Voronoi Diagrams: Definition and Formal Properties

Voronoi Diagram Basics: A Voronoi diagram is a partitioning of a space (typically a plane) into regions around a set of seed points (also called sites or generators). Each **Voronoi cell** consists of all points closer to one specific seed than to any other seed (Voronoi diagram - Wikipedia) (Voronoi diagram - Wikipedia). Figure 1 shows an example: every colored polygon is the Voronoi cell for the black seed point inside it, containing all locations nearer to that seed than to any other (Voronoi diagram - Wikipedia). The boundaries between cells lie along points that are exactly equidistant to two (or more) seeds (Voronoi diagram - Wikipedia). In the plane with distinct point seeds, each boundary segment is a portion of the perpendicular bisector of the line between two seed points (Voronoi diagram - Wikipedia), and where three or more boundaries meet is a point equidistant to three or more seeds (Voronoi diagram - Wikipedia).

(image) *Figure 1: A Voronoi diagram for a set of points. Each colored cell is the region of the plane closest to one black seed point (Voronoi diagram - Wikipedia).*

Convexity of Cells: In a standard Euclidean Voronoi diagram (with point sites and unweighted distance), every Voronoi cell is a convex polygon (Voronoi diagram - Wikipedia). This is because a cell can be constructed as the intersection of half-spaces: for a given seed, consider the half-space that is closer to that seed than to any particular other seed; the Voronoi cell is the intersection of all such half-spaces for that seed (Voronoi diagram - Wikipedia). The intersection of half-planes yields a convex region, so each cell is convex. In general, any point inside a Voronoi cell can be connected by a straight line segment entirely within that cell to the seed (convexity ensures no “dents” in the region). If seeds are in general position (no special coincidences), Voronoi cells in the plane are polygons (possibly unbounded), and if one seed lies on the convex hull of all seeds, its cell will extend to infinity (be unbounded) (). Conversely, interior seeds (not on the outer convex hull) have bounded polygonal cells.

Cell Adjacency and Dual Graph: If two Voronoi cells share a boundary edge, their seed points are **adjacent** in the sense of being nearest neighbors. Such a shared edge is comprised of all points equidistant to those two seeds and closer to them than to any other (Voronoi diagram - Wikipedia). An important property is that the Voronoi diagram’s adjacency relationships are the **dual** of the Delaunay triangulation for the same set of points (Delaunay triangulation - Wikipedia). In fact, connecting each pair of seed points whose cells share an edge produces the Delaunay triangulation (Delaunay triangulation - Wikipedia). Equivalently, if two points are connected by a Delaunay edge, their Voronoi cells touch along a border. Thus, the graph formed by connecting neighboring seeds is planar and is the exact dual graph of the Voronoi tessellation (Delaunay triangulation - Wikipedia). This duality provides a powerful way to analyze adjacency: for instance, each vertex where three Voronoi cells meet corresponds to the circumcenter of a triangle formed by the three associated seeds in the Delaunay triangulation

(Delaunay triangulation - Wikipedia). Because of this, the vertices of the Voronoi diagram are points equidistant to three (in general position) seeds, and the edges of the Voronoi diagram are perpendicular bisectors of Delaunay edges.

Centroidal Voronoi Tessellations: A notable optimization of a Voronoi diagram is the **centroidal Voronoi tessellation (CVT)**. In a CVT, each seed is located at the **centroid** (geometric center of mass) of its own Voronoi cell (Centroidal Voronoi tessellation - Wikipedia). This configuration often represents an optimal partition of the space – for example, it minimizes the variance of the distance from points in each region to the seed (useful for clustering or quantization). One can obtain a centroidal Voronoi tessellation by iteratively adjusting the seeds: compute the Voronoi diagram for an initial set of seeds, then move each seed to the centroid of its cell, and repeat (this process is known as **Lloyd's algorithm**) (Voronoi diagram - Wikipedia) (Centroidal Voronoi tessellation - Wikipedia). Over successive iterations, the cells become more regular and uniform, and the seeds converge to cell centroids, yielding a CVT (Voronoi diagram - Wikipedia). Centroidal Voronoi diagrams have special properties; for instance, in the plane the optimal CVT for many points tends toward all cells being congruent regular hexagons (a hexagonal grid is the most area-efficient partition into equal cells) (Centroidal Voronoi tessellation - Wikipedia). CVTs are used in optimal spatial sampling, meshing, and even generative art, since they produce an even, balanced tessellation.

Computational Construction of Voronoi Diagrams

Constructing a Voronoi diagram from a set of points can be done by several algorithms. The challenge is to determine the correct cell boundaries efficiently for all seeds. Key methods include:

- **Brute-Force / Naïve Methods:** The simplest conceptual approach is to determine each cell by **distance comparisons**. For every location in space (or for a fine grid of sample points), one can compute distances to all seeds and find the nearest seed – effectively coloring space by nearest neighbor. This **distance transform** approach is straightforward but computationally expensive. Similarly, one can construct each cell by intersecting all the half-planes that define the region closer to one seed than another. For *n* seeds, each cell is the intersection of *n–1* half-planes (Voronoi diagram - Wikipedia). A naive implementation would consider *O(n)* boundaries per cell, yielding *O(n^2)* comparisons overall. Indeed, a direct algorithm might run in *O(n^2)* or worse; for instance, a naive algorithm can be *O(n^2 \log n)* if each of the *n* cells is computed by intersecting *n-1* bisector lines (). While too slow for large *n*, brute-force methods are useful for conceptual understanding and are sometimes used for generating Voronoi patterns in pixel-based images (where GPU techniques like the Jump Flooding Algorithm can approximate a Voronoi diagram in near-constant time by propagating distance information across an image grid (Voronoi diagram - Wikipedia) (Voronoi diagram -

Wikipedia)).

- **Fortune's Algorithm (Sweep-Line):** Steven Fortune's algorithm is a classic efficient method to build a Voronoi diagram in the plane in $O(n \log n)$ time (Voronoi diagram - Wikipedia). It uses a **sweep-line** technique: imagine moving a line (typically from top to bottom) across the plane, and "building" the Voronoi diagram as the line encounters each new seed point. The algorithm maintains a **beach line** – an ever-changing curve composed of pieces of parabolas – which represents the frontier of the Voronoi regions that have been finalized so far (Fortune's algorithm - Wikipedia). As the sweep-line moves, new seeds are added (site events), which cause new parabola arcs to appear on the beach line, and circles (circle events) cause existing arcs to disappear when Voronoi vertices are completed (Fortune's algorithm - Wikipedia). The beach line's intersections trace out the edges of the Voronoi cells as the algorithm progresses (Fortune's algorithm - Wikipedia). Fortune's algorithm cleverly handles these events in a priority queue (ordered by the sweep-line position) so that each event is processed in logarithmic time, for a total of $O(n \log n)$. The result is the full set of Voronoi edges and vertices once the sweep is complete. Fortune's method is significant because it was simpler than earlier divide-and-conquer algorithms yet maintained optimal runtime (). It is widely implemented in computational geometry libraries to compute Voronoi diagrams directly from points.

- **Delaunay Triangulation (Dual Graph Method):** Another common approach is to compute the **Delaunay triangulation** of the point set first, then derive the Voronoi diagram from it. The Delaunay triangulation is the planar graph connecting the points such that no point lies inside the circumcircle of any triangle (it maximizes the minimum angle of triangles, avoiding skinny triangles) (Delaunay triangulation - Wikipedia). The Delaunay graph is inherently the **dual** of the Voronoi diagram (Delaunay triangulation - Wikipedia): each Delaunay triangle's circumscribed circle corresponds to a vertex where three Voronoi cells meet, and each edge of the Voronoi diagram is perpendicular to a Delaunay edge connecting two neighboring seeds (Delaunay triangulation - Wikipedia). To construct the Voronoi diagram via Delaunay, one can: (1) find the Delaunay triangulation (using algorithms like Bowyer–Watson insertion, divide-and-conquer, or sweepline, typically $O(n \log n)$ on average (Voronoi diagram - Wikipedia)), and (2) for each Delaunay triangle, compute its circumcenter. These circumcenters become the Voronoi vertices, and connecting circumcenters of adjacent triangles (that share an edge) gives the Voronoi edges (Delaunay triangulation - Wikipedia). In practice, the Delaunay-to-Voronoi approach is convenient: many libraries can compute Delaunay triangulations, and the connectivity directly tells which seeds' cells are adjacent. The **Bowyer–Watson algorithm**, for example, incrementally adds points and re-triangulates, achieving about $O(n \log n)$ in 2D on average (Voronoi diagram - Wikipedia). Once you have the triangulation, outputting the Voronoi edges is linear in n. This indirect method is also easy to extend to higher dimensions (where Voronoi cells are polyhedra and Delaunay forms simplices).

Key Characteristics of Voronoi Cells: In summary, Voronoi cells in Euclidean space are convex polygons (or polyhedra in 3D) (Voronoi diagram - Wikipedia). They **tile** the space without overlap, completely covering the domain (each point in the plane belongs to exactly one cell or on a boundary). Adjacent cells share a straight-line border segment (or a polygonal face in 3D) which lies at equal distance to the two corresponding seeds (Voronoi diagram - Wikipedia). The network of cells is connected and in planar diagrams satisfies Euler's formula for planar graphs (if there are *n* seeds, and assuming general position, the Voronoi diagram will have about *2n – 5 – h* vertices and *3n – 6 – h* edges, where *h* is the number of seeds on the convex hull) – roughly on the order of *O(n)* edges and vertices for *n* seeds () (). Voronoi diagrams also have the **nearest-neighbor property**: if you take any point in a cell and move slightly, it remains closest to the same seed until you cross a boundary. This makes Voronoi diagrams useful for nearest-neighbor search, natural neighbor interpolation, and many other geometric applications.

Voronoi in Parametric Design and Pattern Generation

Voronoi patterns are not just theoretical; they have become a popular motif in **parametric design**, architecture, and generative art due to their natural, cell-like aesthetic and space-filling properties. Designers use Voronoi diagrams to create organic-looking facades, partitions, tiling patterns, and structural grids. In particular, **Grasshopper** (the visual programming plugin for Rhino 3D) offers straightforward tools to generate and manipulate Voronoi diagrams for design purposes.

Grasshopper Workflow: Generating and Modifying Voronoi Patterns

Using Grasshopper, one can interactively create a Voronoi diagram and experiment with its layout. A typical **step-by-step Grasshopper process** is:

1. **Define a Boundary:** Decide the region where the Voronoi cells should exist. This could be a rectangle (for a 2D pattern), a closed curve outline, or even a surface (for paneling a façade). The boundary will constrain the Voronoi diagram so that cells are cut off at the edges of the desired shape.

2. **Generate Seed Points:** Provide a set of points within that boundary to act as Voronoi seeds. Grasshopper can generate points randomly or in a controlled way (e.g. using the **Populate 2D** component to scatter a given number of random points in a region ([Tutorial] Voronoi Gradient Pattern Facade - Grasshopper)). You can also use any point distribution (grid points, attractor-driven points, etc.) depending on the pattern needed.

3. **Create the Voronoi Diagram:** Grasshopper has a **Voronoi** component (in 2D, and a **Voronoi 3D** for volumetric patterns) that takes the boundary curve and the seed points as inputs. This component outputs the line segments or curves delineating each Voronoi cell within the boundary. The result is usually a set of closed polylines (one for each cell) trimmed to the boundary shape.

4. **Explode and Refine Cells:** Once the Voronoi cells (polylines) are generated, you can **explode** them to get individual cell boundaries for further processing. At this stage, designers often scale or offset cells, extrude them (if making 3D panels), or subdivide them further. Each cell's polygon can be treated as a panel for fabrication or as an aperture, etc.

5. **Modify with Attractors (Optional):** A powerful technique in parametric design is using **attractors** to customize the Voronoi pattern. An attractor can be a point, curve, or surface that influences the distribution or size of Voronoi cells. For example, you might weight the density of seed points such that more points are clustered near an attractor (creating smaller, denser cells in that area) and sparser farther away (larger cells). In Grasshopper, one can achieve this by moving existing seed points or culling/adding points based on their distance to attractor geometry. Another approach is to post-process the cells: for instance, use the distance to an attractor to **scale each cell's outline** or to selectively remove cells. A *curve attractor* example might pull nearby Voronoi cell edges inward or outward, creating a gradient effect in cell shape across a facade ([Tutorial] Voronoi Gradient Pattern Facade - Grasshopper). The **Voronoi Gradient Pattern Facade** tutorial, for instance, uses a curve attractor to adjust cell openings gradually across a surface ([Tutorial] Voronoi Gradient Pattern Facade - Grasshopper).

6. **Apply Bounding Constraints:** Ensure the Voronoi pattern cleanly fits the design domain. This often means trimming cells at the edges (which the Voronoi component does with the boundary), but it could also involve constraints like holes or interior boundaries (e.g., avoid placing seeds in certain areas). In Grasshopper, one can supply multiple closed curves as boundaries (to create holes) or intersect the Voronoi output with specific shapes to clip it. The result is a pattern bounded exactly by the design's silhouette or panel shape.

7. **Recursive Subdivision (Optional Advanced Step):** For more complex patterns, designers sometimes create **nested Voronoi** patterns. This means taking one cell (or a set of cells) and subdividing it again by generating new seed points inside it and computing a *Voronoi within that cell*. The effect is a hierarchy of cells – larger cells containing smaller sub-cells. Grasshopper does not have a built-in one-click component for recursive Voronoi, but this can be achieved through scripting or iterative definitions (e.g., using loops or the **Voronoi Groups** add-on, which is noted to produce nested 2D Voronoi subdivisions (Recursive 3D Voronoi Diagram - Grasshopper)). A workflow could be: pick a cell, scatter new interior points, intersect the resulting Voronoi diagram with

the cell's boundary. Repeating this process yields fractal-like patterns or multi-scale cell networks. This is useful for mimicking natural patterns like crack propagation or leaf venation, and allows multi-level detail in designs.

8. **Output and Fabrication:** Finally, the Voronoi geometry can be baked into Rhino for further refinement or exported for fabrication. Voronoi cells can be used as cut-out panels, structural frame layouts, floor plans, etc. The parametric process means that by adjusting the seed points or attractors, one can regenerate a new pattern quickly, exploring design alternatives with ease.

Throughout this process, Grasshopper's visual feedback makes it easy to tweak the pattern – move a seed point and see the cells update, change the number of points and get a whole new tiling, or adjust an attractor's influence to fine-tune the gradient of cell sizes. These capabilities make Voronoi diagrams a favorite in computational design workflows.

Applications in Architecture and Generative Design

Voronoi-based designs appear in many architectural and design contexts, from building facades to interior partitions and even furniture. Key characteristics – like the organic, irregular cell shapes and the ability to scale to any boundary – give designers a rich aesthetic and functional vocabulary. Here are some examples and use-cases:

- **Architectural Facades and Screens:** Voronoi patterns are often used to create eye-catching facades. The cells can function as windows, openings, or simply surface articulation. Designers might convert certain Voronoi cells into glazed openings while others remain solid, producing a random window pattern that still follows a logical partitioning. *For instance, a Grasshopper definition might generate a parametric Voronoi facade and then designate a subset of cells as windows (openings) based on some criteria* (Voronoi Facade - Grasshopper). This approach was used in a facade project where larger Voronoi cells were selectively removed to create window openings, blending structure and fenestration into one irregular pattern (Voronoi Facade - Grasshopper). Another notable example is the **Torre de Especialidades** hospital in Mexico City, whose exterior features a Voronoi-like tiling of geometric panels that double as an air-filtration skin. Each panel is a cell in a giant Voronoi diagram covering the building, giving a visually striking and functionally performative facade (the pattern was chosen for both aesthetics and effective distribution of air-purifying coating on the panels).

- **"Water Cube" – Structural Voronoi 3D Pattern:** One of the most famous architectural applications is Beijing's National Aquatics Center (the *Water Cube*), which showcases a 3D Voronoi-like structure. The exterior and interior walls are formed by a translucent cellular matrix that looks random but is based on an optimized space-filling polyhedral pattern (The Fascinating World of Voronoi Diagrams | Built In). The design was inspired

by the natural formation of soap bubbles, essentially a Voronoi foam in three dimensions (National Aquatics Center (Water Cube) - Arup). The resulting steel space frame and ETFE pillow panels create an efficient, seismically sound structure that glows like a cluster of bubbles (see **Figure 2**) (National Aquatics Center (Water Cube) - Arup). The Voronoi analogy was deliberate – the cells' organic arrangement is visually appealing and mimics nature's way of efficiently partitioning space. At night, when lit in blue, the bubble-like pattern becomes even more pronounced, highlighting how Voronoi geometry can produce both a structural solution and a compelling visual identity for a building (The Fascinating World of Voronoi Diagrams | Built In).

(National Aquatics Center (Water Cube) - Arup) *Figure 2: The **Water Cube** in Beijing. Its walls and roof are composed of a 3D Voronoi (foam) pattern of cells, inspired by soap bubbles (National Aquatics Center (Water Cube) - Arup). The Voronoi diagram analogy was chosen for its efficient, organic structure and visual effect (The Fascinating World of Voronoi Diagrams | Built In).*

- **Tiling and Panelization:** Voronoi diagrams naturally tile a plane (or surface) without gaps, so they are used for tiling treatments and panel layouts. On a flat facade or pavement, a Voronoi tiling can create an irregular mosaic of panels or pavers. On curved surfaces, designers can morph a 2D Voronoi pattern onto the surface (using Grasshopper's Surface Morph or similar, which maps geometry from a flat reference to a target surface ([Tutorial] Voronoi Gradient Pattern Facade - Grasshopper)). This results in a panelized freeform surface where each panel is a curved Voronoi cell. Such techniques have been applied in pavilion designs and can be seen in bespoke ceiling installations, where Voronoi cells become perforated panels allowing light through in a pattern that feels like leaf canopy or cracked earth. Panelization with Voronoi is often combined with practical constraints: for example, using a limited set of cell sizes or ensuring each cell face is planar for constructability.

- **Generative Design and Art:** Because Voronoi patterns can be controlled with parametric inputs, they are a staple in generative art/design. By moving seed points or adding random perturbations, one can create endless variations of a pattern. Some designers use **dynamic attractors** (points that move or weights that change over time) to animate Voronoi patterns for interactive installations. Others use **recursive Voronoi subdivisions** to create fractal-like artwork – for instance, starting with a few large cells and then subdividing certain cells iteratively to produce many levels of detail. The pattern that emerges can resemble natural phenomena (like the crackle glaze on certain ceramics from the Song dynasty, which is essentially a Voronoi crack pattern – an effect admired historically for its randomness and uniqueness (The Fascinating World of Voronoi Diagrams | Built In)). Modern generative designs often incorporate Voronoi algorithms to optimize structures as well – e.g., in lightweighting strategies, a solid volume can be "hollowed out" in a Voronoi lattice fashion to reduce material while maintaining strength, resulting in an organic trusswork. This approach is seen in some 3D-printed furniture or automotive components where a Voronoi-like internal structure

provides an ideal trade-off between weight and rigidity.

- **Customization and Control:** A key advantage of using Voronoi in design is the high degree of control designers have. By adjusting input parameters (point locations, attractor fields, etc.), one can fine-tune cell sizes and adjacencies. For example, using multiple attractor points with varying influence can create a pattern that transitions smoothly from small, dense cells in one region to large, open cells in another. Designers sometimes incorporate programmatic needs into Voronoi patterns – imagine a floor plan where rooms are regions in a Voronoi diagram around key functional centers, or a landscape design where different seed points correspond to trees or lighting poles, and their Voronoi cells define influence zones or paving areas. The **centroidal Voronoi optimization** mentioned earlier can also be applied in design: by relaxing the point distribution (using a tool or script to perform Lloyd's algorithm), the cells become more uniform and hexagon-like. This is useful if a more regular pattern is desired for structural or aesthetic reasons, while still retaining a degree of randomness at the boundaries of a site or form.

In all these applications, Voronoi diagrams bring a blend of **mathematical rigor and organic randomness**. They ensure efficient use of space (each cell claims its region optimally) and every piece of the pattern has a logical relation to a seed (useful if that seed has meaning, like a building module or a feature to emphasize). Yet, the overall pattern avoids obvious repetition, giving a sense of natural complexity. Computational tools like Grasshopper have made it straightforward to harness these qualities – designers can quickly generate a Voronoi layout, visualize it, and iterate, possibly even linking it with environmental or structural analysis. The result is that Voronoi diagrams have become a go-to strategy in parametric architecture for achieving patterns and forms that are **functional, efficient, and visually dynamic**, from building skins and partitions to decorative motifs and beyond.

Sources:

1. Wikipedia: *Voronoi diagram* – Definition and properties of Voronoi diagrams (Voronoi diagram - Wikipedia) (Voronoi diagram - Wikipedia); convexity and construction via half-spaces (Voronoi diagram - Wikipedia); formal definition (Voronoi diagram - Wikipedia); duality with Delaunay triangulation (Delaunay triangulation - Wikipedia); Fortune's algorithm and Bowyer–Watson complexity (Voronoi diagram - Wikipedia); Lloyd's algorithm and centroidal Voronoi tessellation (Voronoi diagram - Wikipedia).
2. Wikipedia: *Centroidal Voronoi tessellation* – CVT defined as Voronoi where each seed is at the cell's centroid (Centroidal Voronoi tessellation - Wikipedia).
3. Mount, D. (2023). *CMSC 754 Lecture Notes* – Voronoi properties and algorithms. Naive construction *O(n^2 log n)* by half-plane intersection (); Voronoi edges as perpendicular bisectors; relationship to convex hull (unbounded cells) (); history of Voronoi algorithms ().

4. Wikipedia: *Delaunay triangulation* – Relationship between Delaunay and Voronoi: dual graphs, circumcenters of triangles are Voronoi vertices (Delaunay triangulation - Wikipedia).
5. Grasshopper3d (Parametric House) – Examples of Grasshopper Voronoi workflows. Voronoi facade with point attractors (Voronoi Facade - Grasshopper); Voronoi facade example turning cells into windows (Voronoi Facade - Grasshopper). Grasshopper tutorial for Voronoi gradient facade with attractor curve ([Tutorial] Voronoi Gradient Pattern Facade - Grasshopper).
6. BuiltIn.com – "The Fascinating World of Voronoi Diagrams" – Discusses Voronoi patterns in nature and architecture. Notes the Water Cube facade uses a Voronoi (bubble) pattern for its walls (The Fascinating World of Voronoi Diagrams | Built In).
7. Arup Project Page – *National Aquatics Center (Water Cube)* – Describes the bubble-inspired Voronoi-like geometry of the structure, combining repetition with organic randomness (National Aquatics Center (Water Cube) - Arup). This illustrates a real-world application of 3D Voronoi principles in an architectural icon.