TDA361/DIT220 Computer Graphics, January 14th 2015

EXAM

(Same exam for both CTH- and GU students)

Wednesday January 14th, 2015, 14.00 – 18.00

Examiner

Ulf Assarsson, tel. 0701-738535

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

Opportunity to review the correction of your exam is provided on Wednesday February 11th at 12.00, room 4117 (next to my office room), 4th floor, west corridor, EDIT-building.

Institutionen för datorteknik
CHALMERS TEKNISKA HÖGSKOLA

Question 1

Pipeline

a) **[3p]** Explain the rendering pipeline and corresponding hardware and draw figure.
Answer: see lecture 1 or full-time wrapup slide 5. 0.5p for each of: application stage, geometry stage with vertex+geometry shader, rasterization stage with pixel shader (+ merge).

b) **[2p]** Explain screen tearing.
Answer: see lecture 1. Solved by double buffering and syncing buffer swapping with the vblank.

Vectors and Transforms

c) **[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).

d) **[1p]** What is the matrix to transform the normals, if matrix **M** is used to transform the positions?
Answer: (M^{-1}**)**^{T}.

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: $\mathbf{q} = (\sin\theta\ \mathbf{u}, \cos\theta)$, $\mathbf{p}_{rot} = \mathbf{qpq}^{-1}$.

Question 2

Shading, Antialiasing and Texturing

a) **[1p]** Which types of filters are common in **real-time** computer graphics for **magnification** of 2D textures? (Each listed wrong filter will give negative score. Sum will however not go below 0p.)
Answer: nearest (box filter), (bi)linear (tent filter). Mipmapping is wrong.

b) **[2p]** Which types of filters are common in real-time computer graphics for **minification** of 2D textures? (Each listed wrong filter will give negative score. Sum will however not go below 0p.)
Answer: nearest (box), biliear filtrering (tent) with and without mipmapping. Trilinear filtering with mipmapping, anisotropic filtering.

Institutionen för datorteknik
CHALMERS TEKNISKA HÖGSKOLA

c) **[3p]** How often are the vertex shader, geometry shader and fragment shader executed when one triangle is sent as input for drawing by the hardware? Explain your answer.
Answer: Three, once, and once per non-occluded fragment (unless supersampling schemes are used, which executes the fragment shader per fragment sample).

d) **[1p]** What is the difference between supersampling and multisampling?
Answer: multisampling only runs the fragment shader **once** per **fragment** (not once per sample-inside-each-pixel).

e) **[1p]** How much more memory does a mipmap hierarchy require compared to just using the base texture?
Answer: ~1/3 extra (or ~33%).

f) **[2p]** Which is fastest to compute for a GPU - Phong shading or Gouraud shading? Motivate!
Answer: Phong shading is more expensive, since full lighting is computed per pixel – instead of just per vertex and interpolated for each pixel. (If triangles are on average smaller than a pixel, then vice versa could be true.)

Question 3

OpenGL

a) **[2p]** Assume that you have enabled backface culling. How does the hardware determine if a triangle will be backfacing or frontfacing? (Drawing an image can be useful.)
Answer: The vertices v0, v1, v2, are projected onto the screen and if they appear in anti-clockwise order (right-hand side rule), the triangle is (by default) assumed to be front facing. Otherwise, backfacing. (The order could be reversed by specifying CW-order.)

Spatial Data structures

b) **[5p]** Draw a simple scene and visually divide it into
1) an Axis Aligned Bounding Box hierarchy,
2) an Octree or Quadtree,
3) an Axis Aligned Binary Space Partitioning Tree,
4) a grid,
5) **and** a recursive or hierarchical grid

Institutionen för datorteknik
CHALMERS TEKNISKA HÖGSKOLA

You can do all your drawings in 2D for simplicity and you may use different scenes for the different hierarchies. (You don't need to show the corresponding tree structures). **The differences** in the spatial divisions between the five types of data structures **should be clear** from your examples, for any score.

Collision Detection

c) [2p] Why is the sweep-and-prune algorithm efficient in course pruning non-colliding objects?
Answer: Uses bubble-sort (or insertion sort) (1p), which have expected runtime of resorting already almost sorted input, in O(1), instead of O(n log n), where n is number of elements. Flip bits of matrices could also give 1p (for a max of 2p).
See sweep-and-prune:
Collision detection for x,y,z-axes, sort start and end of AABBs, active interval list, update sort using bubble sort, flip bits of matrices.

d) [1p] Describe the advantage and disadvantage of using ray tracing for collision detection compared to using bounding volume hierarchies.
Answer: Advantage: ray tracing typically a lot faster. Disadvantage: cannot do exact collision detection (only sampling based).

Question 4

Global Illumination

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. 0.5 points for each correct answer of: BRDF abreviation, ω_i, ω_o and return value.

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) [2p] Why do we want to use Final Gather when using photon mapping?
Answer: Because using the global photon map for primary rays will result in poor low-pass filtering. I.e., give a blotchy (noisy) looking result. A

blotchy result in the second recursion level/for secondary rays is less severe, since indirect illumination is a low-frequency phenomenon.)

d) **[4p]** There are several other algorithms for global illumination than just photon mapping. **Name and explain** at least two of them.
Answer: Path Tracing, Bidirectional path tracing, Monte Carlo Ray Tracing, Metropolis Light Transport, (Radiosity). See lecture slides for explanations.

Question 5

Ray Tracing and Shadows

a) **[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.

b) **[2p]** Describe how to compute soft shadows in **ray tracing**.
Answer: e.g., shoot several shadow rays to different light samples. It is also OK to answer for path tracing – choosing one random light sample.

c) **[4p]** Describe the advantages and disadvantages of shadow maps vs shadow volumes. You should mention at least a total of four important bullets (1p 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, Cons: jagged shadows / resolution problems, biasing.

SV - Pros: sharp shadows. Cons: 3 or 4 rendering passes (and thus often slower than shadow maps), lots of polygons and fill.

Curves

d) **[1p]** Sketch one non-continuous curve and one C^0-continuous curve (and mark which is which).

e) **[2p]** Explain what C^k-continuity means and what G^1-continuity means for curves.
Answer: C^k-continuity means that the derivatives of order 0 (or 1) to k exists and are continuous. G^1-continuity means that the tangent vectors between each curve segment have equal directions, but their lengths (strengths) may vary.

Institutionen för datorteknik
CHALMERS TEKNISKA HÖGSKOLA

Question 6

Hardware

a) **[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: $u_i = (u/w)_i / (1/w)_i$, (i=screen space interpolated value)

Miscellaneous math

b) **[1p]** Assume **p**=(4,2,1,3). Perform the homogenization step on **p**.
Answer: p=(4/3, 2/3, 1/3, 1).

c) **[1p]** Manually normalize the vector **x=**(3,1,2).
Answer: x=x/|**x**| = (3,1,2) / sqrt($3^2+1^2+2^2$) = (3,1,2)/sqrt(14).

Intersections

d) **[2+2p]** 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) Why does not (classic) environment mapping (e.g., the ones taught in the course) work well for **planar** reflections? (Your answer can preferably include an illustration. As an example, you could use a man standing on a floor, where the man should reflect in the reflective floor.)
Answer: the environment map is, for each pixel-to-be-shaded, assumed to be centered on that pixel. Reflection vectors often change little on planar surfaces, and the environment-map pixel is only dependent on the reflection vector (not its origin). Another way to say this is that the environment map represents surrounding geometry **infinitely** far away.

Institutionen för datorteknik
CHALMERS TEKNISKA HÖGSKOLA