EXAM IN

COMPUTER GRAPHICS

TSBK07

(TEN1)

Time: 4st of June, 2021, 14-18, extended to 19
for extra time for digitizing and uploading

Room: Distance

Teacher: Ingemar Ragnemalm, available by E-mail during exam, and by phone at 070-6262628.

Allowed help: Any. Copying text from other sources is not allowed. Copying solutions from other students is also not allowed.

Requirement to pass:
Grade 3: 21 points
Grade 4: 31 points
Grade 5: 41 points

ECTS:
C: 21 points
B: 31 points
A: 41 points

Answers may be given in swedish or english.

Upload answers to https://twokinds.se/liu/fileupload/
For assistance on the upload system, write to susanne@twokinds.se

In case of problems with uploading, you can also mail answers to ingemar.ragnemalm@liu.se, but you will need to keep file sizes low.

- Wish us luck!
- I wish you skill!
[Martin Landau, "Mission Impossible"]

1. OpenGL programming

a) Using glDrawElements rather than glDrawArrays will improve the rendering performance. What is the main benefit for performance? Motivate your answer.

(2p)

b) A student starting out with OpenGL wrote the following initialization code (plus some code for window/context, e.g. MicroGlut etc). But it doesn't work! Suggest what errors there are in the code. A good majority of the errors should be found for full score. You should state what the error is, not just point at a line and say that there is an error.

```
unsigned int vertexArrayObjID;
GLuint program;

void init(void)
{
	unsigned int vertexBufferObjID;
	unsigned int vertexArrayObjID;
	GLuint program;

	glClearColor(0.8,0.2,0.5,0);
	glEnable(GL_DEPTH_TEST);
	glEnable(GL_CULL_FACE);

	glBindVertexArray(vertexArrayObjID);
	glGenVertexArrays(1, &vertexArrayObjID);
	glGenBuffers(1, &vertexBufferObjID);

	glBindBuffer(GL_ARRAY_BUFFER, vertexBufferObjID);
	glBufferData(GL_ARRAY_BUFFER, 36*sizeof(GLfloat), vertices,
GL_STATIC_DRAW);
	glVertexAttribPointer(glGetAttribLocation(program, "in_Position"), 3,
GL_FLOAT, GL_FALSE, 0, 0);
	glEnableVertexAttribArray(glGetAttribLocation(program, "in_Position"));

	glUniformMatrix4fv(glGetUniformLocation(program, "worldToView"), 16,
GL_TRUE, worldToView);
	glUniformMatrix4fv(glGetUniformLocation(program, "projMatrix"), 16,
GL_TRUE, projectionMatrix);

	program = loadShaders("buggycube.vert", "buggycube.frag");
}
```

The array "vertices", describing a cube as a set of triangles, the matrices " worldToView" and " projMatrix" are given elsewhere in the code, and the files "buggycube.vert" and "buggycube.frag" exist and are correct. Additional data like normal vector are omitted for simplicity and does not count as a bug.

(3p)

c) Describe how a texture is accessed, including texture coordinates. The whole chain from disk to shaders should be included. Exact syntax is not important as long as it is clear what the operations are.

(3p)

2. Transformations

a) You wish to render a simple, planar shadow of some geometry (one or more objects) onto a given plane. (No, I am *not* talking about shadow mapping. That is not in this course.) The plane is given by the point **p** in the plane and the normal vector **n**. The shadow will be planar with no perspective, i.e. with a light source on high distance. It is shining along **n**. The plane is large enough to fit all shadows.

A suitable fragment shader is available that can blend or paint as desired. You do not need to specify this.

You should produce a solution as a sequence of 4x4 matrices, either "standard" ones as given by the course book or change of basis, including how it is calculated. The contents of each matrix should be specified as well as the multiplication order. You do not need to multiply the matrices together. *The matrix is one single matrix, calculated on the CPU and common to all models involved.*

Hint: This is related to the usual "rotation around arbitrary axis" kind of questions.

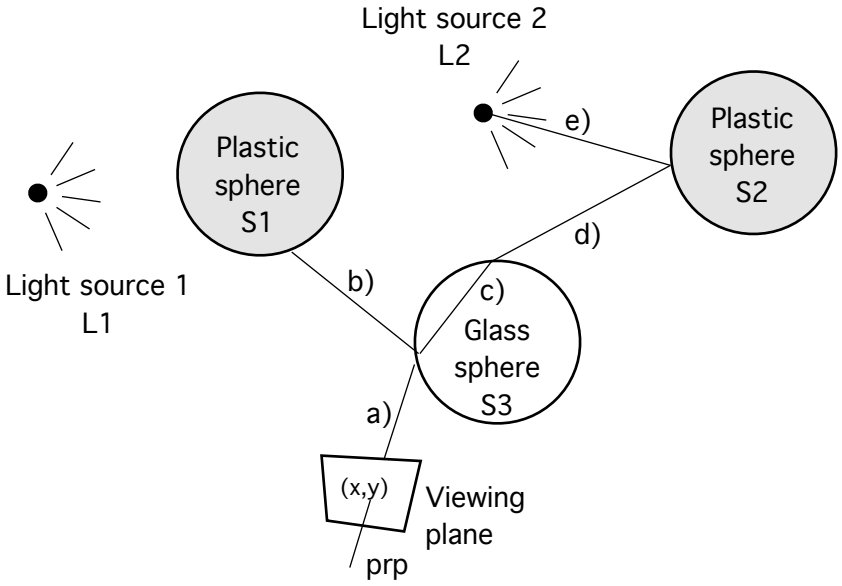
(4p)

b) Specify how to construct a camera placement (world-to-view matrix) using only rotations and translations, in order to place the camera in (-2, 0, 0), looking at (0, 1, 0). Draw a figure explaining what actually happens.

(3p)

3. Light, shading and ray-tracing

a) The following figure shows a simplified case for ray-tracing. Describe how the resulting pixel (x,y) is calculated from the given geometry, including any vital additional rays needed. Note: Everything happens in the same plane. Describe the process using the symbols in the figure.



(3p)

b) What will be the effect if the max() functions are omitted from the 3-component light model?

(1p)

c) How are shadows produced in the radiosity model? Clarify with a figure.

(2p)

4. Surface detail

a) When creating a skybox, you may either use cube mapping or conventional texture mapping. Compare the two methods with pros and cons of each.

(2p)

b) In mip-mapping, mip-mapping can be used on distant objects, but we get linear filtering when close. Why do we get that?

(1p)

c) Explain why cylindrical mapping by vertex gives problems in certain areas. What is the problem and how can it be avoided?

(2p)

5. Curve generation

a) Two segments of a 2D spline is given by the following functions:

$p_x(u) = 2 - u$
$p_y(u) = 1/(2-u)$
$q_x(v) = \sqrt{2} \cdot \cos(v + \pi/4)$
$q_y(v) = \sqrt{2} \cdot \sin(v + \pi/4)$

What continuity criteria do these segments fulfill for u=1, v=0? For full score this should be shown mathematically. Partial score can be given from reasoning from a figure.

(4p)

b) The Bézier curve is always inside the convex hull of its control points. Show that this is the case by using the definition/formulation of the spline.

(3p)

6. Miscellaneous

a) Linear post-filtering is a simple way to supress aliasing. Why can it produce fairly good results (sort of), and why is it still inferior to supersampling?

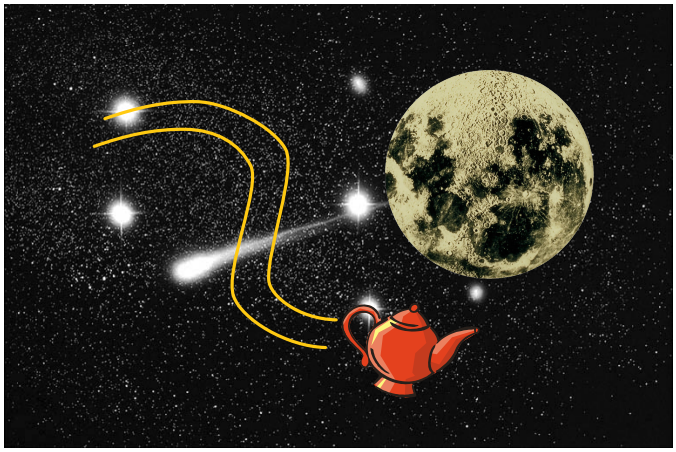
(2p)

b) Square-square is an often ignored noise generation algorithm. Compare it to Diamond-square in terms of performance and results.

(3p)

7. Collision detection and animation

a) You are making a game where a teapot is flying through space, the way teapots do*. This one is controlled by a pixie inside (the player) who can fly the teapot anywhere the pixie wants.



** This refers to Bertrand Russell's famous flying teapot analogy, which, however, is not part of this course, but only gives us a context for the question.*

Suggest a representation and strategy for movement control for this kind of scene, enough information to control the teapot, place it in world coordinates and move it in a suitable way. Camera placement does not have to be included.

(3p)

b) SAT is a popular theorem for basing collision detection on. However, it can not be directly applied as a method. Why?

(1p)

c) Typical SAT-based collision detection in 2D works in testing all sides of one object against all vertices of the second. You also need to test the other way around, sides of the second against vertices of the first. Why? Explanation by figure is recommended.

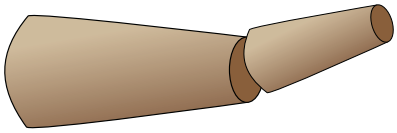
(2p)

8. Visible surface detection and large worlds

a) I claim that frustum culling should usually be made with *five* or *three* planes. Why? In what situations would each of these be suitable?

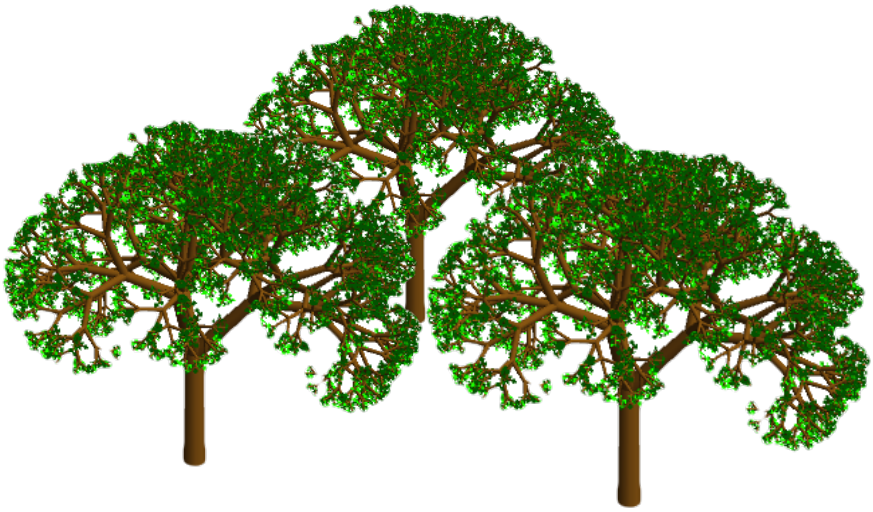
(2p)

b) Linus and Linnea are creating an exciting project, with fractal trees. Their approach is to use a branch model:



Two branch pieces almost fit together

This is scaled, rotated and translated to appropriate places. Leaves are rendered as single polygons. Everything is done with the fundamental methods used in the labs, and the resulting trees are very pleasing.



With their approach, they can only use a few trees in the scene before the frame rate suffers. Suggest *two* ways (other than frustum culling) to significantly improve the performance of the scene. Describe the remedies and motivate why they would help.

(4p)

Note: Some questions may have similarity with recent projects. This is just a matter of inspiration, and not a single project but several. I am not criticising these projects but rather expand on them. Some students have spent much time on these issues. If it makes these question easy for you, then you have deserved it!