Department of Computer Science Vrije Universiteit Dr. T. Kielmann	Exam **Computer Graphics** Exam code: 4001061 03 – 04 – 2007

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)

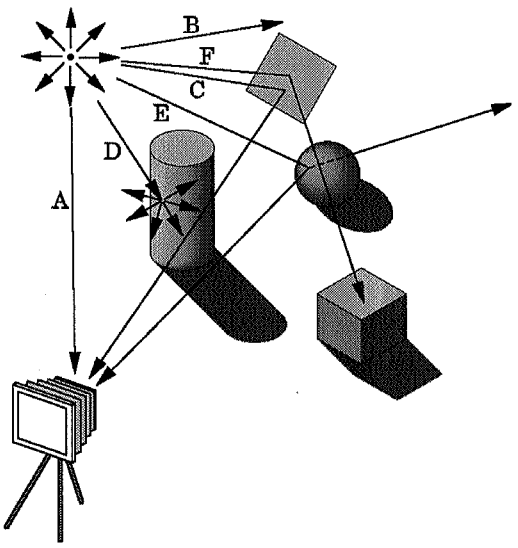
Q	1			2		3		4		5				6			7	8	
	a	b	c	a	b	a	b	a	b	a	b	c	d	a	b	c	a	a	b
P	3	6	2	9	10	3	7	3	7	3	2	5	4	4	4	2	3	5	8

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

1. Ray Tracing

a) Explain how images are created using the ray tracing technique!

b) Look at the rays denoted A to F in the following picture! What happens to each ray and how does it contribute to the image created in the camera?



c) What is the most important disadvantage of ray tracing because of which it is not used, for example, by OpenGL?

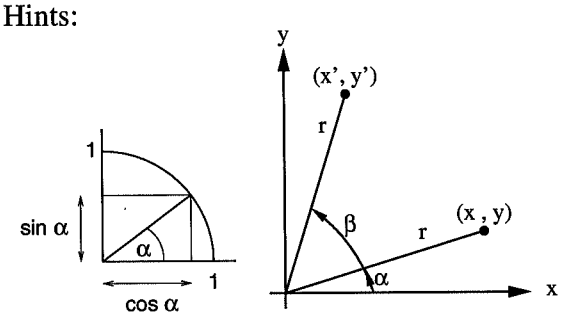
2. Affine Transformations

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 $[t_x, t_y, 0]^T$
- $S(s_x, s_y)$, for a *scaling* with the scalars s_x and s_y (and the fix point in the origin)
- $R(\beta)$, for a *rotation* around the origin by an angle β

Hints:



$$\sin(\alpha \pm \beta) = \sin\alpha \cdot \cos\beta \pm \cos\alpha \cdot \sin\beta$$
$$\cos(\alpha \pm \beta) = \cos\alpha \cdot \cos\beta \mp \sin\alpha \cdot \sin\beta$$

b) Let $T_1, T_2, S_1, S_2, R_1, R_2$ be translations, scalings, and rotations, as defined by the matrices from part a). Which of the following transformation pairs are commutative? Show why!

i) T_1, T_2 ii) S_1, S_2 iii) R_1, R_2 iv) T_1, S_1 v) T_1, R_1

3. Picking

a) Explain (briefly) the logical input operation called *picking*! What is the problem caused by the pipeline rendering architecture for implementing picking?

b) Explain how OpenGL allows to implement picking! Which functionalities have to be added to the mouse callback and to the scene rendering code?

4. Bresenham's Algorithm

a) What is the fundamental idea behind Bresenham's line drawing algorithm that reduces the necessary computation per pixel?

b) Using Bresenham's algorithm, implement a function `draw_line(x0,y0,x1,y1)` that draws a line from individual pixels, where the starting point is `(x0,y0)` and the end point is `(x1,y1)`. Your function should work for all lines with slope m for which holds $0 \leq m \leq 1$. For drawing an individual pixel `(x,y)`, use the function `void plot(int x, int y)`.

Hint: It is OK to use floating point arithmetic for your function.

5. Scene Graphs

Consider the class definition `Node` for scene graphs; 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. The class `Rectangle` is one example of scene graph nodes.

a) Implement the method `AddChild` of class `Node`!

b) Implement the method `Render` of class `Node`!

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

d) Implement the method `Render` of class `Rectangle`!

Where necessary, use OpenGL calls.

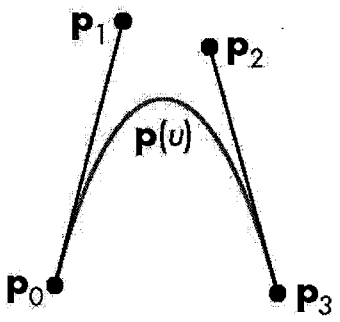
```
class Node{
  public:
    Node();
    virtual ~Node();
    virtual void Render();
    void AddChild(Node *);
    void Traverse();
  private:
    Node *LeftChild;
    Node *RightSibling;
};
Node::Node(){
    LeftChild=NULL;
    RightSibling=NULL;
}
class Rectangle: public Node{
  public:
    Rectangle(float v[4][3]);
    virtual ~Rectangle();
    virtual void Render();
  private:
    float vertices[4][3];
}
```

6. Curves and Surfaces

a) For a parametric curve, explain the degrees of continuity C^0, C^1, and C^2!

b) A curve segment of a Bezier curve is shown in the figure (right). Are Bezier curves C^0, C^1, or C^2? Explain why!

c) For an algorithm that renders a cubic polynomial surface by recursive subdivision, which (simple) criterion can be used to limit the recursion depth?



7. Color Balancing

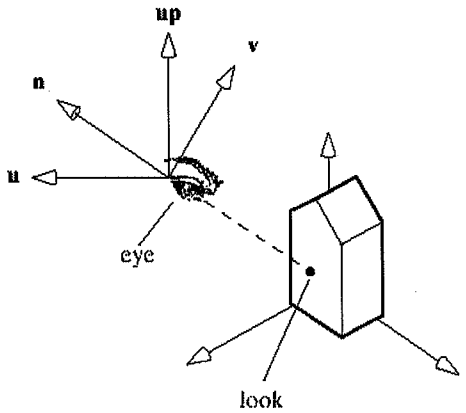
a) What is *color balancing*? How can it be implemented?

8. Viewing

a) In the OpenGL rendering pipeline, the Current Transformation Matrix (CTM) consists of two parts. Which are they? What is their respective role for the rendering process? To which of the two should the function `gluLookAt()` be applied to? (say why!)

b) The function `gluLookAt(eyex,eyey,eyez,lookx,looky,lookz,upx,upy,upz)` internally uses a u-v-n viewing coordinate system:

$$\begin{aligned} \mathrm{n} &= \mathrm{eye} - \mathrm{look} \\ \mathrm{u} &= \mathrm{up} \times \mathrm{n} \\ \mathrm{v} &= \mathrm{n} \times \mathrm{u} \end{aligned}$$



`gluLookAt` first normalizes n, u, v to unit length and then uses the normalized vectors to build up the viewing matrix:

$$V = \begin{pmatrix} u_x & u_y & u_z & d_x \\ v_x & v_y & v_z & d_y \\ n_x & n_y & n_z & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (d_x, d_y, d_z) = (-eye \cdot u, -eye \cdot v, -eye \cdot n)$$

Show that u, v, n are mutually perpendicular (orthogonal)!
Show that the matrix V properly converts object coordinates to eye coordinates by demonstrating that it maps *eye* to the origin $(0,0,0,1)^T$, u to $(1,0,0,0)^T$, v to $(0,1,0,0)^T$, and n to $(0,0,1,0)^T$!

Hint: $\cos 0 = 1$