Department of Computer Science Vrije Universiteit Dr. T. Kielmann	Exam **Computer Graphics** Exam code: 4001061 11 – 01 – 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	a	a	b	a	b	c	d	a	b	c	d	a	b	c	a	b	c	a	b	c
P	3	12	3	7	4	4	5	4	3	2	5	4	3	3	3	4	4	2	5	5	5

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

1. Color Balancing

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

2. Bresenham's Algorithm

a) 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 for which holds `x0 <= x1` and `y0 <= y1`, which is slightly more general than the case shown in the lecture. For drawing an individual pixel `(x,y)`, use the function `void plot(int x, int y)`.

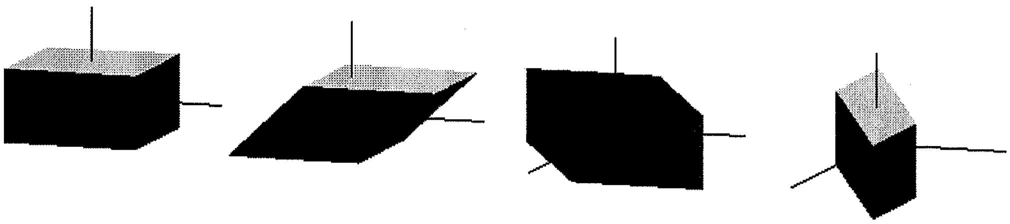
Hint: It is OK to use floating point arithmetic for your function.

3. Picking

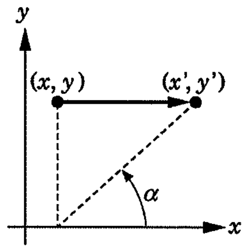
a) Explain (briefly) the logical input operation called *picking*! What is the problem caused by the pipeline rendering architecture for implementing picking?

b) Explain how OpenGL allows to implement picking; do so by explaining the following terms: rendering mode, object stack, pick matrix, clipping, select buffer, changes/additions to the mouse callback and the scene rendering code.

4. Shear The following pictures show (from left to right) a cube, and the same cube sheared along the X axis, the Y axis, and the Z axis.



The following drawing (left) shows how the shear along the X axis can be represented depending on a shearing angle α. H_x is the respective transformation matrix.



$$H_x(\alpha) = \begin{bmatrix} 1 & \cot\alpha & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a) Make similar drawings, one for the shear along the Y axis (angle β), and one for the shear along the Z axis (angle γ), according to the above picture!

b) Determine the shear matrices $H_y(\beta)$, and $H_z(\gamma)$, corresponding to part a!

c) Implement a C function `void glShearX(GLfloat alpha)` that works like `glTranslate` or `glRotate` (affecting the currently active transformation matrix by multiplying $H_x(\alpha)$ to it! You can assume that a function cot() is available.

d) Determine a shear matrix $H(\alpha, \beta, \gamma)$ that combines the effects of $H_x(\alpha)$, $H_y(\beta)$, and $H_z(\gamma)$!

Assume you have implemented the function glShearX, and also (analogously) glShearY and glShearZ. Is it possible to implement a function `glShearXYZ(alpha,beta,gamma)` by a combination of calls to glShearX, glShearY, and glShearZ? Give arguments for your answer!

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. 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?

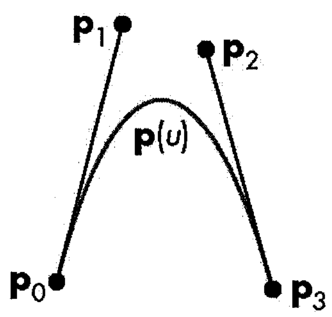
c) Explain *Phong shading* and how it improves over the disadvantage of flat shading!

7. Curves and Surfaces

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

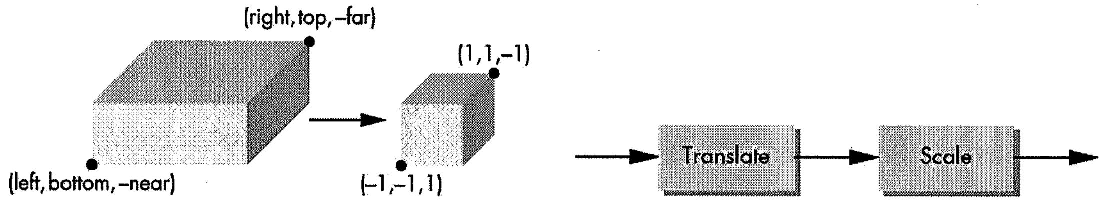
b) A curve segment of a Bezier curve is shown in the figure (right). Are Bezier curves C^0, C^1, or C^2? Explain why!

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



8. 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 1: In the following, it is important to get the signs ($+/-$) right. Unfortunately, Angel's book is not free of mistakes when explaining this subject — so please don't rely too much on *details* that you may remember from the book, just use your own knowledge and understanding!

Hint 2: 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!