Department of Computer Science	Exam **Computer Graphics**
Vrije Universiteit	Exam code: 4001061
Dr. T. Kielmann	20 – 02 – 2012

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	c	a	b	a	b	c	a	b		a	b	c
P	4	3	9	3	3	2	4	6	4	4	9	4	3	6	8	6	6	6

Total: 90 To pass the exam, it is sufficient to get at least 45 points.

1. The Synthetic Camera Model

a) The Synthetic Camera Model is based on the working of an old-fashioned bellows camera. Describe how an image is constructed using the Synthetic Camera Model.

b) One problem of a bellows camera (or the Synthetic Camera Model) is that it produces upside-down images. What 'trick' is used to correct for this problem, and why does this work?

2. Coordinate System Transformations
In OpenGL, the following coordinate systems are used (among others), here listed in alphabetical order:

1. camera coordinates
2. clip coordinates
3. normalized device coordinates (NDC)
4. object coordinates
5. world coordinates

a) Explain the role of each of these coordinate systems. By which mechanisms are coordinates transformed from one system to the other?

b) Which of these are homogeneous coordinates? Which ones are used by a vertex shader? Which ones by a fragment shader?

3. Texture Mapping

a) Roughly speaking, texture mapping associates a discrete texel with each point on a geometric object. Name at least 3 problems that can occur with mapping textures, and explain each of the problems in reasonable detail.

b) Mapping of 2-dimensional textures to non-flat 3-dimensional surfaces can be quite complex. Describe a two-step solution that can simplify the mapping.

c) In a program, you are using the following fragment shader:

```
in varying vec2 tex_coords;
uniform sampler2D my_texture;

void main(){
     gl_FragColor = mix(texture(my_texture, tex_coords),
                        vec4(0.55,0.27,0.07,1.0), 0.3);
}
```

i What does this fragment shader do?

ii Supposed, you are confused by the way the texture is used and you wish to debug your program. Write a version of this fragment shader that displays a visual representation of the texture coordinates instead of the “true” colors.

4. Color

a) Explain the RGBA color model, as used by OpenGL. How does the CMY model relate to RGB? Where are RGB and CMY used, predominantly?

b) Write GLSL code (or similar syntax) for a simple fragment shader that implements a non-photorealistic, “comic style” effect of using only a limited number of colors. More precisely, your fragment shader shall allow n gradations of color intensity in each of the R,G,B channels. (Choose your own n.) The shader has the following parameters:
`in varying vec4 color;`
`out varying vec4 gl_FragColor;`
You can use the following GLSL function: `genType trunc (genType x)` that returns a value equal to the nearest integer to x whose absolute value is not larger than the absolute value of x.

5. Illumination

a) Explain the difference between a *local* illumination model and a *global* illumination model, and give an example of each of these.

b) Explain how specular reflection is computed in the Blinn-Phong lighting model. Which material properties are used in this model? Make a sketch of the involved vectors; which added vector was Blinn's contribution to Phong's model and what is its purpose?

c) With OpenGL3, you can implement an illumination model either in the vertex shader or in the fragment shader. Explain the advantages of the two approaches, compared to each other.

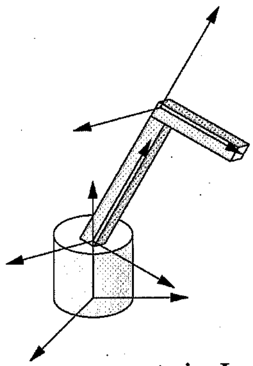
6. Vertices and Fragments

a) Explain the concepts *vertex, fragment, pixel,* related to each other.

b) There are two units actually removing data from the rendering pipeline, clipping and hidden-surface removal. Where are they placed inside the rendering engine, and why? On which type of data do they operate?

7. Robot Components

The robot arm shown on the right can be moved in three ways, as shown in the diagram: The base can be rotated on the ground by an angle α, the lower arm can be rotated by angle β relative to the base, and the upper arm can be rotated by angle γ relative to the lower arm.

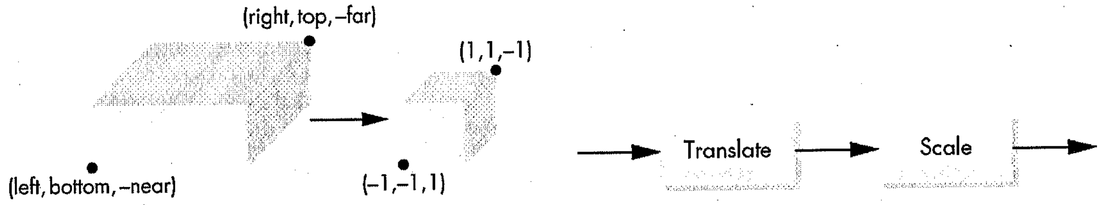


Write a method `void robot(float alpha,float beta,float gamma)` in Java (or in similar syntax) that draws the robot arm. You can assume there are three objects called `base`, `lower`, and `upper` with the following methods:
`void render(Mat4 modelview)` // renders this object based on the matrix
`float get_height()` // returns the height of the object

Hint: all you need to do is compute model-view matrices and call `render()` on the respective objects. You can assume to have a library of matrix operations like the one used in class.

8. Orthographic Projection

A call to `MatrixMath.ortho(left, right, bottom, top, near, far)` defines a viewing volume for orthogonal projection. `ortho` computes a matrix P:



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!