# Flat vs. smooth shading

The decision of whether or not triangles should share vertices

# Normals

- Mathematics: A vector that is perpendicular to the surface at a given point -- particularly, pointing outward
- Graphics: We're only interested in vertex normals -- discrete "approximations" of the normal sampled at points on the imaginary surface
  - True realism would require calculation of normal/derivative at every point along a continuous shape - not feasible

# What are they used for?

- Lighting! The direction of the normal determines how the light will bounce off each surface when modeling the rays

# Our shapes so far have easy normal vectors.

- “Z axis” vector is perpendicular to Triangle and Square
- For a cube, normals would also just be axis-aligned
- For a sphere, we know analytically that the vector away from the center (perpendicular to the formula’s surface) will be the normal.  Equals position coord.

# What do you do when the normals aren’t known?

- Hint: Think per-triangle.

# What do you do when the normals aren’t known?

- Using the indices, collect the positions of the three points of the triangle.
- Create two vectors out of the triangle.
- Cross product
- What to do if the normal points inside the shape instead of out?
- How to detect that?  Assume shape is convex.

# Flat vs Smooth shading

- That algorithm produces a normal per whole triangle
  - Distribute it to all three vertices identically
  - Result is flat lighting across the triangle
- When vertices are shared, we logistically cannot do that
  - A vertex can’t have two normals at once
  - Other triangles will fight over assigning normals to a vertex

# Flat vs Smooth Shading

- As opposed to “Smooth shading”
  - Deriving normals from some source besides the triangle, like the analytic formula of the shape (like Sphere)
  - Lighting formulas can react to the normal being different across a surface
  - Appearance: “morph normals” demo in http://threejs.org/examples/

# Flat vs Smooth Shading

Review:

- Flat shading:
  - All three normals are the same.
- Smooth Shading:
  - All three normals are not the same.

# Flat vs Smooth Shading

- Smooth Shading:
  - All three normals are not the same.
    - Allows interpolation of infinity different normals as you move from one extreme point of the triangle to the other
    - This creates a different lighting formula estimate and different color at each pixel
    - Interpolation done in fragment shader (per pixel in triangle), and fed into the lighting formula

# Flat vs Smooth Shading

- Flat shading
  - Nothing to interpolate. All three extreme points hold the same value. That value gets spread across whole triangle, giving the same color result everywhere.
  - The GPU still most likely does all the same interpolation math
    - (Flat shading doesn't actually speed it up, but just produces repetitive color answers across the triangle).

# But What If…?

- What if we already did share vertices when we built the shape, perhaps for simplicity?
  - Loop through all the index triples and create a new, unique, vertex for every index -- based on what used to be at that index.
  - Overwrite our old lists with this.
  - In your code, class `Shape` already has a short utility function that does this, called `make_flat_shaded_version()`.