Problem set 3

Due **1pm, Tuesday December** 11th, 2001.

Required Part

Problem 1 [20 points]. Exercise 11.6, p. 250. Describe an efficient data structure that allows you to test whether a query point q lies inside a convex polytope in $\Re^3$. Hint: use point location structures.

Problem 2 [10 points]. Prove (in detail) that the Minkowski sum of two convex polygons is convex.

Problem 3 [20 points]. Exercise 14.5, p. 304. It is possible to reduce the size of a quadtree of depth d for a set of points inside a square, from $O(dn)$ to $O(n)$. The idea is to discard any node v that has only one child under which points are stored. The node is discarded by replacing the pointer from the parent of v, to point to the only interesting child of v, not v itself. Prove that the resulting tree has linear size.

Can you also improve upon the $O(nd)$ construction time ?

Optional Part

Problem A [30 points]. Given n points in $\Re^d$, and a parameter r, give a (possibly randomized) $O(d^{O(1)}n)$ time algorithm which does the following:

- if there is a pair of points within distance r from each other, output YES
- if there is no pair of points within distance $d^2 r$ from each other, output NO.
- otherwise, the output can be arbitrary.

Note: The running time should be *polynomial* in d.

Problem B [10 points]. Consider a set P of points in the *discrete* space $\{0 \dots U\}^2$. A *hardcore quadtree* for P is a quadtree constructed in such a way that a node is split as long as it contains some points in P and some not in P. E.g., even if P contains only one point, the hardcore quadtree procedure will continue splitting until it creates a node containing a single point. Given two hardcore quadtrees T_1 and T_2 representing point sets P_1 and P_2, show how to construct a hardcore quadtree T representing $P_1 \cup P_2$. Your procedure should run in time linear in the sizes of T_1 and T_2.

In addition, consider a *naive* quadtree. The latter continues splitting a node, even if all points in the node belong to P. Show that the size of a naive quadtree for P is always at least $|P|$. What is the largest cardinality of a set which can be represented by a hardcore quadtree of constant size ?

Problem C [20 points]. Problem 15.2, page 317.

Programming Exercise.

The programming exercise this time has these objectives:

1. Understand one frequently used line duality.
2. Construct and query an arrangement of lines.
3. Implement interaction subject to geometric constraints.
4. Provide continuous visual feedback.

For this assignment, you'll need a complete implementation of the Convex Hull code from PS1, and the DCEL/BSP code from PS2 – one that explicitly generates a DCEL face for each cell of BSP tree. If your own code does not do this, feel free to adopt or use (with acknowledgement) any of the "exemplary" solutions (from Matt Seegmiller, Daniel Vlasic, or Jason Yang), or the demo from the Arrangements lecture (L9, by Darius Jazayeri), all of which are posted to the course web page.

Briefly, you'll implement primal and dual views of a plane space populated by various interactively specified geometric primitives. You'll construct and query some non-trivial assemblies in both the primal and dual space. Below, you'll find suggestions on how to proceed, but of course you may design and implement things however you wish. The imperative statements below describe the capabilities of your applet that are required for credit.

Visualization Mode.

Implement two drawing regions, labeled Primal and Dual. Whatever interval you choose to display in the Primal, choose the Dual interval appropriately, so that the "interesting behavior" in the Dual will not occur outside the drawing region.

Enable the user to enter points, lines, and segments in the Primal region. For each, draw the Dual (line, point, double wedge respectively). Restrict the line and segment specification to bound the magnitude of the Primal slope so that the Dual point ends up inside the drawing area.

Enable mouse-over highlighting, and selection editing, of both Primal or Dual objects, *with the appropriate update of the corresponding object in the other space.* That is, continuously highlight the object (and its Dual) that would be selected if the user mouse-pressed there; if the mouse-press happens, select and edit the object.

Note that the editing step requires you to implement geometry constraints. For example, when the user edits a segment you should allow a selected endpoint to be moved in 2D, or in 1D – along the segment itself, changing only the segment length. You'll probably need to use two mouse buttons, or a Shift/Control/Alt keypress to discern the user's intent here, unless you can think of a better way.

You should also support selection of a segment at or near its midpoint, and allow rigid translation of the segment with no rotation.

Similarly, when editing a wedge in the Dual space, either boundary line of the wedge should be selectable, after which it can be rotated about the wedge fulcrum (careful: should the user be allowed to rotate either wedge boundary through the vertical?). You should also support highlighting, selection, and translation of the wedge fulcrum itself, which will translate the whole wedge.

Finally, enable the user to specify the boundary of a triangle, and of a disc of radius r, in the Primal or Dual, and display its dual. (Note that both boundaries are one-dimensional point families, so their duals are one-dimensional line families.)

Design your code carefully so you don't end up writing several small variations of your mouse-over, selection, and editing code.

Composition and Query Mode.

Now we're ready to compose primitives.

First, support **insertion** of Primal points and segments. For each, use your BSP data structure to insert the corresponding line or wedge in the Dual. (You'll need to insert one or two "segments" spanning the entire Dual region into the BSP tree.)

Now go back to Visualization mode (i.e., in which sweeping an object doesn't insert it, but just displays it), and enter a Primal line. Highlight the Dual cell in which the line's Dual point lies. The boundary of this cell is an ordered list of Dual lines. To what Primal points do these correspond? Display them in both Primal and Dual.

Second, for an interactive query point in the Primal, highlight the **Zone** of the point's corresponding Dual line. To what set of lines in the Primal does this Zone correspond? Display them in both Primal and Dual.

Third, support selection of a set of one or more Primal segments (for example, by repeated selection holding the Shift key). What is the set of Primal lines that stabs **all** of the segments? Display this set in both the Dual and Primal.

Fourth, come up with your own interesting connection between something in the Primal and something in the Dual, and display it in both spaces. "Interesting" means that it shouldn't be subsumed by, or a trivial consequence of, one of the three relationships above.

Extra Credit:

Use the arrangement to compute form factors between all pairs of segments (i.e., the fraction of lines leaving segment i that arrive at segment j, under an appropriate measure). Select a few of the segments and mark them as light sources (set $E > 0$). Set the reflectivity $\rho > 0$ for the rest of the segments. Use the form factors and the radiosity equation to update the radiosity B on every segment (make sure to do this using "flatland" radiosity; see e.g. Paul Heckbert's papers on this topic). Render each segment with a color equal to its radiosity. Iterate, subdividing segments as necessary, to produce a nice equilibrium lighting distribution.