

Techniques with Convex Polyhedron

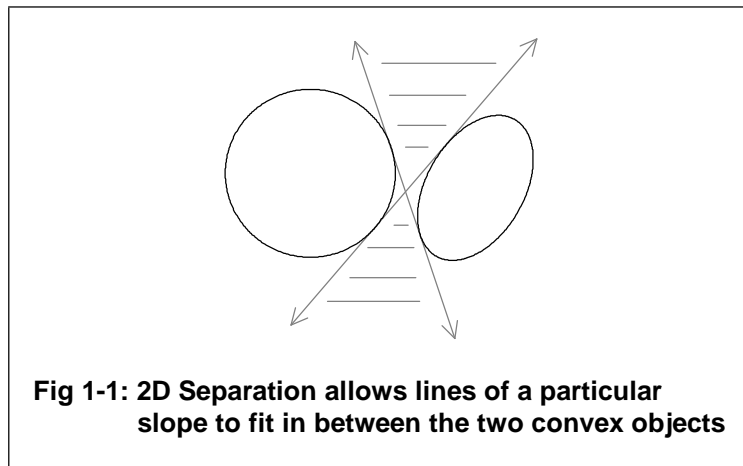
Introduction

One component of drawing 3D objects is sorting them so that they are drawn in the correct order.¹ The closer objects must be drawn first to insure that they hide the more distant objects. This problem led me to study the features of Convex Solids and Polyhedra to find a fast and efficient method of solving this problem.

What's in Front?

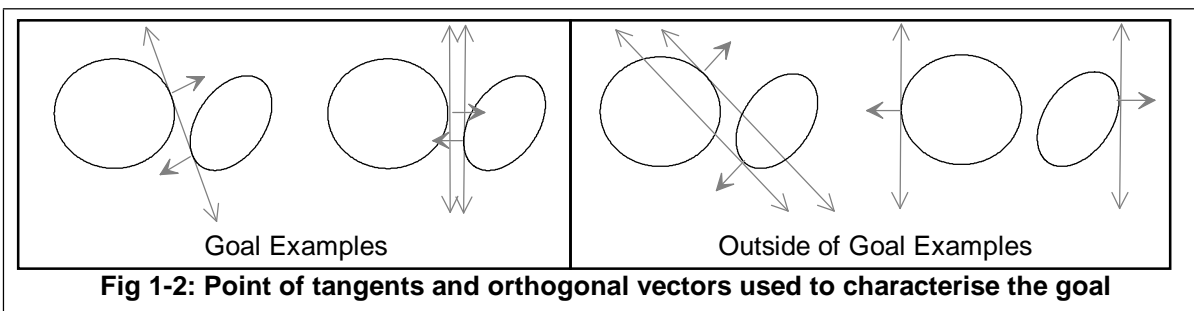
To determine what is in front, we must observe the space that resides between two objects. In 2D, this space can fit lines of particular slopes, but not of others. Given the slope of one of these lines, we can tell which object is seen in front of the other object. Taking the vector representing the slope of the line and taking the cross product of this with the visibility vector will give a positive or negative value depending on which side of the line is being seen (and the forward object will be on the side of the line being seen).

A similar effect can be witnessed in 3D convex objects. There is also a space between them, however instead of filling it with lines with particular slopes, it can be filled with planes with particular orthogonal (normal) vectors. Given this normal vector and the display vector, the dot product can be taken to find out which side of the plane is being seen. The object on the viewed side of the plane is the forward object.



Characteristics of the Goal

To find one of these slopes describing the separation of the two objects, we need to adequately characterize it so that we know we've found it using any given search algorithm. In the 2D case the tangents of each object with the same slope as the lines face each other. In addition, a line drawn from each object's tangent point should not go through the other object. Notice that this relates the objects' orientation to the slope of the lines found in between.



¹ Z-buffering doesn't work for objects that are really close together.

This same statement holds in 3D except that instead of using lines tangent to the object, planes are drawn tangent to each object.

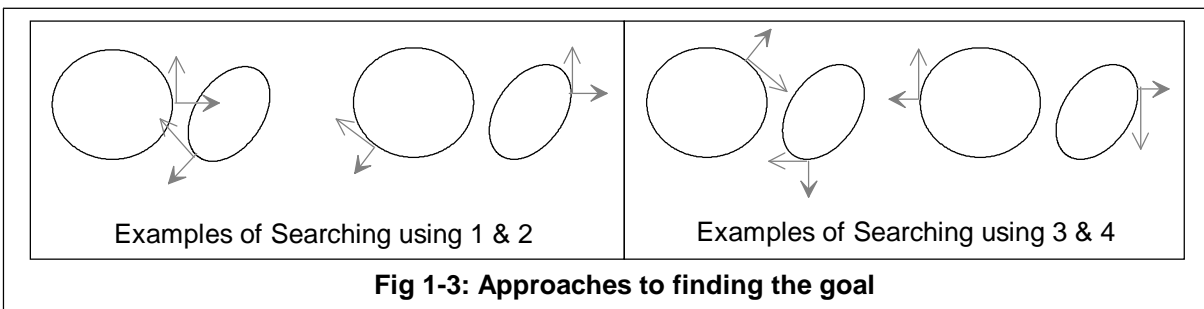
Finding the Goal

Finding the goal starts with the problem of determining a starting point. The most obvious place to start would be a random position on the surface of the object. This certainly does suffice, although by restricting the random position to being on the objects' surfaces that face each other, the points would typically be placed much closer to the goal. Other ideas may involve the vector from the center of one object to the center of the other and finding the places on each object whose tangent normal points in the same direction as this vector. To be general, in my discussion I'll start from any point on the surface of the objects.

Next a plan of action must be devised to reach the goal. This can be difficult because there isn't any really clear *best* way to proceed. It can be a combination of the following steps, but how should they be placed together?

- 1) Move along surface of object 1 in a direction such that the dot products of the vector normal to the tangents to the points of each object approach -1.
- 2) Move along surface of object 2 in a direction such that the dot products of the vector normal to the tangents to the points of each object approach -1.
- 3) Move along surface of object 1 in a direction such that the dot product of the vector between each tangent point approaches zero.
- 4) Move along surface of object 2 in a direction such that the dot product of the vector between each tangent point approaches zero.

In addition to these directions to follow, there should be an additional rule to never violate a condition of the goal once it has already been met.



It can be seen through careful thought, that whether or not we start with 1 & 2 and finish with 3 & 4, or we start with 3 & 4 and finish with 1 & 2, we'll eventually arrive at the desired goal. Notice however that if we do these steps alone, meet a condition of the goal, and finish with the remaining two, there is always the possibility that we'll first move away from the goal, and have to backtrack.

To fix this inefficiency in movement, we can move such that as many rules are satisfied as possible. For example, when choosing between 1 & 2, we try to match either 3 or 4. We must be careful because moving both at the same time could eventually introduce a location where it can't improve. In this case it is best to move as well as possible for 3 and/or 4 and then use the combination of the two.

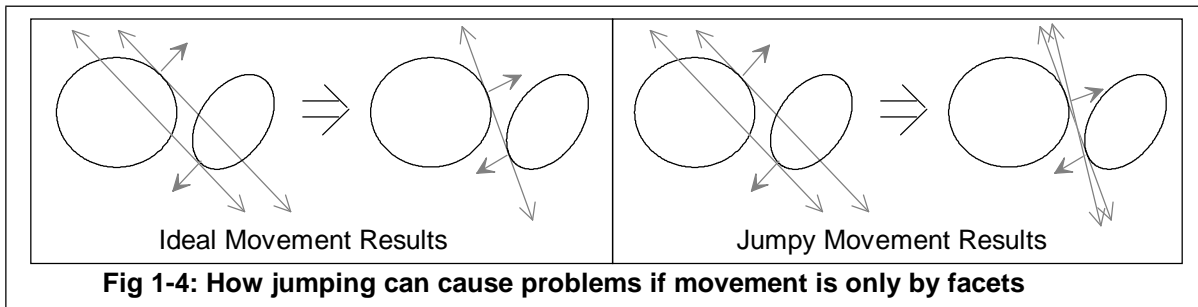
Introducing Polyhedra

So far I've built the concepts using continuous surfaced convex solids. Dealing with polyhedra is the same except that they are extremely flat in particular locations (facets) and have rather sharp lines (edges) and points (vertices). The easiest way to make the transition is by imagining that the edges are actually rounded (many facets which curl from one facet to the next) and the points are also rounded (many facets which gradually form the connection between all the connected edges).

This transition to polyhedra is necessary for multiple reasons. One is that most 3D graphics is inevitably based on drawing polyhedra, not contours or solids based on formulae. Another is that moving along detailed surfaces makes it more difficult to find the goal location because a great deal must be traversed before the exact location is found², so it makes sense to partition it into chunks that approximate the surface. These chunks have enough information to put us in the correct region, and then the *goal* chunk can be replaced by smaller chunks, allowing us to continue searching in a divide and conquer manner until we have come to a satisfactory location.

Moving Between Polygon Facets

Moving between the facets can be challenging simply because taking larger steps can result in passing up critical points on the convex polygon. Recall that before we were considering movement in very small increments, now the steps can be leaps in comparison! For this reason, it is good to consider what would happen if we skipped ahead large amounts in our previous discussion on finding the goal. Then ask ourselves a few questions, such as, "How would we backtrack if we overran some point?" "Should we backtrack if we overrun some point?" "What if a point is in the center of a facet?" "What if a point is on an edge or a vertex?"



There are a few problems we encounter from jumping around. One problem is that we will almost definitely miss the condition where the two normal vectors to the tangent points are pointed directly at each other. We also have to figure out how to test for all of the points on a facet and/or edge so that we know if the points we think reside on them are between the objects and not off to the side of the objects.

² Practically, you'd have to break up the surface anyway, otherwise you'd have an infinite number of steps to make to reach the goal. Just keep in mind Zeno's Paradox, except to the extreme where you'd have to keep dividing your first step in two to insure you were moving in a gradual manner.

Modeling Movement Along a Convex Polyhedron

If a sphere was drawn and considered, the dot product of a particular vector with the normals of its surface would cause a contour of values across its body. The point where the normal is in the same direction as the vector, given that all vectors have a magnitude of one, would have a value of one. Where the vectors pointed in opposite directions, the value would be negative one. An equator would be formed along the center with a value of zero.

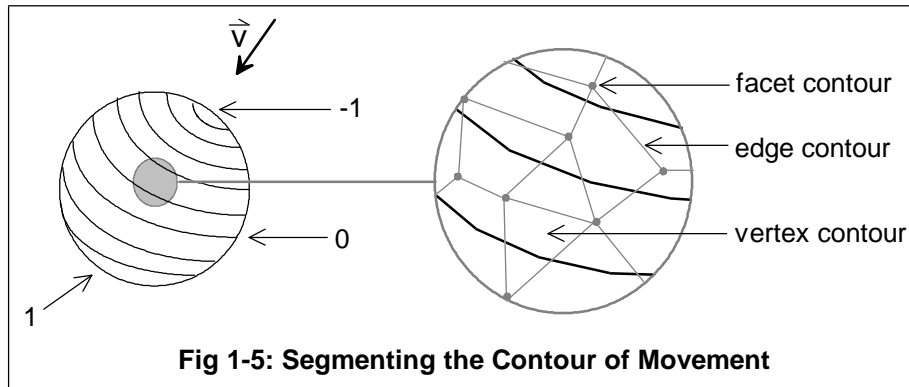


Fig 1-5: Segmenting the Contour of Movement

If we subdivide this sphere into sections, we can make some interesting observations. The wide area regions of contained contours appear to represent points which connect facets and edges together³. The edges of the areas appear to represent edges which join points together, but most importantly are found between facets. The points appear to represent facets which are flat, so they only consist of one value.

This representation of the contour with corresponding relationships can help us visualize the effect of jumping from face to face on a convex polyhedron, at least for movement rules 1 & 2. Faces have a single value and appear far apart. This makes them ideal for jumping around the polygon, however what would represent the proper way of moving between faces? What if the normal is on an edge or vertex?

Moving among the faces resembles a typical graph (points connected by a net of lines) problem. For a complex polyhedron, the points are represented by convex polygons⁴, this means there should always be a path jumping from points up to the top contour point of -1. Once we reach the top, there are only three possible features that can contain that point, a vertex, an edge, or a convex polygon. If it is a vertex, then when we reach that facet, we'll be at -1. If it is an edge, then two consecutive facets of the convex polyhedron will be sharing an edge which contains the desired point. This can be detected by finding the vector which points along the edge and taking the dot product of it, returning a zero if it can contain the needed normal vector. To know for certain, the dot product of the two neighboring faces must be taken to insure that the edge travels across the top of the contour by returning the sum of the contour values of the facets. If it is a convex polygon, then a vertex will contain the point. This can be found by circling about a point with the edges and seeing if the vectors along the edges all have a component pointing in the opposite direction as the given vector (using the dot product).

³ It should also be noted that points must be convex polygons to be on a convex polyhedron. This is very important because the search problem can become complex if a depth first search cannot be used.

⁴ I've not formally proven this, so I *could* be wrong. Picturing it mentally, it seems to make perfect sense.

When to Check More Than a Facet

To operate as quickly as possible, it would be best to jump around on facets for as long as possible. This insures that the tests only require the dot product of the normals of the opposing faces and a test to make sure they are not pointing away from each other, but towards each other. The trouble found by jumping around on facets doesn't appear while we simply move around according to the contour given above for test 1 & 2. It's when we test for 3 & 4, that the situation becomes more complex.

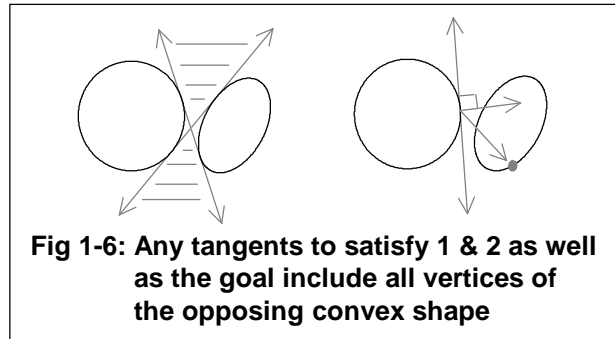
The problem is that there may be a place where moving to satisfy 1 & 2 may conflict with trying to satisfy 3 & 4 and we need to know how this condition should be interpreted. To begin with interpreting this condition, let's start with reexamining how we came up with our 4 rules of finding the goal.

The first characteristic to observe is that before the goal can be reached, the tangent line (plane) must point so as not to intersect the opposite object. The dot product can be used to determine which side of the line (plane) a vertex lays on by using a point on the tangent facet to the opposite object's vertex and the vector perpendicular to the line (plane). Rules 3 & 4 are designed to make the tangent line (plane) fit this condition, and if this condition is not held for all of the points on the opposite object, then there is no point in finding the vectors which lie tangent to each other.

There is a case where rules 3 & 4 cannot hold when jumping from face to face. This is where the tangent lines (planes) all lie on either an edge or a vertex. In the case where the tangent lines (planes) lie on an edge, the test will cause it to go so far and then while trying to satisfy rules 1 & 2, it will want to move back on itself to the previous polygon. In the case where the tangent lines (planes) lie on a vertex, the test could either cause it to try backing up onto a previous polygon or circle around a vertex. In either case, the revisiting of a previously visited facet is the cue to moving onto an edge or vertex.

The second case is where the facet with the closest to the desired normal vector cannot be found. This can be simplified if all of the points in the opposing facet are contained on the correct side of the current facet. If this was true, then we'd know that one of the vertices that was tested had to be the *closest* vertex, and it is contained, and one of the edges that was tested had to be the closest edge, and it is also contained. The result is that we know that the plane which defines the facet properly divides all of the points in both objects. If this test comes up false, then the tests for edges and vertices given in the previous section must be used⁵.

When testing to see if a facet properly divides all of the points in both objects, all of the points need to be detected. A good observation is that 3 & 4 is violated if any points do not exist on the correct side of the plane. This could be another way of finding the correct edge or vertex with the tangent dividing plane.



⁵ I don't think these will fail because of rules 3 & 4, but if they do, then more has to be done.

Finding the Dividing Plane's Normal

Once the planes, edges, and/or vertices have been found which help define the dividing plane, the information must be deciphered to find the normal of that plane. There are four cases which can be easily verified⁶ shown in table 1-1.

Table 1-1: Steps to find the dividing plan's normal.

Answers	Finding the Dividing Plane's Normal
a facet and anything else	Use the normal vector of the plane defining the facet.
an edge and an edge	Use the vector orthogonal to the two lines that make up the edges.
an edge and a vertex	Use the vector orthogonal to the vector of the line that makes up the edge and an orthogonal vector of the vector formed by a point on the edge to the vertex and the vector of the line that makes up the edge.
a vertex and a vertex	Use the vector formed by a line drawn between both vertices.

The way to visualize can be as shown below.

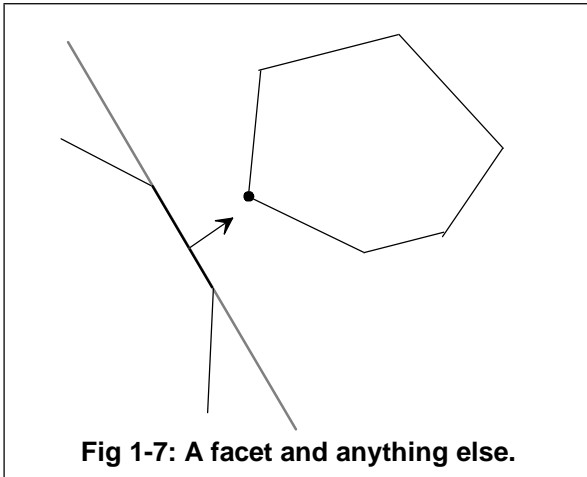


Fig 1-7: A facet and anything else.

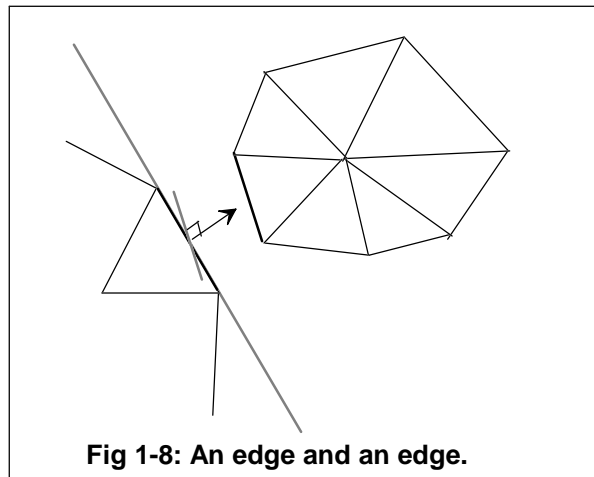


Fig 1-8: An edge and an edge.

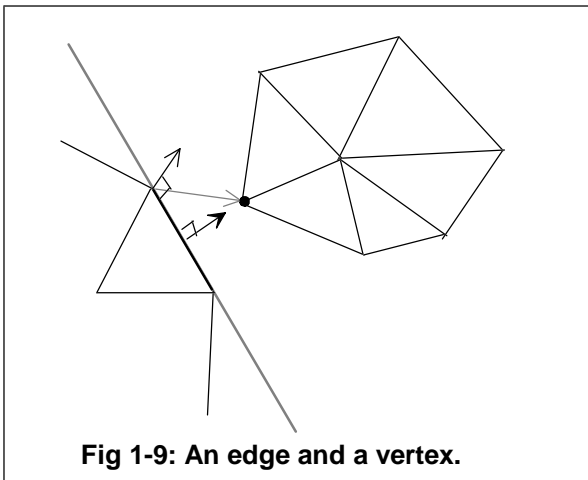


Fig 1-9: An edge and a vertex.

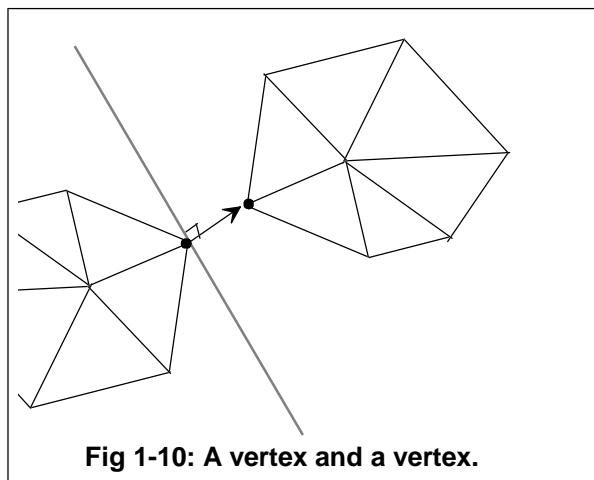


Fig 1-10: A vertex and a vertex.

⁶ Well, maybe not easily verified mathematically, but visually.