ChatGPT

Classification of 2D Tiling Patterns

Tiling patterns can be broadly categorized by symmetry, tile types, and adjacency rules. **Regular tilings** (edge-to-edge, one polygon type) include only the three classic tessellations of equilateral triangles, squares, and hexagons. **Uniform (Archimedean) tilings** have one vertex type but may use two or more regular polygons, while more generally a **k-uniform (demiregular)** tiling has k distinct vertex types [1] [2]. **Isohedral (tile-transitive)** tilings use congruent tiles and are transitive on tiles; these have been classified (Grünbaum & Shephard proved exactly 81 combinatorial types [3]). **Non-edge-to-edge** tilings allow tiles to meet without full edge alignment (e.g. the Cairo pentagonal tiling); these are often treated with extended vertex symbols or by splitting edges. **Aperiodic/substitution** tilings (Penrose, Ammann–Beenker, the "hat" monotile, etc.) lack translational symmetry and are generated by inflation (substitution) rules or projection methods. **Islamic geometric patterns (girih)** use star polygons and strapwork; they are typically constructed by compass-and-straightedge or parametric shape-grammar rules [4]. **Wang tiles** (colored-edge squares) define local matching constraints and can enforce aperiodicity; they form a formal system of tilings [5]. Finally, **polyform** or **puzzle tilings** (polyominoes, polyhexes, etc.) are classified by tile shape and tilability (often solved by exact-cover algorithms). Table 1 summarizes key categories with their notation conventions and generation approaches.

Category	Description / Notation	Generation Methods / Systems
Regular / Uniform Tilings	Edge-to-edge tilings by regular polygons. Vertex configuration given by Schläfli/Cundy–Rollett symbols (e.g. 3^6, $3^4{\cdot}6$) [1]. Orbifold or wallpaper-group notation can classify symmetries. GomJau-Hogg notation (Antwerp app) encodes uniform tilings algorithmically [6] [7].	Enumerated by symmetry (e.g. 11 Archimedean tilings). Software tools like **Antwerp** (GomJau-Hogg) generate tilings from notation [7]. Plantri (Brinkmann–McKay) can enumerate planar tilings by convex faces [8].
k-Uniform Tilings	Multi-vertex uniform tilings (k>1). Extended vertex-symbol notation (e.g. "$3^2{\cdot}4{\cdot}3{\cdot}4$; $3{\cdot}4{\cdot}6{\cdot}4$" for a 2-uniform tiling). GomJau-Hogg notation uniquely names k-uniform tilings [6] [2].	Same methods as above. GJ-H notation (Antwerp v3.0) automatically generates k-uniform tilings up to k=3 [2]. Catalogs exist for known k=1,2,3 tilings.
Isohedral Tilings	Monohedral tilings with transitive symmetry on tiles. Classified by an **adjacency symbol** or ISO-n ($1 \leq n \leq 93$) indices [3]. Conway's "crankshaft" or Delaney-Dress graph can represent these combinatorially [9].	Exhaustive enumeration by combinatorial methods (Delgado-Friedrichs/Huson) or by Plantri enumeration of face-coloured graphs [8]. Known catalogs (81 types) from Grünbaum–Shephard [3].

Category	Description / Notation	Generation Methods / Systems
Aperiodic / Substitution	Tilings defined by inflation (substitution) rules or cut-and-project schemes (e.g. Penrose, Ammann–Beenker, Lançon–Billard). Often described by prototile labels and subdivision rules. Combinatorial “address” or coordinate systems can encode positions [10] [11] .	Generated by recursive subdivision (finite tile-subdivision rules). Tools: *Escher* (Bowers–Lawson) uses Python/KoebePy to apply subdivision rules [12] [13] . Simon Tatham’s Sage code generates substitution tilings and finite-state transducers for Penrose, Spectre, Hat tilings [10] [14] . De Bruijn multigrid algorithm (and libraries like **pynrose** for P3 tilings) use projection methods [11] .
Edge-constraint (Wang) Tiles	Sets of unit-square tiles with colored edges. Tiling rules: abutting edges must match colors [5] . Special sets enforce aperiodicity (Berger, 1966). Notation is simply tile-edge color sequences.	Stochastic or constraint-solving algorithms to place Wang tiles (see UCI algorithm [15]). The domino (Wang-tile) problem is undecidable in general. Techniques for texture synthesis (e.g. in graphics) use Wang tilings [15] .
Islamic (Girih) Patterns	Star-polygon and strapwork designs (often 5-, 6-, 8-, 10-, 12-fold symmetries). Notation typically via star-figure parameters or “girdle” lattice rules. Often related to **girih tiles** (decagon, pentagon, bowtie, etc.) with lines inscribed.	Constructed by geometric algorithms: intersecting circle/line grids or shape grammars [4] . Recent papers use parametric circle/line methods to generate star patterns. No widely adopted compact notation, but rule-based tools (e.g. parametric CAD scripts) support design.
Polyform Puzzles	Tilings by polyominoes/polyhexes/ etc., often used in puzzles. Notation: shape names (e.g. tetromino) and adjacency (e.g. exact cover matrices).	Solved by backtracking/exact-cover algorithms (e.g. DLX for pentomino tilings). Libraries for polyform enumeration exist (e.g. polyform generator code). Visualization as grid colorings.
Voronoi / Random Tilings	Tilings derived from point sets (Voronoi) or stochastic rules. Notation is implicit (e.g. point coordinates).	Generated by computing Delaunay triangulation/Voronoi diagram of a point lattice or random seeds (common in simulations). Lloyd’s algorithm can regularize random tilings.

(Table entries combine several related concepts; references indicate representative sources or methods for each category.)

Notations and Representations

- **Schläfli and Vertex Symbols**: Regular tilings use Schläfli symbols {p,q} and vertex configurations (e.g. *3⁶* for hexagonal tiling) [1] . Cundy–Rollett’s notation lists the polygons around each vertex in order [1] . However, this can be ambiguous for k-uniform patterns.
- **GomJau–Hogg Notation**: An extension of vertex symbols to encode k-uniform tilings unambiguously [6] [7] . It consists of a “polygon placement” sequence (e.g. *6-3-3*) and

transformations (rotations *r*, reflections *m* with indices) that replicate the prototile. For example, "4-3,3,3-4,3/r(c2)/r(h13)/r(h45)" is a GomJau–Hogg code for a specific 3-uniform tiling. Antwerp (v3.0) is a web tool that interprets this code to draw the tiling [7] [16].

- **Orbifold / Wallpaper Notation**: Conway's orbifold symbols (e.g. *p6m,* 442) encode the symmetry group of a periodic tiling. These classify wallpaper symmetries but do not by themselves specify tile shapes.
- **Delaney–Dress Symbols**: A complete combinatorial encoding of a tiling's topology and symmetry [17]. The tiling is subdivided into triangular "chambers" (flags), and neighbor maps s_0, s_1, s_2 record adjacent chambers [18]. Equivalently, one can use a labeled **Delaney graph**: nodes for chamber orbits, edges for adjacency, with labels for tile/vertex cycles [9]. Conway's "crankshaft" notation arranges these adjacency maps in a readable table [19]. In code, Huson & Delgado-Friedrichs use a canonical numeric string to represent the symbol [20]. This framework enumerates all periodic tilings (e.g. Huson generated billions up to given complexity). (Though powerful, Delaney symbols are complex to implement directly.)
- **Substitution / L-System Rules**: Aperiodic tilings are often given by **inflation** or **substitution rules** (each prototile is dissected into smaller copies). These can be encoded in a small grammar (e.g. "tile A → tiles B+C; B → A+B; …"). Alternative "combinatorial coordinate" strings can address tile positions without explicit geometry [10]. Software may store the prototile geometry and a list of subdivided tiles with references to parent vertices [12].
- **Colored-Edge (Wang) Notation**: Wang tiles are described by listing each tile's four edge colors (e.g. *(R,G,B,Y)*). A tiling exists if a matching-edge graph can be built. For isohedral (face-transitive) Wang sets, one can enumerate by state graphs of edge-matching, but in general the problem is undecidable [5].

Generation Algorithms and Software

Practical tiling generation often uses either **combinatorial rules** (substitutions, graph grammars) or **geometric constructions** (projections, packings). Key examples include:

- **Uniform/Periodic Tilings**: Tools like **Antwerp** (Web app) implement GomJau–Hogg rules to draw regular and uniform tilings [16] [7]. The **Plantri** program generates all planar graphs of a given type (triangulations, quadrangulations, etc.) isomorphism-free [8]; its dual graphs correspond to tilings (e.g. Catalan solids or isohedral tilings). These methods can systematically enumerate tilings by constraints (fixed vertex types or transitivity).
- **Subdivision (Inflation) Tilings**: The *Escher* engine (Bowers & Lawson, 2024) provides a Python library and simple text language for hierarchical tilings [12] [13]. One defines prototiles by vertex names, specifies which edges are split and where new vertices lie, and assigns which smaller tiles fill each face. Escher uses a **DCEL** (doubly-connected edge list) data structure to store the subdivision and to compute neighborhoods efficiently [13]. By iterating the rule, one obtains arbitrarily large patches. KoebePy is an open-source geometry library that includes Escher and can render the tilings via circle-packing or Tutte embedding. Conformal-tiling software (e.g. by Bowers–Stephenson) can also produce "true" Euclidean drawings of subdivision tilings using circle packings [13].
- **Aperiodic and Transducer Methods**: Simon Tatham's "combinatorial coordinate" algorithm generates random patches of known aperiodic tilings (Penrose P2/P3, hat, Spectre). His Sage-based code builds finite-state **transducer** automata from a substitution rule, then walks through coordinate strings to place tiles [10] [14]. The code (available on Git) can compute the necessary state machines for these tilings and then output tile coordinates or vector graphics [14] [21]. This approach

handles very large or even infinite tilings in a disciplined way (the heavy lifting is building the transducer, after which tiling generation is linear in output size [21]).

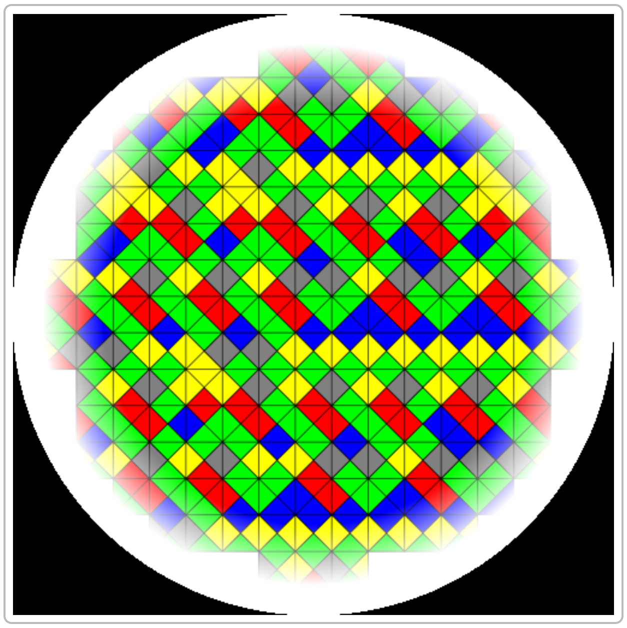
- **Penrose De Bruijn Line Algorithm**: The well-known method for P3 rhombus tiling uses five families of parallel lines spaced at 36° intervals [11]. This "multigrid" construction is implemented in packages like **pynrose** (Python) or other Penrose libraries. It procedurally finds all intersection points of the five grids, then selects points lying inside a decagon to form rhombi. Many tiling programs use this or the equivalent Robinson triangle method.
- **Wang Tile Synthesis**: Simple stochastic algorithms can also non-periodically fill a region with a given set of Wang tiles [15]. In graphics and games, "Wang tiling" is used to generate non-repeating textures by random placement with local color-matching rules. More rigorous methods solve the edge-color constraints via backtracking or constraint solvers (though general decidability is impossible [5]). Specialized code (e.g. Gupta & Papaodysseus, 2019) can tile plane regions with small Wang sets for textures or procedural art.
- **Voronoi / Lattice Tilings**: A periodic point lattice produces a space-filling honeycomb via its Voronoi diagram. In 2D, this yields regular tilings (square lattice → squares, hexagonal lattice → hexagons). In 3D, any lattice's Voronoi cells form a **convex uniform honeycomb** [22] (e.g. the cubic honeycomb from the cubic lattice). Software libraries (CGAL, Qhull) compute Voronoi diagrams in higher dimensions. Random point sets give *Laguerre* or centroidal Voronoi tessellations, widely used in simulations.
- **Polyform Tiling Solvers**: For puzzles (e.g. packing a polyomino set into a shape), exact-cover algorithms (Knuth's DLX) or SAT solvers are standard. These treat tiling as a combinatorial search problem on a discrete grid. Many implementations exist in puzzle/game frameworks but are specialized (e.g. polycube packing).

Visualization: Once tiles and vertices are computed, one can render the tiling by drawing each polygon or by graph-drawing techniques. Escher uses circle-packing/Tutte embedding to produce nice layouts [12]. Cellular-automaton style tools often simply mark cell shapes in a grid. Data structures like DCEL or winged-edge meshes are commonly used to store adjacency for real-time CA or game use [13].

Extensions and Higher Dimensions

Analogous concepts exist in 3D and higher dimensions. **Uniform honeycombs** in 3D (space-filling tessellations by congruent cells) are classified by Coxeter/Wythoff constructions; e.g. the only regular Euclidean honeycomb is {4,3,4} (cubic cells) [22]. In general, any lattice in $\mathbb{R}^3$ yields a convex uniform honeycomb via its Voronoi (zonohedral) cells [23]. Delaney–Dress theory extends to periodic tilings of 3D space (with symmetries of crystallographic groups) and even hyperbolic 3-space. In 4D and beyond, one studies regular and uniform polytopes and honeycombs by Coxeter groups. These higher-dimensional systems suggest that many of the 2D methods (subdivision rules, orbifold notation, Delaney symbols) have analogues, potentially guiding future extensions of tiling software and notation.

Table 1. Summary of 2D tiling categories, their characterization, and generation methods (see text for details and references).



Example Wang-tile tiling (edge-colored squares placed with matching colors) [5] *. Each colored square represents a prototile; the pattern is forced to be aperiodic by the tile set.*

---

[1] [6] Euclidean tilings by convex regular polygons - Wikipedia
https://en.wikipedia.org/wiki/Euclidean_tilings_by_convex_regular_polygons

[2] [7] GomJau-Hogg's Notation for Automatic Generation of k-Uniform Tessellations with ANTWERP v3.0
https://www.mdpi.com/2073-8994/13/12/2376

[3] faculty.washington.edu
http://faculty.washington.edu/moishe/branko/BG106%20Eighty-one%20types%20of%20isohedral.pdf

[4] Girih - Wikipedia
https://en.wikipedia.org/wiki/Girih

[5] Wang tile - Wikipedia
https://en.wikipedia.org/wiki/Wang_tile

[8] plantri-full.dvi
https://users.cecs.anu.edu.au/~bdm/papers/plantri-full.pdf

[9] [17] [18] [19] [20] EPINET Delaney Dress tiling theory
https://epinet.anu.edu.au/page/epinet2_mathematics_delaney_dress

[10] [14] [21] Beyond the wall: working with aperiodic tilings using finite-state transducers
https://www.chiark.greenend.org.uk/~sgtatham/quasiblog/aperiodic-transducers/

[11] Welcome to pynrose's documentation! — pynrose 1.0.1 documentation
https://pynrose.readthedocs.io/

[12] [13] archive.bridgesmathart.org
https://archive.bridgesmathart.org/2024/bridges2024-455.pdf

[15] Wang Tiles for image and texture generation - ACM Digital Library
https://dl.acm.org/doi/10.1145/882262.882265

[16] GitHub - HHogg/antwerp: Application for Nets and Tessellations With Edge-to-edge Regular Polygons
https://github.com/HHogg/antwerp

[22] [23] Convex uniform honeycomb - Wikipedia
https://en.wikipedia.org/wiki/Convex_uniform_honeycomb