CSE167, Introduction to Computer Graphics
Midterm, Tuesday May 15

Matthias Zwicker

Please include all steps of your derivations in your answers to show your understanding of the problem. Try not to write more than the recommended amount of text. If your answer is a mix of correct and substantially wrong arguments we will consider deducting points for incorrect statements. There are thirteen questions for a total score of 100 points.

Your name:

1. Compute the dot-product of vectors $\mathbf{a} = (1, 2, -3)$ and $\mathbf{b} = (4, 1, 2)$. What is the angle between $\mathbf{a}$ and $\mathbf{b}$? (**6 points**)

   $1 \cdot 4 + 2 \cdot 1 - 3 \cdot 2 = 0$. The angle is 90 degrees.

2. Given two vectors $\mathbf{a}$ and $\mathbf{b}$ and their cross product $\mathbf{c} = \mathbf{a} \times \mathbf{b}$. What is the cross product $\mathbf{b} \times \mathbf{a}$ and $\mathbf{c} \times \mathbf{b}$? (**6 points**)

   $\mathbf{b} \times \mathbf{a} = -\mathbf{c}, \mathbf{c} \times \mathbf{b} = -\mathbf{a}$

3. Show that the vectors $(2, 6, 5)$, $(-3, 5, 1)$, and $(12, 8, 13)$ are linearly dependent. (**8 points**)

   $3 \cdot (2, 6, 5) - 2 \cdot (-3, 5, 1) = (12, 8, 13)$

4. Given a rotation and a translation matrix $\mathbf{R}$ and $\mathbf{T}$,

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & t_{14} \\ 0 & 1 & 0 & t_{24} \\ 0 & 0 & 1 & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

   write down the matrix $(\mathbf{RT})^{-1}$, i.e., write down its sixteen elements. (**8 points**)

$$(\mathbf{RT})^{-1} = \mathbf{T}^{-1}\mathbf{R}^{-1} = \begin{bmatrix} 1 & 0 & 0 & -t_{14} \\ 0 & 1 & 0 & -t_{24} \\ 0 & 0 & 1 & -t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{21} & r_{31} & 0 \\ r_{12} & r_{22} & r_{32} & 0 \\ r_{13} & r_{23} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{21} & r_{31} & -t_{14} \\ r_{12} & r_{22} & r_{32} & -t_{24} \\ r_{13} & r_{23} & r_{33} & -t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. What is the geometric interpretation of a homogeneous point with a homogeneous coordinate that is zero, i.e., a point of the form

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}. \quad (\textbf{4 points})$$

   A vector, or a direction.

6. Explain the term *metamer* (1-2 sentences). (**7 points**)

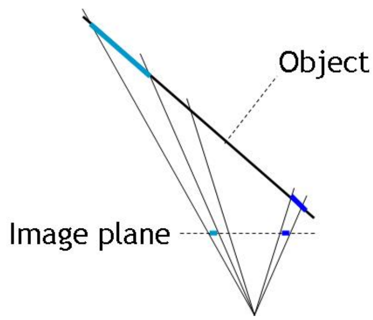
   Metamers are colors that have the same visual appearance, but different spectral composition.

7. Describe the main property of perceptually uniform color spaces (1-2 sentences). (**7 points**)

   In a perceptually uniform color space the Euclidean distance between colors corresponds to the perceived color difference. MacAdams ellipses become circles in a perceptually uniform color space.

8. Why does linear interpolation of texture coordinates in screen space lead to artifacts? Explain using a sketch and one or two explanatory sentences. (**8 points**)

   Linear interpolation in screen space does not correspond to linear interpolation in object space. With linear interpolation in screen space, textures seem to move on the surface as the viewpoint is changed.



9. Write down the four stages of the rendering pipeline and explain their main functionality in one sentence each. (**12 points**)

   Transformation: Transform vertex coordinates from object to camera space.
   Shading: Perform shading computation for each vertex in camera space.
   Projection: Project vertices from 3D camera space to 2D image space.
   Rasterization: Draw triangles and perform z-buffering to resolve visibility.

10. Given a directional light source with direction $\mathbf{d} = (1,1,0)/\sqrt{(2)}$ and strength $c_l = (0.5, 0.5, 0)$. Compute diffuse reflection on a surface with the following properties: The surface normal is $\mathbf{n} = (1,0,1)/\sqrt{(2)}$, and the material has a diffuse reflectance coefficient $k_d = (1,0,0)$. (**8 points**)

$$k_d c_l(\mathbf{n} \cdot \mathbf{d}) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \star \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \end{bmatrix} (1 \cdot 1 + 1 \cdot 0 + 0 \cdot 1)/2 = \begin{bmatrix} 0.25 \\ 0 \\ 0 \end{bmatrix}$$

   Here, $\star$ means element-wise multiplication. The question was ambiguous whether to use $\mathbf{d}$ or $-\mathbf{d}$ as the light direction. Therefore you also got full credit if your result was $(-0.25, 0, 0)$.

11. How many texel values have to be read to perform nearest neighbor texture filtering, bilinear texture filtering, and trilinear mipmapping? (**6 points**)

    1, 4, and 8 texels.

12. What is the advantage of trilinear mipmapping over bilinear filtering (2-3 sentences)? (**8 points**)

    Bilinear filtering leads to aliasing problems, because for each pixel the texture is interpolated at a single location. This means that fine details in the texture can appear as distracting low-frequency patterns. Trilinear mipmapping attempts to compute an average of the texture over the area of the pixel, hence avoiding these problems.

Explaining the difference between mipmapping with bilinear filtering vs. trilinear mipmapping was also considered a correct answer. Mipmapping with bilinear filtering results in visible seams between mipmap levels. In contrast, trilinear mipmapping leads to a smooth transition between levels.

13. Give pseudocode for hierarchical view frustum culling with bounding spheres. (**12 points**)

```
cull(node)
  inside = 0
  for each of the 6 bounding planes of the view frustum
    distance(node.bbox.center, bounding plane)
    if distance > node.bbox.radius
      // bounding sphere is outside view frustum
      // don't need to draw or further cull this node
      return
    else if distance < -node.bbox.radius
      inside++
    end
  end
  if inside == 6
    // node completely within view frustum, draw node
    // without further culling
    draw(node)
  else
    // node intersects view frustum, cull each child
    for each node.child
      cull(node.child)
    end
  end
end

distance(point, plane)
  // plane.d is distance of plane to origin
  return dot(point, plane.normal) - plane.d
end

// start culling at root node
cull(root)
```