# CS52: Homework 3

Out: Oct 22. Due: Nov 5.

## Problem 1: Viewing

Suppose you are looking at a wireframe cube of side length $l$ though two different camera configurations; the resulting images are shown in Figure 1. In each view, the faces of the projected cubes closest to the camera have the same size, while the ratio of the sizes of the back faces' sides is 2.

Each camera configuration has a square image pane, i.e. $t = r$, at a fixed distance $|n| = 1$ from its origin; furthermore the camera is only allowed to move along its local $z$ axis and change its image plane size $2t$.

(a) Describe in words the changes in camera position and field of view necessary to obtain the given effect. For example something like "move camera to the left and shrink field of view" would be ok.

(b) Find the closed form solution that relates the distances of the objects in each camera configuration $(d_1, d_2)$ and the camera image plane sizes $(2t_1, 2t_2)$ for the configurations given above. You can work in the $yz$ plane for simplicity.

The effect of changing field of view and distance simultaneously in order to maintain the size of a given object fixed is known as *Vertigo effect* in cinematography since it was first introduced in the movie Vertigo. Even today it is a widely use trick.

## Problem 2: Triangle Strips

Suppose we have stored a triangle mesh using an indexed triangle data structure implemented as an array of vertex data $v[n_v]$ as well as an array of triplets of face indices $f[n_f][3]$.

Upon closer inspection we realize that the mesh can be represented using one triangle strip stored in a single list $s[n_s]$ and that each vertex can only be shared by a maximum of three triangles.

(a) Give the pseudocode for an algorithm that given the face index of the first triangle on the strip will produce the list of indices describing the strip. Your algorithm should be at

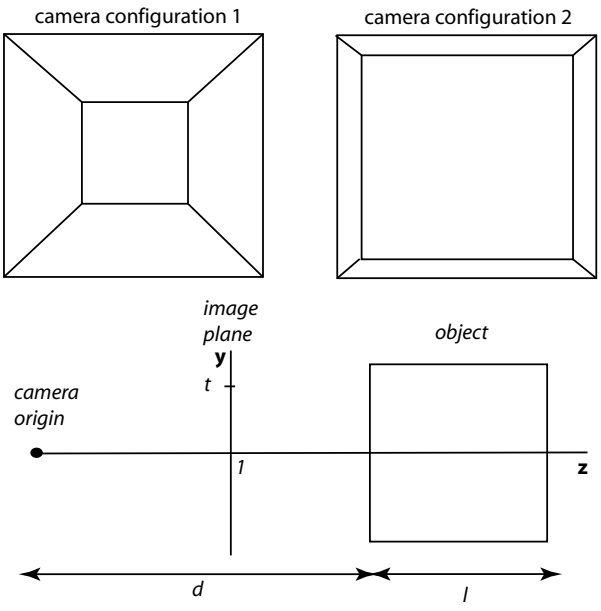


Figure 1: Illustration for Problem 1. Top row: image comaprison. Bottom row: camera configuration in the $yz$ plane.

worst $O(n_f^2)$. When writing the pseudocode, you have to give the details on how to solve the required adjacency queries. If you would like to use any adjacency data structure (which is not required), you have to give the code on how to compute it and this will count toward the running time. Hint: use the fact that you already know that all triangles in the mesh can be arranged on a single unique strip.

(b) Suppose your previous algorithm was not given the starting face. Write pseudocode to find it (note that there can be two of these, so just worry about finding one of them).

## Problem 3: Parametric Curves and Surfaces

(a) Draw a Bezier curve made of two segments where adaptive subdivision would perform better than uniform subdivision.

(b) Suppose we want to obtain a *smooth closed* curve using a Bezier formulation. Which additional constraints we have to set on the control points for this to happen?

(c) Write the parametric equations of a torus centered at the origin and aligned along the $z$ axis. The torus's radius from the center of the hole to the center of the tube is $R$ while the radius of the tube is $r$.

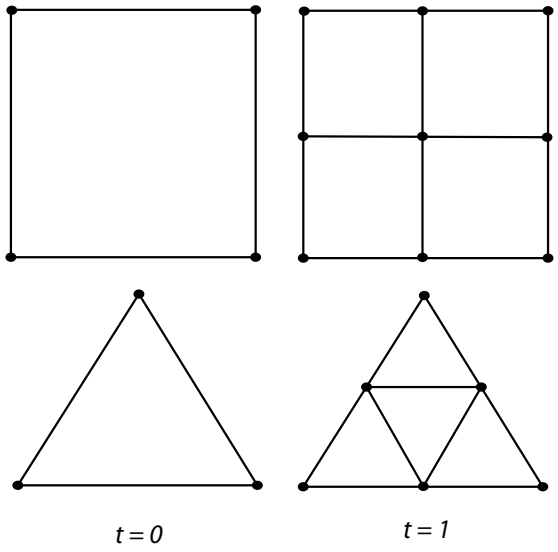


Figure 2: Illustration for Problem 4.

## Problem 4: Subdivision

(a) Suppose we want to subdivide a quad in four quads similarly to the Catmull-Clark subdivision scheme, as in Figure 2. Write the closed form solution for the number of faces and vertices when applying $t$ times the subdivision process.

(b) Suppose we want to subdivide a triangle in four triangle similarly to the Loop subdivision scheme, as in Figure 2. Write the closed form solution for the number of faces and vertices when applying $t$ times the subdivision process.

When solving this problem, shared vertices should be counted only once and $t = 0$ indicates the original face. You are welcome to prove your solution using only geometric arguments.

## Extra credit: Triangle Strips

(a) Give the pseudocode for an algorithm for problem 3.a that runs in $O(n_f)$ time. Obviously this would also be a solution for 3.a. Hint: use a two pass algorithm, where you first fill in some additional adjacency data structure and than use this to speed up the previous algorithm.

(b) Extend the algorithm in 3.a to the general case where a mesh might need multiple strips to be fully covered. A greedy approach is considered ok for this homework.