Prof. Dr. Leif Kobbelt
J. Born, M. Ibing,
J. Möbius, P. Schmidt

Visual Computing Institute | RWTH AACHEN UNIVERSITY

Basic Techniques in Computer Graphics

Assignment 10

Date Published: January 16th 2018, Date Due: January 23rd 2018

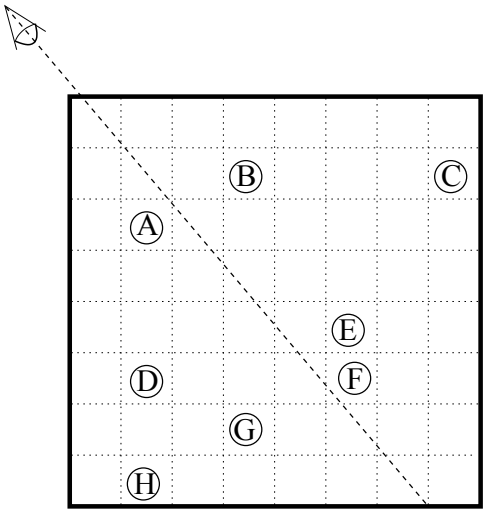
- All assignments (programming and text) have to be done in teams of 3–4 students. Teams with less than 3 or more than 4 students will receive no points.
- Hand in **one solution per team per assignment**.
- Every team must work independently. Teams with identical solutions will receive no points.
- Solutions are due 18:00 on January 23rd 2018. Late submissions will receive zero points. No exceptions!
- Instructions for **programming assignments**:
  - Download the solution template (a zip archive) through the L^{2}P course room.
  - Unzip the archive and populate the `assignmentXX/MEMBERS.txt` file. Any team member not listed in this file will not receive any points! (Also see the instructions in the file.)
  - Complete the solution.
  - Prepare a new zip archive containing your solution. It must contain exactly those files that you changed. **Only change those files you are explicitly asked to change in the task description.** The directory layout must be the same as in the archive you downloaded. (At the very least it must contain the `assignmentXX/MEMBERS.txt`.)
  - Upload your zip archive through the L^{2}P before the deadline.
  - Your solution must compile and run correctly **on our lab computers** using the exact same `Makefile` provided to you. If it does not, you will receive no points.
- Instructions for **text assignments**:
  - Prepare your solutions on paper.
  - If you write your solution by hand, write neatly! Anything we cannot decipher will receive zero points. No exceptions!
  - If you hand in more than one sheet, staple your sheets together. (No paper clips!)
  - Put the names and matriculation numbers of all team members onto every sheet.
  - Unless explicitly asked otherwise, always justify your answer.
  - Be concise!
  - Put your solution into the designated drop box at our chair before the deadline. (1st floor, E3 building.)

Prof. Dr. Leif Kobbelt
J. Born, M. Ibing,
J. Möbius, P. Schmidt

Visual Computing Institute | RWTH AACHEN UNIVERSITY

Exercise 1 Spatial Data Structures [4 Points]

To speed up many algorithms the scene is often partitioned using spatial data structures. In this task we will regard the ones most commonly used.



(a) Quadtree [1 Point]

Construct a Quadtree for the scene depicted above. Stop subdivision as soon as every node contains at most 1 object.

(b) kD-tree [1 Point]

Construct a kD-tree for the scene depicted above. Choose the positions of the splitting planes such that the tree is optimally balanced. The first splitting plane should be horizontal (i. e. split the space into an upper and lower region). Stop subdivision as soon as every node contains at most 1 object.

(c) Painter's Algorithm [1 Point]

Use the kD-tree constructed in exercise (b) to determine the order in which the objects are drawn by the Painter's Algorithm (assuming the camera position indicated in the drawing).

(d) BSP-tree [1 Point]

What differences are there between BSP-trees and kD-trees?
What benefits and disadvantages do BSP-trees have compared to kD-trees?

Prof. Dr. Leif Kobbelt
J. Born, M. Ibing,
J. Möbius, P. Schmidt

Exercise 2 Volume Ray Casting [3 Points]

You are computing the perceived color of a view ray through a volume using Volume Ray Casting. Taking discrete samples along the ray, you find the following colors c_i (represented as monochromatic gray values) and opacities α_i (where 0 is fully transparent and 1 is fully opaque):

i	0	1	2	3	4	5
c_i	0.0	1.0	0.5	0.0	0.8	0.0
α_i	0.5	0.0	1.0	0.5	0.2	0.0

The index $i = 0$ represents the sample closest to the viewer and $i = 5$ the sample farthest away from the viewer.

(a) Back-to-Front [1.5 Points]

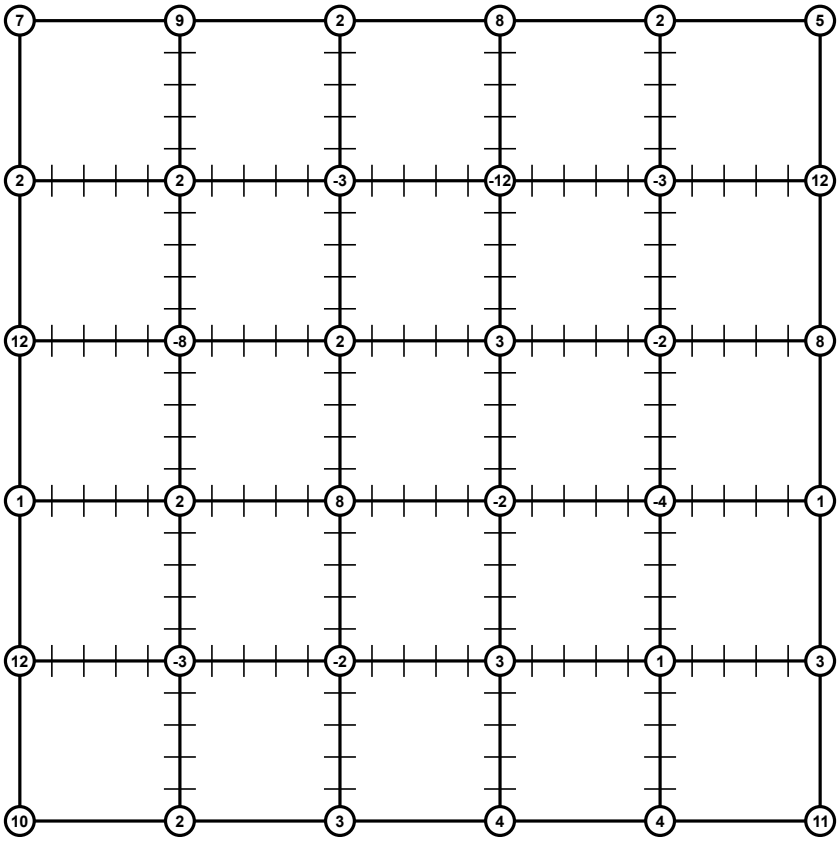
Compute the color along the ray using back-to-front compositing. Write down all computation steps.

(b) Front-to-Back [1.5 Points]

Compute the color along the ray using front-to-back compositing. Write down all computation steps. You may terminate early as soon as no further changes to the computed color value are possible.

Exercise 3 Indirect Rendering of Implicit and Volumetric Geometry [3 Points]

Instead of rendering implicit or volumetric functions directly using ray traversal techniques, one can also take an indirect route and first extract a polygonal mesh representation that is then rendered using the standard triangle rasterization pipeline. For this purpose the Marching Cubes (MC) algorithm has been presented. Given an implicit volumetric representation in form of discrete sample values at grid points, it constructs a polygonal representation of the zero-level surface (or any other iso-level).



(a) Surface Extraction [2 Points]

Execute the Marching Squares algorithm (the 2D equivalent of MC) on the discrete implicit data given above to extract a zero-level representation, i.e. draw the resulting set of line segments (instead of polygons in 3D) into the raster. Whenever there is an ambiguity, assume the square center has a positive value. Make sure to get the geometry right, not only the topology, i.e. position the samples at the correctly interpolated positions.

(b) Topological Equivalence [1 Points]

Is the zero level representation which is extracted from the Marching Squares algorithm always topologically equivalent to the original implicitly defined object? If it is, explain your answer. If not, give a counter example.