# Exam – Computer Graphics

**18 december 2004, 8-13**

1. (a) Which matrix maps the normalized vectors along the positive x-, y, and z-axes on the vectors (3,2,1), (-2,0,4) and (4,-7,2) respectively. *(0.4*)
   (b) How is the transformation of a point different from the transformation of a vector? *(0.3*)
   (c) Rotation matrices are particularly easy to invert. How? *(0.3*)

2 (a) A rasterizing mesh-renderer uses so called depth testing.
Describe its purpose and how it works. *(0.8)*
   (b) Why is depth testing normally not used for wireframe-rendering? *(0.2)*

3 (a) Scene descriptions often uses hierarchical structures. Which are the benefits of this technique? *(0.6)*
   (c) Describe how these structures are traversed at rendering. (*0.4*)

4. (a) Which vertex-information must exist in order to apply bump-mapping? *(0.2)*
   (b) Why is this information needed? *(0.4)*
   (c) Explain how it is use in the algorithm for bump-mapping. *(0.4)*

5. (a) Describe what is displayed on the screen after a call to the funktion draw() below. *(0.8)*

```
def draw():
    glColor(1,0,0)
    glPushMatrix()
    glScale(3,3,3)
    glTranslate(2,0,0)
    glRotate(90, 0,0,1)
    glPushMatrix()
    glTranslate(1,0,0)
    drawSquare()

    glColor(0,1,0)
    glPopMatrix()
    glPushMatrix()
    glTranslate(4,0,0)
    glScale(0.5,0.5,0.5)
    glRotate(270, 0,0,1)
    drawSquare()


def drawSquare():
    glBegin(GL_QUADS)
    glVertex(0,0,0)
    glVertex(0,1,0)
    glVertex(1,1,0)
    glVertex(1,0,0)
```

```
glEnd()
```

(b) OpenGL has a concept called texture object. What is that and which problem does it solve? *(0.3)*

6. A triangle surface with vertices $P_0$, $P_1$, $P_2$ is lit by a point light source with light intensity $L$ and position in the point $P_L$. The viewer position is $P_V$. Ambient light is neglible and the material properties of the surface are given below.

| Material properties | Vertex coordinates | Light source | Viewer |
|---|---|---|---|
| $k_a = 0.2$<br>$k_d = 0.3$<br>$k_s = 0.5$<br>$\alpha = 4$ | $P_0 = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$ $P_1 = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$ $P_2 = \begin{bmatrix} 0 \\ 1 \\ -2 \end{bmatrix}$ | $L = 1.0$<br>$P_L = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$ | $P_V = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$ |

(a) Let $P$ be the point which has the barycentric coordinates *[0.25, 0.25, 0.5]* with respect to the vertices of the triangel. Determine the light reflected from the point $P$ with Phong's reflection model. *(0.5)*

(b) Determine the light reflected from $P$ with Blinn-Phong's reflection model. *(0.2)*

(c) Do the same calculations with both reflection modells for the case where the normal at $P$ has been interpolated from the non-normalized vertex normals given below. *(0.3)*

| Vertex normals |
|---|
| $N_0 = \begin{bmatrix} 1.2 \\ 1.1 \\ 0.8 \end{bmatrix}$ $N_1 = \begin{bmatrix} 1.1 \\ 0.9 \\ 0.7 \end{bmatrix}$ $N_2 = \begin{bmatrix} 0.75 \\ 0.8 \\ 1.3 \end{bmatrix}$ |

THE END!

# Short answers to the exam in Computer Graphics

**18 december 2004**

1. (a) $\begin{bmatrix} 3 & -2 & 4 \\ 2 & 0 & -7 \\ 1 & 4 & 2 \end{bmatrix}$.

   (c) Translation transformations make no sense for vectors..

   (b) The inverse of a rotation matrix is equal to its transpose.

2. (a) The depth of a rendered point is the distance from the point to the camera along the image plane normal. During rendering a buffer is maintained which contains the depth of every rendered pixel. Depth testing means that that you only render a point if it depth is less than the corresponding current value in the depth buffer. By doing so hidden surfaces are eliminated correctly.

   (b) A major feature of wireframe-rendering is that only the edges of the rendered triangles have to be drawn whereas depth testing is an operation which must be done for every pixel of the triangle.

3. (a) One benefit is that the scene description can be made more compact because similar structures which are used in multiple places only needs to be described once. Another advantage is that changes, for example animations, of structures containing multiple elements can be specified in one place.

   (b) In general a hierarchy is here a directed acyclic graph which is traversed top-down. When the primitives at the bottom, typically triangle meshes, are reached, these are rendered to the screen with the current rendering attributes and the the current geometric transform. On the way down the attributes and the transform is updated on each intermediate level.

4. (a) Bump-mapping requires that vertices in addition to normal, position och texture coordinates also contains tangent and binormal.

   (b) Bump-mapping means that you take the surface normal from a surface texture. This normal typically deviates from the geometric normal which gives the surface a bumpy appearance. The problem is how to interpret this deviation when the surfaces has no orientational reference. The tangent and the binormal provides this reference by, together with the normal, define a coordinate system for the surface, the so called tangent space.

   (c) Bump-mapping:
   - For each vertex, transform tangent, binormal and normal to the camera frame.
   - Interpolera them over the triangle surface and normalize.
   - Use texture coordinates to look up the new normal in the bump-map.
   - Transform this normal from tangent space to camera space.
   - Use the transformed normal in the shading calculations.

5. (a) A call to the function `draw` produces a red square with side 3 and lower left corner in (3,3) and a green square with side 1.5 and lower left corner in (6,12).

(b) A texture object is texture data with associated parameters which has beed retained in the graphic memory and which can be referenced by an identifier from the application program. Thereby you don't have to do time consuming download of texture data for each frame.

6. The solution is given by the program `phongquestion.py` given below. Executing the program gives these answers:

```
>>> import phongquestion
Phong:  0.356180236983

Blinn:  0.568266809152

Phong:  0.322131833014
Blinn:  0.547706048252
>>>
```

The program:

```
from Numeric import *
from LinearAlgebra import *

def length(v): return sqrt(dot(v,v))
def unit(v): return v/length(v)
def cross(u,v):
    return array((u[1]*v[2]-u[2]*v[1],
                  u[2]*v[0]-u[0]*v[2],
                  u[0]*v[1]-u[1]*v[0]))
ka = 0.2
kd = 0.3
ks = 0.5
alpha = 4

ps = array(((-1,-1,1), (1,-1,-1), (0,1,-2)))
N = unit(cross(ps[1]-ps[0], ps[2]-ps[0]))

P = ps[0]*0.25 + ps[1]*0.25 + ps[2]*0.5

L = 1.0
pL = (1,0,-1)
pV = (1,2,1)

def phong(p):
    dL = unit(pL-p)
    dV = unit(pV-p)
    dReflect = 2*dot(dL,N)*N - dL
    ambient = 0
    diffuse = kd*L*dot(dL,N)
    specular= ks*L*dot(dV,dReflect)**alpha
    return ambient + diffuse + specular

def blinn(p):
    dL = unit(pL-p)
    dV = unit(pV-p)
    dH = unit(dL+dV)
    ambient = 0
    diffuse = kd*L*dot(dL,N)
    specular= ks*L*dot(dH,N)**alpha
    return ambient + diffuse + specular


print "Phong: ",phong(P)
print
print "Blinn: ",blinn(P)
print

ns = unit(array(((1.2, 1.1, 0.8), (1.1,0.9,0.7), (0.75,0.8,1.3))))
N = unit(ns[0]*0.25 + ns[1]*0.25 + ns[2]*0.5)

print "Phong: ",phong(P)
print "Blinn: ",blinn(P)
```