TDA362/DIT223 Computer Graphics

EXAM

(Same exam for both CTH- and GU students)

Friday, April 26th, 2019, 14:00 - 18:00

Examiner

Ulf Assarsson, tel. 031-772 1775

Permitted Technical Aids

None, except English dictionary

General Information

Numbers within parentheses states the maximum given credit points for the task. Solutions shall be clear and readable. Too complex solutions can cause reductions in the provided number of points

Questions to examiner during exam

will be possible approximately one hour after the start of the exam. If anything is unclear – remember what has been mentioned on the lectures, in the slides and course book and do your best.

Grades

In order to pass the course, passed exam + exercises (or exam + project) are required. The final grade is calculated from the exam grade. The exam is graded as follows

CTH: 24p ≤ **grade 3** < 36p ≤ **grade 4** < 48p ≤ **grade 5**

GU: 24p ≤ **G** < 45p ≤ **VG**

Max 60p

Grades are announced by the LADOK system ~3 weeks after the exam

Solutions

will be announced on the course home page.

Review

Review date (granskningsdatum) is announced on the course home page.

Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

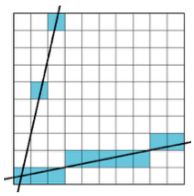
Question 1

a) **[1p]** What does the fragment shader do? (Hint: what is its input and output and when is it executed?)
Answer: Receives interpolated values from vertex shader and computes fragment color. E.g., refines lighting, adds textures, may modify depth.

b) **[2p]** What is the ModelViewProjectionMatrix? What does it do?
Answer: The matrix that transforms from modelspace (via worldspace) into viewspace and finally into unit- or projection space (or normalized device coordinates or homogeneous coordinates).

c) **[1p]** What is the z-buffer for?
Answer: to resolve visibility / avoid depth sorting.

d) **[1p]** Assume you have implemented a DDA algorithm for line drawing that only can draw lines with a low slope correctly, but has problems with steep lines (see image). How can you in a simple way make it plot steep lines as well?
Answer: Swap roles of x and y. I.e., loop over y, instead of x, to plot the (x,y)-value.



e) **[2p] Quaternions:** Describe how to perform a rotation of a point or a vector **p**, by 2θ degrees, around an axis **u** using quaternions.
Answer: $\mathbf{q} = (\sin\theta\,\mathbf{u}, \cos\theta)$, $\mathbf{p}_{rot} = \mathbf{q}\,\mathbf{p}\,\mathbf{q}^{-1}$.

f) **[2p]** Assume that you are creating a system where you have a rotation matrix **R**, a scaling matrix S, and a translation matrix T, to define an object's transform. For a vertex **v** of the object, show how you compute the transformed vertex **v'**. (Use the recommended order of the matrices. A one-line answer **v'**=... is enough.)
Answer: $\mathbf{v}' = (\mathbf{TRS})\,\mathbf{v}$

g) **[1p]** State a simple 4x4-matrix **M** that performs a scaling (s_x, s_y, s_z).
Answer: $\mathbf{M} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

Question 2

a) **[3p]** Describe flat shading, Gouraud shading and Phong shading.
Answer: Flat shading: illumination computed using the triangle-plane's normal.
Gouraud-shading: illumination computed per triangle vertex and for each pixel, the color is interpolated.
Phong Shading: For each pixel, the normal is interpolated from the 3 vertex-normals. Full illumination is done with this interpolated normal.

b) **[1p]** While a normal might be normalized in the vertex shader, why do we generally need to normalize it again in the fragment shader?
Answer: As values get interpolated between the vertices from the vertex shader to the fragment shader, the interpolated normal in not necessarily normalized.

c) **[2p]** What is 2x2 grid, 2x2 RGSS, and 8 rooks. Draw them!
Answer:

d) **[2p]** How much extra memory does it take to store a mipmap-hierarchy of a texture, compared to just the base texture itself?
Answer: ~1/3 extra (or ~33%).

e) **[2p]** Can you do displacement mapping with the vertex- and fragment shader? Motivate!
Answer: You can do vertex-based displacement mapping in the vertex shader. You cannot simply displace the output from the fragment shader (although workarounds could be possible nowadays using raycasting or OpenGL side effects.)

Question 3

a) **[2p]** To draw transparent objects using for instance OpenGL, you typically divide the triangles in two groups: the transparent ones and the opaque ones. 1) Which of these two groups needs to be sorted, 2) why, 3) and in which order?
Answer: 1) the transparent triangles, 2) for correct blending, 3) back-to-front.

b) **[1p]** How can you implement fog using the fragment shader?
Answer: E.g., (linearly/exponentially) blend between the fragment color and fog color based on the fragment's depth. To get full points, draw or explain in more detail.

Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

c) **[1p]** Normally, you would want to draw all geometry that is in front of the camera. So, why is a near and far plane used for the view frustum?
Answer: Near plane is used to avoid a degenerate projection plane. Far plane can be used to get better z-precision in the z buffer.

d) **[4p]** Describe two methods/algorithms to determine if a **2D** point is inside a **2D** polygon (i.e., point in polygon test). Convex polygon can be assumed.
Answer: (Your answers of course have to be more detailed than below and also explain how this is calculated):
1) angle sum = 360 instead of 0.
2)*The Crossings Test* (s 583). Count crossings between (horizontal) line and the polygon edges. Odd number = inside, even number = outside.
3) Check if point is inside the triangles formed by one vertex and the other vertices. If inside odd number -> point is inside polygon, else outside. (See Bondesson's OH-slides on the course home page, p: 227).
4) Split into triangles and check them (naive way but OK)
5) (Use Plücker coordinates - but for 2D.)
6) Test point against edges with 2D cross product.

e) **[1p]** Describe a test which determines whether or not two spheres intersect.
Answer: $||c_1 - c_2|| <= r1 + r2$.

f) **[1p]** Describe a test which determines whether or not a sphere and a plane intersect.
Answer: Insert the sphere center into the plane equation. If the result is smaller than the radius, there is an intersection.
I.e., $||\mathbf{n} \bullet \mathbf{c} + d|| <= r$.

Question 4

a) **[2p]** Describe a top-down approach to build an Axis Aligned BSP-tree for a scene.
Answer: Create minimal AABB around the whole scene. Split along an axis. Recursively split the 2 parts alongs a new axis. Terminate if empty node, #triangles < threshold or level >= max recursion depth. (Axis aligned: x-,y-,z-axis are the split planes.)

b) **[1p]** Why is bubble sort efficient when we have high frame-to-frame coherency regarding the objects to be sorted per frame?
Answer: Bubble-sort (or insertion sort) has expected runtime of resorting already almost sorted input in O(n) instead of O(n log n), where n is number of elements.

c) **[1p]** Why can't we use ray tracing to do an exact object-object collision detection test?
Answer: Only tests a discrete subset of surface points and directions. A finite set of rays may miss small intersections.

Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

d) **[1p]** What is the point of using **dynamic** collision detection tests?
Answer: Objects may collide in space and time although they never collide in any of the frames rendered at discrete time steps.

e) **[1p]** What is a shadow cache? Explain what it is good for and how it works.
Answer: pointer to previous intersected triangle (primitive) by the shadow rays. Null, if last shadow ray had no intersection. Reason: speedup, potentially avoiding tree traversal.

f) **[2p]** Describe a common recursive pattern for adaptive super sampling (suggestively the one that was taught on the lectures). Draw start samples, samples after one level of recursion and criteria to terminate or continue the recursion.
Answer: see lecture Raytracing 1

g) **[2p]** Explain how a skippointer tree works and what the advantage is?
Answer: The (BVH)-tree stored in depth first traversal order, sequentially in an array, with a pointer/index to the next element if traversal of subgraph should be skipped. **Advantage:** gives good cache coherence.

Question 5

a) **[2p]** What is a BRDF? (You can answer by stating what the abreviation BRDF stands for and assuming *f*() is a BRDF and describe its input parameters and what it returns.)
Answer: Bidirectional Reflection Distribution Function. The function $f(\omega_i, \omega_o)$ returns how much of the radiance from the incoming direction ω_i that is reflected in the outgoing direction ω_o.

b) **[2p]** Two separate photon maps are constructed during photon mapping. **Which** and **why**?
Answer: Caustics map and Global map (slowly varying illumination). The reason is that for correctness, the global map uses an even distribution of shot photons from the light source. For efficiency reasons, this is violated for the caustics map, when instead photons are fired in the directions of specular objects to capture the caustics effects. Physical correctness of the light intensity is typically much less visible in the latter case.

c) **[1p]** What is caustics?
Answer: specular reflections or refractions that focus light.

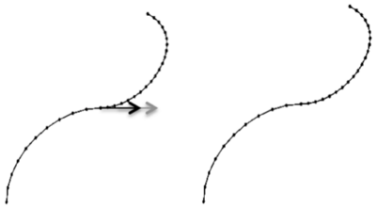
Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

d) **[2p]** Describe the advantages and disadvantages of shadow maps vs shadow volumes. You should mention at least a total of four important bullets (0.5p per unique bullet).
Answer: SM Pros: any rasterizable geometry, constant cost per rasterized fragment from the camera's view (basically just a texture lookup), fast.
SM Cons: jagged shadows / resolution problems, biasing.
SV Pros: sharp shadows.
SV Cons: 3 or 4 rendering passes (and thus often slower than shadow maps), lots of polygons and fill.

e) **[3p]** Describe the shadow map algorithm.
Answer: 1. Render a shadow (depth) map from the light source.
2. Render image from the eye. For each generated pixel, transform/warp the x,y,z-coordinate to light space and compare the depth with the stored depth value in the shadow map (at the pixel position (x,y).
If greater → point is in shadow.
Else → point is not in shadow.
(Bias/offset is necessary due to disretization and precision problems.)

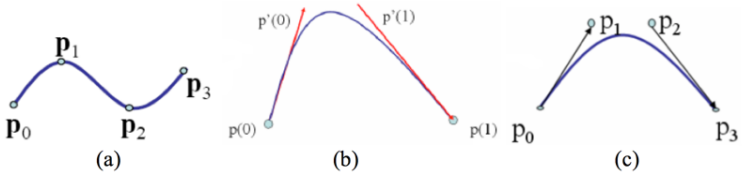
Question 6

a) **[1p]** What continuity does the curve below have? Motivate. (I.e., the two curves are exactly the same curve, but with and without arrows to depict the type of gradient discontinuity).



Answer: G^1 continuity. Different gradients, but same direction.

b) **[3p]** Tell which type of curve each image corresponds to. You can choose between Bezier curve, Interpolation-curve, and Hermite-curve. To get any points, you must also motivate your choice!



Answer: a) interpolation curve since the curve goes through the control points. b) Hermite-curve since gradients are specified per control point. c) Bezier-curve since two intermediate points are used to approximate the gradients in the end points.

c) **[1p]** Assume **p**=(1,2,8,7). Perform the homogenisation step on **p**.
Answer:

d) **[1p]** Manually normalize the vector **x**=(0,2,1).
Answer:

e) **[2p]** What is direct illumination and what is indirect illumination? (No formulas needed - words are enough.)
Answer: Direct illumination is the light that comes directly from a light source. Indirect illumination is the inter-reflected light, such as reflections, irradiance, shadows, caustics, etc.

f) **[2p]** Linear interpolation of (u,v) in screen space does not give perspective correct texturing. Describe how perspective correct texture interpolation can be achieved.
Answer: In screen space, linearly interpolate (u/w, v/w, 1/w) from each vertex.
Then, per pixel: $u_i = (u/w)_i / (1/w)_i$, where *i* means screen space interpolated value.

Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY