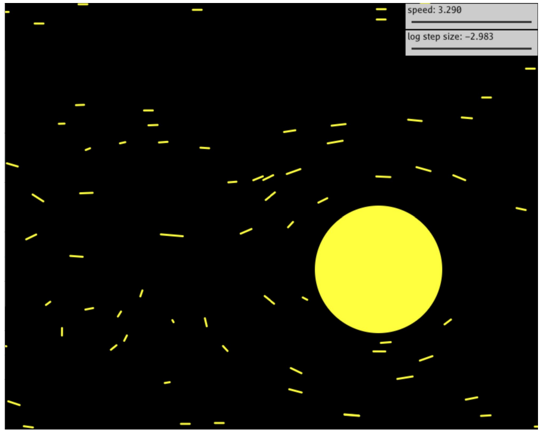


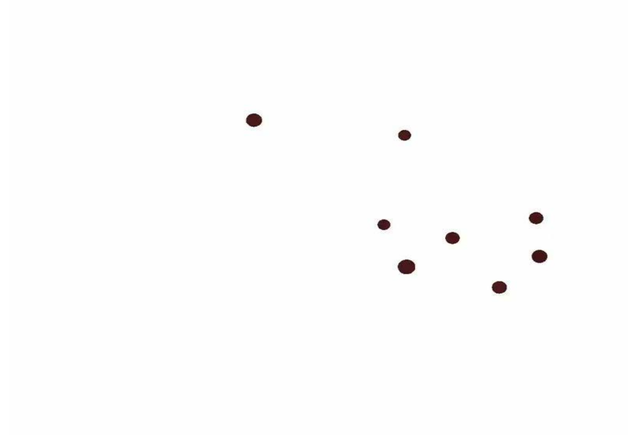
Project Overview

- Voronoi Diagram generate pre-separated mesh for glass shattering
- Generate mesh according to the Voronoi Diagram
- Particles flow through a 3D curl noise field
- 3D Collision Detection of the particle and the glass
- 3D Collision Response of the mesh being hit updating angular and linear velocity
- Influence of the mesh being hit on its surrounding mesh determined by a spring algorithm with threshold/radius of effect
- Update of the affected mesh shattering pieces and particle velocity

3D Curl Noise

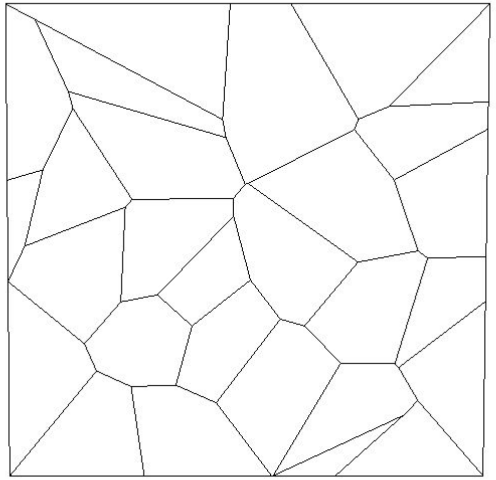
- We referenced the implementation from PA1.
- The 3D Curl Noise field is generated by taking the curl of a 3D potential vector field.
- The 3D potential field is made up of 3 randomly generated numbers through a hash function.





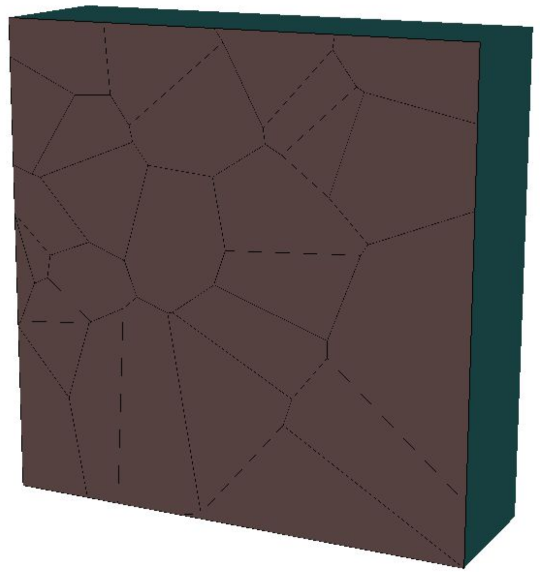
Fracture pattern generation

- Fracture pattern generated with Voronoi
  - Jump flooding algorithm
  - Scipy Voronoi API
- Data structure hierarchy
  - Vertices
  - Triangles
  - Polygon
  - Edges
- Shared vertices vs. Unique vertices



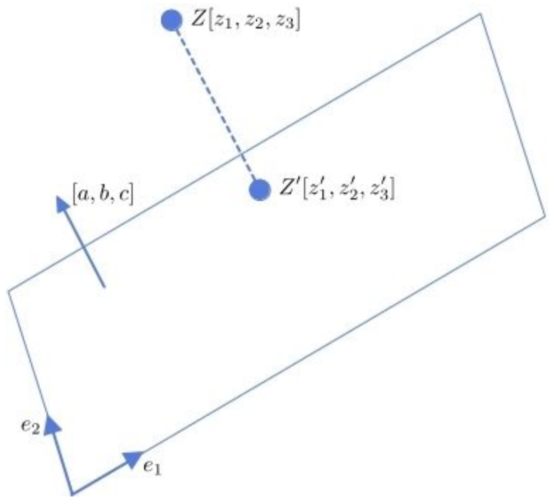
Mesh generation

- 2D -> 3D: thickness
- Meshes and lines
- Storing and updating fractures



3D Collision Detection

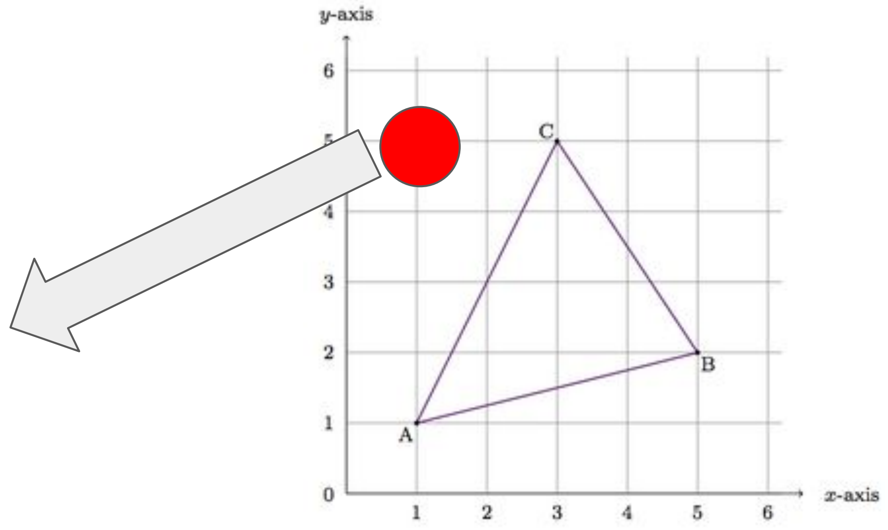
- The 3D position of the incoming particle is projected onto the surface of glass brick for initial comparison.



3D Collision Detection

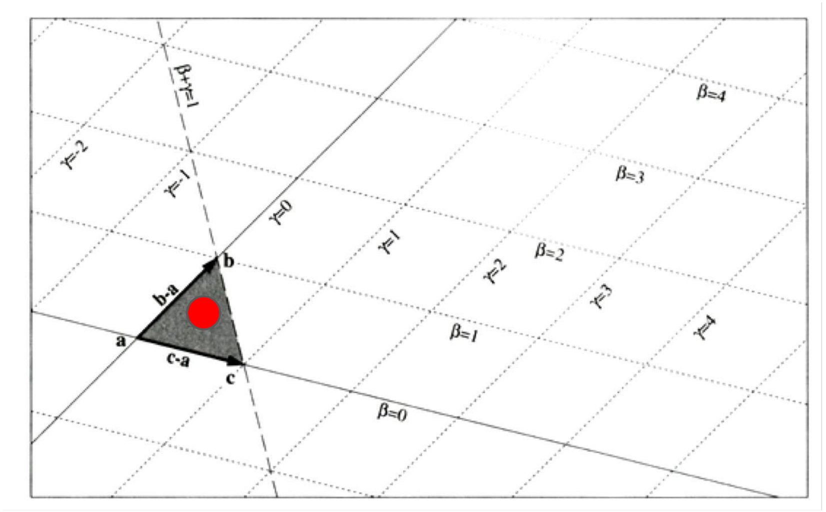
- Barycentric Coordinate Test

The center of the particle is projected onto the plane of the glass brick for comparison.



3D Collision Detection

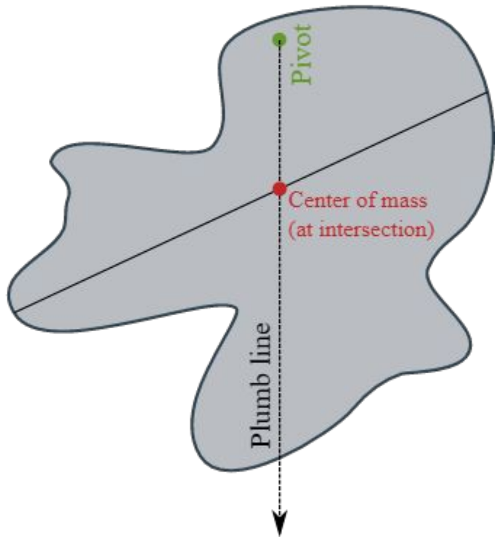
- Barycentric Coordinates Test



$$\beta > 0; \quad \gamma > 0; \quad \beta + \gamma < 1$$

3D Collision Detection

- Find Center of Mass for each segment in the glass brick



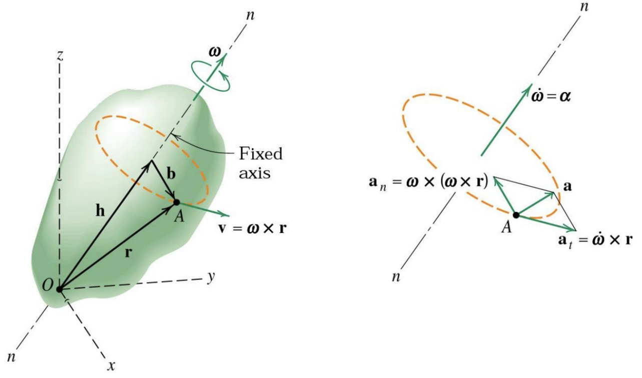
$$\bar{x} = \frac{\sum_{n=1}^{N} m_n x_n}{\sum_{n=1}^{N} m_n}, \quad \bar{y} = \frac{\sum_{n=1}^{N} m_n y_n}{\sum_{n=1}^{N} m_n}$$

3D Collision Response

- Linear Velocity
- Angular Velocity
- Calculating which parts of the voronoi mesh it influences
- Update position of corresponding meshes

Angular Velocity

- Obtain the initial angular velocity vector of the object before the collision.
- Determine the point of impact and the collision normal vector (a vector perpendicular to the collision surface).
- Calculate the impulse vector: This vector represents the change in angular momentum due to the collision. It depends on factors such as the collision force, the mass distribution of the object, and the point of impact.
- Apply the impulse vector: Add the impulse vector to the object's current angular velocity vector to obtain the new angular velocity vector.
- Using Euler Integration to update the Orientation of the Object



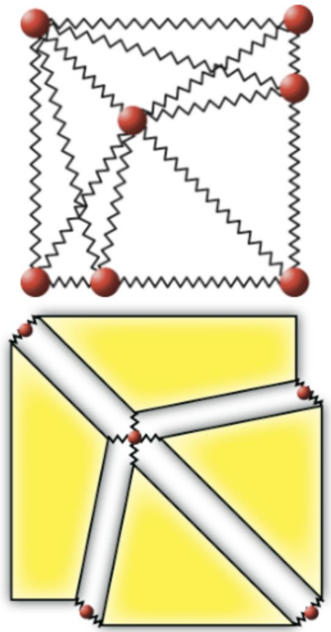
n
ω
z
Fixed axis
b
h
A
$\mathbf{v} = \omega \times \mathbf{r}$
r
O
y
n
x
n
$\dot{\omega} = \alpha$
$\mathbf{a}_n = \omega \times (\omega \times \mathbf{r})$
a
A
$\mathbf{a}_t = \dot{\omega} \times \mathbf{r}$
n

$$\mathbf{j} = \gamma \hat{\mathbf{n}}$$

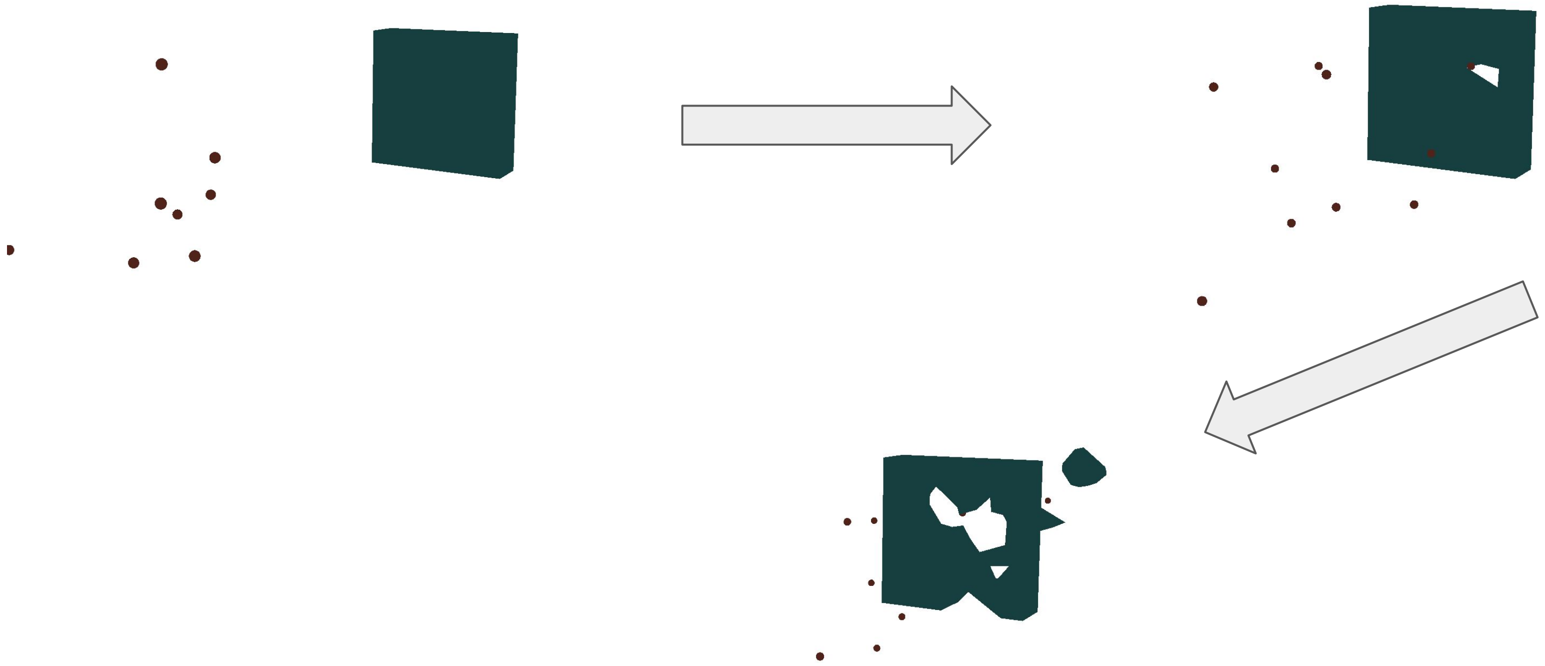
$$\Delta\omega_b = -I_b^{-1}\mathbf{r}_b \times \mathbf{j}$$

Influencing multiple meshes

- Spring between neighboring vertices
- Influencing a certain area depending on the magnitude of impulses
- Decreasing force based on damping and springs



Demo



Reflections on our Methods

- Region of influence of ball on the shattering glass
- Angular velocity and usage of quaternion
- Convex polygon collision detection and response
- Voronoi Diagram and its usage