Computer Graphics 2024
U3087 / M5222
National Taipei University

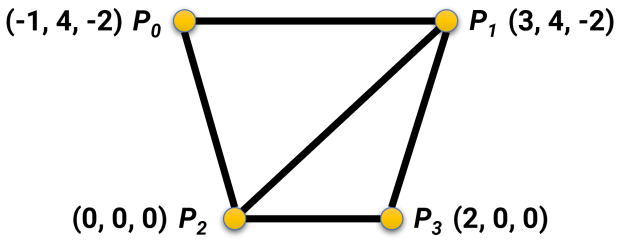
Midterm

1. Please answer the following questions about normals:

   (a) Describe what is the normal of a surface. (3%)
   The normal of a surface is a unit vector that is perpendicular to the plane in which the surface is situated.

   (b) Vertex normals are commonly used to achieve smooth shading rather than using surface normals. Describe how vertex normals are computed. (3%)
   The normal of a vertex is calculated by averaging the surface normals of the triangles that share that vertex.

2. Please answer the following questions for rendering the mesh shown below:



   (a) If you render the mesh **without** an index buffer, what is the content of the **vertex buffer**? (4%)
   [-1, 4, -2, 3, 4, -2, 0, 0, 0, 3, 4, -2, 0, 0, 0, 2, 0, 0]. Note there are other answers.

   (b) If you render the mesh **with** an index buffer, what is the content of the **vertex buffer** and **index buffer**? (6%)
   Vertex buffer: [-1, 4, -2, 3, 4, -2, 0, 0, 0, 2, 0, 0]. Index buffer: [0, 2, 1, 1, 2, 3]. Note there are other answers.

3. Imagine you are asked to create a forest scene containing many trees and grass. What is your strategy? Highlight as many advantages of your approach as possible. (8%)
   I will create some different trees and grass with the vertex coordinate defined in Object Space. Then, I will place them into the World Space of a scene using various world transformations, including translation, scaling, and rotation (4%). This approach allows me to build a scene with fewer unique models, making it more memory-efficient, as a single model can achieve diverse appearances through the use of a 4×4 matrix (4%).
   Note: you can answer using simple geometry with masked texture maps (alpha test).

4. When applying transformations, why do we use the **matrix representation** and **homogeneous coordinate**? (6%)
Matrix representation enables us to manage all types of transformations in a consistent way. Additionally, multiple transformations can be combined into a single matrix, improving efficiency. Matrix multiplication is also more optimized for GPUs. (3%) Homogeneous coordinates are used because certain transforms, like translation, require an extra dimension. (3%)

5. Please answer the following questions about camera transform:

   (a) Why is it necessary to transform a vertex into **Camera Space** during rendering? (4%)
   Transforming vertices into Camera Space simplifies the math of projection by eliminating the needs to account for the camera's location and orientation.

   (b) Describe how a **camera matrix** is constructed. (6%)
   The camera matrix is formed by combining translation and rotation matrices. First, We apply the inverse translation of the camera's position to shift all models, then rotate the World Space to align with the camera's local frame. As a result, the camera is positioned at the origin (0, 0, 0) and oriented to face the negative Z-axis.

6. When users change the width or height of an OpenGL window, which transform(s) need to be recalculated? Why? (6%)
We need to recalculate the projection matrix because it takes the aspect ratio of the screen window as parameters.

7. Please describe how modern graphics engines determine the nearest surfaces to a camera during rasterization. You should write down the technique's **name** and describe how it works. (8%)
Modern rasterization-based rendering engines use a Z-buffer for hidden surface removal (2%). The Z-buffer is an additional buffer, the same size as the color buffer, that stores the depth value of the nearest surfaces to the camera for each pixel (2%). During rendering, a fragment compares its depth to the Z-buffer. If the fragment's depth is closer than the existing value in the Z-buffer, the color and depth buffers are updated; otherwise, the fragment is discarded (4%).

8. Please answer the following questions about **rasterization**:

   (a) Describe the **scanline rasterization** process used to generate fragments from a shape and perform interpolation. (6%)
   Scanline rasterization identifies the edge pixels using DDA or Bresenham algorithms. Pixel values along the edges are interpolated first, followed by interpolation of the pixels between edge pixels for each row.

   (b) Describe how **barycentric coordinates** are used for rasterization. What is the major advantage? (6%)
   We can begin by determining the 2D bounding box of the triangle. For each pixel inside this boundary, we calculate its barycentric coordinate to check if it lies inside the triangle (all coefficients must be within the range [0, 1]). These coefficients can then be used for interpolation.

9. Please answer the following questions about the GPU graphics pipeline:

   (a) What is the purpose of **clipping**? How does OpenGL achieve this task? (4%)
   The goal of the clipping stage is to trim portions of triangles that extend beyond the screen window. If a triangle remains a triangle after clipping, no further

action is needed. However, if clipping results in a polygon, it must be subdivided into triangles.

(b) What is the purpose of **back-face culling**? How does OpenGL achieve this task? (4%)
The goal of the back-face culling stage is to skip rendering triangles that are not visible to the camera. These invisible triangles can be identified by calculating the dot product (angle) between their surface normals and the viewing direction. If the dot product is positive, the face is oriented away from the camera.

10. Please explain how lighting can be computed per fragment instead of per vertex in OpenGL 2.0. Additionally, provide an example where per-fragment lighting produces better results than per-vertex lighting. (8%)
In OpenGL 2.0, rasterization allows us to interpolate geometric properties, like 3D position and normals. This enables us to compute the interpolated geometric data for each fragment during lighting computation. For example, when rendering a low-polygon mesh with specular, per-vertex lighting may miss the highlights if none of the triangle's vertices produce them. Per-fragment lighting solves this issue.

11. How does an **environment map** store the radiance of a scene that varies with direction? (6%)
Each pixel's two-dimensional coordinate in an environment map corresponds to a specific spherical direction, and its pixel value indicates the incoming radiance from that direction.

12. Please answer the following questions about **Phong lighting model**:

(a) The **Lambertian term** is the key for the **diffuse component**. Describe how to compute it. (4%)
Calculate the cosine value of the angle between the surface normal and the direction of the incoming light.

(b) Explain why **specular term** is computed by: $L_s = k_s \cdot I \cdot \max(0, vE \cdot vR)^n$, where vE is the viewing direction and vR is the perfect reflected direction. Additionally, describe how this equation can simulate different shapes (sizes) of highlights. (8%)
The dot product of vE and vR quantifies the angle of deviation from the perfect reflected direction (vR). Specular lighting diminishes as this deviation increases. The exponent n governs the rate of this decline. A higher value of n results in a faster reduction, producing a smaller and sharper highlight.