Department of Computer Science
Vrije Universiteit
Dr. T. Kielmann

Exam **Computer Graphics**
Exam code: 4001061
04 – 04 – 2006

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	a	b	a	b	a	b	c	a	b	c	d	a	b	c	a	b	c	a	b
P	3	3	4	4	9	10	3	4	4	3	2	5	4	3	3	3	4	4	2	5	8

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

1. Human Visual System

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 can a computer-graphics application do to work around the following properties of the human visual system

- standard observer curve,
- lateral inhibition ?

2. 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?

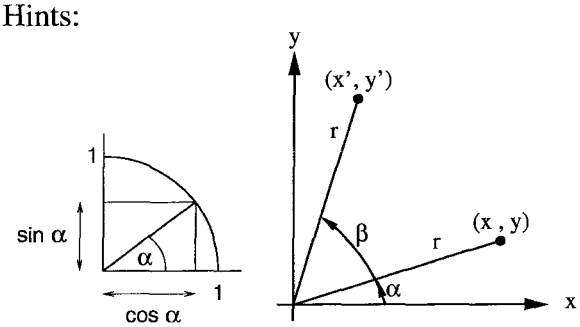
3. Affine Transformations

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 $[t_x, t_y, 0]^T$
- $S(s_x, s_y)$, 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

4. 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?

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 *);
  private:
    void Traverse();
    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. 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?

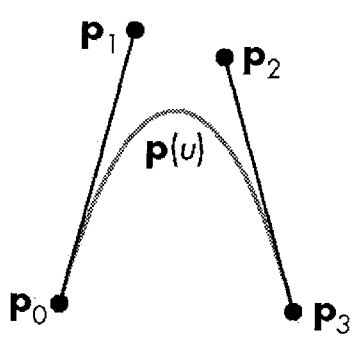
c) Explain *Phong shading* and how it improves over the disadvantage of flat shading!

7. 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 a 1024×1280 pixel window, what is the maximum number of subdivisions that are needed to render a cubic polynomial surface?

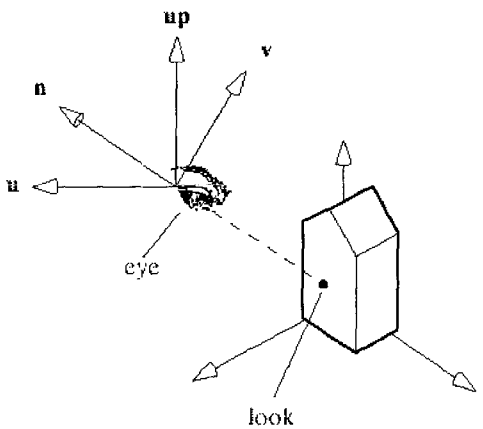


8. Viewing

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:

n = eye − look
u = up × n
v = n × u



`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: $\cos 0 = 1$