UNIVERSITY OF WATERLOO

CS 759 Report: Killing Time

Theo Belaire
20415730
Bryan Coutts
20428420

April 7, 2015

Some terminology, I use the term *included* point to refer to points that are to be included inside the blob, and *excluded* points to refer to points that must be on the exterior of the blob. Unless specified, polygon can mean non-convex polygon.

Our algorithm works as follows:

1. Take the convex hull of the included points.
2. Fix the convex hull, so that no excluded points are in its interior.
3. Compute the radii for each point.
4. Induct nearby points to the polygon.
5. Remove "crossover" points.
6. Draw blob around the vertices of the polygon.

1. GENERATION OF THE CONVEX HULL

Our code uses the giftwrap algorithm to generate the convex hull. This runs in $\emptyset(i^2)$ time, where i is the number of points included in the set.

It works by picking the leftmost point, and calculating the angle that is formed with each other point in the set, and picking the point that forms the angle closest to a straight line with the previous point.

At the conclusion of this step, we have a list of included points in clockwise order, which forms our polygon P. Currently it is convex.

2. Point Exclusion

In this step, we will fix P, so that no excluded point is in its interior. We will do so by removing triangles from P; this will cause it to no longer be convex.

To test if an excluded point is within P, we use the Jordan Curve Theorem. One of its consequences is that if we trace a ray from the point in any direction, it will cross an edge of P an even number of times if and only if the point is outside P. This is computationally simple to check.

For each excluded point p in the interior of P, we find the edge closest to p. We consider only the edges whose endpoints p is between. We then insert p into our list of polygon vertices, between the endpoints of this edge. This removes the triangle formed by the edge and p from P.

3. Calculating Radii

The *radius* of a point determines how large a circle is used when drawing the blob around it. If all the radii were very small, the blob would appear the same as the polygon. However, the radii cannot be too large, otherwise the circles draw around included and excluded points will intersect with each other. Hence, the radius of an included point is bounded by its distance to an excluded point, and vice-versa. Moreover, there are situations in which the radius of an included point must be bounded by its proximity to another included point, as illustrated in 3.1.

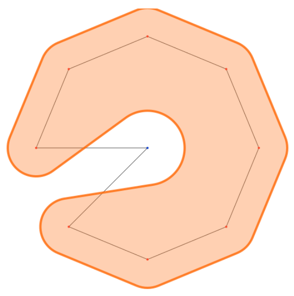

Figure 3.1: "If the radii of a, b are too large, the blob will self-overlap"

For included points a, b, a must bound the radius of b if and only if the line segment from a to b is contained entirely within P. Similarly, for excluded points a, b, a must bound the radius of b if and only if the line segment from a to b is disjoint from P. We check this condition by checking if any edges of the polytope intersect with the line segment from a to b.

Given this notion of which points bound each others' radii, we choose the radius of a point to be 1/3 of its distance to the nearest bounding point. This ratio must be at most 1/2 to avoid overlap. The smaller the ratio is, the thinner and blockier our blob will be. 1/3 allows for a curvy blob, while avoiding some pinching effects observed with higher ratios.

4. INDUCTING NEARBY POINTS

Once we have all the radii, we almost know the final shape of the blob. However, there may be excluded points near the edge of the polygon, whose circles (or even the points themselves) may intersect the blob, as seen in 4.1. There may also be included points near the border, whose circles would intersect the exterior of the blob. We add all such points to the polygon.

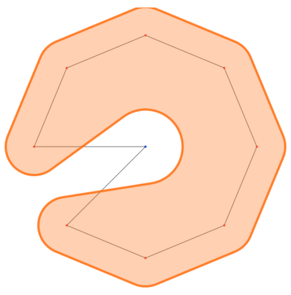

Figure 4.1: "The circle of the bottom-right point is cut by the blob"

We iterate over all excluded points p. Then, for each edge $e = (a, b)$ of P, we check if the line from the edge of the circle of a to the edge of the circle of b intersects the circle of p. If it does, we add p to P, between a and b (effectively adding the trangle with vertices a, b, p to P).

5. UNCROSSING

6. ARC COMPUTATION

Once we have a final boundary, and radii, we can draw the blob.
For each edge uv, we consider a pair of circles centered at u and v, with the radii r_u and r_v. Consider as if we were to take a piece of string and wrap it around the outside of these circles. It would have three segments, first when it was in contact with the first circle, then it would leave and travel along a line tangent to both circles, then it would arrive at the second circle and travel along it before leaving somewhere on the other side.