CSE 167:
Introduction to Computer Graphics
Lecture #19: Bump Mapping

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2020

Announcements

- Sunday, December 6th at 11:59pm:
  - Homework Project 3 late deadline
- Next Wednesday, December 9th at 1pm:
  - Discussion Project 4 and Final Exam
- Sunday, December 13th at 11:59pm:
  - Homework Project 4 due
- Thursday, December 17th 2:30pm until Dec 18th 2:30pm
  - Final Exam
  - Timed 3-hour Canvas quiz, to be taken within 24h
- Sunday, December 20thth at 11:59pm:
  - Homework Project 4 late deadline

Bump Mapping with Normal Maps

Consider Modeling an Orange

- Start with an orange-colored sphere
  - Too simple
- Replace sphere with a more complex shape
  - Does not capture surface characteristics (small dimples)
  - Takes too many polygons to model all the dimples

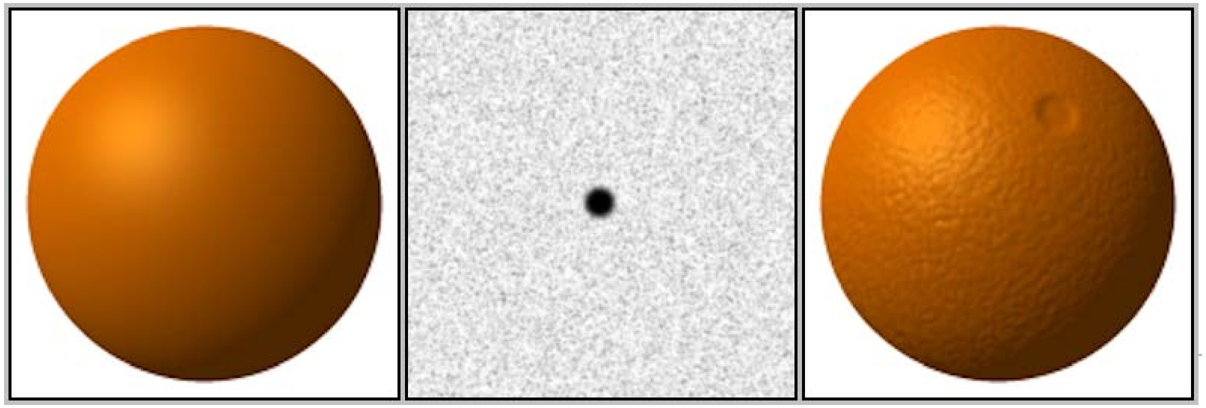
Texture Mapped Orange

- Take a picture of a real orange
- “Paste” pixels of the image onto simple geometric model
  - This process is known as texture mapping
- Still might be problematic…
  - Looking at the orange in a rendered scene: shading of dimples won’t match lighting environment

Bump Mapped Orange

Use an image that specifies the normal to use to render the surface

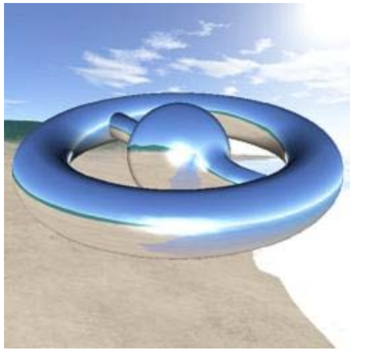
- This way, can render a “bumpy” surface during shading without subdividing the surface into lots of tiny triangles



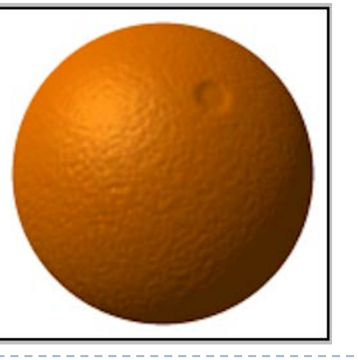
Three Types of Mapping

- Texture Mapping
  - Paste images onto polygons

- Environment Mapping
  - Uses a picture of the environment for texture maps
  - Allows simulation of mirror-like surfaces

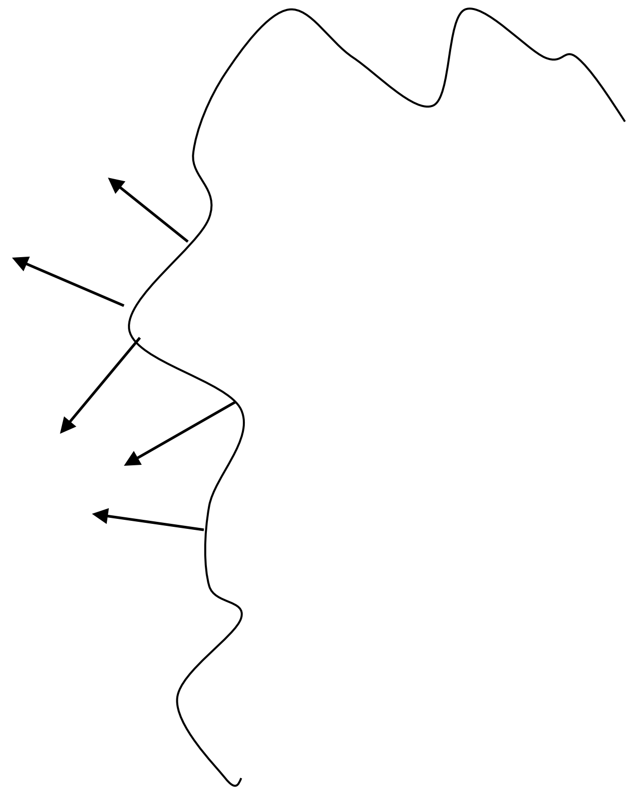


- Bump mapping
  - Alters normal vectors during the rendering process
  - Generates bumpy looking surfaces



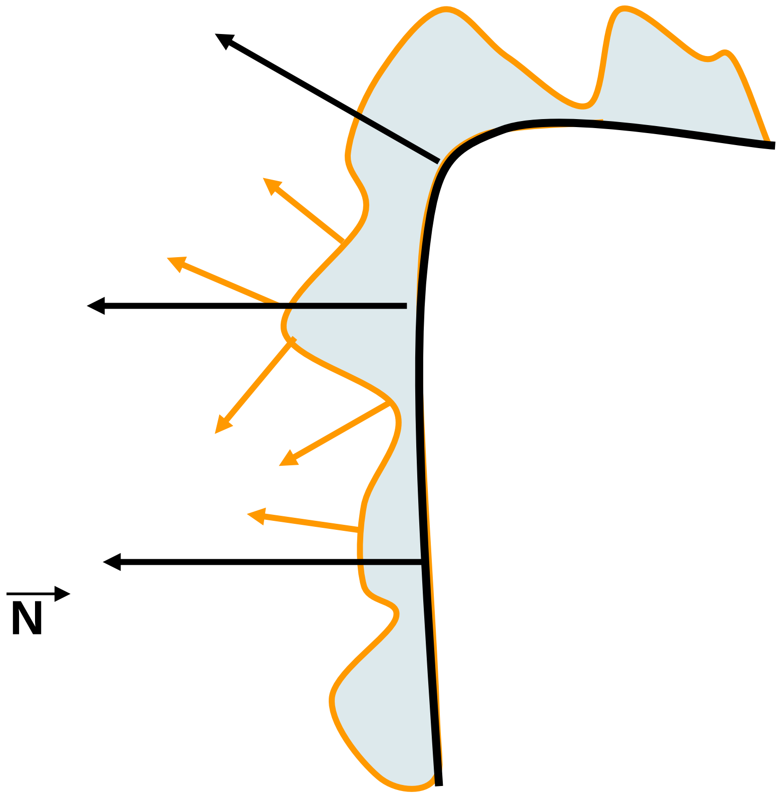
Surface Shading

- Consider the lighting for a modeled surface.

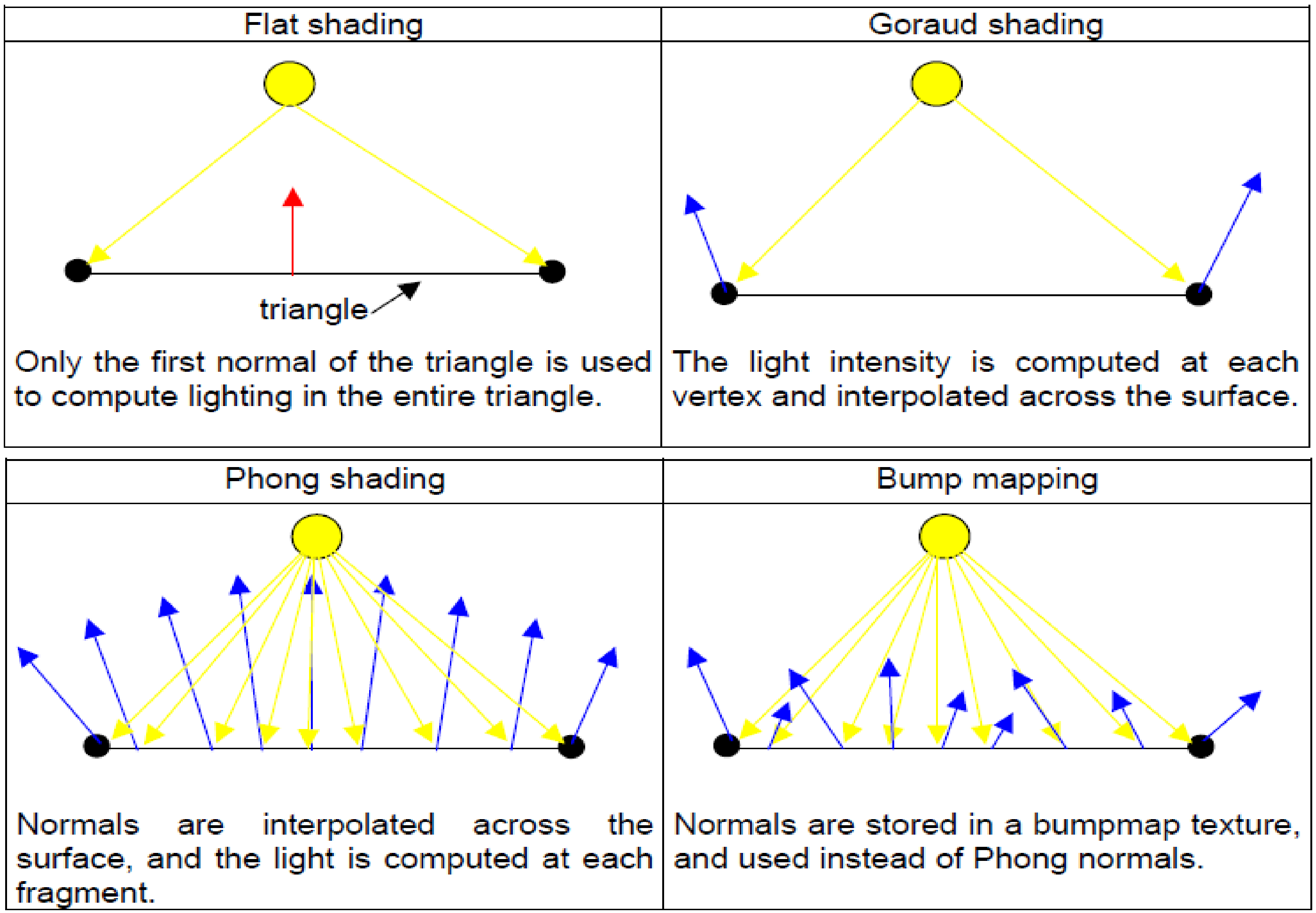


Surface Shading

- We can model this as deviations from some base surface.
- The question is then how these deviations change the lighting.



Bump Mapping



Bump Mapping with Normal Maps

Just texture mapped

Texture and normal maps

Notice: The geometry is unchanged. There's the same number of vertices and triangles. This effect is entirely from the normal map.

Normal Maps

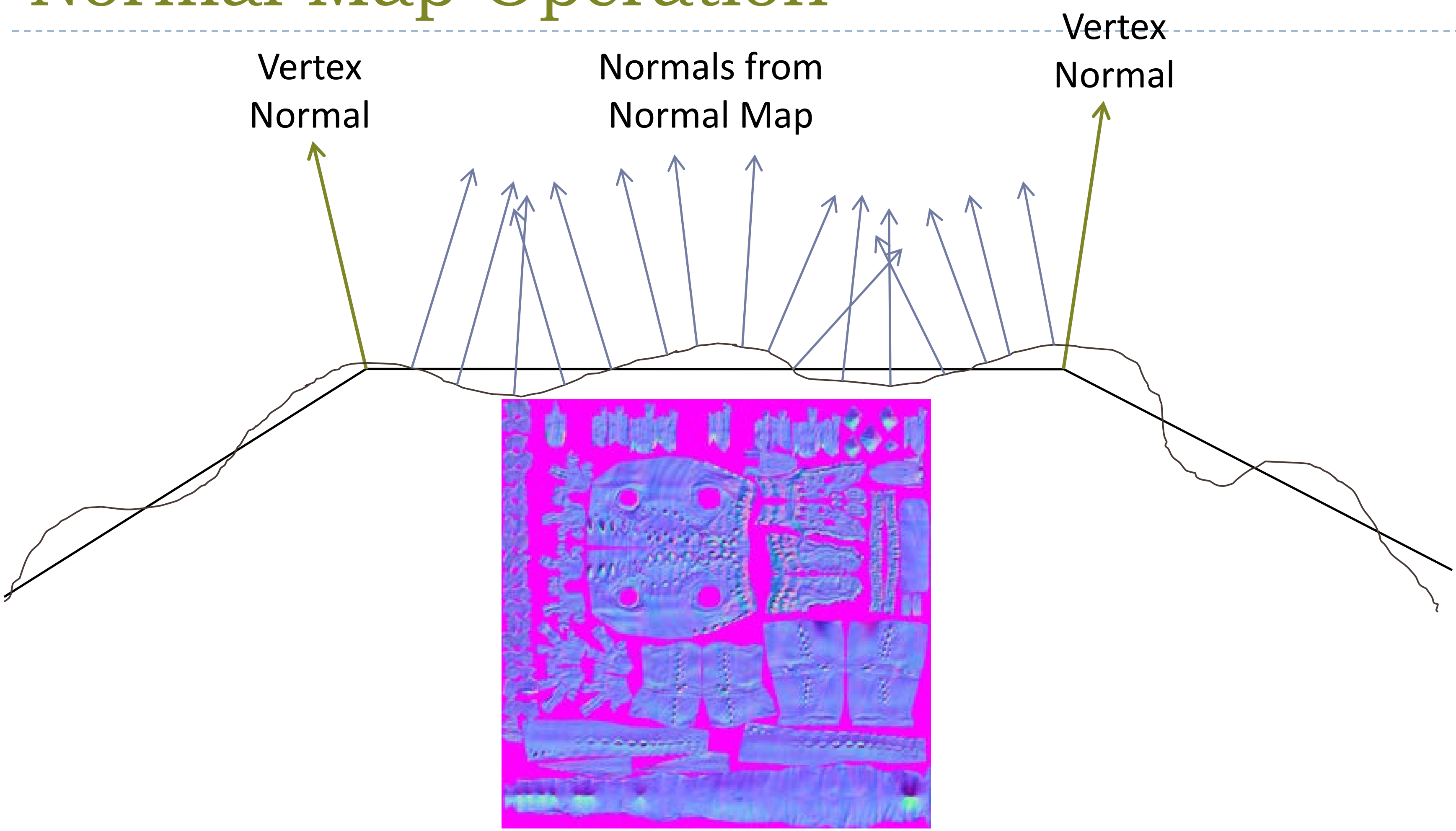
Diffuse Color Texture Map

Normal Map

Each pixel represents a normal vector relative to the surface at that point. -1 to 1 range is mapped to 0 to 1 for the texture so normals become colors.

→ Inverse of Normal Coloring

Normal Map Operation

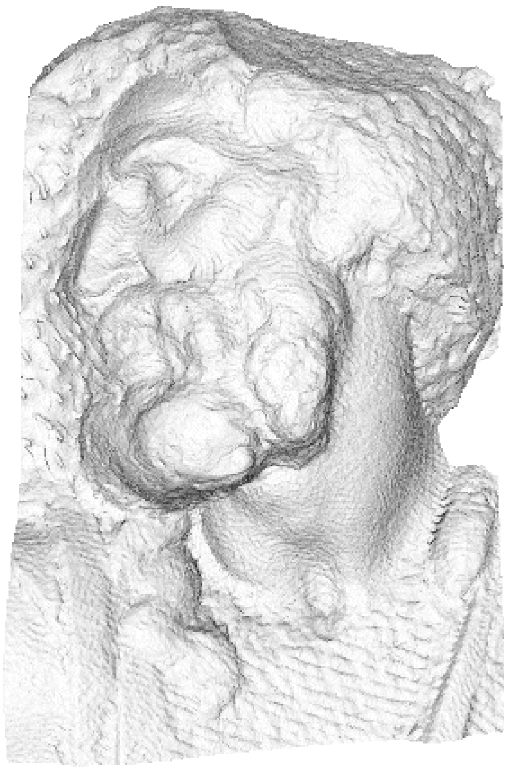


For each pixel, determine the normal from a texture image. Use that to compute the color.

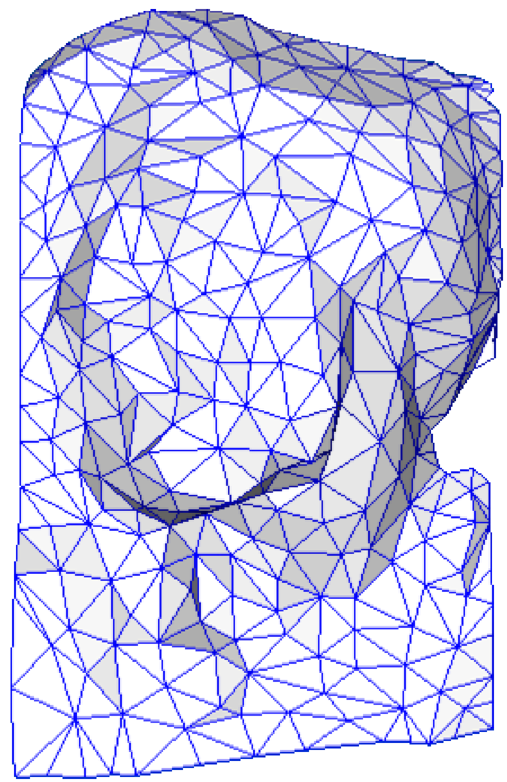
Normal Map

- Normal vector encoded as rgb
  - $[-1,1]^3 \rightarrow [0,1]^3$: rgb = n*0.5 + 0.5
- RGB decoding in fragment shaders
  - vec3 n = texture2D(NormalMap, texcoord.st).xyz * 2.0 – 1.0
- Normal maps typically map direction out of image to +z
  - Hence RGB color for the straight up normal is (0.5, 0.5, 1.0).
  - This is why normal maps are mostly a light blue color
- Normals are then used for shading computation
  - Diffuse: n•l
  - Specular: $(n \bullet h)^{\text{shininess}}$

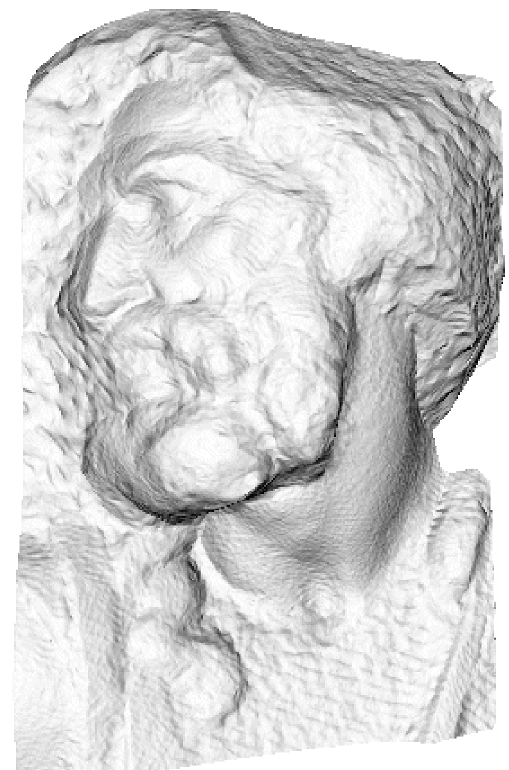
Normal Mapping Example



original mesh
4M triangles



simplified mesh
500 triangles



simplified mesh
and normal mapping
500 triangles

Normal Mapping

Bump Mapping:
Perturbing mesh normals to create the appearance of geometric detail

Normal Mapping:
A way of implementing bump mapping

What's Missing?

- There are no bumps on the silhouette of a bump or normal-mapped object

→ Displacement Mapping can model that
(not covered in CSE 167)