CS3451 — Fall 2008 — Q03 NAME: **Solution to Q03** Grade: out of 20

Write legibly. Use points, vectors, and topological operators discussed in class, such as .x, n(), U(), V(), R(), +,–,• .i, b, .k... R(V) denotes the version of vector V obtained by rotating it 90 degrees to the right. Let **P** be a simply connected polygon with vertices {P[0], P[1],… }. P.length denotes the number of entries in P. Given index i of P[i], let p(i) and n(i) return the indices of the previous and next vertex around **P**. Hence P[p(i)], P[i], P[n(i)] are three consecutive vertices along the bounding loop of **P**. Each one of the 6 questions is worth 4 points . Hence, you need only 5 correct answers to get 20. Your grade will be capped to 20 points.

---

1) <4 points> Assuming **P** is clockwise, provide a detailed algorithm to test whether P[i] is convex?

boolean convex(int i) { return R(V(P[p(j)],P[j]))•V(P[j],P[n(j)])>0;} i.e., B is convex when R(AB)•BC>0. If P was counterclockwise, then the test would be R(AB)•BC<0.

---

2) <4 points> Provide a detailed algorithm for testing whether **P** is clockwise.

boolean clockwise() { int j=0; for (int i=1; i<P.length; i++) if (P[i].x<{P[j].x) j=i; return convex(j);}

Here we compute the index j to the left-most vertex, which we know must be convex.

From question 1, we know that convex(j)==clockwise. Hence, we return convex(j).

Another possible approach is to compute the signed area and check the sign.

---

3) <4 points> Let A and B be two closed sets. Use Boolean and topological operators (.i, .b, .e …) to decide whether A and B

- are disjoint when: A.i∩ B.i=∅ && A.b∩ B.b=∅ or more concisely when A∩ B=∅
- touch when: A.b∩ B.b≠∅ (boundaries intersect) && A.i∩ B.i =∅ (interiors do not overlap)
- interfere when: A.i∩ B.i ≠∅ (interiors overlap)

---

4) <4 points> A photon starts at point P and moves with constant velocity V in the plane. Write the pseudocode for testing whether t will hit a disk of center C and radius r

boolean hit(pt P, vec V, pt C, float r) { return (PC•V>0) && (U(R(V))•PC≤r); }

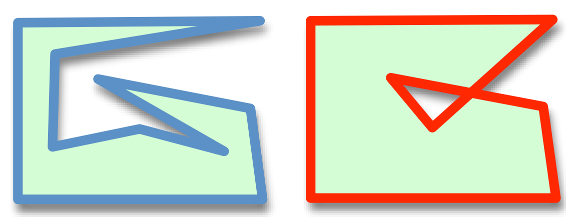
The above solution has two problems: (1) It will produce the wrong result if P in the disk. (2) It correctly rejects hits when the disk lies on the right side of the trajectory, but not on the left. We may fix (1) by stating that we assume that the photon is initially out of the disk or by testing for it. We may fix 2 by using absolute value or squares. Hence

boolean hit(pt P, vec V, pt C, float r) { return d(P,C)<r) || ((PC•V>0) && (abs(U(R(V))•PC≤r)); }

---

5) <4 points> Show (figure with counter-example and brief explanation) that the following algorithm may fail to compute the convex hull of **P**.

```
void hull() {
  while(P.has_a_concave_vertex()) {
        for (int i=1; i<P.length; i++) if(!convex(i)) mark(i);
        P.remove_all_marked_vertices() ;};}
```



The red (self-intersecting) polyloop is obtained by removing the 3 concave vertices of P (blue). All the vertices of the new red polygon are right turns and hence are marked as convex by our local test. We can fix this algorithm by removing concave vertices one by one, but only when doing so does not create a self-intersection. The corresponding test amounts to testing whether there is any other vertex in the triangle spanning the candidate concave vertex and its two neighbors.

---

6) <4 points> Given 3 rectangles (A, B, and C) as shown. Express the shaded area S both in CSG and BSP form, such that each symbol A, B, and C appears only once in each form.

CSG: Desired solution A–(B–C). Less compact solution: (A–B)∪(A∩C)

BSP: < < < 1 , C , 0 > , B , 1 > , A , 0 >, with convention < A.i , A.b , A.e >