TDA362/DIT224/DIT223 Computer Graphics

EXAM

(Same exam for both CTH- and GU students)

Friday, January 14th, 2022, 08:30 - 12:30

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.

© Ulf Assarsson, 2022
Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

Question 1

a) **[2p]** Explain what the z-buffer is used for and how it works.
Answer: For resolving visibility per pixel. The z buffer stores, for each pixel, the depth of the closest fragment so far. For each new fragment, compare its depth to the depth in the z buffer. If less, replace pixel color and update z buffer.

b) **[2p]** Which are the four buffers a default frame buffer typically (or often) consists of?
Answer: Front Buffer, Back Buffer, Depth Buffer, Stencil Buffer

c) **[1p]** What is the advantage of Bresenham's line drawing algorithm?
Answer: uses only integers

d) **[3p]** Let **A** be the model-to-world matrix, **B** be the world-to-camera matrix, and **C** be your projection matrix. What matrix do you use to transform the per-vertex *normals* to camera space?
Answer: Model-to-view-Matrix is **BA**.
Normal matrix is transpose of inverse (or inverse of transpose) of ModelView matrix. I.e., $((\mathbf{BA})^{-1})^T$.

e) **[1p]** For what do we typically use quaternions in computer graphics? Is it scalings, rotations, translations, skewings, projections or something else?
Answer: To do rotations.

f) **[1p]** What is a rigid body transform?
Answer: Only rotations and translations (no change of size or shape).

Question 2

a) **[2p]** Give two reasons for gamma correction.
Answer: 1) screen output is non-linear so we need gamma to counter that. 2) Textures/images can be stored with better precision (for human eye) for low-intensity regions.

b) **[1p]** Does the rendering order of transparent objects (e.g., triangles) matter? Motivate for any point.
Answer: Yes. Back-to-front order (see lecture 3).

© Ulf Assarsson, 2022
Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

c) **[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), bilinear filtrering (tent) with and without mipmapping. Trilinear filtering with mipmapping, anisotropic filtering

d) **[1p]** Up to how many texel accesses are required for highest-quality 16x anisotropic filtering of one texture-lookup request? And why?
Answer: 8x16 = 128.
I.e., 8 texel accesses per trilinear-filtered lookup, and up to 16 texture such trilinear mipmap lookups along the line of anisotropy.

e) **[2p]** Describe how the texture coordinates are computed for cube-mapping. I.e., how to compute the texture coordinates (u,v) and how to decide which of the 6 texture sides in the cube map that is to be used, given the eye position, **e**, and the normal, **n** for a point **p**.
Answer: Compute reflection vector, **r**.
Largest abs-value of component determines which cube face to use.
Example: **r**=(5,-1,2) gives POS_X face. Divide **r** by 5 gives (u,v)=(-1/5,2/5).
Possibly also normalize to [0,1] by adding [1,1] and multiplying by [0.5, 0.5].

f) **[2p]** Assume that you are rendering an impostor (i.e, billboard) as a rectangular quad. The impostor's texture represents a tree and contains lots of fully transparent texels. How do you assure that objects that are later (for the same frame) rasterized behind the transparent parts of the impostor will show on the screen? (**Hint:** use the fragment shader.)
Answer: alpha test, fragment kill, i.e., kill fragment if alpha = 0 in fragment shader.

Question 3

a) **[2p]** Assume that you have enabled backface culling. How does the hardware determine if a triangle will be backfacing or frontfacing? Draw an explanatory image!
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.)

© Ulf Assarsson, 2022
Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

b) **[3p]** What is a vertex shader, geometry shader and fragment shader? Explain their functions, respectively.
Answer:
Vertex shader: per vertex operations like projection to unit-cube (2D) and per-vertex attributes to be interpolated (color,normal).
geometry shader: take input primitives and output possibly another amount and possibly another type of primitives.
Fragment shader: compute and output pixel color, typically from interpolated data from vertex shader.

c) **[3p]** Explain an efficient method for determining intersection between a ray and a box (in 3D).
Answer: Find t_{min} and t_{max} for the intersections against each of the three slabs. Keep *max* of the three t_{min} and *min* of the three t_{max}. If $t_{min} < t_{max}$, there is an intersection. (Check for degenerate case when ray parallel to slab.)

d) **[2p]** Describe the separating axis theorem **and** when it can be used.
Answer:

Question 4

a) **[4p]** For some scene, draw:
- an Axis-Aligned Bounding Box hierarchy
- a Sphere hierarchy
- an OBB hierarchy
- an Octree or Quadtree
- an Axis-Aligned Binary Space-Partitioning Tree
- a Polygon-Aligned Binary Space-Partitioning Tree
- a grid
- a recursive and a hierarchical grid

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 all the types of data structures **must be clear** from your examples, for any score.
Answer:

b) **[3p]** Explain Occlusion Culling, Detail Culling, View Frustum Culling, Portal Culling, Backface Culling and Levels of Detail.
Answer:

c) **[1p]** Name a sorting algorithm that is efficient when we have high frame-to-frame coherency (regarding the objects to be sorted per frame)?
Answer: Bubble-sort (or insertion sort), 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.

© Ulf Assarsson, 2022
Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

d) **[2p]** Describe an efficient and conservative method for course pruning of non-colliding objects.
Answer: E.g., Use a grid with an object list per cell, storing the objects that intersect that cell. For each cell with list length > 1, test those against each other with a more exact method. Explaining the sweep-and-prune algorithm is also OK.

Question 5

a) **[4p]** Describe the algorithm for path tracing (2p). Remember to include how light sources are handled. Also explain what makes the path-tracing algorithm efficient compared to more naive Monte-Carlo sampling strategies (2p)
Answer: Shoot many paths per pixel, by randomly choosing one new ray at each interaction with surface + one shadow ray per light. Terminate the path with a random probability. Spawning many rays at each bounce would result in a ray tree where most work would be spent on the many rays at the bottom of the tree which have least impact on the pixel. Path tracing only follows one randomly chosen spawned ray per bounce, resulting in a ray path. Instead, many paths are traced per pixel. Better balance between spent work and importance of rays.

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

c) **[2p]** Why do you need to use a bias in the shadow map algorithm? What is the cause of the problem?
Answer: We compare two different discretizations - one from the eye and one from the camera. (To avoid z-fighting (incorrect self shadowing) and light leaking.) The view sample can lie further from the light than the shadow map sample (due to discrete sampling).

d) **[1p]** How do you modify the z-pass algorithm into the z-fail algorithm?
Answer: The problem with the eye located within shadow volumes (stencil count gets wrong) is solved. Counting is done to infinity instead of to the eye. Invert the depth test and stencil inc/dec. I.e., use depth test GREATER and inc for backfacing shadow quads instead of vice versa. Also render near- and far capping shadow volume planes.

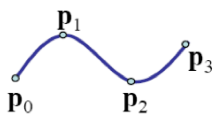
e) **[1p]** We have talked about two major classes of real-time shadow algorithms in this course. Which of these is percentage closer filtering related to?
Answer: Shadow maps

f) **[1p]** Assume you have an **incoming** light ray **l** at a surface point **p** with the normal **n** (which is normalized). Compute the reflection vector **r**. Draw figure!
Answer: $\mathbf{r} = \mathbf{l} - 2(\mathbf{n} \cdot \mathbf{l})\mathbf{n}$.

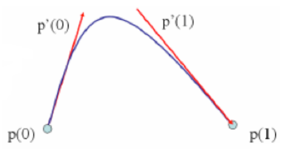
© Ulf Assarsson, 2022
Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

Question 6

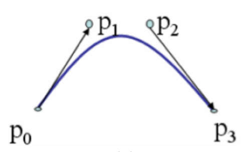
a) **[3p]** Tell which type of curve each image corresponds to. You can choose between Bezier curve, Interpolation-curve, and Hermite-curve. To get any points, you must also motivate your choice!



(a)



(b)



(c)

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.

b) **[2p]** In which ways are NURBS more general than B-Splines?
Answer: control points can be set at non-uniform intervals and they can have different weights.

c) **[2p]** Compute how far above or below point **p**=(-2,-7,4) is with respect to the plane with **n**=(3,4,4) and d=-5. (Remember to use a normalised normal.)
Answer: Signed distance = n • p + d = -23 / sqrt(41).

d) **[3p]** Draw the graphics-pipeline's major functional blocks (i.e., logical layout) and their relation to hardware, for a modern graphics card. (Hint: preferably I want the major functional blocks as described in the hardware lecture, e.g. with the different parallel shader units and the other functional units).
Answer:

© Ulf Assarsson, 2022
Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY