Orienting a Scribble

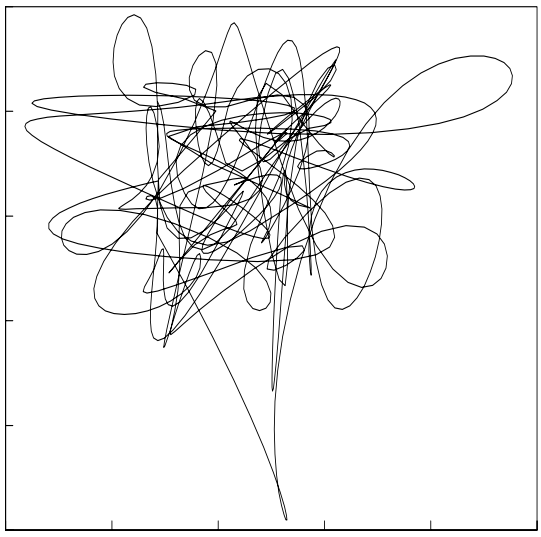
Gary D. Knott, Ph.D.

Civilized Software Inc.
12109 Heritage Park Circle
Silver Spring MD 20906
Tel: (301)962-3711
URL: http://www.civilized.com
email: knott@civilized.com

abstract: A constructive proof is given for the fact that a suitably-nice self-intersecting curve can always be traversed without crossing itself.

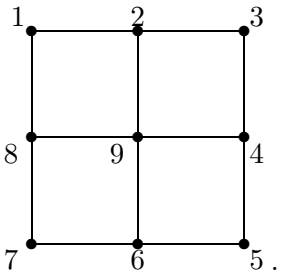
I recently discovered that you can traverse a "scribble" like that shown below, *without crossing your path*. (You might enjoy imagining an algorithm by which you could generate such "random" scribbles at will.) You may need to *touch* the previously-traced path at a point, or even retrace a section of it, but you need not cross it. It seems likely to me that this fact must have been observed before, perhaps many times, but I don't know of any prior report. It is convenient to restrict our attention to scribbles that are closed planar continuous curves, and moreover we want the curves we consider to self-intersect at only a finite number of points, or to overlap themselves along a finite number of disjoint connected segments (or both.) Such curves are suitably nice for our purposes.

The existence of non-crossing traversal paths for such curves is linked to an interesting, and not very deep, structural regularity that is possessed by such closed planar curves. Describing these structural features is the purpose of this story.



The reason we want our curve to intersect itself non-pathologically is that we want to approximate our closed curve by a polygon with a finite number of straight line-segment edges connecting the polygon vertices. You can probably already see that the self-intersection points of our curve will all be vertices of the representing polygon, and overlapping segments will become overlapping edge sequences. We could easily let polygons have curved edges, but it is just as easy to introduce additional degree-2 vertices as needed to adequately approximate a curved edge with a sequence of straight-line segments.

As an example, consider:



We can traverse this curve without crossing it by following the path: 1, 2, 9, 2, 3, 4, 9, 4, 5, 6, 9, 6, 7, 8, 9, 8, 1.

Polygons

A *polyline sequence of directed line segments* is a finite ordered list of directed line segments, called *edges* in many contexts, where the second endpoint of each line segment is equal to the first endpoint

of the next line segment in the list. A *closed* polyline sequence is a polyline sequence such that the second endpoint of the last line segment is equal to the first endpoint of the first line segment in the sequence. Each directed line segment in a polyline sequence must have a positive length, except that the single line segment of a one-member closed polyline sequence must be a degenerate line segment of length 0. A closed polyline sequence corresponds to a polygon in $\mathcal{R}^2$.

We may define an *n-sided polygon* by its piecewise-linear *polygon-boundary curve*. The polygon-boundary curve of the n-sided polygon $P = \mathit{polygon}[p_1, p_2, \ldots, p_n]$ in $\mathcal{R}^2$ is the closed polyline sequence of n directed line segments $\langle \mathit{segment}(p_1, p_2),\ \mathit{segment}(p_2, p_3),\ \ldots,\ \mathit{segment}(p_n, p_1)\rangle$. As a point-set, the polygon-boundary is $\cup_{1\leq i\leq n}\ \mathit{segment}(p_i, p_{i+1})$, with the definition $p_{n+1} := p_1$. We denote the polygon-boundary curve of P by ∂P.

The n points $p_1, p_2, \ldots, p_n$ are the *sequence vertices* of the polygon P, while the n line segments, $\mathit{segment}(p_i, p_{i+1})$ for $i = 1, \ldots, n$, are its edges. The *sequence* of points $\langle p_1, p_2, \ldots, p_n\rangle$ determine the sequence of edges that make-up the boundary curve of P. Remember $p_{i+1} \neq p_i$ for $1 \leq i \leq n-1$, but we may have $p_i = p_j$ where $j \neq i + 1$. Thus, in general, the collection of sequence vertices of P may be a multi-set. We often want to focus on the set of distinct vertices of P; we shall generally mean *distinct vertices* when we use the word 'vertices' without qualification. Note we do not insist that the edges of P be distinct, so, in general, the edges of P are also a multi-set. Usually we write $\mathit{edge}[a, b]$ as a synonym for $\mathit{segment}[a, b]$. Also, we define $\mathit{begin}(\mathit{edge}[a, b]) := a$ and $\mathit{end}(\mathit{edge}[a, b]) := b$.

Note that the use of the descriptive term "boundary" does not imply that a polygon necessarily has obvious inside and outside regions; the piecewise-linear polygon-boundary curve may intersect or cross itself and/or overlap itself many times. A closed polyline sequence that does not cross itself encloses a finite-area region. This region may be disconnected, and it may even be empty. However when our closed polyline sequence is of length greater than one and does not intersect itself, the finite-area region is connected and has positive area. Such a closed polyline sequence has its finite-area region either entirely to its left or entirely to its right as one traverses the closed polyline sequence.

For a non-self-crossing polygon-boundary curve, ∂P, we define the associated polygon P in $\mathcal{R}^2$ as the union of the polygon-boundary curve and the finite-area region it encloses. This finite-area region, not including any points of its enclosing polygon-boundary curve is called the *interior* of P, so that $P = \mathit{interior}(P) \cup \partial P$. For a self-crossing polygon-boundary curve, ∂P, we will take $\mathit{interior}(P) = \emptyset$, so that we may still have $P = \mathit{interior}(P) \cup \partial P$. (Note this definition allows a polygon to be non-convex, unlike defining a polygon as a convex hull or intersection of half-planes.)

A closed polyline sequence ∂P does not cross itself when no edge of ∂P intersects another edge in its interior, but this is only a sufficient condition, not a necessary condition. A closed polyline sequence ∂P does not cross itself is exactly when ∂P passes the test for being *oriented* as discussed below.

Generally we use the word 'polygon' to mean the point-set consisting of the polygon-boundary *and* the finite-area point-set that it encloses; however this presupposes that there is such a finite-area "interior", and this is *not* always the case, so we need to take care to only use the word polygon to refer to a region when this makes sense; otherwise we must perforce be referring to the

polygon-boundary curve.

All polygons considered here are *planar* polygons embedded in $\mathcal{R}^2$. A closed polyline sequence in $\mathcal{R}^3$ is a piecewise-linear space-curve, but it doesn't have much in common with polygons in $\mathcal{R}^2$. The more appropriate generalization for $\mathcal{R}^k$ is to replace edges with hyperplane patches. Also, note a polygon has an obvious abstraction as a directed graph, but this graph is *not* necessarily planar in the graph-theoretic sense.

A polygon is *simple* if it has at least two distinct vertices, and if no pair of nonconsecutive edges share a point, and if each pair of consecutive edges share exactly one point. The essential property of a simple polygon is that its polygon-boundary curve is non-self-intersecting. A simple polygon in the xy-plane is thus defined by a finite set of edges, each of which has positive length and intersects exactly one other edge at and only at each of its end-points. The union of these edge line-segments is the polygon-boundary point-set of the simple polygon. If we allow edges to overlap or intersect arbitrarily at polygon-boundary points besides the end-points, we have a general, not necessarily simple, polygon. A simple polygon P partitions $\mathcal{R}^2$ into three parts: The finite-area interior of P, the boundary, ∂P, of P, and the infinite-area exterior of P.

When the vertices of a simple polygon in $\mathcal{R}^2$ with respect to a right-handed coordinate system are listed in an order such that traversing the edges of the polygon from each vertex to the subsequent adjacent vertex in the sequence winds clockwise around the points in the finite-area interior of the polygon, then that polygon is said to be assigned a *clockwise orientation* and its vertices are said to be listed in *clockwise order*. A simple polygon in the right-handed plane thus has a clockwise orientation if and only if its vertices are ordered in a clockwise order. We shall see that a precise definition of clockwise orientation for a simple polygon can be given in terms of the signed area of a polygon. If the polygon's vertices are listed in the opposite order, the polygon has a *counterclockwise* orientation. Thus if a simple polygon $polygon[p_1, p_2, \ldots, p_n]$ has a clockwise orientation, the reverse simple polygon $polygon[p_n, p_{n-1}, \ldots, p_1]$ has a counterclockwise orientation.

Vertex-Complete Polygons

A polygon P is *vertex-complete* if every pair of its edges either are disjoint, or intersect at endpoints, or are identical as sets.

Any polygon P can be converted into a vertex-complete polygon $VC(P)$ by introducing additional vertices into the sequence of original vertices at the intersections of pairs of non-colinear edges of P where vertices are missing. For a pair of intersecting colinear edges, we must add vertices that match the end-points of the two intersecting edges at the places where such end-points lie in the interior of one of the edges.

The polygon Q defined below has one point that needs to be categorized as a vertex and added to the defining closed polyline sequence, and two points that are already present as sequence vertices that need to be replicated and added to the defining closed polyline sequence, in order for Q to become a vertex-complete polygon.

$Q = polygon[(1,0),(7,0),(6,-1),(5,1),(4,0),(3,0),(2,1)]$.

A general polygon whose edges are directed line-segments is *oriented* if its boundary curve does not

cross itself. For a general vertex-complete polygon P, specified by a given sequence of vertices [p_1, p_2, ..., p_n], wherein edges may coincide, it is possible, but tricky, to check to see if P is oriented. A way to test to see if an arbitrary vertex-complete polygon P is oriented is presented below. This test also serves to *define* an oriented vertex-complete polygon. The basic idea in this test is that, if we "walk-around" an oriented polygon starting with its infinite-area region to our left, then the infinite-area region will constantly remain to our left during the traversal, or if we start with the infinite-area region to our right, then the infinite-area region will constantly remain to our right during the traversal.

Let P be a vertex-complete polygon with two or more edges. Recall that such a polygon, by definition, has no zero-length edges, so that no two sequential sequence vertices are identical. Consider a pair of sequential edges of P: $edge[a, b]$ and $edge[b, c]$ which span the three sequential sequence vertices a, b, c in P. If the vertex point c lies to the left of the directed line: $line[a, b]$, then we say there is a *left-turn* at (a, b, c). If c lies to the right of $line[a, b]$, there is a *right-turn* at (a, b, c). If c lies on $line[a, b]$ then there may be either a left-turn or a right-turn at (a, b, c). An assignment of the descriptors "left-turn" or "right-turn" to each sequential pair of edges: $edge[a, b]$ and $edge[b, c]$ of P which satisfy the above constraints is called an *admissible turn-type assignment* for P.

Suppose there is a left-turn at (a, b, c). Then define the open region $S(a, b, c)$ to be the set of points which are to the left of $line[a, b]$ *and* to the left of $line[b, c]$. The open region $C(a, b, c)$ is defined to be the set of points which are either to the right of $line[a, b]$ *or* to the right of $line[b, c]$ or both. Alternatively, if there is a right turn at (a, b, c), then the open region $S(a, b, c)$ is defined to be the set of points which lie to the right of $line[a, b]$ *and* to the right of $line[b, c]$, and the open region $C(a, b, c)$ is defined to be the set of points which are either to the left of $line[a, b]$ *or* to the left of $line[b, c]$ or both.

Now consider a vertex b which occurs in the vertex sequence of the vertex-complete polygon P as ..., a, b, c, ..., and which also occurs elsewhere in the polygon vertex sequence as ..., d, b, e, ... If there is no such vertex, then P is an oriented vertex-complete polygon. Otherwise, if the edge $edge[d, b]$ lies in the region $S(a, b, c)$, then we label $edge[d, b]$ with the label "$S(a, b, c)$" and also, if $edge[b, e]$ lies in $S(a, b, c)$ we label $edge[b, e]$ with the label "$S(a, b, c)$". Similarly, if the edge $edge[d, b]$ lies in the region $C(a, b, c)$, then we label $edge[d, b]$ with the label "$C(a, b, c)$", and if $edge[b, e]$ lies in $C(a, b, c)$ we label $edge[b, e]$ with the label "$C(a, b, c)$". If $edge[d, b]$ coincides with $line[a, b]$ or with $line[b, c]$, then $edge[d, b]$ may be assigned either the label "$S(a, b, c)$" or "$C(a, b, c)$" as we choose. Similarly, if $edge[b, e]$ coincides with $line[a, b]$ or with $line[b, c]$ then $edge[b, e]$ may be assigned either the label "$S(a, b, c)$" or "$C(a, b, c)$" as we choose. (Actually, in a practical algorithm, it is convenient to label such "choice" edges with a label such as "$B(a, b, c)$" indicating either possiblity.)

Now suppose each sequential pair of edges in P has been assigned an admissible turn-type: left-turn or right-turn, and that, for every vertex b, each sequential pair of edges with the center vertex b are both individually labeled with the labels "$S(x, b, y)$" or "$C(x, b, y)$". with respect to *each* other sequential pair of edges $edge[x, b]$ and $edge[b, y]$ centered at b. Such a labeling is *orientation-consistent* if no sequential pair of edges, $edge[d, b]$ and $edge[b, e]$, has non-matching labels with respect to any other sequential pair of edges centered at b. That is: if $edge[d, b]$ is labeled with "$S(a, b, c)$", then $edge[b, e]$ is not labeled with "$C(a, b, c)$", and symmetrically, if $edge[d, b]$ is labeled

with "$C(a,b,c)$", then $edge[b,e]$ is not labeled with "$S(a,b,c)$". (Essentially, if the edge $edge[d,b]$ has the label "$S(a,b,c)$" while $edge[b,e]$ has the label "$C(a,b,c)$", our polyline crosses itself.)

A vertex-complete polygon P is *oriented* if there exists an admissible turn-type assignment for P which has an associated orientation-consistent labeling. Otherwise P is *unorientable*.

If there is a consistent-labeling such that the turn-type at a left-most (least x-coordinate) vertex of a vertex-complete oriented polygon is a left-turn then the polygon is counterclockwise (CCW). If the turn-type is a right-turn then the polygon is clockwise (CW).

An oriented vertex-complete polygon cannot have crossing edges, but it can have coincident vertices and collinear edges and many closed polyline sequences of edges. Any polygon P can be converted into a vertex-complete polygon $VC(P)$ by introducing additional vertices at the intersections of edges of P where vertices are missing. Then, a non-vertex-complete polygon P is oriented if its vertex-complete form $VC(P)$ is oriented.

A vertex-complete oriented polygon is "almost" a simple polygon; that is: if P is a non-simple vertex-complete oriented polygon, then there is a near-by simple polygon which may be constructed from P by moving each instance of each vertex a distinct distance $\leq \varepsilon$, where ε is any fixed positive value so that the vertices become distinct. We may also introduce new edges between the new distinct vertices obtained from copies of identical vertices in a "faithful" manner so that these edges correspond to the edges in the original boundary sequence $\langle p_1, \ldots, p_n \rangle$, maintaining the original orientation.

Finding a Non-Crossing Path

Theorem: Every unorientable vertex-complete polygon can be converted into a coincident oriented vertex-complete polygon by reversing the directions of some of its edges and reassembling its edges in a suitable order to form a closed polyline-sequence.

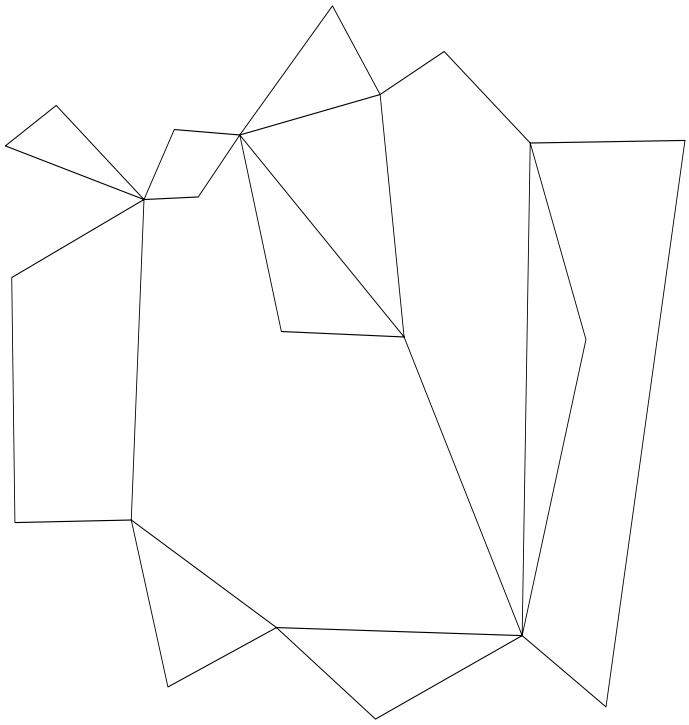
Note this construction corresponds to suitably shuffling the list of vertices that defines the originally-given polygon so that the pairs of vertices that define edges remain adjacent. Given this theorem, it is straightforward to establish the fact that a suitably nice closed curve can be traversed with no crossings. Also note that reversing some edge directions is sometimes necessary (consider a "figure-8" polygon.)

Consider a vertex v in the directed graph corresponding to a vertex-complete polygon P; an edge e is *in-directed to* v if $end(e) = v$ and *out-directed from* v if $begin(e) = v$. The number of in-directed edges of the vertex v is called the *in-degree* of v, and the number of out-directed edges of the vertex v is called the *out-degree* of v. The *degree* of v is the sum of the in-degree and the out-degree.

Note every distinct vertex, v, of a vertex-complete polygon P has an even number of (possibly non-distinct) edges (*i.e.*, v has even degree). This is because our polygon is defined by a closed polyline corresponding to a given edge-sequence so that whenever we have an edge entering a vertex, we have the next edge in the edge-sequence leaving that vertex, and since no edge can have zero length, the edges of the vertex can be classified as entering edges (in-directed edges) and leaving edges (out-directed edges) and there must be the same number of both. Thus each vertex of the directed graph corresponding to P likewise has even degree with its in-degree equal to its out-degree.

In general we can speak of a vertex-complete polygon or of the corresponding directed graph using essentially the same language, and we shall generally not distinguish between them.

You may want to try your hand at orienting a vertex-complete polygon by assigning directions to the edges in the graph shown below, and then ordering these edges in an admissible manner to produce an oriented vertex-complete polygon.



To prove the theorem, let P be a vertex-complete polygon. Let us assume for now that the polygon P has no coincident edges. (Later, we shall reduce the case of coincident edges to the case of no coincident edges.) Then the polygon P is directly interpretable as a geometrically-represented planar directed graph. Thus we have dual views of P as a geometric figure specified by various associated point-sets in $\mathcal{R}^2$, and also as an abstract directed graph. The directed graph for a vertex-complete polygon P necessarily has an *Eulerian circuit*, *i.e.* there is a path which contains every edge of P exactly once and ends-up at the starting vertex of the initial edge of the path. One such path is the polygon-boundary curve of P!

Every connected directed graph such that the in-degree of any vertex equals the out-degree of that same vertex possesses an Eulerian circuit. Clearly a vertex-complete polygon corresponds to such a *degree-balanced* directed graph.

It is a well-known fact that a connected degree-balanced directed graph G posesses at least one Eulerian circuit. To see this, consider a longest possible path of adjacent edges $e_1, e_2, \ldots, e_k$, where each edge in the path occurs only once. The last edge e_k *must* enter the vertex s that begins

the first edge, *i.e.* $end(e_k) = begin(e_1) =: s$. This is because when we enter a degree-balanced vertex, we are guaranteed to be able to leave it along an untraversed edge, except for the vertex we started at. Thus every visit to a vertex w corresponds to two edges in sequence of the form $edge[\cdot, w], edge[w, \cdot]$ in our path, *except* for a terminating visit to the starting vertex s, which thus must be the vertex that terminates our path. (Hence our longest Eulerian path is a *circuit*.) Also, every edge that exits from s is present in our path, since otherwise we could append such an edge after e_k and get a longer path contrary to our hypothesis that $e_1, e_2, \ldots, e_k$ is as long as possible.

Now suppose there is an edge $edge(a, b)$ of G not present in our longest Eulerian path. If some other edge exiting from the vertex a is in our path then, since our path is, in fact, a circuit, we can rotate our path to bring that edge to the front. Then the first edge in our rotated path is of the form $edge(a, \cdot)$, and just as we argued earlier, we can see that the last edge in our rotated path is of the form $edge(\cdot, a)$. But then we could add the edge $edge(a, b)$ at the end to form a longer path contrary to our hypothesis; thus *no* edge of our path exits from a. And moreover, no edge of our path enters a or we could rotate it to the end and then again conclude that we can append $edge(a, b)$ at the end to form a longer path contrary to our hypothesis.

Similarly, if some other edge entering the vertex b or leaving the vertex b is in our path then, just as above, we can arrange to add the edge $edge(a, b)$ to form a longer path contrary to our hypothesis; thus *no* edge of our path enters b or leaves b.

Thus, it must be that our degree-balanced graph is not connected, since otherwise we could enter the connected sub-graph containing the vertices a and b from some vertex v in our path and return to v (since every vertex is degree-balanced,) thus forming a longer Eulerian path. But this conclusion is contrary to our assumption that our entire graph is connected; thus it must be that every edge *is* in our path, and that we have an Eulerian circuit. $\boxed{\sqcap}$

Consider a vertex v in the geometrically-represented directed graph P; all the edges that come in to or go out from v are arrayed as the spokes of a wheel extending from v at various angles. And since P is a closed curve, there is an even number of such "spokes", ı.e., the degree of each vertex v of P is even with identical in-degree and out-degree. (Remember several edges can overlap, *i.e.*, emanate from v at the same angle.)

An edge e starting or ending at the vertex v has a *clockwise neighbor* edge defined as the next edge extending from v encountered as we rotate in the clockwise direction about v starting from a point on the edge e. Similarly, the edge e has a *counterclockwise neighbor* edge defined as the next edge extending from v encountered as we rotate in the counterclockwise direction about v starting from a point on the edge e. Note when the vertex v has only one in-directed edge and one out-directed edge, the clockwise neighbor edge and the counterclockwise neighbor edge of e at v will be one and the same edge. The clockwise neighbor edge and the counterclockwise neighbor edge will be called the *neighbors* of e at v.

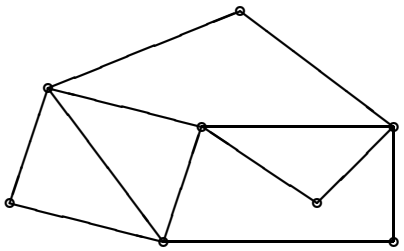
Now the process of constructing an oriented polygon coincident with the given vertex-complete polygon P consists of two major steps. In the first step, we assign potentially-new directions to each edge of P, and in step 2, we assemble these edges into an Eulerian circuit which never crosses itself.

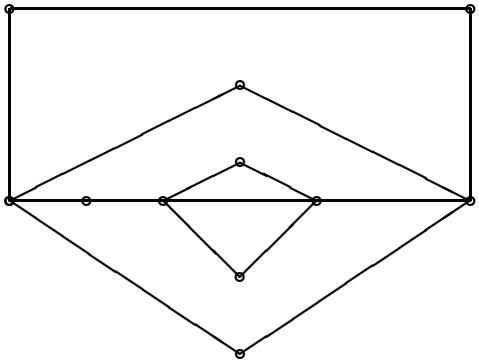
Let us discard knowledge of the directions of the edges of P so as to reduce P temporarily to an *undirected* graph P^u. Now we may choose any edge e of P^u and assign it a direction; then e becomes an out-directed edge of the vertex $begin(e)$ and e becomes an in-directed edge of the vertex $end(e)$. Then we can assign directions to all the other edges extending from the vertex $begin(e)$ by specifying that each neighbor of an out-directed edge is an in-directed edge, and vice-versa. Similarly, we can assign directions to all the other edges extending from the vertex $end(e)$ by following this same *alternating edge-direction rule*. By iteratively assigning directions to edges following this alternating edge-direction rule at each vertex where some edges have been assigned directions, we will end-up assigning directions to every edge in P^u, thus creating a new directed graph P'.

We must show that this edge assignment process is consistent. That is, we must show that no edge will receive conflicting directions. We will demonstrate this consistency by describing another way to assign directions to the edges of P^u which is clearly consistent, and which can be seen to be equivalent to the alternating edge-direction rule introduced above.

To begin, note that the vertex-complete polygon P *partitions* the plane into a collection of disjoint connected point-sets. The polygon-boundary itself is one such partition component and the infinite-area "outside"-region is another; the partition components that remain after excluding the polygon-boundary and the infinite-area region partition components are called the *open face* partition components of P, and their closures are called the *face polygons* of P; these are simple polygons. We will initially treat the boundary of a face polygon as an undirected graph; later we will introduce directions for its edges so as to form an oriented simple polygon.

Let $P_1, \ldots, P_h$ be the face polygons of P. We suppose that $h \geq 2$; otherwise P is a simple polygon, and there is nothing to be proved. Each face polygon P_i shares either a single vertex or exactly one entire polyline-sequence of one or more edges with some other face polygon; there may be several other face polygons with which P_i shares a set of vertices or one or several sequences of edges, but, of course, no edge is shared by more than two face polygons. Each shared edge is associated with an edge of P itself. Note, if a face polygon P_i shares several vertices and no edges with some other face polygon P_j, then there is necessarily at least one interposing face polygon P_k which shares a single polyline-sequence of edges with P_i. Also, if a face polygon P_i shares several separate polyline-sequences of edges with some other face polygon P_j, then there is another face polygon P_k which shares just a single polyline-sequence of edges with P_i. The examples below show instances of these various adjacency relationships among face polygons.





If no pair of face polygons share an edge, then each intersecting pair of face polygons meet solely at a vertex. Let us suppose this latter situation is not the case. Then there are two face polygons P_i and P_j which share a single polyline-sequence of edges. Let us select an edge e shared between these two face polygons P_i and P_j, and let us assign a direction to the edge e, thereby specifying the vertices $begin(e)$ and $end(e)$. The direction chosen for the shared edge e determines the orientation of the adjacent face polygons P_i and P_j. If P_i lies to the left of the directed edge e then P_i is counterclockwise-oriented, and P_j is clockwise-oriented. Otherwise, P_i is to the right of the edge e and is clockwise-oriented, and P_j lies to the left of e and is counterclockwise-oriented. All the edges of P_i and P_j thus have directions imposed which correspond to the orientations required for P_i and P_j.

Now, given the postulated shared directed edge e, consider traversing the edges that bound the face polygon P_i starting with the edge e. Suppose P_i is to the left of the edge e. Let us start at the vertex $begin(e)$ and traverse e to enter the vertex $end(e)$. The counterclockwise neighbor of edge e at the vertex $end(e)$ must be the edge of P_i that follows the edge e on the boundary of P_i, and thus this edge must be out-directed. Similarly we see that the clockwise neighbor of the edge e at the vertex $end(e)$ is the following edge in the polygon-boundary curve of the face polygon P_j and must be out-directed from $end(e)$. Note the alternating edge-direction assignment rule is consistent with the requirement that both the clockwise neighbor and the counterclockwise neighbor of $end(e)$ (which may coincide) be out-directed. Also note the outer-boundary of the assembly of P_i and P_j is the boundary of an oriented polygon.

Now we can build the directed graph P' by joining together the face polygons in $\{P_1, \ldots, P_h\}$ one by one, so that each added face polygon P_a joins the already-assembled face polygons either along a polyline-sequence of edges on the outer-boundary of this assembly or solely at a single vertex on its outer-boundary. We begin by selecting one face polygon and assigning directions to its edges that orient it with either a clockwise or counterclockwise orientation. Then we successively join additional face polygons that meet our assembly constructed so-far, either at a vertex or along a sequence of edges.

Thus we have two cases: either the newly-joining face polygon being added meets the current assembly solely at a vertex, or it meets the current assembly along a polyline sequence of edges. Suppose the newly-joining face polygon P_a being added meets the current assembly solely at one

vertex. Then we will assign the unique direction to the edges of P_a which is consistent with the alternating edge-directions rule at the shared vertex.

Alternately, suppose the newly-joining face polygon P_a being added meets the current assembly along a single polyline sequence of edges. The direction of the edges of the newly-joining face polygon P_a is then determined by the direction already assigned to the edges where P_a is joined.

At the conclusion of this assembly process, we have the directed graph P' whose edge-directions are locally consistent with the alternating edge-direction assignment rule. Clearly, the construction of P' consumated by assembling the face polygons of P is incapable of introducing an edge that requires conflicting directions be assigned to it, and hence the alternating edge-direction assignment rule is likewise globally consistent. Indeed, constructing P' by assembling the face polygons of P is just executing the process of assigning edge-directions to the edges of P^u via the alternating edge-direction assignment rule one edge at a time at the vertices of each added face polygon which meets the current asssembly along a sequence of edges, and two edges at a time at the joining vertex for each added face polygon that meets the current assembly at only at a vertex.

Note we can use an alternate approach to constructing the directed graph P' by assigning an orientation to the outer-boundary of P^u which, in turn, induces orientations for each interior face polygon, starting with those adjacent to an outer-boundary face polygon. Thus, instead of constructing P' from its face polygons, we can construct P', by decomposing P into its face polygons starting from those that define the outer-boundary of P.

Finally, we can accomodate the case where P has coincident edges by separating these edges with our mathematical scapel to make them distinct, and placing new degree-2 vertices within them to allow them to be represented by pairs of straight-line segments. This reduces our polygon P to a vertex-complete polygon without coincident edges and the construction above then applies. (We may then collapse these near-coincident edges back to form a "stack" of 0-area face-polygons.) This concludes step 1 of our proof.

Let P be a vertex-complete polygon with no coincident edges, and let V be the number of vertices, E be the number of edges, and F be the number of face polygons associated with P. The construction of P' by assembling face polygons presented above provides a proof of Euler's relation: $F-E+V = 1$, since Euler's relation holds at each step throughout the assembly of the coincident graph P'. (If we add an m-sided face polygon that shares a polyline sequence of k edges with the existing assembly, then we add $m-(k+1)$ vertices, $m-k$ edges, and 1 face, leaving $F-E+V$ unchanged.)

Note if we include the infinite area partition component as a face polygon, we have a graph model for a polyhedron, where the infinite-area face polygon represents the "back" face. In this situation, we have the variant Euler relation for a polyhedron: $F-E+V=2$.

Now we may proceed with step 2 of our proof. If we can order the edges of P' to form an Eulerian circuit that never crosses itself then we will have constructed an oriented polygon coincident with P. To do this, let us construct a father-directed *spanning tree* for the directed graph P', where each edge of the spanning tree is an edge of P', and is directed to the father node in the spanning tree. Each vertex of P', except the vertex r that serves as the root of the spanning tree, has exactly one *tree-edge* exiting from it. Given this spanning tree, we can construct an Eulerian circuit in the

directed graph P' which can be taken as the polygon-boundary curve of a polygon P' coincident with P which is a vertex-complete oriented polygon! (A father-directed spanning tree can always be found for a degree-balanced directed graph possessing an Eulerian circuit. This can be shown by considering an Eulerian circuit of P, and collecting all the edges e in the Eulerian circuit which are the *last* exits from their beginning vertex $begin(e)$. See [Knu73].) Another effective way to construct the desired spanning tree is to pick an arbitrary vertex as the root, and take all its incoming edges as the edges that connect to the next level of vertices in the tree. We proceed in this way for each subtree, but we exclude incoming edges that originate at vertices already incorporated in the tree.

To construct the desired Eulerian circuit, we start at the vertex r which has been chosen as the root of the spanning tree. There is at least one out-directed edge at r (which is necessarily not a tree-edge *i.e.* an edge in the spanning tree.) We may begin our Eulerian circuit with any out-directed edge from r, and honor the following rule as we travel from vertex to vertex along previously-untraversed edges.

Suppose we have traveled along an edge f to arrive at a vertex p; the traversal rule is: to leave the vertex p, we must select an out-directed edge which is an untraversed neighbor of f at p, with the constraint that we *do not* select the tree-edge unless no other untraversed out-directed edge exists at p. An edge g is an *untraversed neighbor* of an edge f at the vertex p if g is the next edge encountered which has not been traversed earlier as we rotate either clockwise or counterclockwise about p starting at the edge f. (Careful study of this traversal rule might allow us to count the number of non-crossing circuits of P in the same way we can count the number of Eulerian circuits [Knu73].)

Theorem D in [Knu73] tells us that this traversal process is guaranteed to terminate at the root vertex r after having visited all the edges of P'. Moreover, although the Eulerian circuit thus constructed may touch itself, it does not cross itself, so the associated vertex-complete polygon is oriented. The proof of Theorem D that we have traversed an Eulerian circuit of P' goes as follows. Let $e_1, e_2, \ldots, e_k$ be the edges we have traversed until we stopped due to being unable to proceed further. Now suppose an in-directed edge $edge(\cdot, v)$ entering the vertex v is not in our path. Then there must be at least one out-directed edge from v which is also untraversed, and, in particular, the tree-edge $edge(v, z)$ exiting from v must be an untraversed edge, since it is required to be the last out-directed edge traversed from v. But now there is an untraversed in-directed edge entering the vertex z. By repeating this argument, vertex by vertex, we may conclude that there is an untraversed in-directed edge entering the root vertex r, and hence there is also an untraversed out-directed edge from r, and therefore the path we have generated does not correspond to a complete traversal, which we assumed we had! In order to avoid this contradiction, it must be that no edge of P' is untraversed. $\boxed{\sqcap}\boxed{\sqcap}$

Note that the face polygons of a vertex-complete oriented polygon P, with their edge-directions inherited from their directions in P, are oriented polygons, and that each clockwise-oriented face polygon is bordered at each *edge* by a counterclockwise-oriented face polygon, and conversely. This ensemble of face polygons form a tiling of the region enclosed by the outer-boundary of P (potentially including 0-area face polygons resulting from coincident edges.) and this tiling can be 2-colored. It may be that the constructions discussed here can help in elucidating aspects of the four-color theorem. (Note two α-colored regions are separated by a "stack" of one or more

0-area face polygons colored $\beta, \alpha, \beta, \ldots, \alpha, \beta$ in sequence. Is there a way to use this information to locally change the color of one of the α-colored regions to have color γ or δ in order to achieve a 4-coloring? Perhaps the decision is related to the parity of the number of such stacks between the two α-colored regions?)

References

[Knu73] Donald E. Knuth. *The Art of Computer Programming: Volume 1, Fundamental Algorithms, second edition.* Addison-Wesley, New York, 1973.