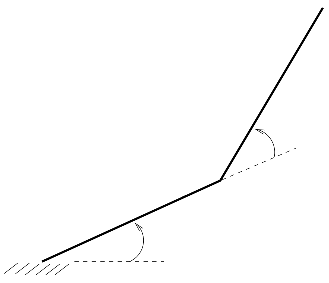
Algorithmic Robotics and Motion Planning — Fall 2006/7 — Dan Halperin

Assignment no. 2

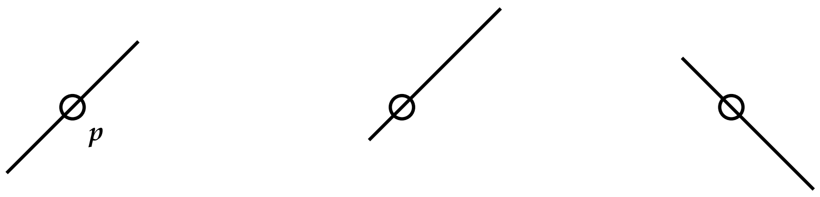
http://www.cs.tau.ac.il/~danha/courses/robotics07.html

new due date: Thursday, December 7th, 2006

Exercise 2.1 **(a)** What is the maximum combinatorial complexity (give asymptotic lower and upper bounds) of the free space of the following planar motion-planning problem. An anchored 2-link arm consisting of two segments and two rotational degrees of freedom (see the figure) moving among polygonal obstacles with a total of n vertices.
(b) Devise an efficient algorithm to compute the free space.



Exercise 2.2 Answer the same questions as in Exercise 2.1, items (a) and (b) for the following motion planning problem. A robot arm with two degrees of freedom moving in the plane among polygonal obstacles with a total of n vertices. The arm consists of a line segment that passes through a point p in the plane. It can rotate around p and translate through p, but at all times it coincides with p. See the following figure for an illustration.



Exercise 2.3 (p) Write a program that finds a path for the following motion-planning problem, whenever such a path exists. We are given a simple polygon A, the robot, which can translate in the plane, and a set of pairwise interior-disjoint simple polygons, the obstacles, which the robot has to avoid. We are also given a pair of planar points s and g denoting the reference point of A at a start position and a desired goal position respectively. The program decides whether a semi-free path for A from start to goal exists and if so outputs such a path.

In the TA's webpage you will find ample helpful information for solving this problem. In particular you will find there a CGAL program that reads the data of the robot and the obstacles from a file, computes the Minkowski sum of the obstacles and a rotated copy of the robot, draws the C-obstacles on a screen, and more.

Exercise 2.4 is on the other side of the page.

Exercise 2.4 (p) (bonus) Same as Exercise 2.3, only this time you are required to produce a *good* path. Choose your criterion: length, clearance, smoothness, etc. and devise a path that is good according to this criterion; not necessarily optimal, but better than an arbitrary path that the method that puts a point inside each trapezoid and on each vertical edge produces.