CS 362: Computer Graphics

Texture Mapping & Intensity Representation

Dr. Samit Bhattacharya
Dept. of Comp. Sc. & Engg.,
IIT Guwahati, Assam, India

Texture Mapping

- We have limited ability to generate complex surfaces with geometry
- Illusion of geometry can be achieved without using analytical methods
  - Such techniques are generally called texture mapping
- Three types
  - Projected texture
  - Texture map
  - Solid texture

Projected Texture

- We have a texture image -- a 2D array of color values (texels)
  - Texture image called "texture map"
  - Usually synthesized/scanned images
- At each screen pixel, texel can be used to substitute a polygon's surface property (color)

Projected Texture

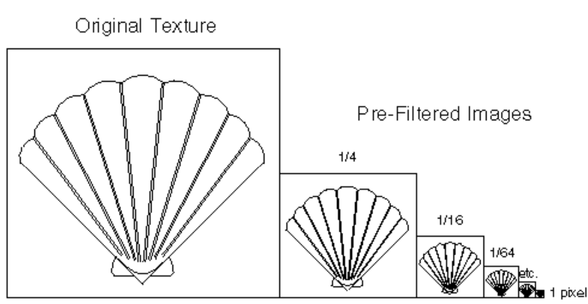
- There are three ways the substitution can be done
  - Replace surface pixel color with the color of the texel
  - Apply the following function for a smooth blending $C'' = (1-k).C + k.C' \quad 0 \leq k \leq 1$
  - Perform logical operation (AND, OR etc) between the two pixel values

MIPMAP

- A special projected texture technique
- **Multum In Parvo** -- many things in a small place
  - Pre-specify a series of pre-filtered texture maps of decreasing resolutions
  - Requires more texture storage
  - Eliminates shimmering and flashing as objects move

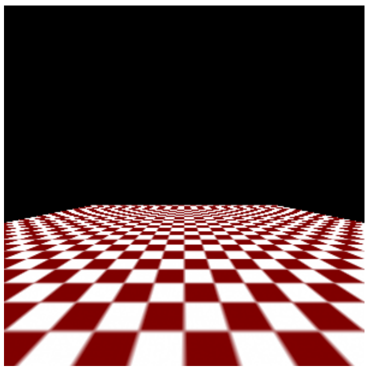
MIPMAP

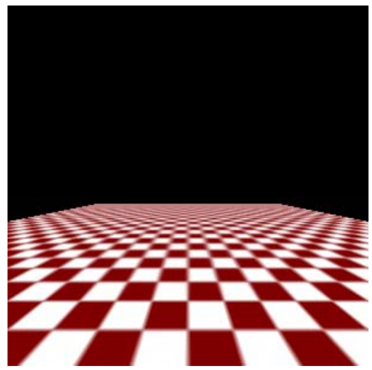
- Arrange different versions into one block of memory



MIPMAP

- With versus without MIPMAP





Texture Mapping

- Useful for curved surfaces
- Mapping from texture space to surface space
  - We define a 2D function in a texture co-ordinate system (u, w)
  - Surface in parametric form (θ, φ)
  - Define mapping functions from texture to object space and vice-versa

Texture Mapping

- Usually linear mapping function is used

  $\theta = A.u + B$

  $\varphi = C.w + D$

  A, B, C, D are constants, can be obtained from relations between known points in the two spaces (corners of the texture map and corresponding surface points)

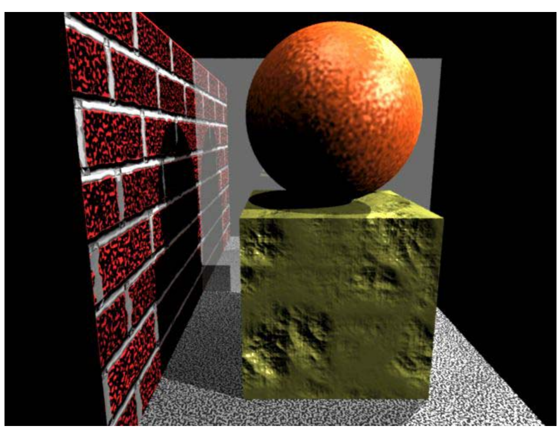
Texture Mapping

- Texture map better than projected texture for curved surfaces
  - However, for complex surfaces, it is difficult to find mapping function
- Both methods are weak when feature of texture on one surface should *match* those on other
  - i.e. simulating an object curved out from a material (e.g. a block of wood)

Solid Texture

- Texture defined in 3D texture space (unlike the previous case) – represents the structure of the material such as wood, marble
- The texture definition is called solid texture
  - Often defined procedurally
- Map surface point (x,y,z) to (u, v, w) in texture space
  - Use transformations to place object into the coordinate system that defines the texture

Bump Mapping



Bump Mapping

- Texture mapping not sufficient to introduce "roughness" to a surface
- Modify surface geometry
  - "Perturbation" function to change the surface normal directions
- Calculate intensity with the modified surface normals
  - Modified normals model rough surface

Intensity Representation

- Illumination model gives intensity as any value in the range of 0.0 to 1.0
  - A graphics system can display only a limited set of intensity values
- A calculated intensity must be converted to one of the allowable system values
  - Also, the allowable system intensity levels should be distributed so that they correspond to the way our eyes perceive intensity differences

Perception: Relative Intensity

- We are not good at judging absolute intensity
- Let's illuminate pixels with white light on scale of 0.0 - 1.0
  - Intensity difference of neighboring colored rectangles with intensities:
    - 0.10 -> 0.11 (10% change)
    - 0.50 -> 0.55 (10% change)

  will look the same
- We perceive relative intensities, not absolute

Representing Intensities

- If the ratio of two intensities is same as ratio of two other intensities, we perceive the difference between each pair of intensities as the same
  - Preserve ratio
- How to represent n+1 successive intensity levels with equal brightness?
  - Use photometer to obtain min and max brightness of monitor
  - This is the *dynamic range*
  - Intensity ranges from min, I_0, to max, 1.0

Representing Intensities

- $I_1/I_0=I_2/I_1=\ldots=I_n/I_{n-1} = r$
- $I_k = r^k I_0$, $k > 0$
- ex: B/W monitor with 8 bits/pixel
  - $n = 255$
  - $r = 1.0182$ (typical)
  - $I_0 = 0.01$ (say)
  - Ints = 0.0100. 0.0102, 0.0104…1…

$I_0=I_0$
$I_1 = rI_0$
$I_2 = rI_1 = r^2I_0$
…
$I_{255}=rI_{254}=r^{255}I_0$

Gamma Correction

- Illumination model produce linear range of colors
  - RGB (0.25, 0.25, 0.25)=1/2 intensity of RGB (0.5, 0.5, 0.5)
- Video monitors are non linear
  - Intensity (*brightness* of the electron gun) = *a*(voltage applied)$^{\gamma}$
    - i.e., brightness * voltage != (2*brightness) * (voltage/2)

Gamma Correction

- Thus, if we set voltage proportional to calculated pixel value
  - Due to non-linearity, there will be a shift in displayed intensity
- Common solution: *gamma correction*
  - Adjust voltage before applying to E-gun using inverse function $V = \left(\frac{I}{a}\right)^{\frac{1}{\gamma}}$
    - In other words, the transmitted signal is deliberately distorted so that, after it has been distorted again by the display device, the viewer sees the correct brightness
  - Can have separate γ for R, G, B
  - A video lookup table is used to store different V for different I's (pre-computation)

Gamma Correction

- a and γ depends on monitor characteristics
  - $1.7 \le \gamma \le 2.3$ (typical)
- Some monitors perform the gamma correction in hardware (SGI's)
- Others do not (most PCs)
- Tough to generate images that look good on both platforms

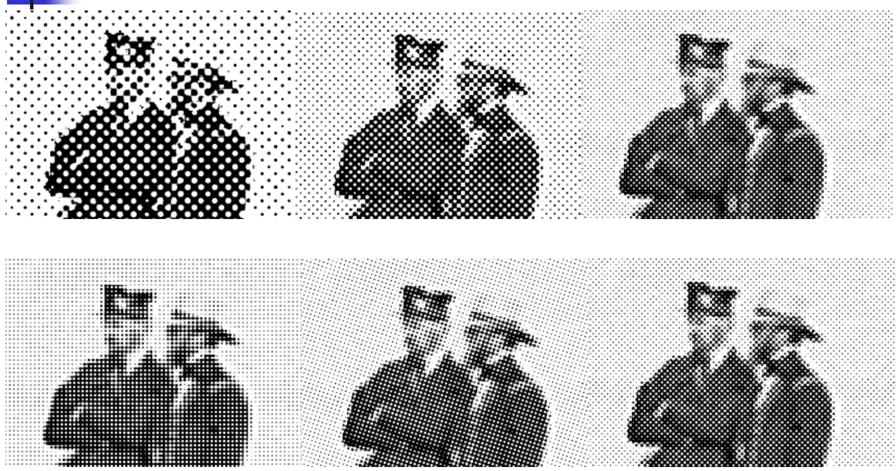
Intensity – IM to Device

- Let I = intensity values calculated by an illumination model (IM)
- Calculate nearest intensity level I_k supported by the device (from a table of pre-computed intensity values)
- Calculate electron gun voltage taking into account Gamma correction
  - Keep the calculated voltage in the look-up table

Halftoning

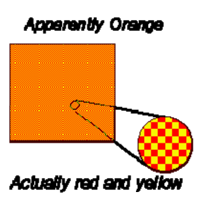
- Used to represent intensities on bi-level devices
- Pixel grid is used to represent intensity, instead of single pixel
  - n×n pixel grid = $n^2 + 1$ intensity levels
  - For example, a 2×2 pixel grid can represent 5 intensity levels
- Drawback
  - Reduces resolution

Halftoning



Dithering

- Halftoning for color images
- Dithering also refers to halftoning without reducing resolution
  - Floyd-Steinberg error diffusion technique



Floyd-Steinberg Error Diffusion

- A pixel is printed using the closest intensity device supports
- The error term propagated to neighboring pixels (yet to be scanned)

  S(x,y) = original pixel value at x,y

  e = S(x,y) - approximated intensity value

Then,

S(x+1,y) += ae	a=7/16
S(x-1,y+1) += be	b=3/16
S(x,y+1) += ce	c=5/16
S(x+1,y+1) += de	d=1/16

- Scan order: left→right, top→bottom
- Origin: top left corner
- +ve Y-axis in downward direction