Department of Computer Science
Vrije Universiteit
Dr. T. Kielmann

Exam **Computer Graphics**
Exam code: 4001061
09 – 04 – 2008

This is a *"closed book"* exam.
No printed materials or electronic devices are admitted for use during the exam.
You are supposed to answer the questions **in English**.

Wishing you lots of success with the exam!

Points per question (maximum)

Q	1		2			3			4		5				6			7		
	a	b	a	b	c	a	b	c	a	b	a	b	c	d	a	b	c	a	b	c
P	3	3	4	4	4	6	4	6	4	4	4	3	5	7	3	4	4	6	6	6

Total: 90 (+10 bonus) = 100 To pass the exam, it is sufficient to get at least 45 + 10 = 55 points.

1. Color Perception and Production

a) Explain the following properties of the human visual system as far as they are relevant for the perception of images generated by computer graphics!

- human color perception
- CIE standard observer curve
- lateral inhibition

b) What is *color balancing*? How can it be implemented?

2. Polygon Shading

a) Explain the basic idea of the *Phong reflection* model! Draw a simple figure that shows the vectors involved in computing the shade of a given point on the surface of an object!

b) Explain how *flat shading* works, for example for a polygonal mesh! What are the advantage and the disadvantage of flat shading?

c) Explain *Phong shading* and how it improves over the disadvantage of flat shading!

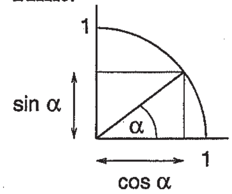
3. Affine Transformations

In a 2D homogeneous coordinate system, each point P can be represented as $P = P_0 + xv_1 + yv_2$

a) For this coordinate system, identify the **matrices** for the following transformations:

- $T(t_x, t_y)$, for a *translation* by the vector $[t_x, t_y, 0]^T$
- $R(\alpha)$, for a *rotation* around the origin by an angle α

Hint:



b) For a rotation around an arbitrary point (x, y), describe the matrix $R(\alpha, x, y)$ as a suitable concatenation of the simple transformations $T(x, y)$ and $R(\alpha)$.

c) Draw a simple 2D coordinate system and show the point $(1, 0)$ and where it is rotated to by both $R(90°)$ and $R(90°, 1, 1)$.

Show that your matrix $R(\alpha, x, y)$ performs the same rotations.

4. Hidden Surface Removal

a) Explain *briefly* the painter's algorithm! In which cases does the algorithm fail?

b) Explain *briefly* the z-buffer algorithm! Which issues does the application programmer have to deal with that the algorithm cannot handle by itself?

5. Scene Graphs

In a program, scene graphs shall be built from objects of a class `Node`; all specific classes of nodes (geometric objects, transformations, lights, material properties, etc.) are supposed to be subclasses of `Node`. The scene graph shall be organized as a left-child, right-sibling tree.

a) Define a (C++) class `Node` with methods `Render` for rendering a node, `AddChild` for adding a child node, and `Traverse` for traversing the tree. Class `Node` shall have (only) those data members that are needed to maintain the tree structure.

b) Implement the method `AddChild` of class `Node`.

c) Implement the method `Traverse` of class `Node`.

d) Define a (C++) class `Rectangle` as a subclass of `Node`. Implement its `Render` method.

Where necessary, use OpenGL calls.

6. Viewports

a) Explain the terms *viewport* and *aspect ratio*! Give a formula that expresses the aspect ratio for a given viewport!

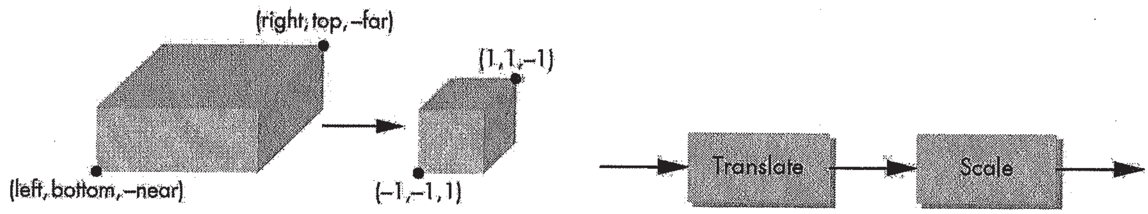
b) Assume, an OpenGL application shall maintain the aspect ratio of its output a_v, even when a user resizes the window. In that case, the application shall use the maximal possible viewport that maintains a_v and that still fits into the reshaped window with its aspect ratio a_w.
The viewport shall be centered in the window.

Given a_v and a_w, how many different cases have to be distinguished for finding such a maximal viewport? For each case, draw a simple sketch that shows the window, the viewport, and their respective width and height!

c) Write a callback function in the C language using OpenGL (the GLUT library) that selects the viewport according to part b! Which (GLUT) function has to be used to register this callback?

7. glOrtho

A call to `glOrtho(left, right, bottom, top, near, far)` defines a viewing volume for orthogonal projection. The implementation (the code) of `glOrtho` multiplies a matrix P to the current transformation matrix (CTM).



As shown in the diagram (left), the effect of the maxtrix P is to map the viewing volume to the canonical volume. The right side of the diagram indicates that this mapping can be done in two steps, first a *translate* (moving the center of the viewing volume to the origin) and then a *scale*, bringing the volume to the size $2 \times 2 \times 2$.
Hint: To save work, simply abbreviate *left, right, bottom, top, near, far* by their first letters: *L, R, B, T, N, F*.

a) The translation step can be performed by using a translation matrix $T(dx, dy, dz)$. Compute dx, dy, dz and show that your $T(dx, dy, dz)$ translates the points $[L, B, -N, 1]^T$ and $[R, T, -F, 1]^T$ to coordinates that lie symmetrically around the origin!

b) The scaling step can be performed by using a matrix $S(sx, sy, sz)$. Compute sx, sy, sz and show that your $S(sx, sy, sz)$ scales the translated corners of the viewing volume (the results from part a) to the respective corners of the canonical viewing volume, namely $(-1, -1, 1)$ and $(1, 1, -1)$!

c) Compute the overall matrix P from $S(sx, sy, sz)$ and $T(dx, dy, dz)$! Does it matter if you compute either $P = S \cdot T$ or $P = T \cdot S$? Explain why!