Project 2 Math

Lighting a surface

Important Assumption

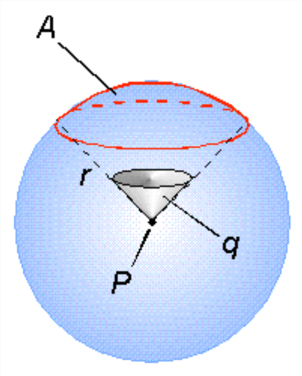
- We'll assume that our surfaces are not dull (matte), not shiny
  - Why? Because reflections off a specular surface are rather complex to model
- Reflections of matte surface are fairly straight-forward
  - Matte surfaces are known as Lambertian surfaces

Side Point: Observed Brightness

- Brightness at Observer/Camera is dependent on the angle between the incoming light's rays and the surface, but, conveniently, is **independent of viewing angle** for Lambertian surfaces
  - Light from individual point **falls off** with cos(viewing_angle)
  - Amount of surface seen **increases** with cos(viewing_angle)
  - Net effect is that a surface appears same brightness no matter what angle you view it from.

Result

- Brightness only a function of illumination level, not of the observer's position
- Total illumination (watts/square meter) on surface is proportional to:
  - cosine of the angle between the surface and the ray from the light source
  - Illumination level (watts/steradian)
    - "Steradian" (sr, aka 'solid angle' or 'square radian') is the 3-D analog to the 2-D angular measure of radian.
    - There are 2 PI radians in a circle
    - there are 4 PI steradians in a sphere



Steradian

Intuitively, this makes sense. For a given illumination level L (in watts/sr), if the surface is at an angle to the incoming rays, then the energy is spread over a wider area: 1/cos(angle) * (perpendicular area): less light falling on each patch of surface, so less light to reflect.

http://whatis.techtarget.com/definition/steradian

Observed brightness

L = I*r * cos(theta)

Where

- I is the illumination source (watts/sr)
- L is observed brightness
- r is reflectance of the object
- theta is the angle between the surface normal and the incoming ray

Reflectance

- We'll assume the reflectance is the same everywhere
- Not true, but good enough for our purposes
  - Granite is granite, so if we're looking at a big, granite mountain with no vegetation, the reflectance should be the same everywhere
    - Mt. Washington and Mt. Dartmouth are granite
  - Also a reasonable assumption for craters on Mars.
    - It's all sort of a dusty red…

Bottom Line

- To illuminate a digital elevation map, all we need to do is:
  1. choose a reasonable angle for the incoming sunlight, and
  2. calculate the angle between the surface at each point and the incoming ray.
- For us, this means all we need is a little geometry wrapped in a for loop
  - (and preferably packaged in a few functions)

Angle between 2 vectors

Dot product of 2 vectors (A_x, A_y, A_z) and (B_x, B_y, B_z) is:

$$A \cdot B = ||A|| * ||B|| * \cos(\text{theta})$$

Where the magnitude of a vector A is given by:

$$|| A || = \text{sqrt}(A_x^2 + A_y^2 + A_z^2)$$

where A is the vector (A_x, A_y, A_z)

Or:

$$\cos(\text{theta}) = A \cdot B / (||A||*||B||) = A_{norm} \cdot B_{norm}$$

Where "·" is the dot product, and :

$A_{norm} = A / || A ||$ and $B_{norm} = B / || B ||$

http://tutorial.math.lamar.edu/Classes/CalcII/DotProduct.aspx

Dot Product Example

Let vectors A & B be:

$A = (5, 6, 7)$

$B = (2, 4, 6)$

$A_{norm} = (5, 6, 7)/\mathrm{sqrt}(5*5 + 6*6 + 7*7)$

$= (5, 6, 7)/10.4881 = (.4767, .5721, .6674)$

$B_{norm} = (2, 4, 6)/7.4833 = (.2673, .5345, .8018)$

Cosine of the angle between A & B:

$A \cdot B = .4767*.2673 + .5721*.5345 + .6674*.8018$

$A \cdot B = .9683$

So, if light (at an angle of (2,4,6)) is falling on a surface with normal of (5,6,7), then it would be just a tiny bit less bright (*.9687) as if the surface were perpendicular to the incoming light.

Now we Just need the Surface Normal

- Given the sun's angle, we can find the angle between it and the surface's "normal" using the dot product.
- Surface Normal is a vector perpendicular to the surface
  - You can get a vector normal to the surface using any 2 vectors that lie on the surface itself
- Cross product of 2 vectors is a 3rd vector which is perpendicular to the first 2 vectors
  - and therefore perpendicular to the surface the 2 vectors line in – just what we need.

Cross Product

- Given 2 vectors:

$$A = (A_x, A_y, A_z)$$
$$B = (B_x, B_y, B_z)$$

- Then the cross product A x B is:

$$A \times B = (A_yB_z - B_yA_z, A_zB_x - B_zA_x, A_xB_y - B_xA_y)$$

(note: original slide, shown in class, had sign wrong for middle element.)

http://tutorial.math.lamar.edu/Classes/CalcII/CrossProduct.aspx

Example

Let:

$A = (1, 2, 3)$

$B = (4, 5, 6)$

Then:

$A \times B \quad = (A_yB_z - B_yA_z, A_zB_x - B_zA_x, A_xB_y - B_xA_y)$

$A \times B \quad = (2*6 - 5*3, 3*4 - 6*1, 1*5 - 4*2)$

$= (12 - 15, 12 - 6, 5 - 8)$

$= (-3, 6, -3)$

Note: Direction Important

- Direction is important: surface normal could be up or down, depending on relative locations of the 2 vectors.
- For standard (x,y,z) axes, direction of A x B is given by "right-hand rule"
- If A & B have same starting point, and B is counter-clockwise from A, then direction of A x B is up
- R H Rule valid for standard (x,y,z) axis, but digital images have y direction reversed, so we need "Left Hand Rule":
  - So we choose B clockwise from A, not CCW.

Why is direction important?

- If the cosine of the angle between the two vectors is positive, then the light is falling on the front (up-side) of the surface
  - it is tilted towards the sun
  - i.e., a point on the surface is in the sunlight
- If the cosine of the angle is negative, then the light would be falling on the “underneath” of the surface
  - it is tilted away from the sun
  - i.e., it is in shadow.

So we need to know 2 things

- Direction of sunlight
  - Normally given by azimuth & elevation (spherical coordinates)
    - Need (az,el) to (x,y,z) equation
- Surface normal at each point on surface
  - We have elevation at each point
  - Just need to find 2 vectors on surface at each point
    - Cross-product of these 2 vectors gives us the surface normal
- Given above, we just normalize the sun vector and the normal vector and take their dot-product, and this gives us a scaling factor for the light reflected from that point.
    - Reminder: normalizing is simply dividing a vector by its length, so we have a unit vector (vector of length 1)

(Az,El) to (x,y,z)

(Az,El) to (x,y,z):

- The mapping from spherical coordinates to three-dimensional Cartesian coordinates is
- x = r * cos(elevation) * cos(azimuth)
- y = r * cos(elevation) * sin(azimuth)
- z = r * sin(elevation)

Example

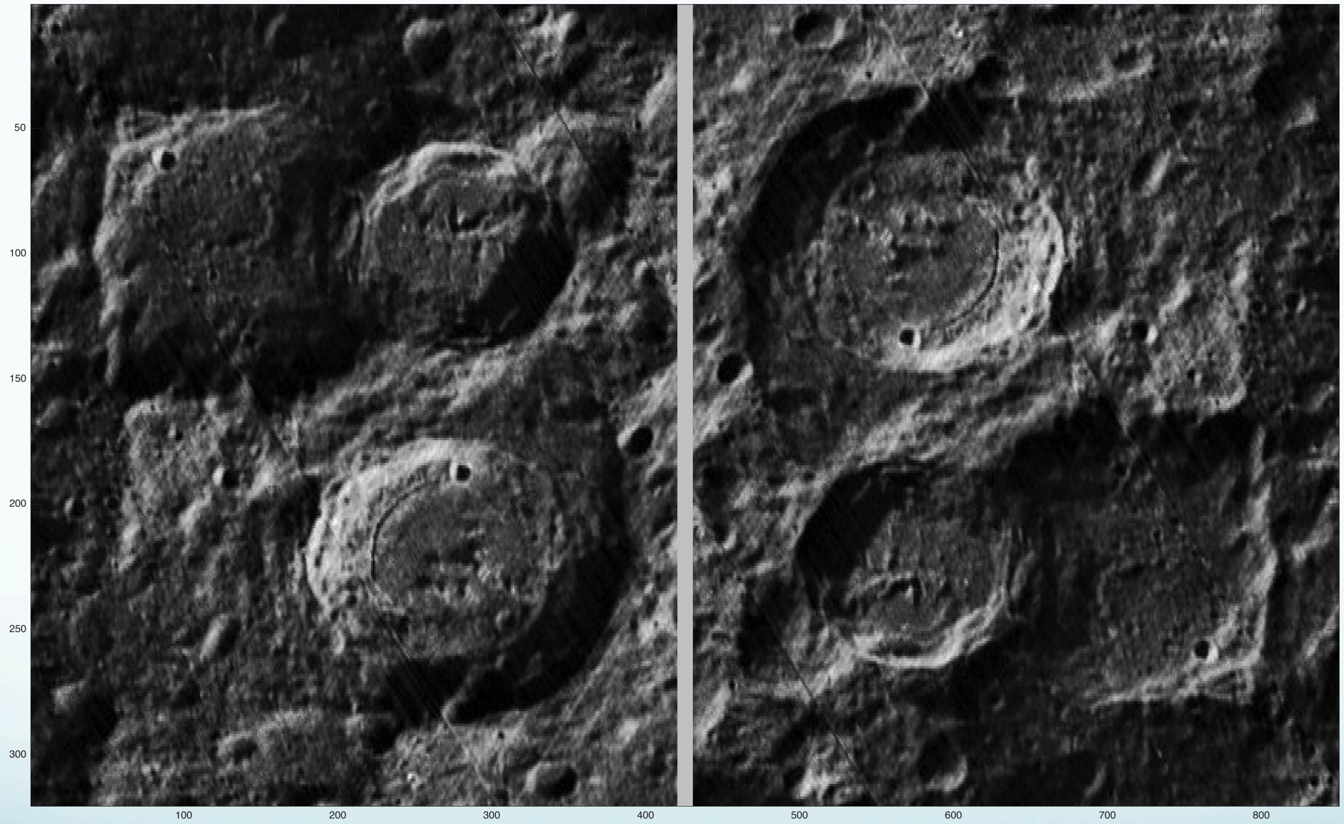
- Assume the sun is at an azimuth of 30 degrees and elevation of 60 degrees
  - az, 30 degrees = 0.5236 radians = (30 * PI/180)
  - el, 60 degrees = 1.0472 radians = (60 * PI/180)
- Then, a ray from the sun is coming in at:
  - S = (cos(1.0472)*cos(.5236), cos(1.0472)*sin(.5236), sin(1.0472))
  - S = (.4330, .25, .8660)
    - Note: S is normalized (i.e., its length is one):
    - $\mathrm{sqrt}(S \cdot S) = \mathrm{sqrt}(.4330^2+.25^2+.8660^2) = 1.0$

Note: this is a vector from the **ground to the sun**, and is the vector we want to dot-product with the surface normal to get the brightness

Asides about positioning the sun

- In latitudes above the tropics, the sun is generally south of due east or west (except for early morning and late afternoon in the summer)
- Because our image coordinate system is left-handed, azimuth values between 0 & 180 degrees put the sun in the south (bottom of our images), where it generally belongs.
- BUT: because our brains expect the sun to be UP, you may have trouble making sense of a shaded image of mountains with the sun coming up from the bottom (in the south when viewed directly overhead as we are doing.)
  - Valleys can look like ridges, and ridges like valleys, and a crater can look like a bump instead of a hole in the ground.
  - This is why a face looks 'scary' when one shines a flashlight up from below the chin
- SO: either:
  - turn your computer screen upside down (not recommended), or
  - use an (unrealistic) sun azimuth position in the north(az between 180 & 360, or between 0 & -180 degrees) to put the sunlight coming from "up" in the image.

Lunar Crater (rotated 180°)



Let's Find 2 vectors at a point of a Digital Elevation Map

- For point at (x,y) in our DEMs, we have 2 obvious vectors:
  - Choose vector A as the vector from:

    (x,y) to (x+1,y)
  - Choose vector B as:

    (x,y) to (x, y+1)

This is estimating the slope at point (x,y) using the triangle defined by the points (x,y), (x+1,y) and (x, y+1)

(close enough for our purposes!)

- Vector A is (Ax, Ay, Az) where

  Ax = 1
  Ay = 0
  Az = change in elevation from (x,y) to (x+1,y)

- Vector B is (Bx, By, Bz) where:

  Bx = 0

  By = 1

  Bz = change in elevation from (x,y) to (x,y+1)

Don't Forget Scaling

- For Mt Washington, Mt Dartmouth and Milankovic craters, header information has elevation.
- Mt Washington & Mt Dartmouth:
  - elevation is in feet
  - X & Y are 30 feet per pixel
- Milankovic:
  - Elevation is in meters
  - X is 300 meters/pixel, Y is 600 meters/pixel (approx.)
    - Note: elevations are negative; area of crater is below mean elevation of mars' surface.
- Mt. Eden (volcano.egm)
  - Elevation is in **tenths of meters**. (Peak is 195 m above sea lvl)
  - X & Y are 10 meters/pixel

So, to illuminate Mt Washington

- Choose an Az, El for the sun
  - Convert to (x,y,z)
- For each pixel in DEM
  - Calculate surface normal using cross product
  - Take dot-product of the normal with sun angle
- Result is brightness of pixel (on scale of 0 to 1)
  - Multiply by 255 to get brightness for output PGM

Example Part 1: Find Vectors A & B

Mt Dartmouth elevations, col 18-19, rows 0-1*:
1117 1118
1122 1120

Vector A (unscaled): (1, 0, 1118-1117)
Vector B (unscaled): (0, 1, 1122-1117)

Using 30 ft per pixel:

$A = (30, 0, 1) \quad A_{norm} = (30,0,1)/30.0167 = (.9994, 0, .0333)$

$B = (0, 30, 5) \quad B_{norm} = (0,30,5)/30.4138 = (0, .9864, .1644)$

$N = A_{norm} \times B_{norm} = (0*.1644 - .9864*.0333, .0333*0 - .1644*.9994, .9994*.9864 - 0)$

$N = (-.0329, -.1643, .9859)$

So, the unit vector perpendicular to the ground for the pixel at (col 18, row 0) of the Mt.Dartmouth DEM is the vector (-.0329, -.1643, .9859)

** The surface is flat pixel at (0,0), so I've chosen a sample pixel with a little slope to it.*

Example Part 2: find its brightness

Surface Normal, N is:

$N = (-.0329, -.1643, .9859)$

Using Sun at (az,el) of $(30^0, 60^0)$:

$S = (.4330, .25, .8660)$

Take the cross-product:

$S \cdot N = (-.0329)*.4330 + (-.1643)*.25 + .9859*.8660$

Brightness = .7985

So, the ground in the upper left corner of Mt.Dartmouth is reflecting .7985 of the incoming sunlight. If we rescale it so ground perpendicular to the sun (brightness = 1) is at 255, then the upper left pixel will be at

.7984*255 = 203.6014

So our grayscale PGM of Mt Dartmouth would have 203 in col 18, row 0:

P2 346 473 255

...203...

A little more realistic

- In the real world, some of the sun's energy is scattered by the particles in the sky, which also illuminates the surface
  - (a good thing, or the world would be dark on a cloudy day!)
- Since the sky is an extended source, the ground reflects all of the sky brightness, regardless of the surface angle
- A reasonable approximation is to use .9 of the sunlight as direct light, and .1 as sky-brightness
- Including the sky brightness, our pixel becomes:

brightness = (1-sky)*S·N + sky = .9*.7984 + .1 = .8186

pix = 255 * brightness = 255 * .8186 = 208

You can Add Color

- Use color-table “colormap_mountains.txt”
- Assign color based on elevation
  - Lowest elevation = 0
  - Highest elevation = 255
  - Scale elevation of pixel to value between 0 & 255
  - Get (r,g,b) color for that elevation
    - Colormaps in DEMS zip file are tables w/ 256 (r,g,b) values, one per row
  - Multiply r, g & b by pixel brightness.
- Write out as PPM (P3 instead of P2) with 3 values per pixel

Reminder

- From the intro to digital images in lab 3, the color version of a grayscale PGM (extension PPM), is almost identical to a PGM, except it:
  - Has “P3” instead of “P2”
  - Has 3 values (red, green & blue) for each pixel

See pgm_basics.pdf, linked to in part 3 of lab 3 assignment

Example

- Let's color the upper left pixel of Mt Dartmouth, using colormap_mountains.txt

1. Get the relative elevation of the pixel:

For a pixel at height h, we can calculate its relative height (on a scale of 0 to 1) of

$$h_{rel} = (h-h_{min})/(h_{max}-h_{min})$$

And if there are 256 colors in our color table, we want the row at h_{rel} * 255.

Example, cont.

- From header: elevation range: h_{min} = 1085, h_{max} = 3725
- Pixel at col 0, row 18 is at 1117, so h_{rel} is
  - h_{rel} = (1117-1085)/(3725-1085) = .0121
  - Row of color table for height of 1117 = .0121*255 = row 3
- So we want to use the color in row 3 (4th row):
  - 1 105 51 (red=1, green=105, blue=51)
    - This is a not-too-bright green with almost no red and tinge of blue.

- Since the brightness of the pixel is .7984 (.8186 w/ sky), we color it in with

  .7984*(1, 105, 51) = (.80, 83.83, 40.72) -> (0, 83, 40) or

  .8186*(1, 105, 51) = (.82, 85.95, 41.75) -> (0, 85, 41) with sky

Result

If we want a grayscale PGM of Mt.Dartmouth, the file would begin with (hdr & 19th pixel):

```
P2 346 473 255
… 203 …
```

(208 with sky)

If we want a color PPM of Mt.Dartmouth (using colormap_mountains.txt), the file would begin with (hdr & 19th pixel):

```
P3 346 473 255
… 0 83 40 …
```

(0 85 41 with sky)

You can add Contours

- To draw contours on your virtual image:
  1. Calculate contours from the DEM
  2. Illuminate your DEM
  3. Add the two images
     - Combined = .8 * illuminated + .2 * contours

Note: you'll need to be intelligent about coloring the image if you wish to have a colored, contoured image.

Sample Images

- Sample images are in the zip file downloadable from the Project 2 assignment on Blackboard