The University of New Mexico

Programming with OpenGL Part 4: Color and Attributes

Ed Angel

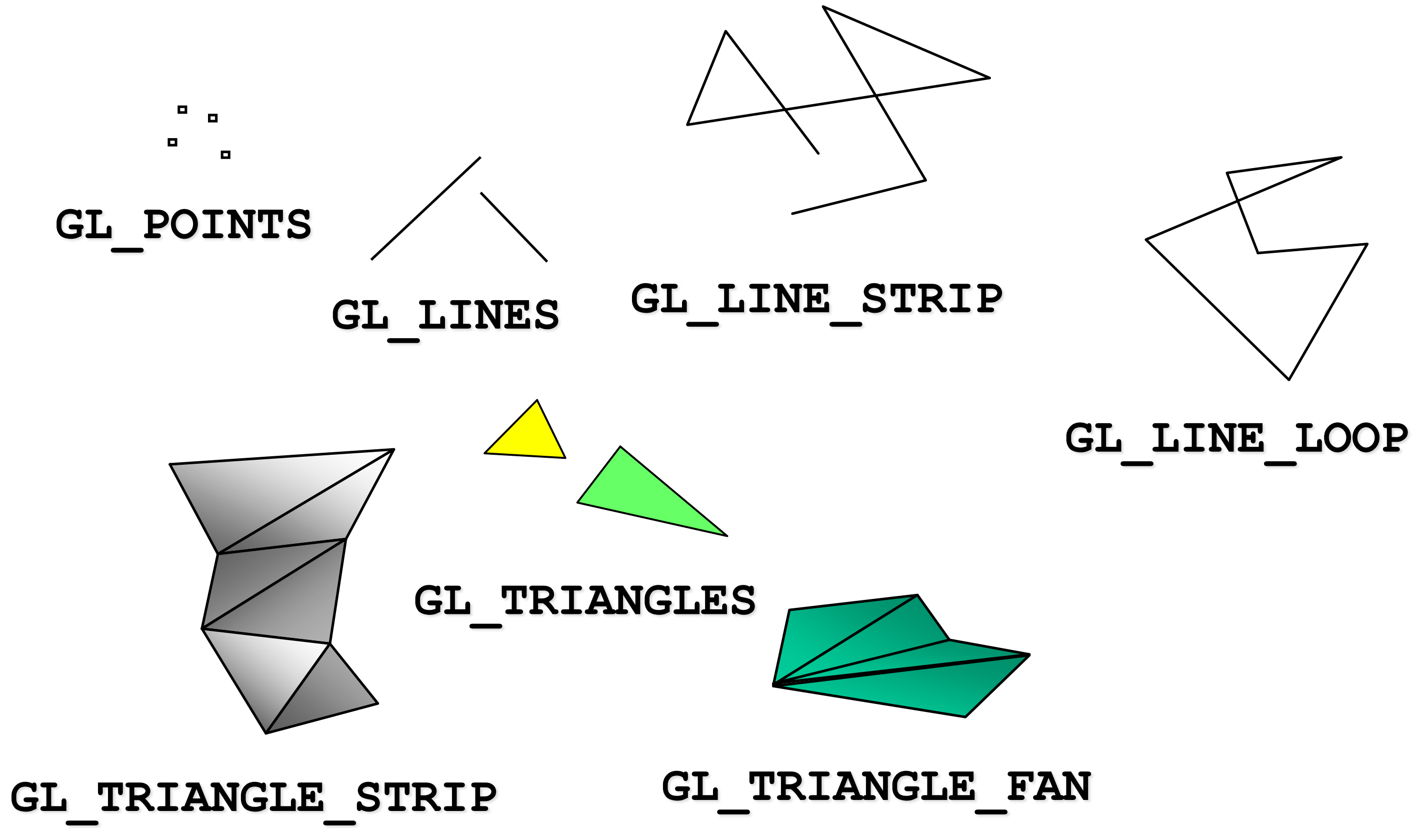
Professor of Emeritus of Computer Science

University of New Mexico

Objectives

- Expanding primitive set
- Adding color
- Vertex attributes
- Uniform variables

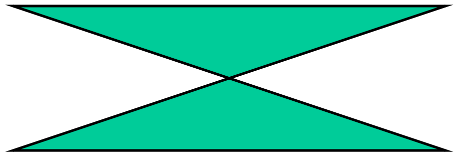
OpenGL Primitives

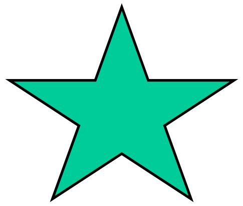


The University of New Mexico

Polygon Issues

- OpenGL will only display triangles
  - Simple: edges cannot cross
  - Convex: All points on line segment between two points in a polygon are also in the polygon
  - Flat: all vertices are in the same plane
- Application program must tessellate a polygon into triangles (triangulation)
- OpenGL 4.1 contains a tessellator


nonsimple polygon

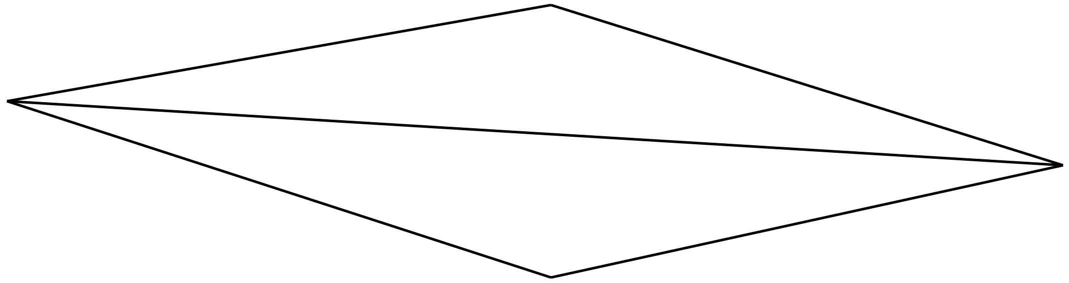

nonconvex polygon

Polygon Testing

- Conceptually simple to test for simplicity and convexity
- Time consuming
- Earlier versions assumed both and left testing to the application
- Present version only renders triangles
- Need algorithm to triangulate an arbitrary polygon

Good and Bad Triangles

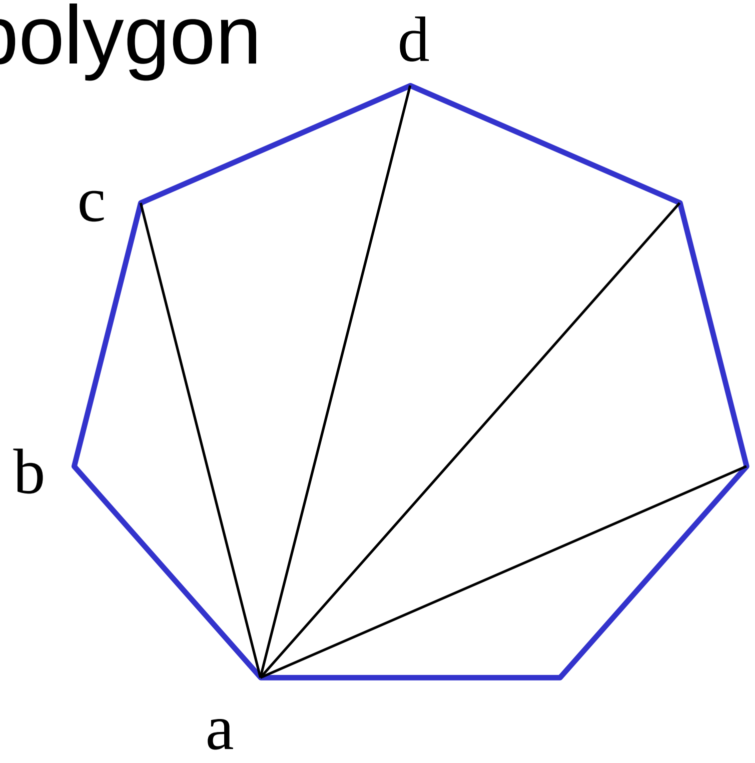
- Long thin triangles render badly



- Equilateral triangles render well
- Maximize minimum angle
- Delaunay triangulation for unstructured points

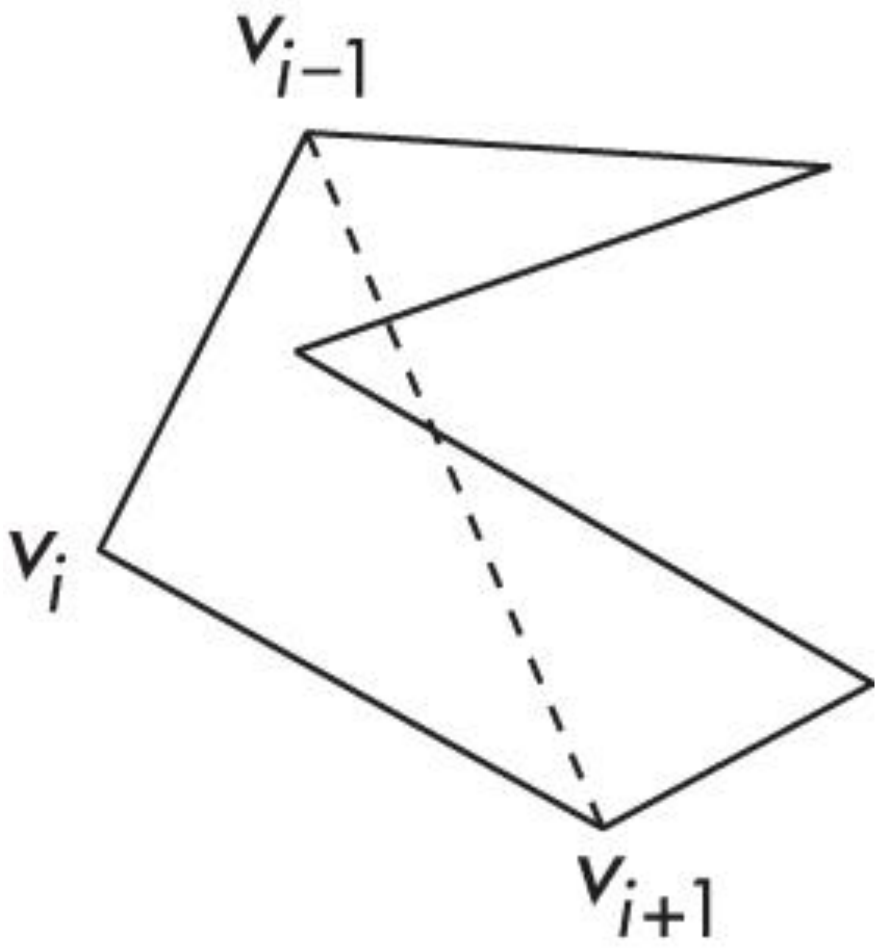
Triangularization

- Convex polygon



- Start with abc, remove b, then acd, ….

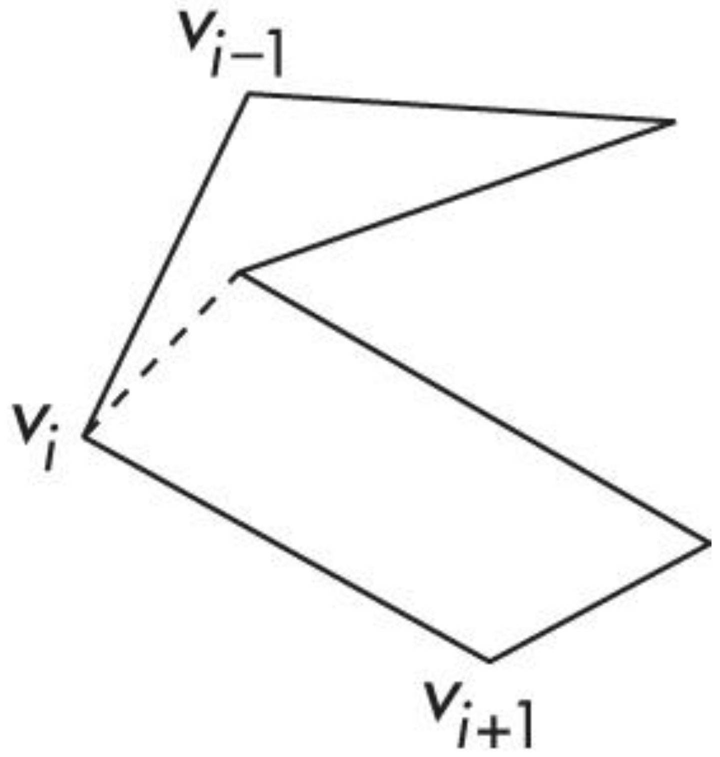
Non-convex (concave)



The University of New Mexico

Recursive Division

- Find leftmost vertex and split

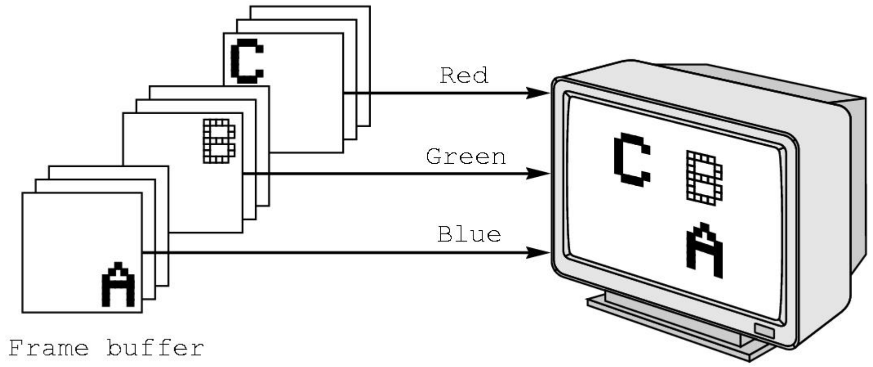


Attributes

- Attributes determine the appearance of objects
    - Color (points, lines, polygons)
    - Size and width (points, lines)
    - Stipple pattern (lines, polygons)
    - Polygon mode
        - Display as filled: solid color or stipple pattern
        - Display edges
        - Display vertices
- Only a few (glPointSize) are supported by OpenGL functions

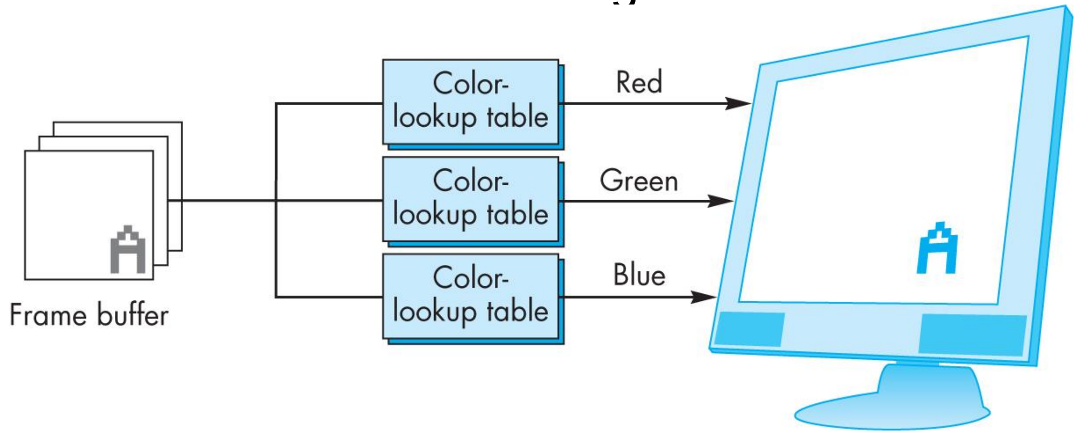
RGB color

- Each color component is stored separately in the frame buffer
- Usually 8 bits per component in buffer
- Color values can range from 0.0 (none) to 1.0 (all) using floats or over the range from 0 to 255 using unsigned bytels



Indexed Color

- Colors are indices into tables of RGB values
- Requires less memory
  - indices usually 8 bits
  - not as important now
    - Memory inexpensive
    - Need more colors for shading

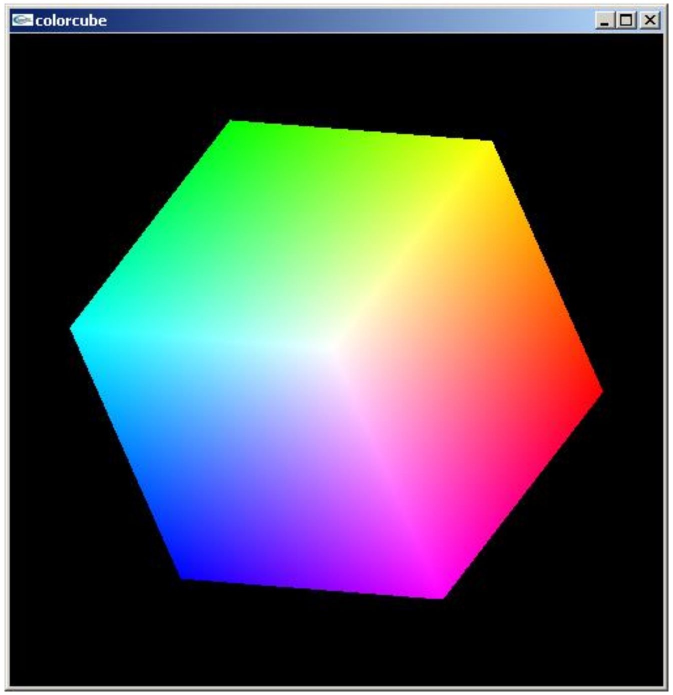


The University of New Mexico

Smooth Color

- Default is *smooth* shading
  - OpenGL interpolates vertex colors across visible polygons
- Alternative is *flat shading*
  - Color of first vertex

    determines fill color
  - Handle in shader



Setting Colors

- Colors are ultimately set in the fragment shader but can be determined in either shader or in the application
- Application color: pass to vertex shader as a uniform variable (next lecture) or as a vertex attribute
- Vertex shader color: pass to fragment shader as varying variable (next lecture)
- Fragment color: can alter via shader code