TDA361 Computer Graphics

EXAM

Monday Month XX:th, 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 and project are required. The final grade is calculated from the project grade and exam grade. The exam is graded as follows

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

Max 60p

Solutions
provided Friday May 26 outside room 4220.

Grades
are announces no later than Wednesday June 7 at the institution's standard notice-board..

Review
Opportunity to review the correction of your exam is provided on Wednesday June 13:th, room EE, 12.00-13.00 (the standard lecture hall for this course).

Institutionen för datorteknik
CHALMERS TEKNISKA HÖGSKOLA

Task 1 – Pipeline

a) **[1p]** The real-time graphics **pipeline** consists of three parts. Which?
Answer: application stage, geometry stage, rasterization stage

b) **[1.5p]** Give examples of what is done in each part.
Answer: Application stage – e.g. VFC, animation.
Geometry stage: transformation + per vertex shading (lighting).
Rasterization stage: rasterization, texturing, interpolation of per-vertex values from vertex shader, z-test, fragment shading.

c) **[1.5p]** For each part, describe how you can determine if this step is the performance bottle-neck for the rendering.
Answer:

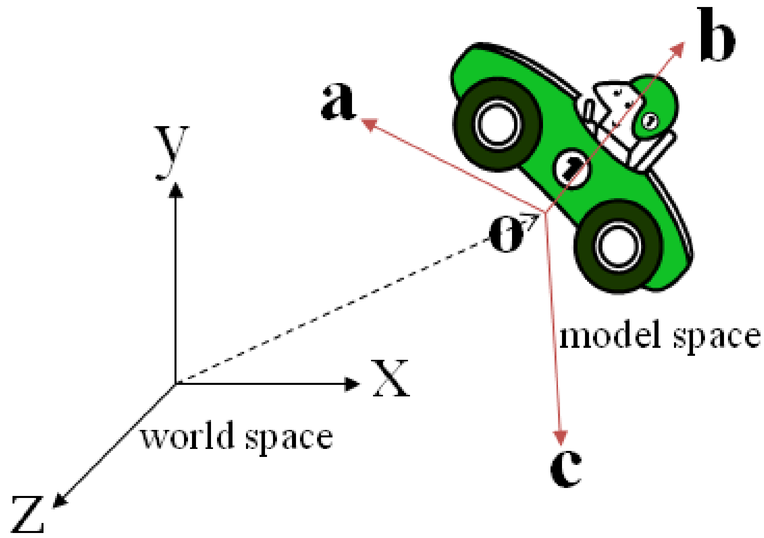
Application stage:	swap glVertex to glColor
Geometry stage:	remove all light sources
Rasterization stage:	Change window size

d) **[1p]** Give examples what the accumulation buffer can be used for? I.e., give (at least) two examples of effects that can be accomplished using the accumulation buffer.
Answer: Motion blur – blend together several frames, depth-of-field - blend images from slightly modified camera positions, (soft shadows – render hard shadow from several point light sources and blend the shadow contributions)

Task 2 – Transforms

a) **[2p]** Which two classes of transformations are part of Rigid Body Transformations?
Answer: translation, rotation

b) [2p] Compute the object's model-to-world matrix.



Answer:

$$M_{\text{model-to-world}} = \begin{bmatrix} c_x & b_x & a_x & o_x \\ c_y & b_y & a_y & o_y \\ c_z & b_z & a_z & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) [1p] Create a 4x4-matrix that performs a uniform scaling, *d*, in x-, y- and z , by utilizing the homogenization step. You must show how the homogenization step leads to the scaling.

Answer: scaling: $M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1/d \end{bmatrix}$, p = (x,y,z,1), Mp = (x,y,z,1/d),

Homogenization: => $p' = \left(\frac{x}{1/d}, \frac{y}{1/d}, \frac{z}{1/d}, \frac{1/d}{1/d}\right)$ = (xd, yd, zd, 1)

Task 3 - Illumination and Visual Appearance

a) [1p] Which are the 3 components in the real-time illumination model? It is sufficient to just state the names. (**Emission** is often included as the fourth component.)

Answer: ambient, diffuse, specular,

b) **[2p]** Compute the reflection ray, **r**, given **n** and **l**, where **n** is the surface normal and **l** is the incoming ray with direction towards the surface.
Answer: $\mathbf{r} = \mathbf{l} - 2*(\hat{\mathbf{n}} \cdot \mathbf{l})\hat{\mathbf{n}}$, (**n** needs to be normalized, **l** does not necessarily)

c) **[1p]** Is alpha channel in the color buffer required for correct rendering of transparent objects? Motivate your answer.
Answer: No, you state the transparency using the alpha value of the color of the object. The alpha value, *a*, decides the interpolation factor between the source color *c* (the object's color) and the destination color *d* (the color of the pixel in th frame buffer). E.g.: Color = $ac + (1-a)d$. The alpha channel in the color buffer does not need to be involved. For correct blending of the transparency, draw the transparent objects in back-to-front order.

d) **[1p]** Is the rendering of transparent objects order dependent? Motivate.
Answer: yes, the blending operator is order dependent (unless you have a pure additive or multiplicative blending – but both are used for classic transparency)

Task 4 – Aliasing

a) **[2p]** Draw RGSS, FlipQuad, Quincunx and 8 rooks. State the weights for each sample point.
Answer: see lecture slides

b) **[1p]** Explain jittering. What is the advantage compared to other sampling schemes?
Answer: see lecture slides

c) **[1p]** Explain how tri-linear filtering works. Draw picture.
Answer: see lecture slides

d) **[1p]** Explain how anisotropic filtering works.
Answer: see lecture slides

Task 5 –Intersection Tests

a) **[2p]** Compute, analytically, the intersection between a ray and a sphere.

Answer: Write down the equations for both objects and find the common solution(s).
(1) ray: p = o + td
(2) sphere: ||**p**-**c**|| = r;
Square: $(\mathbf{p}-\mathbf{c})\bullet(\mathbf{p}-\mathbf{c})=r^2$
Insert (1) into (2) and solve for t.

b) **[3p]** Describe a method/algorithm for computing the intersection between a ray and a triangle (both defined in 3D).
Answer:
Compute the ray's intersection point with the triangle plane (similar to above but with a plane instead of a sphere). Then, do a point-in-polygon test (in 2D) according to one of the following:
1) sum of angles = 360 if point is inside. Otherwise = 0.
2)"The Crossings Test" (see slides or RTR-book). Compute the number of crossings between a (horizontal) line and the edges of the polygon. Odd number = inside, even number = outside.
3) Use a point-triangle test. Check the point against all triangles formed by one vertex and each of the other vertices. If inside odd number -> point is inside polygon, otherwise outside. (See Bonus OH page: 269). Requires that you state how to solve point-triangle intersection test.

Task 6 – Collision Detection, Spatiala Data Structures and Speedup Techniques

a) **[3p]** Shortly explain Occlusion Culling, Detail Culling, View Frustum Culling, Portal Culling, Backface Culling and Levels of Detail.

b) **[1p]** Write pseudo-code for hierarchical view frustum culling.

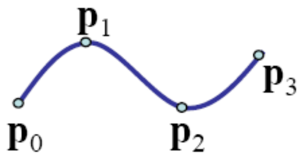
c) **[1p]** What is the difference between a kD-tree and an Axis aligned BSP-tree?
Answer: Axis aligned BSP-tree with fixed split plane order, e.g.: x-,y-,z-,x-,y-,z,...

d) **[2p]** Describe the Separating Axis Theorem.

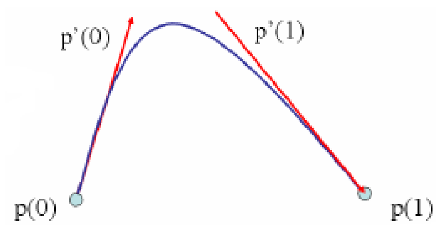
e) **[3p]** Give the names of six of the most common spatial data structures within computer graphics, except the three already mentioned above.
Answer: (Axis Aligned BSP tree, kd-tree, BVH), Polygon Aligned BSP tree, grid, octree, quadtree, hierarchical grids, recursive grids, Bounding Volume Hierarchies such as Sphere hierarchy/AABB/OBB/(kDOP)

Task 6 – Curves

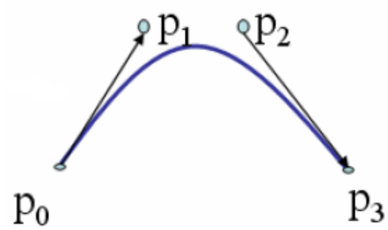
a) **[5p]** State the type of curve that each image represents. Choose between Interpolation-curve, Hermite-curve and Bezier-curve. Motivate your choice:



I.



II.



III.

b) **[5p]** Describe how curve drawing using **B-splines** work, given a set of control points. (I.e., explain B-splines). **Hint**: state the expression for the curve **p**(u) = Σ…, sketch the basisfunction and give the intervals, sketch and/or state the blendfunctions B_i for the points $\mathbf{p}_i$.

Task 7 – Ray Tracing

a) **[2p]** Draw a common recursive pattern for adaptive super-sampling (it is appropriate to draw the one taught in this course). Provide start samples and samples after one recursion step.
Answer: see ray tracing lecture slides

b) **[2p]** Explain *skippointer tree*. What is the advantage?

c) **[1p]** Explain ”shadow cache”

Task 8 – Global Illumination

a) [5p] Describe the algorithm for photon mapping.
Answer: Caustics map, Global Map (for indirect lighting), shoot photons from light source, store at diffuse surfaces. Ray trace from eye and use the maps for their respective contribution. Grow spheres to estimate intensity. Russian Roulette to decide if semi-diffuse materials are considered specular or diffuse.

b) [2p] Describe path tracing and explain what the benefit is (compared to standard Monte Carlo Ray Tracing).
Answer: Randomly shoot one path per ray – no exponential ray tree. Reason: to have equally many of the important primary rays as rays at the end of the paths.

c) [3p] This is ”the rendering equation”. Explain this equation by describing all the components of the equation.

$$L_o = L_e + \int_{\Omega} f_r(\mathbf{x}, \omega, \omega') L_i(\mathbf{x}, \omega')(\omega' \cdot \mathbf{n}) d\omega'$$

f_r is the BRDF, **w**’ is incoming direction, **n** is normal at point **x**, Ω is hemisphere ”around” **x** and **n,** L_i is incoming radiance, **x** is position on surface, **w** is outgoing direction vector.
$L_o(\mathbf{x},\mathbf{w})=L_e(\mathbf{x}, \mathbf{w})+L_r(\mathbf{x}, \mathbf{w})$
I.e.,: outgoing radiance =emitted + reflected radiance. The integral is the reflected radiance in direction **w**.

Task 9 – Collision Detection, Shadows and Reflections

a) [5p] Describe the z-fail-algorithm.
Answer: Create shadow quads from the silhouettes of the shadow caster (as seen from light source). Cap shadow volume with the shadow caster’s front facing polygons (as seen from light source). Also cap at far end (often done by pushing the back-facing triangles towards infinity). Render ambient contribution to color buffer. Set depth test to GL_GREATER and turn off updating of z-buffer, Render backfacing (as seen from camera) shadow volume polygons to stencil buffer, incrementing stencil values. Render front facing shadow volume polygons to stencil buffer, decrementing stencil values. Render diff+spec lighting contribution to color buffer where stecil buffer = 0.

b) [2] Write pseudo code for collision detection between 2 bounding volume hierarchies.

Task 10 – Hardware

a) [2p] State at least 4 bandwidth reducing techniques used on graphics cards.
Answer:
– Texture caching with prefetching
– Texture compression
– Z-compression
– Z-occlusion testing (HyperZ)
– (Pipelining)
– threads

b) [3p] Sketch the architecture for a modern graphics card, e.g. the Geforce 8800. You can use the major functional blocks that we have mentioned in the course.
