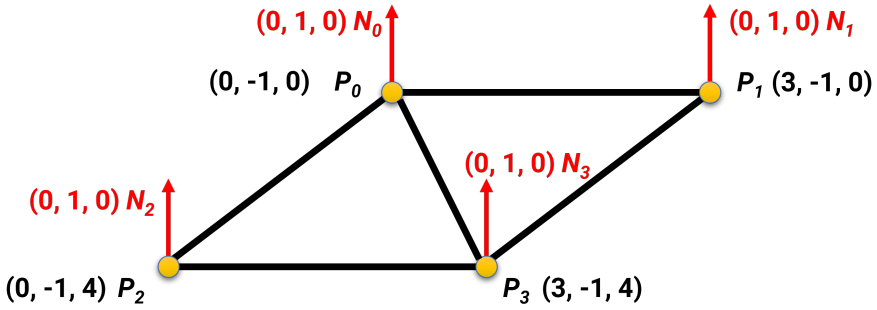
Computer Graphics 2023
U3087 / M5222
National Taipei University

Midterm

1. What are the three pillars (major topics) of computer graphics? (6%)
   modeling (2%), animation (2%), and rendering (2%)

2. If you would like to render the following quad with a **vertex buffer** and an **index buffer**:



   (a) Illustrate the content of the **vertex buffer**? (the vertex data should include vertex normals) (4%)
   [0 -1, 0, 0, 1, 0, 3, -1, 0, 0, 1, 0, 0, -1, 4, 0, 1, 0, 3, -1, 4, 0, 1, 0] (4%) there are other answers

   (b) Illustrate the content of the **index buffer**? (4%)
   [0, 2, 3, 0, 3, 1] (4%) there are other answers

3. Why do we define the attributes (position, normal) of vertices in **Object Space** instead of in **World Space**? State the **two** advantages of using a world transformation (8%)
   (1) reuse model (4%); (2) save memory (store a transformation matrix instead of the full geometry, 4%)

4. When representing the coordinate of a 3D point (x, y, z), what is the advantage of using the **Homogeneous Coordinate** $(x, y, z, 1)$? (4%)
   we can handle all transformation in a united manner (and different transformations can be concatenated, 4%)

5. Assume you want to scale an object which is centered at $(3, 6, 9)$, describe how to construct a scaling matrix that can scale the object by 2 times and keep its center unchanged (6%)
   we need to first translate the object to origin with a translation transformation T(-3, -6, -9). Then we scale the object by a scaling transformation S(2, 2, 2). Finally, we translate the object back to its original position by a translation transformation T(3, 6, 9). (6%)

6. Please answer the following questions about camera and projection:

(a) In **interactive graphics**, what is the most common **camera model** used for rendering (3%)
pinhole camera model (3%)

(b) Before projection, why do we need to transform a vertex to **Camera Space**? (6%)
to keep the math of projection simpler (without considering the location and orientation of the camera, 6%)

(c) Describe how a **camera matrix** is constructed (6%)
the camera matrix is a combination of translation and rotation matrix. We first move all models with the inverse translation of the camera's position, then rotate objects to match the camera's local frame (6%)

(d) What are the major differences between an **orthographic projection** and a **perspective projection**? (6%)
the perspective projection mimics human's perception system in which farther objects will look smaller (3%); the orthogonal projection preserves distance and angle (3%)

7. Please answer the following questions about **hidden surface removal**:

(a) Please describe **Painter's algorithm** and state its disadvantages. (6%)
Painter's algorithm draws objects from far to near. The colors of the latter objects overwrite the content in the frame buffer (3%). It has two drawbacks. First, objects need to be rendered from far to near (thus need sorting). Second, even though objects have been sorted according to their distance, sometimes we cannot find a global order due to cyclical overlapping (3%)

(b) Please describe how modern graphics engines determine the closest surfaces to a camera. You should write down the technique's name and describe how it works. (6%)
using z-buffer (3%) to keep the depth value of the closest surfaces from the camera. Only the fragments that have smaller depth values will update colors to the frame buffer and update depth to the z buffer (3%)

8. Please answer the following questions about shaders:

(a) What is the major changes from **OpenGL 1.1** (fixed function pipeline) to **OpenGL 2.0**? (6%)
to improve flexibility, some of the fixed stages have been replaced by programmable stages including vertex and fragment shaders (6%)

(b) What are the major goals of a **vertex shader** and a **fragment shader**? (6%)
vertex shader: transform a vertex to Clip Space (3%); fragment shader: determine the fragment color (3%)

9. **Scanline rasterization** and **barycentric coordinates** are two methods for a **rasterizer** to generate fragments and interpolate per-fragment data. Please describe how these two methods work (8%)

- **Scanline rasterization: first find the edge pixels using DDA or Bresenham algorithms. Pixel values on edges are first interpolated. After that, pixels in-between edge pixels are interpolated for each row (4%)**
- **Barycentric coordinate: first find the 2D bounding box of the triangle. For each pixel inside the bound, compute its barycentric coordinate to determine whether it is inside the triangle (all coefficients should be within $[0, 1]$). The coefficients can be used for interpolation directly (4%)**

10. Please briefly describe the purposes of the following stages in the graphics pipeline:

    (a) Clipping (3%)
    to fit a triangle within the viewport boundaries (3%)

    (b) Back-face culling (3%)
    to avoid rendering triangles that are facing away from the camera (3%)

    (c) Stencil test (3%)
    to control the screen regions for rendering (used to render mirror, 3%)

11. Please describe the reason that **Gouraud shading (per-vertex lighting)** might **blur out the highlights** on surface. How do you solve this problem? (6%)
    Gouraud shading only computes lighting on vertices. If the three vertices of a triangle do not produce highlights, the interpolated color inside a triangle will not produce hightlights (3%) One can solve this problem by using Phong shading (per-fragment lighting) by interpolating geometry proerties using rasterization and compute lighting in the Fragment shader (3%).