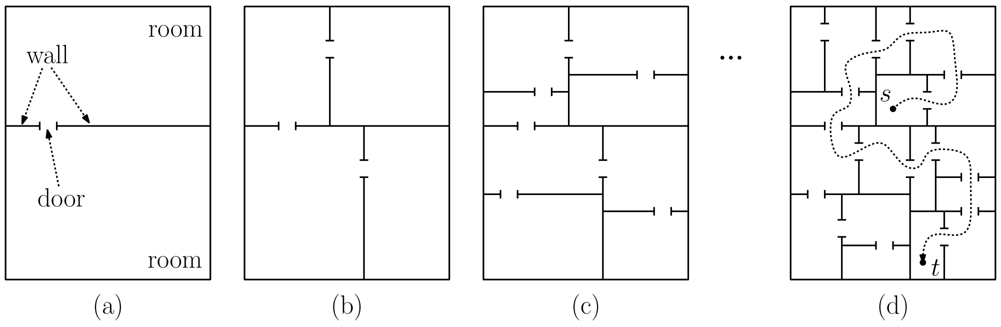
CMSC 425 : Fall 2018 Dave Mount

Homework 2

Handed out Sun, Dec 2. Due: **Tue, Dec 11 at 11:59pm**. (The standard late penalties for programming assignments apply.)

Problem 1. (15 points) Your new game engine has procedurally generated buildings. The method works recursivley. Start with a rectangle, and cut it by a horizontal or vertical line, called a *wall*, that contains a single gap, called a *door*. This splits the original rectangle into two subrectangles, called *rooms*, that are connected by this door (see the figure below (a)).

Apply the algorithm recursively to each of the rooms, with the added restriction that each new wall cannot overlap an existing door. The recursion ends when the rooms are of a desired size. (The figure shows successive recursive levels and the final structure in (d).)



Answer each of the following questions.

(i) In order to generate n rooms in the final structure, what is the total number of walls (that is, splitting lines) that need to be generated?

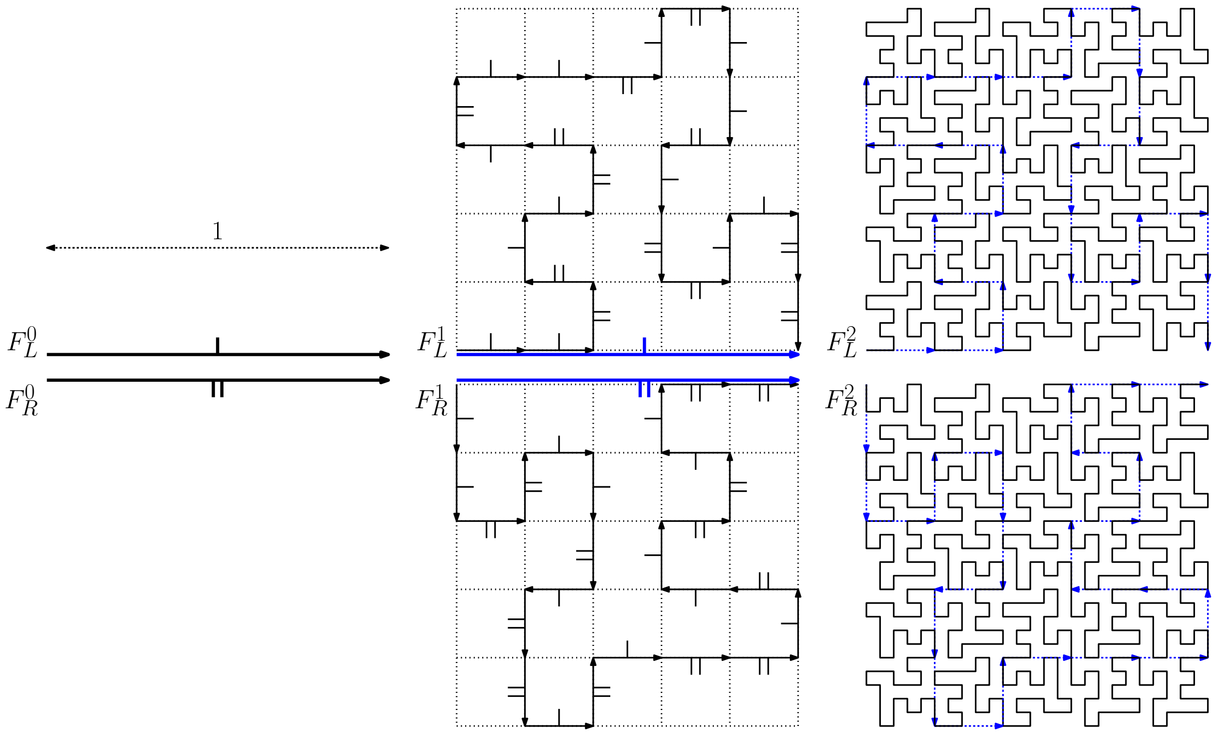
(ii) We say that a path is *simple* if it does not pass through a door more than once (see the figure below (d)). **True or False:** Given any structure formed by this algorithm, and given any two points s and t in different rooms, there exists at least one simple path between them.

(iii) Two simple paths are said to be *equivalent* if they pass through the same doors in the same order. **True or False:** Given any structure formed by this algorithm, and given any two points s and t in different rooms, all simple paths are equivalent.

In each case provide a short justification (perhaps a sentence or two). A formal proof is *not* required.

Problem 2. (13 points) In addition to producing plant-like models, L-systems can be used for generating maze-like structures. In the figure below, we show the first two generations of such a structure. The L-system involves two variables F_L and F_R. The initial segments, F_L^0 and F_R^0 are just line segments. (We have added some embellishments to help distinguish the two segments, but these are not part of the final drawing.)

Intuitively, F_L fills in a square that lies to the left of a directed line segment and F_R fills in a square that lies to the right of the directed line segment. The recursive rules for F_L and F_R have been cleverly designed so that the resulting curves do not intersect each other, no matter how many generations we produce. Observe that with each generation distances are scaled by $\sigma = 1/5$, and each individual segment of the basic length is replaced by 25 segments of the next smaller size. The recursive rule for F_L generates 13 instances of F_L and 12 instances of F_R, while the rule for F_R generates 12 instances of F_L and 13 instances of F_R. Thus, the first generations F_L^1 and F_R^1 each consist of $13+12 = 25$ segments, each of length $1/5$. The second generations F_L^2 and F_R^2 each consist of $25^2 = 625$ segments, each of length $1/5^2 = 1/25$.



(a) Derive an L-system that generates F_L and F_R. (Hint: The rules will be *long* because each needs to generate 25 segments, with the associated turns.) In particular, please provide the recursive rules for F_L and F_R.

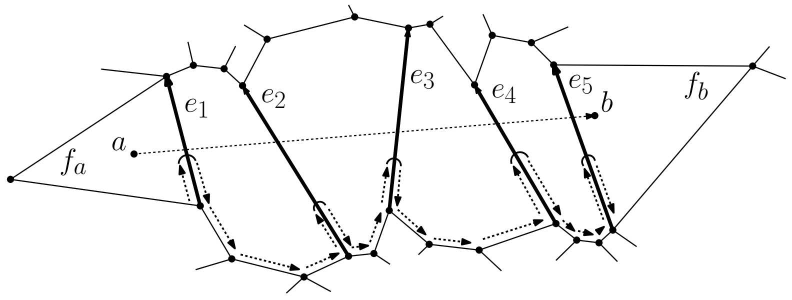
(b) Consider the limiting curve $F_L^* = \lim_{k\to\infty} F_L^k$. Derive its fractal dimension. (Because each F_L^k is a 1-dimensional curve embedded in 2-dimensional space, the fractal dimension will be in the range between 1 and 2, and it may equal 1 or 2.)

(c) Based on the definition that we gave of a fractal in the class lecture notes, is F_L^* a fractal? Explain why briefly.

Problem 3. (10 points) A fundamental algorithmic task involving navigation meshes is how to determine which edges and faces of the mesh are traversed by a path. We will consider how to solve this task. Suppose that we are given a navigation mesh that is represented as a doubly-connected edge list (DCEL). Assume further that every face of the DCEL is a convex polygon, and has no holes.

Given two point a and b, let $\overline{ab}$ be the line segment between these points. Our objective is to compute a list $L = \langle e_1, e_2, \ldots, e_m\rangle$ of the edges of the navigation mesh that this line segment passes through (see the figure below). In this problem, we will implement an efficient algorithm for computing this sequence of edges. First, let us assume that we are given the faces f_a and f_b that contain a and b respectively. Let us also assume that each edge e of the mesh has an associated boolean function $e.\text{Crosses}(a, b)$, which indicates whether edge e is crossed by the line segment $\overline{ab}$. We could simply check every edge of the mesh, but this is not efficient.

Here is an outline of the method that we would like you to implement. Given f_a and f_b as arguments, the algorithm first enumerates all the edges about f_a until it finds the edge e_1 that crosses $\overline{ab}$. (Because the faces are all convex, there can be at most one such face.) Add this edge to the list L. Next, access e_1.twin to find the oppositely directed edge, and follow this sequence of edges around the face to left of this edge until arriving at the next edge (e_2 in the figure) that crosses the line segment. Add e_2 to the list. Repeat this process until we arrive at the final edge that has f_b to its left (which will be e_5.twin in our figure). The edges visited by this algorithm are illustrated in the dotted path in the figure.

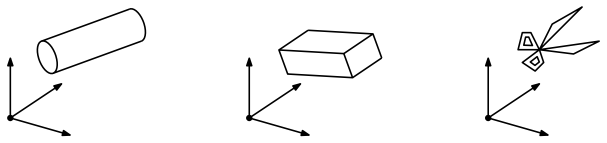


Problem 4. (12 points) How many dimensions are there in the configuration spaces for each of the following motion-planning problems. Justify your answer in each case by explaining what each coordinate of the space corresponds to.

(i) Moving a cylindrical shape in 3-dimensional space, which may be translated and rotated (see the figure below (a)).

(ii) Moving a brick in 3-dimensional space, which may be translated and rotated (see the figure below (b)).

(iii) Moving a pair of scissors in 3-dimensional space, which may be translated, rotated, and swung open and closed (see the figure below (c)).


(a) (b) (c)