Design Overview

The simulation simplifies the full 3D cell into a 2D triangular mesh (analogous to a tetrahedral volume in 3D), and vesicles are modeled as moving circular particles with discrete sample points on their membrane.

Key components include:

- `Cell`: generates a 2D hexagonal triangle mesh
- `Triangle`: stores geometry and outward normals for side tests
- `Vesicle`: holds the center, radius, and sample points
- `Vesicles`: manages multiple vesicles and their diffusion
- `Visualizer`: uses Vispy to render the entire system interactively

Method: Overlap Detection via Vectorized Side Test

For each vesicle:

- A number of sample points are placed on its membrane (`n = 8`)
- The algorithm uses vectorized 2D side-tests to check whether any sample lies inside a triangle
- Triangle normals are precomputed; the test computes:

$$\text{dot}(P - V_i, n_i) \leq 0 \quad \forall i = 1,2,3$$

If any point lies inside, the triangle is considered overlapping / occupied.

This structure is:

- Fully vectorized using NumPy broadcasting
- Easily extendable to 3D tetrahedrons (by testing against 4 face planes)
- Ready for GPU acceleration using CuPy or PyTorch if needed

Diffusion Model

Diffusion is modeled as Brownian motion:

$$\Delta x = \sqrt{2D\Delta t} \cdot \xi$$

- `D` = vesicle-specific diffusion coefficient
- `Δt` = timestep
- `ξ` = Gaussian random direction vector

Vesicle movement is rejected if the new position overlaps with an already occupied triangle (simple exclusion rule).