Exam

Computer Graphics

21 october 2010, 8-13

Answers may be given in either Swedish or English
Electronic calculator not allowed

1 (a) Give the formula for the *projection* of vector v onto another vector b. *(0.4)*
(b) Give an example of an *affine transform* which is not a *similarity transform. (0.6)*

2 Given three points in a plane $\boldsymbol{p}_0$, $\boldsymbol{p}_1$, and $\boldsymbol{p}_2$

(a) Give the definition for the *barycentric coordinates* for a point $\boldsymbol{p}$ with respect to $\boldsymbol{p}_0$, $\boldsymbol{p}_1$, and $\boldsymbol{p}_2$ *(0.5).*
(c) Give a formula which computes the barycentric coordinates. *(0.5).*

3 The techniques of *ray tracing* and *environment mapping* can be used to achieve similar effects. Explain the two techniques. In particular describe how they are different and what advantages they have with respect to each other. (*1.0*)

4 (a) Explain the term *tangent space*. (*0.4*)
(b) What is *Catmull-Rom interpolation*? (*0.3*)
(c) Describe the problems that can occur when you do *texture minification*? *(0.3).*

5. Describe the world created by the following RenderChimp code. *(1.0)*

```
w  = ... // World node
n1 = ... // Red quad node (0,0,0)-(1,1,0) in model space
n2 = ... // Green quad node (0,0,0)-(1,1,0) in model space
n3 = ... // Blue quad node (0,0,0)-(1,1,0) in model space
n4 = ... // Group node
n5 = ... // Group node
w->attachChild(n1);
w->attachChild(n5);
n1->attachChild(n4);
n5->attachChild(n2);
n4->attachChild(n3);
n1->setTranslate(5, 0, 0);
n2->setTranslate(-1, -1, 0);
n3->setTranslate(2, 0, 0);
n3->setScale(2, 1, 1);
n4->setRotateZ(0.5 * fPI);
n5->setTranslate(1, 1, 0);
n5->setRotateZ(fPI);
```

continue on the next page

6. Use the incomplete GLSL program below to create a Phong shader where one of the diffuse and specular components is implemented in the vertex shader and one is implemented in the fragment shader. Choose the placement of the components in such a way that artifacts are minimized. Motivate you choice. *(1.0)*

Vertex shader:

```
attribute vec3 Vertex;  // in model space
attribute vec3 Normal;  // in model space

uniform mat4 World;
uniform mat4 WorldViewProjection;
uniform mat4 WorldInverseTranspose;
uniform vec3 ViewPosition;

uniform vec3 WorldLightPosition;
uniform vec3 LightColor;

uniform float DiffuseMaterial;
uniform float SpecularMaterial;
uniform float Shininess;


// Your varying declarations here


void main() {

    // Your vertex shader code here

    gl_Position = WorldViewProjection * vec4(Vertex.xyz, 1.0);
}
```

Fragment shader:

```
// The same declarations as in the vertex shader


void main() {

    // Your fragment shader code here

    gl_FragColor = vec4(color, 1.0);
}
```

THE END!