Graphics 2008/2009

T2

Final exam

Thu, Nov 06, 2008, 14:00–16:00

Comments and partial solutions

Note: these are not sample solutions, but rather comments which do not necessarily represent what was expected for the grading.

- **Do not open this exam until instructed to do so.**
- **Read the instructions on this page carefully.**

- You may write your answers in English, Dutch, or German.
  Use a pen, not a pencil. Avoid usage of the color red.
- You may *not* use books, notes, or any electronic equipment
  (including your cellphone, even if you just want to use it as a clock).
- Please put your student ID on the table so we can walk around and check it during the exam.
  You also have to show it to the instructor when you turn your exam in before leaving the room.
- Write down your name and student number on every paper you want to turn in.
  Additional paper is provided by us. You are not allowed to use your own paper.

- The exam should be doable in less than 1.5 hours. You have max. 2 hours to work on the questions.
  If you finish early, you may hand in your work and leave, except for the first half hour of the exam.
- The exam consists of 7 problems printed on 5 pages (including this one).
  It is your responsibility to check if you have a complete printout.
  If you have the impression that anything is missing, let us know.
- The maximum number of points you can score is 18.
  You need at least 17 points to get the best possible grade.

Good luck!

1

Problem 1: Perspective projection

Subproblem 1.1 [1.5 pt] Given an arbitrary camera position, we want to display the objects of a 3D model in a 2D image using perspective projection. List all the steps that are involved in the procedure we discussed in the lecture. (Note: Requested are the steps of the initial graphics pipeline (part I) we discussed. Just name the involved transformations. It is *not* necessary to provide detailed descriptions, equations, etc.)

Solution: *(cf. lecture notes, lecture 6: perspective projection, slides 3-10)*
1. define view frustum
2. move camera viewpoint to the origin
3. transform clipped view frustum to an axis-parallel box (i.e. the orthographic view volume)
4. transform the orthographic view volume to the canonical view volume
5. apply windowing transform

Subproblem 1.2 [1.5 pt] In the lecture, we used a matrix M_o in the procedure for calculating the perspective projection of a 3D model onto a 2D image. It was defined by the following product of three matrices (note: l, r, t, b, n, f specify the left, right, top, bottom, near, and far plane of the orthographic view volume, respectively, and *mxn* is the size of the projected image):

$$M_o = \begin{pmatrix} \frac{m}{2} & 0 & 0 & \frac{m}{2} - \frac{1}{2} \\ 0 & \frac{n}{2} & 0 & \frac{n}{2} - \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2}{t-b} & 0 & 0 \\ 0 & 0 & \frac{2}{n-f} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -\frac{l+r}{2} \\ 0 & 1 & 0 & -\frac{b+t}{2} \\ 0 & 0 & 1 & -\frac{n+f}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(a) Which step(s) that you listed in subproblem 1.1 are done by M_o?
(b) Shortly describe what each of the three matrices does.

Solution: *The matrix M_o combines the transformation from the orthographic view volume to the canonic view volume and the subsequent windowing transformation. From right to left:*

- *move coordinates to the center of the orthographic view volume*
- *scale everything into the canonical view volume*
- *project everything to the final image*

2

Problem 2: Hidden surface elimination

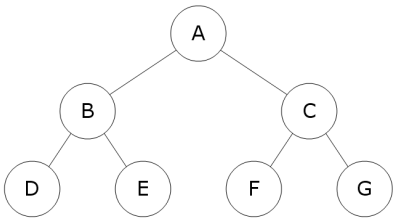
Subproblem 2.1 [0.5 pt] Which of the following statements is correct? (no explanation required)
(a) The calculation of the BSP and the drawing order are both independet from the camera position.
(b) Only the calculation of the BSP is independent from the camera position.
(c) Only the drawing order is independent from the camera position.
(d) The calculation of the BSP and the drawing order are both dependent from the camera position.

Solution: *Answer (b) is correct.*

Subproblem 2.2 [1 pt] Assume a camera position e and the implicit representation of a plane in $\mathbb{R}^3$. Let's further assume that the normal vector from the implicit equation of the plane is on the front side of the plane. How can we easily specify if the camera is located in front of or behind the plane?

Solution: *The implicit representation of a plane is defined by a function $f : \mathbb{R}^3 \to \mathbb{R}$. A point $p = (x, y, z)$ is on the plane if and only if $f(p) = 0$. The function f splits $\mathbb{R}^3$ into a positive half containing all points for which $f(p) > 0$ and a negative half containing all points for which $f(p) < 0$. The normal vector of the plane defined by the implicit equation points to the positive half. Hence, if $f(e) > 0$, we know that the camera is on the positive side (i.e. in front of the plane). If $f(e) < 0$, it is on the negative side (i.e. behind the plane).*

Subproblem 2.3 [1.5 pt] Assume a scene containing seven triangles A, B, C, D, E, F, and G, and the following related BSP tree:



(a) Shortly explain how we can use such a BSP tree to get a correct drawing order of the triangles.
(b) Write down the correct drawing order of the seven triangles with respect to a camera position e which is on the positive side of the planes defined by triangles A, B, C, and on the negative side of the planes defined by triangles D, E, F, G.

Solution: *See lecture notes and tutorials for the explanation of how to draw the triangles based on the given BSP.*
In this example, the correct drawing order is D - B - E - A - F - C - G, because e is on the positive side of A, so we go to the negative subtree first. Then we continue recursively: e is on the positive side of B, so we draw the negative subtree first (i.e. we draw D), then we draw the root of the subtree (i.e. we draw B), then we draw the positive subtree (i.e. we draw E), ...)

Problem 3: Triangle rasterization

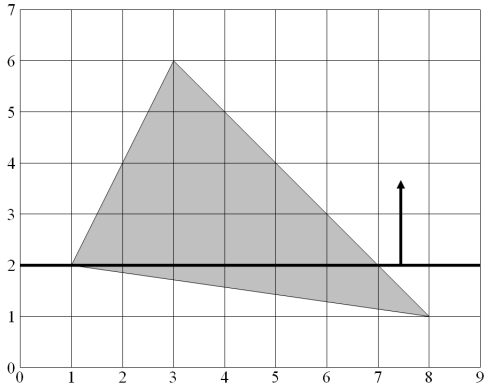
Subproblem 3.1 [1 pt] In the lecture, we used two data structures in the algorithm for triangle rasterization via scan-line conversion: The *edge table* and the *active edge table*. The entries of the edge table are called edge records. Assume the following example of an edge record:

$$2 : (9, 6, -1/2)$$

Explain what this means and describe each of the three values specified by this edge record.

Solution: *For each scanline, the edge table contains an edge record for each edge that starts on this scanline. In the example, one edge starts at scanline number 2. The entries of the edge record specify the x-coordinate of the lowest vertex (here 9), the y-coordinate of the highest vertex (here 6), and the value of Δ_x for the edge which specifies the increase or decrease in x-coordinate from one scanline to the next (here -1/2).*

Subproblem 3.2 [2 pt] Look at the image below. Assume that the scan-line is horizontal and moves vertically from the bottom of the image to the top as illustrated by the black line and the associated arrow.
(a) Give the values for the edge table that are stored for scan-line number 1.
(b) Give the values for the active edge table that are stored for scan-line number 3.



Solution: *At scanline 1, two new edges start. They both have the same x-value (8). One ends at a vertex with y-value 2, the other at one with y-value 6. Δ_x is -7 for the first edge and -1 for the second one. Hence, the correct answer is:*

$$1 : (8, 2, -7), (8, 6, -1)$$

For scanline 3, there are two active edges. The x- and y-value of the lowest and highest vertex respectively are 2 and 6 for the first edge, and 8 and 6 for the second edge. Δ_x for the first edge is 1/2, for the second edge it's -1. Since the first edge started at scanline 2, we have to add it's Δ_x value one time to get the correct x-value for this edge at scanline 3. The second edge started at scanline 1, so we add it's Δ_x value twice to get the correct x-value for this edge at scanline 3. Hence, the correct answer is:

$$3 : (1\tfrac{1}{2}, 6, \tfrac{1}{2}), (6, 6, -1)$$

Problem 4: Ray tracing

Subproblem 4.1 [1 pt] In the lecture, we discussed how we can create new objects based on a given one by *instancing*. Suppose we have an object O, a matrix M which does a basic transformation, and a ray r. Now we want to create a new object O' by applying the matrix M to the object O. Assume that intersections with the object O are much easier to calculate than with object O' (e.g. because it is complied of circles whereas O' is compiled of ellipses). Explain how we can easily compute the intersection p of r and O'.

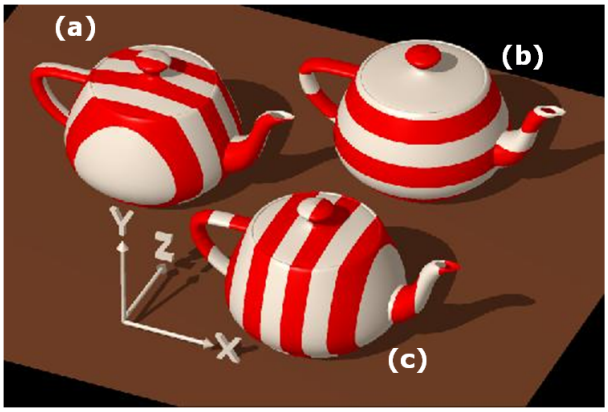
Solution: *Because basic transformations preserve intersections, we can calculate a ray r_0 by applying the inverse transformation matrix M^{-1} to the original ray, i.e. $r_0 = M^{-1}r$. Then we calculate the intersection points p_i of the original object O with this ray r_0. Applying the original transformation matrix M to the intersection points p_i gives us the desired intersection points p'_i of the ray r with the transformed object O'.*

Subproblem 4.2 [1 pt] Assume we have a 2D image containing many triangles. Explain how you can use the Octree approach to separate the space in a way that each of the resulting cells contains only two triangles at the most. (Note: for simplification, you can assume that the triangles in the image are distributed in a way that there is no need to split them when new cells are created.)

Solution: *First, we create a bounding box around all triangles. Then we split this bounding box into four equally sized cells. For each cell, we keep splitting it recursively into four equally sized cells untill it contains only two triangles at the most.*

Problem 5: Texture mapping

Subproblem 5.1 [1 pt] In the following image, different procedures have been used to create the stripe textures for each of the three teapots:



(a) Which texture was created with the following procedure?

```
stripe(point (x_p, y_p, z_p)) {
   if (sin x_p > 0)
      return color0;
   else
      return color1;
   }
}
```

(b) How do you have to change it to create the texture on the other two teapots?

Solution: *The procedure creates texture (c). To create textures (a) and (b), we have to replace* $\sin x_p$ *with* $\sin z_p$ *and* $\sin y_p$*, respectively.*

Subproblem 5.2 [1.5 pt]
(a) Give a short description of bump mapping.
(b) Give a short description of displacement mapping.
(c) In which situation(s) should one prefer bump mapping over displacement mapping?

Solution: *For (a) and (b) see lecture notes. Displacement mapping has the disdvantage of being more computationally complex and requiring more storage. The biggest disadvantage of bump mapping is that it only creates an apparent change of the geometry thus creating unrealistic silhouettes and shadows. Hence, if speed and storage is more important than a perfect silhouette, we usually prefer bump mapping over displacement mapping.*

Problem 6: Radiosity

Subproblem 6.1 [0.5 pt] Assume you have to create an animated movie with a walk-through of the interior of some buildings. Give a short explanation why you might use radiosity to compute diffuse global illumination instead of, for example, ray tracing and ambient lighting.

Solution: *Radiosity is view indendent whereas ray tracing is view dependent. Hence, when creating a walk-through using ray tracing and ambient lighting, we have to do a new calculation for each single frame whereas for radiosity one single, view independent calculation of the global lighting conditions is enough.*

Subproblem 6.2 [1.5 pt] Explain the meaning of the following formula, which is used to calculate the radiosity B_i of a patch A_i:

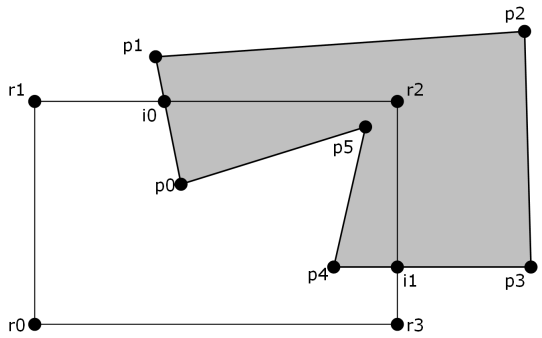
$$B_i = E_i + \rho_i \sum_j B_j F_{ij}$$

Solution: *The radiosity of a patch is the sum of*

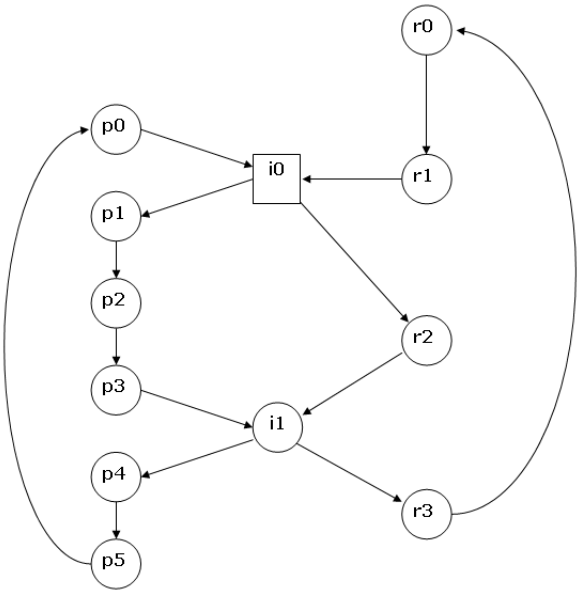
- *the energy E_i emitted by the object itself (e.g. if it is a light source) and*
- *the reflection of the light which in turn is reflected by the other objects and defined by*
  - *the sum of the radiosities emitted by all objects B_j multiplied with the form factor F_{ij} which specifies "how well to objects see each other",*
  - *weighted by the reflective factor ρ_i of patch A_i (which depends on the material of the object's surface and specifies how much light is reflected).*

Problem 7: Clipping

Subproblem 7.1 [1 pt] We want to clip the gray polygone against the clipping area represented by the rectangle in the image below. Construct the graph used by the Weiler-Atherton algorithm.



Solution:



Subproblem 7.2 [1.5 pt] Give a general explanation about how the graph from the Weiler-Atherton algorithm is used to determine the clipped polygones. Then use the graph you have drawn in the previous subproblem to create the resulting polygone.

Solution: *See lecture notes for the general explanation. The resulting polygone in the example is i0-r2-i1-p4-p5-p0-i0.*