computers

Article

Constructing and Visualizing Uniform Tilings

Nelson Max

Department of Computer Science, University of California, Davis, CA 95616, USA; max@cs.ucdavis.edu

Abstract: This paper describes a system which takes user input of a pattern of regular polygons around one vertex and attempts to construct a uniform tiling with the same pattern at every vertex by adding one polygon at a time. The system constructs spherical, planar, or hyperbolic tilings when the sum of the interior angles of the user-specified regular polygons is respectively less than, equal to, or greater than 360°. Other works have catalogued uniform tilings in tables and/or illustrations. In contrast, this system was developed as an interactive educational tool for people to learn about symmetry and tilings by trial and error through proposing potential vertex patterns and investigating whether they work. Users can watch the rest of the polygons being automatically added one by one with recursive backtracking. When a trial polygon addition is found to violate the conditions of a regular tiling, polygons are removed one by one until a configuration with another compatible choice is found, and that choice is tried next.

Keywords: uniform tiling; Archimedean tiling; hyperbolic plane

Citation: Max, N. Constructing and Visualizing Uniform Tilings. *Computers* **2023**, *12*, 208. https://doi.org/10.3390/computers12100208

Academic Editor: Paolo Bellavista

Received: 22 July 2023
Revised: 28 September 2023
Accepted: 7 October 2023
Published: 17 October 2023

1. Introduction

Symmetry and tilings have long fascinated mathematicians, physicists, artists, and architects. The Greek mathematician Archimedes found the uniform tilings of the sphere over 2200 years ago, the French mathematician Henri Poincaré studied groups of symmetries of the hyperbolic plane over 140 years ago, and M.C. Escher used these symmetries in many of his sculptures, drawings, and prints. As explained in Gomez et al. [1], hyperbolic tilings have applications in communication theory. There are now methods of categorizing, analyzing, and enumerating all uniform tilings of the sphere, plane, and hyperbolic plane.

The goal of the present work is to create an interactive computer graphics system that allows users to experiment with possible patterns of polygons around an initial vertex and determine whether they can be extended to uniform tilings of either the sphere, the Euclidean plane, or the hyperbolic plane. This research was developed in association with a freshman seminar on "Symmetry in the World Around Us" at the University of California, Davis, in which students were asked to discover tilings and symmetry groups. This system can serve as an educational tool about symmetry. Users receive positive feedback from the beauty of the tilings they create, and are able to change the view in order to examine a tiling more closely.

Several science museums contain collections of cardboard or plastic regular polygons that visitors can place on a table or wall to create planar tiling patterns or join together to build polyhedra; however, this cannot be easily accomplished for hyperbolic tilings. This system can supplement such hands-on experiments by providing an automatic way of extending a proposed small initial tiling to a much larger one when possible, and works for hyperbolic tilings as well.

1.1. Definition of a Uniform Tiling

A tiling of a surface is a collection of polygons with disjoint interiors with a union that is the whole surface and with the intersection of any two polygons being either empty, a common vertex, or a common edge. A uniform tiling satisfies three additional conditions. (1) Regular polygons: the polygons are all regular, meaning that all edges have the same length and all vertex angles are the same; different regular polygons in the tiling may have different numbers of edges. (2) Archimedean: the pattern of polygons around any vertex must be the same as or a mirror image of the pattern around any other vertex. (3) Transitivity: a stronger version of Archimedean in which there must be a global symmetry of the tiling (a rigid mapping of the whole surface to itself, taking polygons to polygons) that takes any vertex to any other vertex. This symmetry may be a reflection that takes a counterclockwise pattern of polygons around the first vertex to a clockwise one around the second vertex. Note that in Figure 1 there are n lines of mirror reflection symmetry through the center of each red n-gon; thus, half of the vertices have a counterclockwise pattern (n, 6, 4) and half have a clockwise pattern (n, 6, 4), corresponding to a counterclockwise pattern (n, 4, 6).

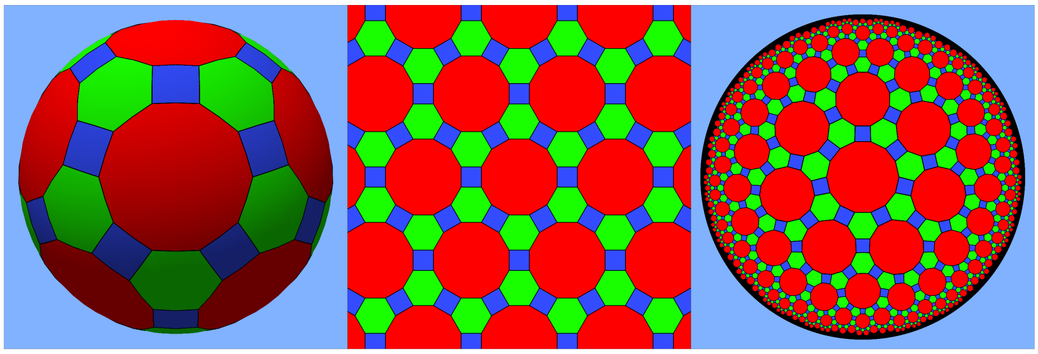

Figure 1. Uniform tilings of the sphere, plane, and hyperbolic plane formed from red n-gons for n = 10, 12, and 14, respectively, surrounded by alternating hexagons and squares, with vertex patterns (n, 6, 4).

If only conditions (1) and (2) are satisfied, such that for any pair of vertices there is a local mapping which matches the pattern of polygons around the first vertex to that around the second (allowing for reflections), the tiling is called Archimedean, after the ancient Greek mathematician Archimedes who discovered all such tilings of the sphere in terms of their corresponding inscribed Archimedean polyhedra. For a uniform tiling, such a local mapping must extend to a global symmetry of the whole tiling matching all the polygons, as stated in condition (3). In this paper, condition (3) is called the transitivity condition, as it is the same as saying that the symmetry group of the tiling acts transitively on the vertices.

1.2. Behavior of the System

The user lists the pattern of polygons around an initial vertex as a sequence of integers, for example, (12, 6, 4) for the center image in Figure 1, specifying the number of sides of the polygons in counterclockwise order around the vertex, then presses the Enter key to indicate that the list is complete. The specified pattern around the first vertex is displayed as the user types. As explained in Section 2.1 below, the sum of the interior angles of the polygons in the list determines whether the tiling (if it exists) will tile the sphere, the plane, or the hyperbolic plane. In the sphere or the hyperbolic plane, the length of the polygon edges is dictated by the initial pattern of polygons, as explained in Section 2.1. For the planar case, the size does not matter and it may be scaled using the scroll wheel.

The program adds polygons one by one, spiraling in counterclockwise order in loops around the outer contour of the existing collection, in order to spread symmetrically outward from the first polygon in the center of the screen and provide a pleasing appearance. It attempts to extend the uniform tiling by matching the pattern specified by the user with the partially complete pattern at the vertex where the new polygon is to be added. As there may be several compatible choices in either clockwise or counterclockwise order, a list is made of the possibilities for each vertex and the first is tried. If a compatible choice is found to extend the existing pattern, the program then checks whether the transitivity condition is satisfied as well. If not, the next choice is tried. If at any stage the list of possibilities is exhausted and no compatible choice is possible, that list is discarded, the previous polygon is removed, and its next possible choice is tried. When no more choices remain, one further previous polygon is removed, its next choice is tried, and so forth. If this backtracking attempts to remove one of the user-specified polygons around the initial vertex, then the program stops, and may be restarted to try a different pattern. This is basically a depth-first search of the tree of possibilities. However, because the tree is infinite in the planar and hyperbolic cases, where the tilings are infinite, it is up to the user to decide when to stop adding polygons when there are enough that the tiling arrangement is clear and/or pleasing. At that point, the user can cause to program to backtrack and find other tilings, if there are any, with the same pattern around the initial vertex. The rest of this paper describes the details of this tiling construction process.

1.3. Basic Information

There are eleven uniform plane tilings, consisting of three regular ones with a single kind of polygon: all equilateral triangles with pattern (3, 3, 3, 3, 3, 3), all squares with pattern (4, 4, 4, 4), all regular hexagons with pattern (6, 6, 6), and eight semiregular ones which mix different kinds of regular polygons with patterns: (6, 3, 3, 3, 3), (4, 4, 3, 3, 3), (3, 4, 3, 3, 4), (6, 4, 3, 4), (6, 3, 6, 3), (12, 12, 3), (12, 6, 4), and (8, 8, 4), as illustrated in [2].

There are five Platonic regular polyhedra, each of which uses only one kind of regular polygon. As discovered by Archimedes, there are thirteen semiregular polyhedra with regular polygons of mixed type, as illustrated in [3]. There are also two infinite families of prisms and antiprisms, which are uniform as well. These polyhedra can be projected radially outwards onto their circumscribing spheres, resulting in uniform tilings of the sphere by regular spherical polygons with sides that are great circle arcs, as shown on the left of Figure 1.

The unit sphere has a constant +1 positive curvature, while the Euclidean plane has a curvature of 0. The hyperbolic plane is a surface with a constant -1 negative curvature, and can be visualized on the Euclidean plane as the interior of the unit disc. In this Poincaré disk visualization, the geodesic lines are circular arcs which meet the bounding circle of the disc at right angles (see [4]), as on the right of Figure 1. The mapping from the hyperbolic plane to the Poincaré disk distorts the lengths, making the polygons closer to the circle bounding the disc appear smaller; however, in the hyperbolic plane itself all of the polygons have the same edge lengths. There are infinitely many regular tilings with only one kind of regular hyperbolic polygon, as both the number of sides of the polygon and the number meeting at a vertex can be arbitrarily large.

1.4. Related Work

Certain hyperbolic tilings can be constructed interactively on the web page [5]; these can be rotated and translated by a fairly large but ultimately limited amount. One particular hyperbolic tiling can be translated an apparently unlimited amount of times using the KaleidoTile program, downloadable from [6].

The web pages [7,8] present the mathematics for the hyperbolic geometric constructions which were used in this paper. The web page [9] contains Python programs for drawing hyperbolic tilings based on reflecting a fundamental triangular region repeatedly in three mirrors along its edges; however, this allows only three different kinds of regular hyperbolic polygons plus truncation. Moreover, it is slow, as the software performs multiple reflections for each pixel, making the rendering time proportional to the resolution. Thus, there is a role for the kind of algorithm described here for creating and interactively viewing a uniform tiling with an arbitrary specified polygon pattern around each vertex, if one exists.

There is a complete theory cataloging uniform 2D tilings available in [10] as well as in chapter 19 of [11], including tables and illustrations of the simpler ones. These references involve complex mathematical methods which will be inaccessible to most readers. The methods could, in principle, enumerate all uniform tilings, in the sense that if an appropriate "breadth-first" algorithm implementing them were run forever it could eventually generate any specific one in a finite amount of time. In contrast, the current system allows a user who is not interested in the detailed mathematics to generate any specific tiling and visualize its sequential trial-and-error construction, including any necessary backtracking.

Chapter 10 in [12] contains a discussion of spherical and hyperbolic uniform tilings, while later chapters show how to create Escher-like tilings with animal- or insect-shaped tiles. Ouyang et al. [13] similarly created such tilings on the hyperbolic plane. Ouyang et al. [14] created symmetrical colored patterns on regular polhedra or spheres. KaleidoTile by Jeff Weeks [6] lets users paint patterns on the regular polygons of uniform tilings. These references aim to exploit the artistic opportunities of specific tilings rather than to explore the geometry of all possible tilings, as we do here.

2. Materials and Methods

This section describes the geometric tiling construction in the planar, spherical, and hyperbolic cases, the combinatorial/topological process of adding tiles one by one, the verification of transitivity, and the user interface. The geometric constructions are described briefly in [15] as well, which additionally considers polygons with an infinite number of edges and describes combinatorial methods for matching vaguely related to those in Section 2.3 below.

2.1. Planar, Spherical, and Hyperbolic Geometry

Planar. If we move counterclockwise around a regular planar polygon with n sides, we make a left turn by the exterior angle α at each of the n vertices and return to the initial side's direction after turning by a total angle of 2π; thus, we have

$$n\alpha = 2\pi, \tag{1}$$
$$\alpha = 2\pi/n. \tag{2}$$

Therefore, the interior angle γ of a regular n-gon is

$$\gamma = \pi - \alpha, \tag{3}$$
$$= \pi - 2\pi/n. \tag{4}$$

To form a planar tiling with a vertex pattern $(n_1, n_2, \ldots, n_k)$ and with no gaps or overlaps between these polygons at the vertex, we must have the interior angle sum

$$S = \sum_{i=1}^{k} (\pi - 2\pi/n_i) \tag{5}$$

add up to exactly 2π. For a given a pattern $(n_1, n_2, \ldots, n_k)$ satisfying this equation, simple plane geometry and trigonometry can be used to construct the tiling geometry.

Spherical. If the sum $S < 2\pi$ and the specified regular polygons are placed in a plane next to each other around a vertex B, there will be a gap between the last polygon and the first. We can rotate the planes of the respective polygons in 3D around their shared edges to create dihedral angles between them and close this gap. These dihedral angles are calculated in Equation (12) below. If there are only three polygons in the pattern, then the triangle formed by the other ends W_i of the edges meeting at B has a unique rigid shape, so there is only one choice of these dihedral angles. If there are four or more polygons, however, the configuration is flexible and there are multiple choices. With the correct choice, the $k + 1$ points $B, W_1, W_2, \ldots,$ and W_k all lie on a single sphere, which is the circumscribed sphere of the uniform polyhedron we are trying to construct. Radial projection of this polyhedron onto the unit sphere then provides a spherical tiling. The internal angles of the spherical polygons are measured in the tangent plane at B, and add up to 2π. They are larger than the internal angles of the polyhedron's faces at B, as the face planes are tilted away on one side of this tangent plane.

Suppose that we have a vertex pattern $(n_1, n_2, \ldots, n_k)$ at vertex B, meaning that the sum S in Equation (5) is less than 2π, and that we want to construct the corresponding spherical tiling to find the correct dihedral angles between the face planes of the corresponding inscribed polyhedron. For this, we can use spherical trigonometry. Figure 2 shows a closeup of the central regular spherical decagon in the tiling of the unit sphere at the left of Figure 1. The user-specified vertex pattern is (n_1, n_2, n_3) = (10, 6, 4); thus, the decagon, hexagon, and square appear in counterclockwise order around the initial vertex B. Let A be the point on the unit sphere at the center of the decagon and let C be the midpoint of the great circle arc BW_2. In Figure 2, the lines AB and AC are great circle arcs as well; however, they appear straight because the view is from directly above A. Based on the mirror symmetry of the decagon across the plane of the great circle arc AC, the angle ACB is a right angle; thus, the triangle ABC is a right spherical triangle. By convention, A, B, and C refer to the angles of the spherical triangle as measured in the tangent planes of the respective vertices, while a, b, and c denote the arclengths of the sides opposite them, that is, sides BC, AC, and AB, respectively. Again, based on the symmetry of the spherical decagon, angle A is $2\pi/20 = \pi/10$, or in the general case, π/n_i. Letting x = cos(a), if we know x, then we can determine $2a$, the side arclength of each of the regular spherical polygons.

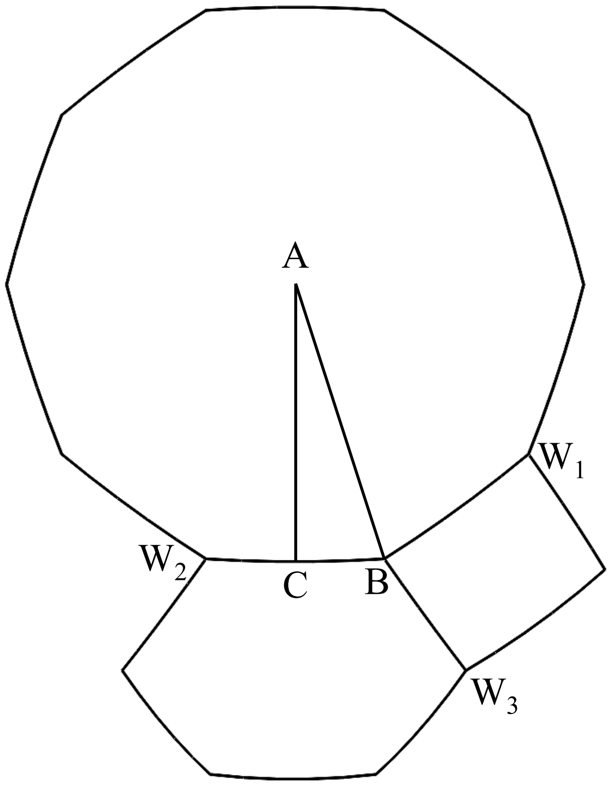

Figure 2. Three spherical polygons around an initial vertex B.

From the spherical trigonometry rules for right spherical triangles (see Equation (19) on page 206 of [16]),

$$\cos(A) = \sin(B)\cos(a), \tag{6}$$

meaning that we have

$$B = \arcsin(\cos(A)/\cos(a)), \tag{7}$$
$$B_i = \arcsin(\cos(\pi/n_i)/x). \tag{8}$$

The angle B_i is half the internal angle of the ith spherical polygon at vertex B, and these internal angles add up to 2π; thus, we have

$$\sum_{i=1}^{k} \arcsin(\cos(\pi/n_i)/x) = \pi. \tag{9}$$

This is a transcendental equation that can be solved for the unknown x by the secant method. Knowing x, we can determine B_i from Equation (8). Then, in (θ, ϕ) spherical coordinates on the unit sphere (with the north pole at the vertex B and the X axis along the direction BW_1) we obtain the spherical coordinates (θ_i, ϕ_i) of W_i by

$$\theta_i = 2\arccos(x) \tag{10}$$

$$\phi_i = 2\sum_{j=1}^{i-1} B_j \tag{11}$$

and these determine the Cartesian coordinates of W_i. A similar construction is provided in [17].

The normal N_i to the plane of the ith polygon around B in the inscribed polyhedron is the normalized cross product of the vector BW_i with BW_{i+1} if $i < k$ (or with BW_1 if $i = k$). The dihedral angle δ_i between consecutive faces i and $i+1$ around B can be found using a dot product:

$$\delta_i = \arccos(N_i \cdot N_{i+1}). \tag{12}$$

Hyperbolic. If the angle sum S in Equation (5) is greater than 2π, then the tiling should lie in the hyperbolic plane. The method of computing the appropriate length of the hyperbolic polygon edges to make them fit around the initial vertex B follows the same method as above, using hyperbolic trigonometry instead of spherical trigonometry. From the hyperbolic right triangle formula in Proposition 8.2.2 of [18] analogous to Equation (6), we have

$$\cos(A) = \sin(B)\cosh(a). \tag{13}$$

Thus, letting $x = \cosh(a)$, where a is half the hyperbolic polygon edge length e, Equations (8) and (9) apply as above, and after solving Equation (9) for x, the polygon edge length is

$$e = 2\text{arccosh}(x). \tag{14}$$

As the Poincaré disc visualization of the hyperbolic plane preserves angles, Equation (11) determines the directions in which the circular arcs representing the hyperbolic lines for the edges BW_i leave vertex B. Thus, we need to construct a circle C leaving B in a given direction D, then find the point W_i on this circle at a hyperbolic distance e from B. To construct C, we can use Proposition 3.1.7 from [18], which implies that the circle C passes through the inversion U of B in the unit circle, which is the boundary of the Poincaré disc with center O and radius $OE = 1$, as shown in Figure 3. Let O be at the origin (0, 0) of the plane, and let $s = \|OB\|$ be the distance of B from O. Then, from the definition of inversion in the unit circle, U lies on the ray OB at a distance $1/s$ from O, and the distance $\|BU\| = \|OU\| - \|OB\| = 1/s - s$. The center M of circle C must be equidistant from B and U; thus, it must lie on the perpendicular bisector SM of the segment BU.

The midpoint S of the segment BU is at a distance $s + \|BU\|/2 = s + (1/s - s)/2 = (s + 1/s)/2$ from O, meaning that if $E = (1/s)OB$ is the unit vector in direction OB, then $S = ((s + 1/s)/2)E$. Also $\|BS\| = \|BU\|/2 = (1/s - s)/2$. In order to leave B in direction D, the circle C must be tangent to the ray between B and $B + D$ shown in Figure 3; thus, its center M must lie on the line BM from B perpendicular to this ray at the position where it intersects the perpendicular bisector of the segment BU. The angle β between BU and BM

is the complement of the angle α between the vectors D and E. Therefore, if $d = \cos(\alpha)$, we have the following:

$$d = D \cdot E \tag{15}$$

$$\alpha = \arccos(d) \tag{16}$$

$$\beta = \pi/2 - \alpha \tag{17}$$

$$\tan(\beta) = \cot(\alpha) \tag{18}$$

$$= d/\sqrt{1-d^2} \tag{19}$$

$$\|SM\| = \tan(\beta)\|BS\| \tag{20}$$

$$= \left(d/\sqrt{1-d^2}\right)(1/s - s)/2 \tag{21}$$

$$M = S + \|SM\|G \tag{22}$$

where G is the unit vector E rotated clockwise by 90°. In the situation shown in Figure 3, α is an acute angle. If α is obtuse, β, d and $\|SM\|$ are negative, and M is on the other side of BU. When we know the center M of C, its radius is $r = \|BM\|$.

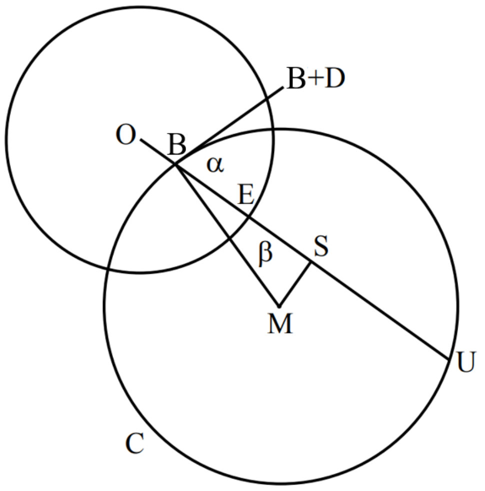

Figure 3. Construction of a circle C perpendicular to the unit circle when leaving point B in direction D.

The next problem is to compute a point W along this circle that is a hyperbolic distance e from B. According to [4], the hyperbolic distance e from B to W is

$$e = \operatorname{arccosh}\left(1 + 2\frac{\|B-W\|^2}{(1-\|B\|^2)(1-\|W\|^2)}\right), \tag{23}$$

and we have

$$\cosh(e) - 1 = 2\frac{\|B-W\|^2}{(1-\|B\|^2)(1-\|W\|^2)}. \tag{24}$$

If we parameterize point W on circle C using the angle θ around the circle:

$$W(\theta) = M + r(\cos(\theta), \sin(\theta)) \tag{25}$$

then Equation (24) becomes a transcendental equation in θ which can be solved by the secant method. Because the right side of Equation (24) increases to infinity as $\|W\|$ approaches 1 and is invalid when $\|W\| > 1$, careful stepping is needed to find the θ interval where it crosses $\cosh(e) - 1$, which can then be used to initialize the secant method iteration.

2.2. Topology

The connectivity or topology of a tiling describes the adjacency relationship of all the polygonal tiles. Here, it is developed by adding new polygons one at a time counterclockwise around the outer contour of the already assembled ones. This effectively adds circle-like layers around the user-specified cycle of polygons at the initial vertex. A more abstract method of doing this in the hyperbolic plane case is described in [19] using a regular expression production system.

As polygons are added, a cyclic ordered list is maintained of edges shared by only one polygon instead of two, in counterclockwise order around the outer contour of the tiling, with a next addition pointer indicating the edge E of the contour across which the next polygon P should be added. Normally, E is an edge of the most recently added polygon; however, see the last paragraph of this subsection for cases when it is not.

The number of vertices of P must be chosen such that at the first vertex V of edge E in counterclockwise order around the outer contour it is compatible with the pattern $S = (n_1, n_2, \ldots, n_k)$ specified by the user; there may be more than one way to do this or none at all. For example, suppose that pattern S is (3, 4, 3, 3, 4) and the counterclockwise polygons already added at V have a partial vertex pattern T = (4, 3). Then, the next polygon could have three vertices, as (4, 3, 3) is a subsequence of (3, 4, 3, 3, 4). Alternatively, it could have four vertices, because (4, 3, 4) is a subsequence of the cyclic permutation (3, 3, 4, 3, 4) of (3, 4, 3, 3, 4). Thus, every possible matching position of T within S considered as a cycle must be tried. A list of such possible choices is saved for each vertex V for trial and backtracking purposes. Only the vertex count choice is saved, not the matching position. If T = (3, 4), there are two positions to match it as a subsequence of (3, 4, 3, 3, 4); both of them are followed by a 3, and would add a triangle, which is all that matters. The pattern (3, 4, 3, 3, 4) is reversible or palindromic, as its reversal, (4, 3, 3, 4, 3), is one of its cyclic permutations. However the pattern (10, 6, 4) shown at the left of Figure 1 is not. Because clockwise matching of the pattern is also allowed, the list of possible polygon types must include those resulting from matching the reversal of pattern S as long as S is not palindromic. When the reversed order is used at a vertex V, this is recorded by setting `vertex_sign`[V] to -1 instead of 1.

The type of the new polygon P must be added to the existing vertex pattern at the vertex W at the opposite end of the edge E from V. Thus, the compatibility of the augmented vertex pattern at W with S must be checked. If the addition of P completes the vertex pattern at V or W, in order to provide a closed loop of polygons there may be further adjacencies of edges of P with edges of the contour. The addition of a polygon may complete the patterns at multiple vertices along the outer contour, resulting in several consecutive edges being deleted from the contour and replaced by the counterclockwise sequence of edges of the added polygon P that do not touch the contour; the next addition pointer points to the last edge in this sequence. An example of several construction stages of the (12, 6, 4) plane tiling in the center of Figure 1 is shown in Figure 4. When the blue square is added at the lower left of to the first image in this figure to produce the second image, two edges are removed from the outer contour, as the addition completes the pattern at vertex V. The next addition pointer indicates the lower right edge of this tilted square. The new dodecagon in the third image is added across this edge; actually, however, four of its edges coincide with contour edges. The next addition pointer indicates the last free edge of the dodecagon. A square is added across this indicated edge in the fourth image. In the last image, six edges of the added dodecagon match the contour edges.

Every time another vertex of the new polygon P other than the two vertices V and W of the edge E is found to coincide with a vertex on the contour, the two are pasted together in the topology data structure by removing the vertex on the new polygon and redirecting all pointers pointing to it to the corresponding vertex on the contour. When P is added, it uses the vertices V and W rather than creating new copies, as it is already known at the time of its construction that these vertices belong to it.

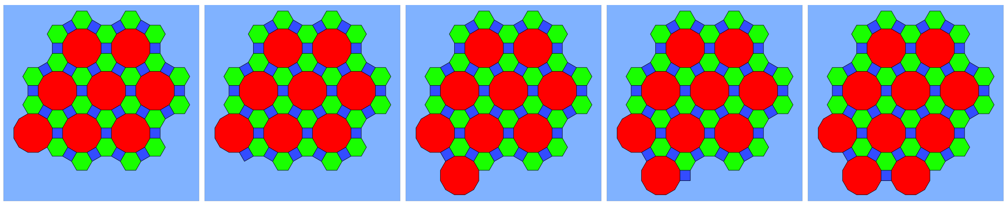
Figure 4. Several polygon addition steps in the construction of the (12, 6, 4) planar tiling.

In the spherical tiling case, viewers can choose to view the developing tiling either as spherical polygons on the sphere or as an inscribed uniform polyhedron with flat faces. Figure 5 shows the addition of the last few polygons for the (8, 8, 3) pattern polyhedron (the truncated cube); the interior sides of the faces are rendered darker, with only ambient illumination. In the second image the last red octagon is added across the free edge of the green triangle at the upper right of the first image, creating three triangular holes geometrically. However, only three of the octagon's edges are actually pasted onto the previous contour. The procedure in the previous two paragraphs does not recognize that the bottom and left sides of the octagon coincide with edges of the contour, because it deals only with the topology (as can be found by completing the patterns at vertices) and does not consider geometric position. Therefore, the contour after this addition has five edges of the octagon, two of which align geometrically with other remaining edges of the contour in the first image. When the green triangle is added at the lower right in the third image, it is found that all of its edges coincide with edges of the contour. This implies that the lower left vertex of this new triangle, which belongs to the newest (front) octagon, coincides with the vertex of the bottom octagon that is matched by another edge of the triangle. Therefore, these vertices are pasted together and the next addition edge pointer indicates the bottom edge of the new octagon, that is, the edge before those which were deleted, as there is no free edge left on the triangle. At this stage it is discovered that the pattern at the previous pasted vertex is complete; thus, the vertex at the left end of the bottom edge of the new octagon is pasted together with its counterpart on the bottom octagon, both are removed from the contour, and the next addition edge pointer is moved one position further back along the new octagon to an edge adjacent to the triangular hole at the bottom left. This causes the next triangle to be added in order to fill in this hole and the same pasting process repeats, meaning that in the last image the hole at the upper left is filled by a triangle. At this point, the contour becomes empty and the polygon addition process terminates. This kind of termination happens only for the spherical case when the tilings are finite. In the planar or hyperbolic case, the tilings are infinite and the addition of new tiles can continue indefinitely.

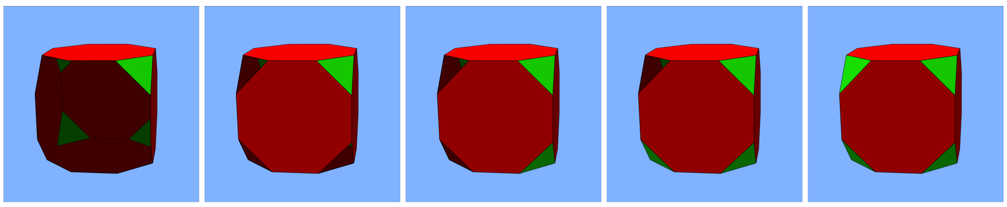
Figure 5. Several polygon addition steps in the construction of the (8, 8, 3) polyhedron (the truncated cube).

2.3. Transitivity

The transitivity condition (3) for uniform tilings states that there is a symmetry of the whole tiling that takes each vertex A to any other vertex C. It is sufficient to verify that there is a symmetry T_{BD} taking the initial vertex B to every other vertex D. If T_{BA} takes B to A, then the inverse symmetry $T_{AB} = T_{BA}^{-1}$ takes A to B and the symmetry T_{BC} takes B to C; thus, the symmetry $T_{AB}T_{BC} = T_{BA}^{-1}T_{BC}$ takes A to C. Therefore, the existence of T_{BD} only needs to be checked for every vertex D other than B.

Because the geometry of the regular polygon tile shapes depends only on the number of vertices in the polygons and their common edge lengths, to construct a symmetry of the tiling taking B to D it is sufficient to find a match T of the edge graph of the tiling, that is, a map taking vertices to vertices, that makes the edges correspond as well. This map is constructed through a breadth-first search (**BFS**) of the edge graph, starting at the initial vertex B and attempting to extend the match between B and D to the new parts of the edge graph discovered during the search. Every possible match of the polygons around B to the polygons around D is tried. The symmetry T may be a reflection, taking polygons in counterclockwise order around B to matching ones in a clockwise order around D. After T has been chosen to be a reflection at B, the order of polygons must be reversed around any other vertex. A sign `signv` for T at B is saved, which is -1 for a reflection and 1 otherwise.

Each initial match of polygons around B determines a match of the edges meeting B, and consequently of the vertices W at the other ends of these edges as well. As the match $X = T(W)$ of each vertex W is determined, it is saved in an array entry `vmatch`$[W]$ defining T. The index `w_bfs_edge`$[W]$ of the edge in the loop of polygons around W along which it was discovered in the BFS is saved, as is the index `x_bfs_edge`$[X]$ at X of the corresponding edge under the symmetry T. These indices are needed to anchor the edge correspondence when extending the symmetry T to new neighbor vertices surrounding W. In addition, newly discovered vertices W are added to a BFS queue of vertices at which $T(W)$ has been determined but for which neighboring vertices have not yet been processed.

In the BFS loop, a vertex W is removed from the queue and X is set to `vmatch`$[W]$. Using `w_bfs_edge`$[W]$, `x_bfs_edge`$[X]$, `signv`, and `vertex_sign`$[W]$ (defined in Section 2.2 above) to force the correspondence between polygons at W and at X, the program checks whether the polygon type patterns around W and X are consistent with this correspondence. If not, the initial match at B cannot be extended, and another one is tried until all are exhausted and failure of transitivity is reported. If consistency is verified, the vertices at the far ends of the edges meeting W can be matched one by one to the vertices at the far ends of the edges meeting X as long as they have not already been matched via another route through the edge graph. If $T(Y) = Z$ is such a new match, `vmatch`$[Y]$ is set to Z, indices `w_bfs_edge`$[Y]$ and `x_bfs_edge`$[Z]$ are set as above, and Y is added to the BFS queue. This ends the BFS loop iteration on vertex W, and the next vertex is removed from the queue. If the queue ever becomes empty, the verification that T is a symmetry is successful and the next vertex D is checked. If T_{BD} can be constructed for all D, then transitivity is verified.

The above process is only applied if there are complete closed cycles of polygons around vertices W and X. If this is not the case, it remains possible to determine that the polygon sequences around them cannot be aligned as subsequences of the specified vertex pattern; however, this involves trying multiple possibilities. I did not do this, so an incompatible choice for an added polygon may not be discovered until a loop of additions around the contour comes back to it and completes the polygon cycles at all of its vertices. A full loop of removals occurs early in the construction of the (4, 4, 4, 5) tiling shown at the left of Figures 9 and 17.

2.4. User Interface

The system only allows polygons of up to 29 sides. Because there are no polygons with one or two sides, it is possible to have the user the vertex counts $(n_1, n_2, \ldots, n_k)$ as a sequence of digits without spaces or delimiters by considering a '1' or a '2' as the first digit of a two-digit number. The user presses the Enter key to terminate the list. If the sum S in Equation (5) of the internal angles of the specified polygons is less than 2π, the dihedral angles for constructing a 3D polyhedron are computed and an animation shows continuous rotation of the planes of the polygons towards these dihedral angles. At any time after this, the user can press the 'p' key to toggle between the polyhedron representation, shown in Figure 6, and the spherical tiling representation, shown in Figure 7.

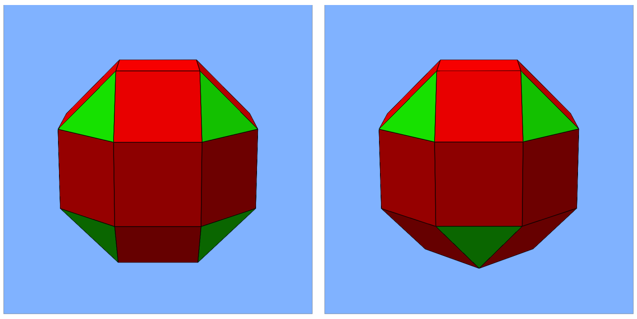

Figure 6. (**left**) The rhombi-cubeoctaheron and (**right**) the pseudo-rhombi-cubeoctaheron.

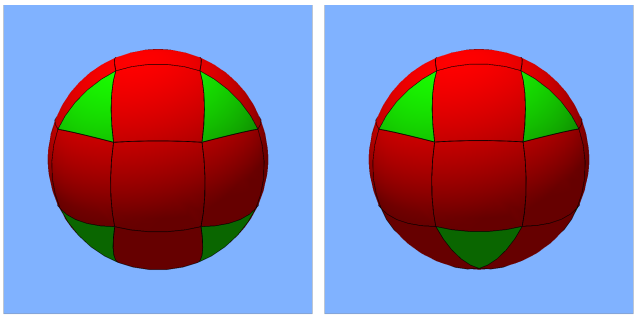

Figure 7. The spherical tilings corresponding to the polyhedra in Figure 6.

If adding an entry n_i makes the sum S exceed 2π, a black Poincaré disc is faded in and the polygons in the list $(n_1, n_2, \ldots, n_{i-1})$ are animated to their hyperbolic shapes, opening up the gap between the first and the last just wide enough to add the new hyperbolic polygon with n_i vertices. If further polygons are specified, the existing ones are again animated to create gaps for the new ones, this time by lengthening their edges to decrease their internal vertex angles.

Polygons with different edge counts are rendered with different colors. Anti-aliased black lines are drawn on the polygon edges to distinguish adjacent polygons with the same number of sides, as in Figure 8.

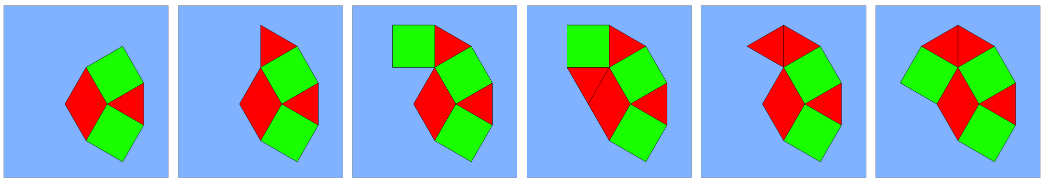

Figure 8. Several polygon addition and removal steps in the construction of the (3, 3, 4, 3, 4) plane tiling starting from the five user-specified polygons.

After the user terminates the vertex pattern input by pressing the Enter key, each subsequent press of that key adds one new polygon. Pressing 'f' adds up to 500 new polygons, which appear one-by-one on the screen.

To obtain the sphere on the left of Figure 1, starting with the pattern (10, 6, 4) around the initial vertex, the user would type "1064εf", where ε represents the Enter key and 'f' specifies the automatic addition of up to 500 polygons. Figure 8 shows a sequence of additions, backtracking removals, and further additions in creating the plane tiling (3, 3, 4, 3, 4) starting from the five user-specified polygons around the initial vertex. Polygons are removed automatically if backtracking is necessary. In this case, the backtracking was caused by a violation of the Archimedean condition (2). Pressing 't' toggles the transitivity test off and on, allowing Archimedean tilings which are not uniform to be constructed, as shown on the right of Figures 6, 7, and 9.

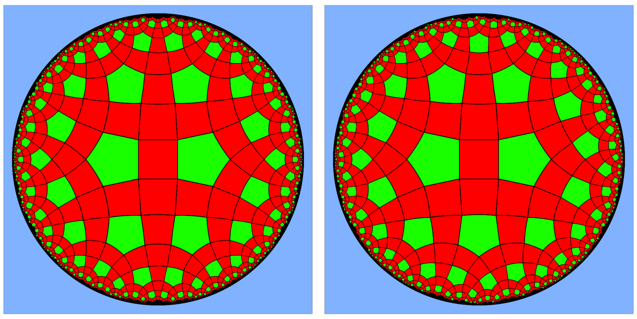
Figure 9. (**left**) A uniform tiling with pattern (4, 4, 4, 5) and (**right**) a non-uniform Archimedean tiling with pattern (4, 4, 4, 5).

The scroll wheel changes the magnification of the 2D image in the planar or hyperbolic case, or alternatively translates a spherical tiling or polyhedron towards or away from the viewer. The right mouse button translates the pattern in two directions. Translation of a hyperbolic tiling is interesting, as certain tiles seem to grow larger and others to shrink, which is due to the length distortion in the Poincaré disc. By moving the mouse with the left button pressed down, the user can rotate a spherical tiling or polyhedron in 3D, as in the Inventor scene viewer interface [20]. The left button acts similarly on the planar or hyperbolic tiling in 2D, with motions along lines through the window center resulting in translations, circular motion touching the outside edges of the window providing rotation about its center, and intermediate motions rotating the view about other centers, thereby specifying all rigid motions of these spaces.

A finite part of an infinite hyperbolic tiling is constructed by concentric loops of polygons around the outer contour of the user-specified polygons based on the initial vertex B. The first polygon is translated in order to place its center at the center O of the Poincaré disc, which makes the polygon appear to have equal edge lengths and makes the tiling around it appear more symmetrical. With only finitely many tiles drawn, the missing tiles appear as small black regions near the outer boundary circle of the disc. However, if the tiling is translated to the right, the missing tiles on the left become very evident as a large black region in the Poincaré disk. To fill in a few of the missing tiles, the vertex W closest to the center O is found at every frame; when W changes index, a tile-aligning 2D mapping T taking B to W, as in Section 2.3 above, is used to provide a matching and re-centered pattern of tiles. While translating, there is cross-dissolve from the translated pattern to the re-centered pattern. For translation to the right, this causes groups of small tiles on the right to fade out and groups of new tiles on the left to fade in, resulting in a sense of unlimited translation similar to that in KaleidoTile [6].

3. Results

The interactive system was originally written in C++ using OpenGL, then converted to JavaScript using WebGL to allow it to be run from a web page. It can be accessed at https://web.cs.ucdavis.edu/~max/tiling.html. Please read the instructions on the web page https://web.cs.ucdavis.edu/~max/UserInstructions.pdf and consult the keyboard and mouse input guide at https://web.cs.ucdavis.edu/~max/QuickReference.pdf.

Figure 6 shows the rhombicubeoctahedron constructed from the pattern (4, 4, 4, 3) on the left; on the right is the pseudo-rhombicubeoctahedron constructed from the same pattern after turning off the transitivity verification. Note that the bottom third of the structure has been rotated 45°, which produces the same local polygon pattern around every vertex while destroying the transitivity; thus, while the resulting polyhedron is Archimedean, it is not uniform. Figure 7 shows these polyhedra projected into spherical tilings.

Figure 9 shows the uniform hyperbolic tiling with pattern (4, 4, 4, 5) on the left; on the right is a non-uniform Archimedean tiling that results when the transitivity test is turned off. Backtracking is still potentially required in constructing such Archimedean tilings in order to enforce the Archimedean condition. A method of constructing Archimedean hyperbolic tilings is provided in [19] that uses the production rules for a regular language to enforce the Archimedean condition and uses backtracking in the case of choosing and applying an incorrect production rule.

There can be more than one uniform hyperbolic tiling with the same pattern of polygons around a vertex, as shown by Lučić and Molnár [10], who presented the example shown in Figure 10. The example on the left of Figure 10 is the tiling with vertex pattern (6, 4, 4, 4) first produced by this system. Note that in addition to bi-infinite strips of adjacent green squares there are other strips consisting of only three squares. If the user wants to check whether there are any other uniform tilings with this same vertex pattern, he or she can type 'n' to (falsely) indicate that the choice of the last polygon was invalid. The forced backtracking process then removes polygons until a different choice of what kind of polygon to add is possible and continues adding polygons from there. In this case, it backtracks to the uniform tiling in the second image of Figure 10, adds a hexagon instead of a square (as shown in the third image), and proceeds to obtain the tiling with only bi-infinite strips of squares shown on the right. Several other polygon additions towards this tiling are shown in Figure 11. The backtracking was caused by a violation of the transitivity condition (3) when trying to add the next polygon to the third image in Figure 11. Figure 12 shows successive addition attempts in the construction of the hyperbolic plane tiling (4, 6, 6, 6). Note that a sequence of several polygons must sometimes be removed before a successful addition. To create the last image in Figure 12, six polygons had to be removed before the red square at the lower right was added.

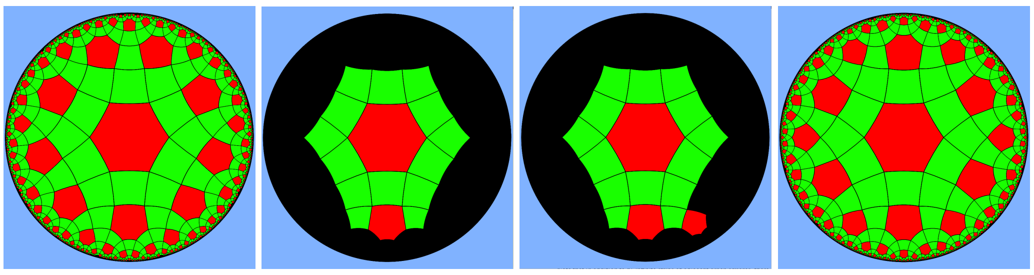
Figure 10. A uniform hyperbolic tiling with pattern (6444), backtracking to add a hexagon instead of a square, and a different uniform tiling with the same vertex pattern.

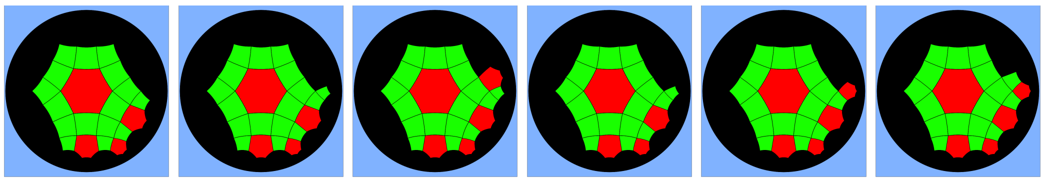
Figure 11. Several polygon addition and removal steps in the construction of the tiling at the bottom right of Figure 10.

When no further alternate choices exist, the system will backtrack all the way to the vertex pattern polygons specified by the user and stop with a pop up window saying this. When backtracking from successful tilings to check whether there are others, the successful tilings are not saved; thus, no comparisons can be made between them and new ones with the same initial vertex pattern. In the example in Figure 10, if the user backtracks from the tiling on the bottom right the program constructs one more tiling, which is a 60° rotation of the first one on the left, and is not fundamentally different. Forced backtracking can produce a mirror image tiling. Figure 13 shows the initial result with vertex pattern (6, 3, 3, 3, 3) on the left and the result of typing 'n' on the right. The illustration of the eleven regular plane tilings in [2] shows both of these mirror images, making for a total of twelve.

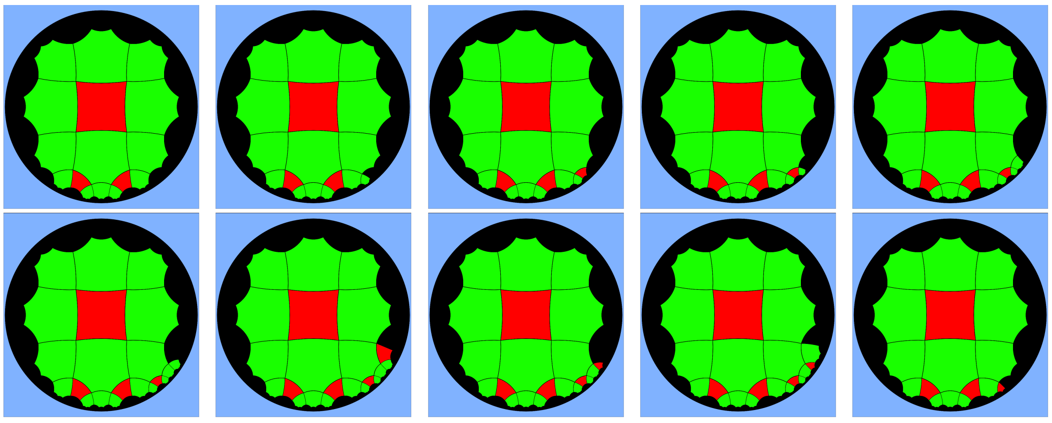
Figure 12. Several polygon addition and removal steps in the construction of the hyperbolic plane tiling (4, 6, 6, 6).

These uniform plane tilings are related to the seventeen wallpaper groups discussed in chapters 2 and 3 of [11], in the sense that the set of symmetries of every uniform tiling is one of these groups. However, certain groups are not represented in uniform tilings, while other groups are the symmetries of more than one tiling. For example, the (12, 6, 4) tiling shown at the center of Figure 1, the (6, 3, 6, 3) tiling, the (6, 6, 6) tiling with only hexagons, and the (3, 3, 3, 3, 3, 3) tiling with only triangles all have symmetry group *632, generated by the reflections across the sides of a 30°–60°–90° triangle with vertices that are six-fold, three-fold, and two-fold mirror points. The tilings in Figure 13 have symmetry group 632, with only six-fold, three-fold, and two-fold rotations, with translations but without reflections. There are symmetry groups for spherical patterns, discussed in chapter 4 of [11], and hyperbolic plane patterns, discussed in chapters 17 and 18 of [11]. For example, the tiling at the left of Figure 1 is in spherical group *532 and the one at the right is in hyperbolic group *732. The tiling at the left of Figure 14, with vertex pattern (5, 3, 3, 3, 3), is in spherical group 532, while the one at the right, with vertex pattern (7, 3, 3, 3, 3), is in hyperbolic group 732. Each of these two tilings has a mirror image tiling, as shown in Figure 13.

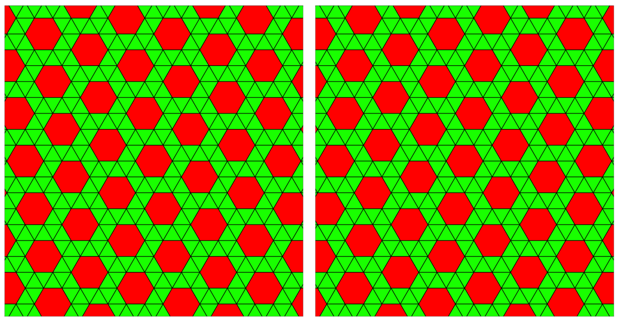
Figure 13. Two mirror image tilings for vertex pattern (6, 3, 3, 3, 3).

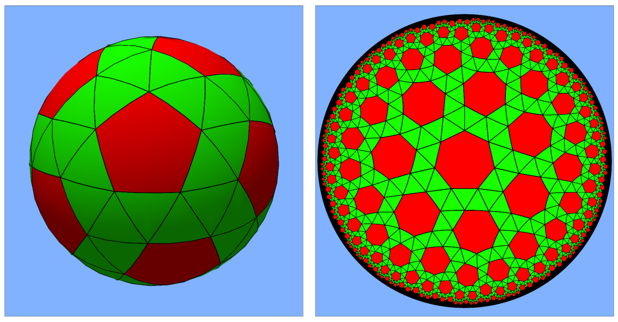
Figure 14. The spherical tiling (5, 3, 3, 3, 3) (**left**) and hyperbolic tiling (7, 3, 3, 3, 3) (**right**).

Figure 15 shows the hyperbolic tiling with the pattern (4, 6, 8, 10) using four colors before and after being translated upwards. Note that the central square appears regular prior to the vertical translation, with equal sides, while after the translation its side lengths appear shorter and unequal; this is because of the length distortion of the Poincaré disc representation of the hyperbolic plane, which also now causes the hexagon below it to appear regular. The Supplementary Materials Video hyperboloc1000.mp4 show the user interaction, including a long translation of the hyperbolic pattern (4, 5, 4, 6) with cross-dissolves.

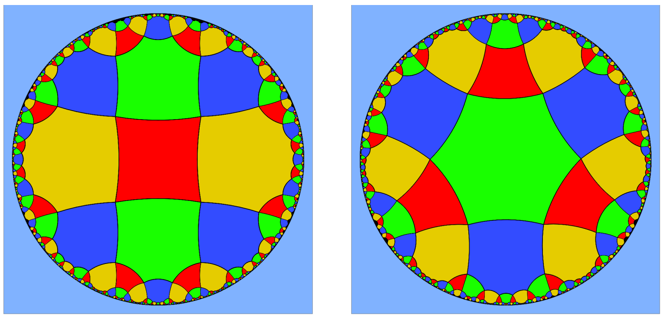
Figure 15. The uniform hyperbolic tiling (4, 6, 8, 10) before (**left**) and after (**right**) being translated upwards.

There is a compromise between the kind of uniform tiling shown on the left of Figure 9 and the kind of completely non-uniform tiling shown on the right. In a *k*-uniform tiling, there are a finite number *k* of types of vertices, meaning that there is a global symmetry when taking any vertex to any other of the same type. Similarly, a *k*-uniform *m*-Archimedean tiling has *k* vertex types with the same global context and *m* vertex types considering only the local pattern of polygons around the vertex. A *k*-uniform 1-Archimedean tiling is called a *k*-uniform Archimedean tiling, or pseudo-Archimedean. Čtrnáct et al. [21] presented an algorithm that enumerates all *k*-uniform tilings of the Euclidean plane using the concepts and notation presented in chapter 19 of [11]. Their method can be extended to the hyperbolic plane to generate all pseudo-Archimedean tilings with a specific vertex pattern, as shown in [22] for the pattern (3, 5, 5, 5). Figure 16 shows the 3-uniform Archimedean tiling at the top of this web page constructed by the algorithm presented here after turning off the overall transitivity test and then specifying that there should be global symmetries which take the rightmost of the three vertices marked by the dark green squares to each of the other two. In the naming convention from [11], this tiling has symmetry group 32* with a three-fold rotation point at the center of the central triangle, a two-fold rotation point at the center of an edge between two pentagons, and a line of mirror symmetry. Users can experiment to create other *k*-uniform tilings as well.

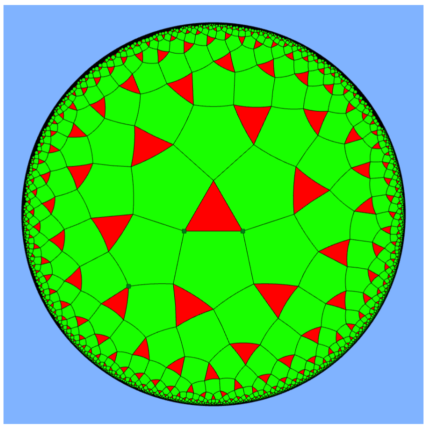
Figure 16. A 3-uniform Archimedean tiling with vertex pattern (3, 5, 5, 5).

Performance

The execution speed was evaluated on a Dell Precision 3550 laptop with an Intel® Core™ i5-10310U CPU @ 1.70 GHz with Mesa Intel® UHD Graphics and 7.4 Gigabytes of memory. Both the Google Chrome and Mozilla Firefox web browsers were used to interpret the JavaScript code. While the code uses only WebGL 1 calls, WebGL 2 was enabled on both browsers. Table 1 shows the results.

Table 1. Time in seconds for various tasks on Chrome and Firefox.

	Chrome	**Firefox**
Adding hyperbolic polygons without display	75.94	118.89
Hyperbolic translation in frames per second	31.3 FPS	33.7 FPS
Adding planar polygons without display	170.67	212.22
Adding planar polygons without transitivity test	8.90	16.89
Planar rotation	31.9 FPS	20.8 FPS

The first two rows of numbers refer to the hyperbolic tiling (4, 4, 4, 5) shown on the left of Figure 17, with 1561 polygons and 2322 vertices, while the last three rows refer to the planar (12, 6, 4) tiling at the right, with 1005 polygons and 3029 vertices.

The cost of the transitivity verification was $\Omega(n^2)$, where n is the current number of vertices, which is because the BFS search to determine whether there is a symmetry taking the initial vertex B to a target vertex D must touch $\Omega(n)$ vertices on average before the vertices that can be matched are exhausted and there are $n-1$ target vertices D to be considered. Thus, this step represents a bottleneck. The first and third rows of the table show the total time in seconds for the combinatorial process of adding all the polygons and performing the transitivity test without displaying a new frame for each polygon addition or backtracking deletion. Normally the individual additions or deletions are displayed so the image changes at a rapid rate, and a Supplementary Materials Video shows the addition of 1000 polygons at a reasonable speed.

The second row in Table 1 shows the frame rate for continuous hyperbolic translation. The fourth row shows the cost of performing only the geometric construction of the planar (12, 6, 4) tiling, as this tiling is uniquely determined by conditions (1) and (2) without testing the transitivity condition (3). The fifth row shows the refresh rate during a rotation of the plane (12, 6, 4) tiling by one degree per frame averaged over 800 frames, and is dominated by the display cost of this many polygons at a 940 by 940 pixel resolution.

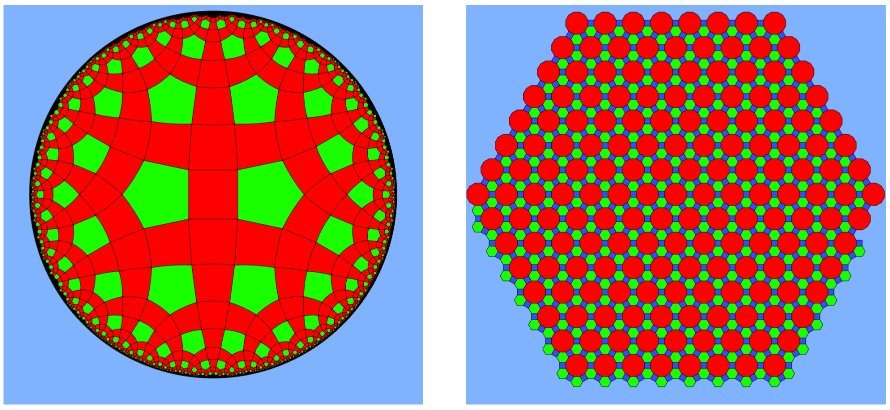
Figure 17. The hyperbolic (4, 4, 4, 5) tiling and the planar (12, 6, 4) tiling.

4. Discussion

4.1. Limitations and Future Work

This system has several limitations and possibilities for improvement. When one or more polygons are removed because of backtracking, this method requires reconstructing the former state of the contour edge list and its next addition pointer in order to add alternative ones. This is currently accomplished by saving the state of the list after each polygon addition, which wastes a great deal of memory. It should be possible to reverse the effect on the contour edge list from the addition of a polygon; however, initial attempts to do this were unsuccessful, so this is future work.

The geometry of the polygons occasionally becomes distorted as a result of floating point errors in the positions of vertices accumulating when the coordinates of the two vertices V and W of an edge E from the last polygon added are used to construct the positions of the other vertices of the next polygon P. The algorithm constructs edge E using successive alignments of common edges of polygons added in a full loop around the outer contour. When the topological connectedness analysis determines that vertices of P other than V and W should be pasted to existing vertices on the contour created in the previous loop, their positions may not agree. The program terminates polygon addition when the difference in these pasted positions exceeds a threshold in order to prevent the process from creating grossly distorted tile shapes. Because the errors are compounded, they grow exponentially in the number of polygons that are added; thus, switching from 32 bit floats to 64 bit doubles only approximately doubles the number of polygons that can be added. This problem does not arise for uniform spherical tilings, as they all have too few polygons. It was solved for uniform planar tilings by saving the 24 possible edge direction vectors and using them to construct the new edges instead of basing them on the direction of an existing edge E from the previous polygon.

Unfortunately, the problem remains for hyperbolic tilings. A potential solution when the vertex pattern around V is completed such that there is a pasting to the next edge F on the contour is to construct the polygon P using F, which is determined in the previous loop of polygon additions and as such will not have accumulated as much error as edge E. For the (4, 6, 6, 6) tiling in Figure 12, where an edge E on the just-previously-added polygon was always used, only 1448 polygons could be added before large floating point errors accumulated. When an edge F on the contour from the previous loop of polygon additions (if available) was used, however, up to 4469 polygons could be added using the same error threshold.

The transitivity verification could be sped up by saving and reusing the edge graph matching information from testing transitivity at each target vertex when the tiling is smaller; however, this would take a great deal of memory. Additional savings could be realized by amortizing the cost over multiple polygon additions, say, by testing the transitivity only after every fourth addition, and if the test fails, redoing it for the last three additions to find the offending choice. In adding 1007 polygons to the (4, 4, 4, 5) pattern tiling on the left of Figures 9 and 17, including those added after failures of transitivity, there were only 66 transitivity failures; thus, such limited testing would result in at least 55% savings in this case.

A future extension could be to the regular tilings of three-dimensional spaces by uniform polyhedra. Such tilings of flat 3-space have been enumerated by Grünbaum [23] and in the PhD thesis of Norman Johnson. Those of the 3-sphere correspond to the boundaries of the four dimensional polytopes, and are listed and illustrated in chapter 26 of [11]. There is room for additional exploration of such tilings of three dimensional hyperbolic space, and the methods of this paper could possibly be extended to that case.

4.2. Concluding Remarks

While other publications have enumerated all possible tilings of a particular type, the method presented in this paper allows users to interactively discover such tilings themselves by trial and error. The user specifies a cycle of regular polygons around the initial vertex by typing their numbers of sides. The system then adds new polygons one-by-one while checking whether all conditions for a uniform tiling (or, optionally, for an Archimedean tiling) are satisfied. If not, that new polygon is removed, and prior polygons may be removed as well, until a new compatible choice for adding a polygon is found.

Each of the eleven uniform plane tilings can be constructed from translated copies of a minimal cell of polygons. Outlined in white in Figure 18 is this minimal cell C for the (6, 4, 3, 4) plane tiling, consisting of one hexagon, two triangles, and three squares. Using two basis vectors V_1 and V_2 joining nearest-neighbor hexagon centers, with V_1 being horizontal and V_2 at 60° above the horizontal, we can construct a translation T_{ij} for every pair of integers i and j:

$$T_{ij} = iV_1 + jV_2. \tag{26}$$

Then, the translated copies $T_{ij}(C)$ of C, outlined in thick black lines in Figure 18, have disjoint interiors, and cover the plane with the (6, 4, 3, 4) tiling. Other cases are similar. For example, the (12, 6, 4) plane tiling shown at the center of Figure 1 has the same symmetry group *632 as the (6, 4, 3, 4) tiling and has a minimal cell with one dodecagon, three squares, and two hexagons. This method of constructing tilings is more complicated for spherical or hyperbolic plane tilings, as rotations are required for the sphere and combinations of rotations and translations for the hyperbolic plane. This would probably be faster than the method presented here, as backtracking and transitivity testing would not be required, and it might decrease the accumulation of numerical error; however, it would require specification of the minimal cell and copying transformations for each tiling. In contrast, the method presented in this paper lets the user experiment freely, with backtracking being a natural part of such experimentation.

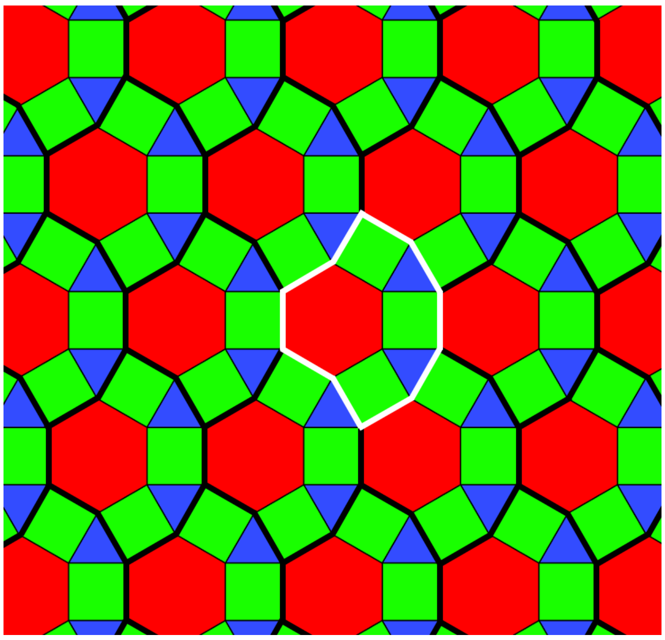

Figure 18. Constructing the (6,4,3,4) tiling from copies of a minimal cell of polygons.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/computers12100208/s1, Video S1: PlaneAndSphere.mp4 and Video S2: hyperbolic1000.mp4.

Funding: This research received no external funding

Data Availability Statement: Not applicable.

Acknowledgments: The author wishes to thank Marek Čtrnáct, Chaim Goodman-Strauss, Don Hatch, and Namo Podee for helpful email discussions, and the IEEE and MDPI reviewers for helpful suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gomez, E.; de Carvalkho, E.; Martins, C.; Soares, W.; da Silva, E. Hyperbolic Geometrically Uniform Codes and Ungerboeck Partitioning on the Double Torus. *Symmetry* **2022**, *14*, 449. [CrossRef]
2. Grünbaum, B.; Shephard, G.C. *Tilings and Patterns*; W. H. Freeman: New York, NY, USA, 1987.
3. Ball, W.W.R.; Coxeter, H.S.M. *Mathematical Essays and Recreations*, 12th ed.; Univertisy of Toronto Press: Toronto, ON, Canada, 1974.
4. Available online: https://en.wikipedia.org/wiki/Poincare_disk_model (accessed on 13 October 2023).
5. Christersson, M. Available online: http://www.malinc.se/noneuclidean/en/poincaretiling.php (accessed on 13 October 2023).
6. Weeks, J. Available online: http://geometrygames.org/KaleidoTile/index.html (accessed on 13 October 2023).
7. Christersson, M. Available online: http://www.malinc.se/noneuclidean/en/circleinversion.php (accessed on 13 October 2023).
8. Christersson, M. Available online: http://www.malinc.se/noneuclidean/en/poincaredisc.php (accessed on 13 October 2023).
9. Tamfang. Available online: https://commons.wikimedia.org/wiki/User:Tamfang/programs (accessed on 13 October 2023).
10. Lučić, Z.; Molnár, E. Fundamental Domains for Planar Discontinuous Groups and Uniform Tilings. *Geom. Dedicata* **1991**, *40*, 125–143. [CrossRef]
11. Conway, J.; Burgeil, H.; Goodman-Strauss, C. *The Symmetries of Things*; A K Peters, Ltd.: Wellesley, MA, USA, 2008.
12. Fathauer, R. *Tessellations: Mathematics, Art, and Recreation*; CRC Press: Boca Raton, FL, USA, 2022. [CrossRef]
13. Ouyang, P.; Fathauer, R.W.; wai Chung, K.; Wang, X. Automatic generation of hyperbolic drawings. *Appl. Math. Comput.* **2019**, *347*, 653–663. [CrossRef]
14. Ouyang, P.; Wang, L.; Yu, T.; Huang, X. Aesthetic Patterns with Symmetries of the Regular Polyhedron. *Symmetry* **2017**, *9*, 21. [CrossRef]
15. Renault, D. The uniform locally finite tilings of the plane. *J. Comb. Theory Ser. B* **2008**, *98*, 651–671. [CrossRef]
16. Kells, L.M.; Kern, W.F.; Bland, J.R. *Plane and Spherical Trigonometry*; McGraw Hill: New York, NY, USA, 1951.
17. Har'el, Z. Uniform solution for uniform polyhedra. *Geom. Dedicata* **1993**, *47*, 57–110. [CrossRef]
18. Stahl, S. *A Gateway to Modern Geometry: The Poincaré Half-Plane*, 2nd ed.; Jones and Bartlett: Burlington, MA, USA, 2008.
19. Goodman-Strauss, C. Regular Production Systems and Triangle Tilings. *Theor. Comput. Sci.* **2009**, *410*, 1534–1549. [CrossRef]
20. ThermoFisher Scientific. Available online: https://www.openinventor.com/ (accessed on 13 October 2023).
21. Čtrnáct, M. (unaffiliated, marek14@seznam.cz) ; Griffin, J. (unaffiliated, james.griffin@cantab.net); Kopczynski, E. (Institute of Informatics, University of Warsaw, Poland). Enumeration of *k*-uniform Euclidean Tilings. Unpublished manuscript .
22. Čtrnáct, M. Available online: https://zenorogue.github.io/tes-catalog/?c=pseudo-Archimedean%2F3555%2F (accessed on 13 October 2023).
23. Grünbaum, B. Uniform Tilings of 3-Space. *Geombinatorics* **1994**, *4*, 49–56.