

---

## Chapter 2. Digital Image Representation

CS 3570

---

## Main topics in the Chapter 2

- Three main types of creating digital images
  - Bitmapping, Vector graphics, Procedural modeling
- Frequency in digital image
- Discrete Cosine Transform (DCT)
- Aliasing
  - Moiré patterns
  - Demosaicing
  - Color aliasing
- Different color models
- Vector graphics

---

## Three Types of Digital Image Creation

- **Bitmaps**
  - Pixel-by-pixel specification of points of colors
  - Created by digital cameras, scanners, paint programs, image processing programs...
- **Vector graphics**
  - Created in program like Adobe Illustrator and Corel Draw - use object specifications and mathematical equations to describe shapes to which colors are applied
- **Procedural modeling**
  - Algorithmic art

---

## Bitmaps - digitization

- **Pixel (picture element)**
  - A number representing the color at position  $(r, c)$  in the bitmap, where  $r$  is the row and  $c$  is the column.
- **Sampling**
  - Under-sampled images may contain blocky and unclear effect.
- **Quantization**
  - Low-bit depth can result in patchiness of color



Original



Sampling



Quantization

## Bitmaps – pixel dimensions & resolution

- **Pixel dimensions**

- The number of pixels horizontally (*i.e.*, width,  $w$ ) and vertically (*i.e.*, height,  $h$ ), denoted  $w \times h$

- **Resolution**

- The number of pixels in an image file per unit of spatial measure
- Resolution can be measured in pixels per inch, abbreviated *ppi*. A 200 ppi image will print out using 200 pixels to determine the colors for each inch
- Resolution of a printer is a matter of how many dots of color it can print over an area. A common measurement is **dots per inch** (DPI).

## Bitmaps – image size

- **Image size**

- The physical dimensions of an image when it is printed out or displayed on a computer
- Relation of image size, pixel dimension and resolution
  - $a = w/r$ ,  $b = h/r$
  - Resolution  $r$ , Pixel dimensions  $w \times h$ , Printed image size  $a \times b$
- Different image size

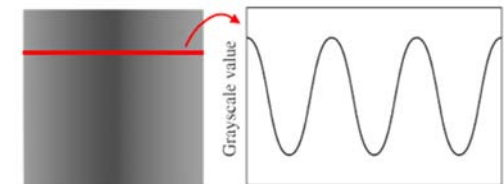


## Frequency in digital image

- An image as a function that can be represented in a graph
  - $y = f(x)$
  - $x$ : the position of one point
  - $y$ : the color value at position  $x$
  - $f$ : **a function over the spatial domain** when the  $x$  values correspond to points in space
- In the realm of digital imaging, *frequency refers to the rate at which color values change*.

## Frequency in digital image

- An image in which color varies continuously from light gray to dark gray and back again

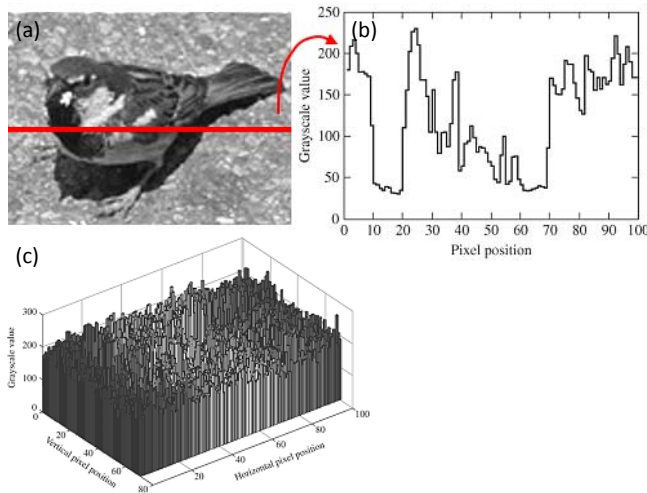


- Discretized version of gradient



## Frequency in digital image

- (a) A grayscale bitmap image
- (b) Graph of one row of sparrow bitmap over the spatial domain – you can imagine that each row is a complex waveform
- (c) Graph of two-dimensional bitmap of sparrow



## Discrete Cosine Transform

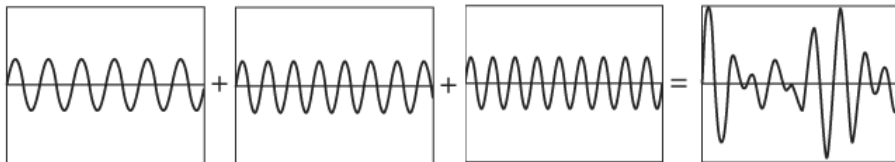
- Any complex periodic waveform can be expressed as an infinite sum of simple sinusoidal wave

$$f(x) = \sum_{n=0}^{\infty} a_n \cos(n\omega x)$$

- $f(x)$ : a continuous periodic function over the spatial domain
- $\omega$ : represent angular frequency,  $\omega = 2\pi f$
- $f$ : the fundamental frequency of the wave
- $n$ : move through these frequency components
- $a_n$ : the amplitude for the  $n$ th cosine frequency component


## Discrete Cosine Transform

- Example
  - a wave that is the sum of three simple sine waves



## 1D Discrete Cosine Transform (DCT)

- Any row of  $M$  pixels can be represented as a sum of the  $M$  weighted cosine functions evaluated at discrete points

 A 1D image of eight pixels

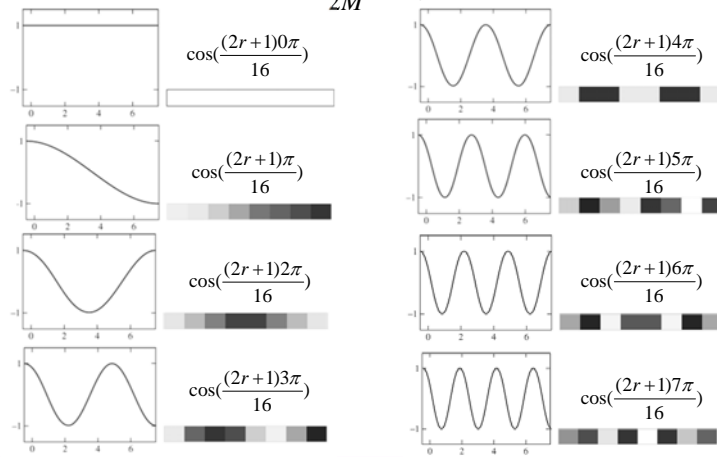
$$f(r) = \sum_{u=0}^{M-1} \frac{\sqrt{2}C(u)}{\sqrt{M}} F(u) \cos\left(\frac{(2r+1)u\pi}{2M}\right) \quad \text{for } 0 \leq r < M$$

$$\text{where } C(u) = \frac{\sqrt{2}}{2} \quad \text{if } u = 0 \quad \text{otherwise } C(u) = 1$$

- $f(r)$ : a one-dimensional array
- $F(u)$ : one-dimensional array of coefficients

## 1D DCT

- Basis function:  $\cos(\frac{(2r+1)u\pi}{2M})$ ,  $M=8$



## 1D DCT

- Coefficient  $F(u)$

$$F(u) = \sum_{r=0}^{M-1} \frac{\sqrt{2}C(u)}{\sqrt{M}} f(r) \cos(\frac{(2r+1)u\pi}{2M}) \quad \text{for } 0 \leq u < M$$

$$\text{where } C(u) = \frac{\sqrt{2}}{2} \quad \text{if } u = 0 \quad \text{otherwise } C(u) = 1$$

- $F(0)$ : DC component
- $F(1), \dots, F(M-1)$ : AC components
- Example

1. Input: [0,0,0,153,255,255,220,220]



2. Compute  $F(u)$  by using the above equation

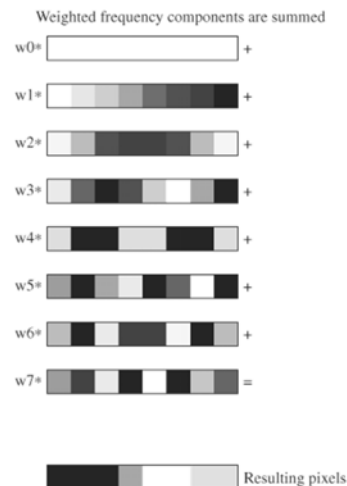
$$F(u) = [389.97, -280.13, -93.54, 83.38, 54.09, -20.51, -19.80, -16.34]$$

## 1D DCT

3. Each pixel is the combination of frequency component

$$f(r) = \frac{389.97}{\sqrt{M}} \cos(0) + \frac{\sqrt{2}}{\sqrt{M}} (-280.13 \cos(\frac{(2r+1)\pi}{2M}) - 93.54 \cos(\frac{(2r+1)2\pi}{2M}) + 83.38 \cos(\frac{(2r+1)3\pi}{2M}) + 54.09 \cos(\frac{(2r+1)4\pi}{2M}) - 20.51 \cos(\frac{(2r+1)5\pi}{2M}) - 19.80 \cos(\frac{(2r+1)6\pi}{2M}) - 16.34 \cos(\frac{(2r+1)7\pi}{2M}))$$

- The first element  $F(0)$  is called the **DC component**. Other components are **AC components**



## 2D DCT

- Coefficients of the frequency components (2D DCT)

$$F(u, v) = \sum_{r=0}^{M-1} \sum_{s=0}^{N-1} \frac{2C(u)C(v)}{\sqrt{MN}} f(r, s) \cos(\frac{(2r+1)u\pi}{2M}) \cos(\frac{(2s+1)v\pi}{2N})$$

$$\text{where } C(\delta) = \begin{cases} \frac{\sqrt{2}}{2} & \delta = 0 \\ 1 & \text{otherwise} \end{cases}$$

- 2D inverse DCT

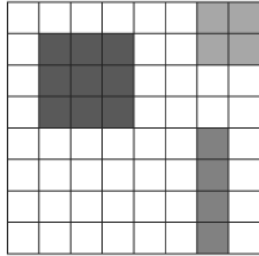
$$f(r, s) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \frac{2C(u)C(v)}{\sqrt{MN}} F(u, v) \cos(\frac{(2r+1)u\pi}{2M}) \cos(\frac{(2s+1)v\pi}{2N})$$

$$\text{where } C(\delta) = \begin{cases} \frac{\sqrt{2}}{2} & \delta = 0 \\ 1 & \text{otherwise} \end{cases}$$

## 2D DCT

- Example

- Input:  
8x8 bitmap image



|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 255 | 255 | 255 | 255 | 255 | 255 | 159 | 159 |
| 255 | 0   | 0   | 0   | 255 | 255 | 159 | 159 |
| 255 | 0   | 0   | 0   | 255 | 255 | 255 | 255 |
| 255 | 0   | 0   | 0   | 255 | 255 | 255 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 | 100 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 | 100 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 | 100 | 255 |
| 255 | 255 | 255 | 255 | 255 | 255 | 100 | 255 |

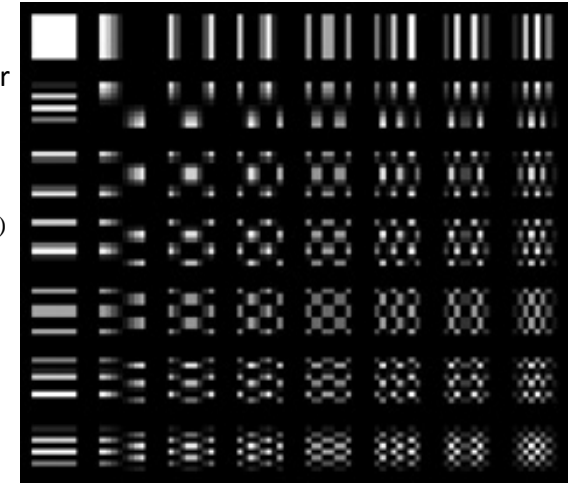
|      |      |      |      |     |      |     |      |
|------|------|------|------|-----|------|-----|------|
| 1628 | -61  | 39   | 234  | 173 | -128 | 171 | 22   |
| -205 | -163 | 74   | 222  | 1   | 74   | -30 | 111  |
| 81   | 150  | -95  | -82  | -42 | -11  | -6  | -53  |
| 188  | 231  | -135 | -188 | -53 | -36  | -2  | -103 |
| 96   | 71   | -42  | -78  | -32 | 2    | -17 | -32  |
| 25   | -42  | 25   | 3    | -14 | 27   | -26 | 19   |
| 70   | 15   | -6   | -51  | -17 | 7    | -16 | -13  |

## 2D DCT

- Example

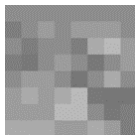
- Base frequencies for discrete cosine transform

$$\cos\left(\frac{(2r+1)u\pi}{2M}\right)\cos\left(\frac{(2s+1)v\pi}{2N}\right)$$

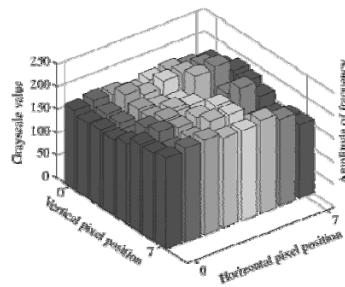


## Example of 2D DCT

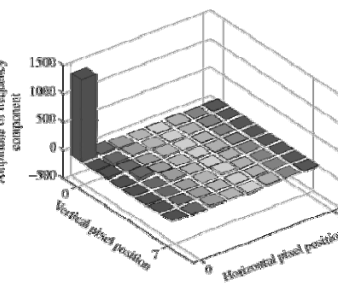
Close-up of 8 x 8 pixel area



Graph of pixel values over the spatial domain

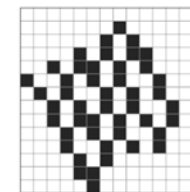
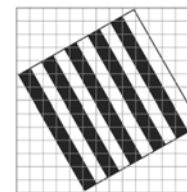


Graph of amplitudes over the frequency domain



## Aliasing - Moiré patterns

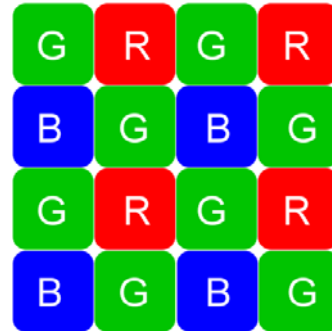
- Aliasing is caused by the discrete nature of pixels (*Sampling Error*).
- Moiré patterns (Also called moiré effect)
  - The sampling rate for the digital image is not high enough to capture the frequency of the pattern





## Bayer Color Filter

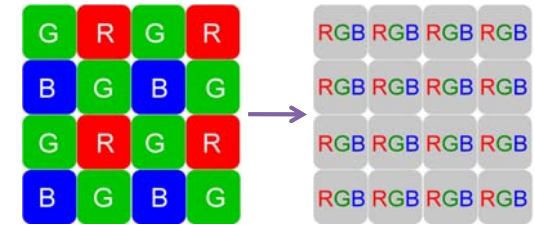
- A less expensive method of color detection uses an array to detect only one color component at each photosite.
- Interpolation is then used to derive the other two color components based on information from neighboring sites.
- A color filter array (CFA) for arranging RGB color filters on a square grid of photosensors.
- Twice as many green sensors as blue or red sensors. Because the human eye is more sensitive to green.



Bayer color filter

## Demosaicing

- The interpolation algorithm for deriving the two missing color channels at each photo-site is called **demosaicing**
- Algorithm
  - Nearest neighbor
  - Linear
  - Cubic
  - Cubic spline



## Nearest neighbor algorithm

### ALGORITHM 2.1 - NEAREST NEIGHBOR ALGORITHM

```

algorithm nearest_neighbor{
  for each photosite (i,j) where the photosite detects color c1 {
    for each c2 ∈ {red,green,blue} such that c2 ≠ c1 {
      S = the set of nearest neighbors of site (i,j) that have color c2
      set the color value for c2 at site (i,j) equal to the average of the color values
      of c2 at the sites in S
    }
  }
}
    
```



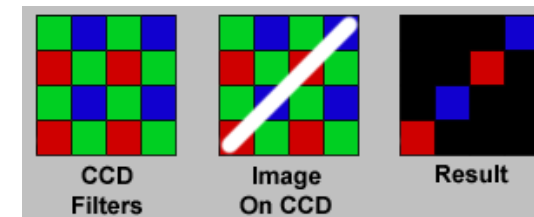
(a) Determining R or B from the center G photosite entails an average of two neighboring sites



(b) Determining B from the center R photosite entails an average of four neighboring sites diagonal from R

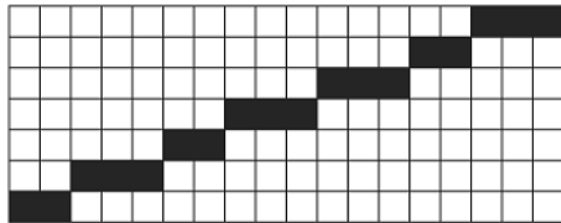
## Color Aliasing

- Interpolation can't give a perfect reproduction of the scene being photographed, and occasionally color aliasing results from the process, detected as moiré patterns, streaks, or spots of color not present in the original scene.
- Example: you photograph a white line on the black background



## Aliasing - Jagged Edges

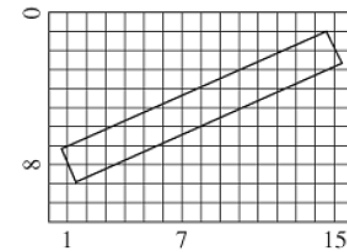
- **Aliasing** used to describe the jagged edges along lines or edges that are drawn at an angle across a computer screen
- A line on a computer screen is made up of discrete units: **pixels**



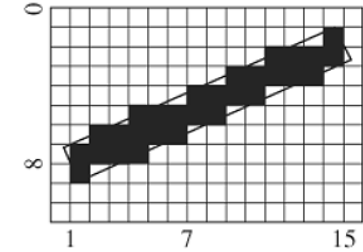
Aliasing of a line that is one pixel wide

## Algorithm 2.2 for drawing a line

- Left figure shows a line that is two pixels wide going from point (8, 1) to point (2, 15)
- In right figure a pixel is colored black if at least half its area is covered by the two-pixel line



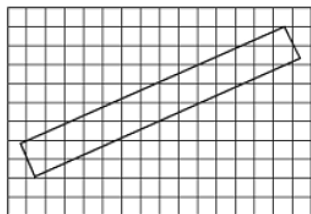
Drawing a line two pixels wide



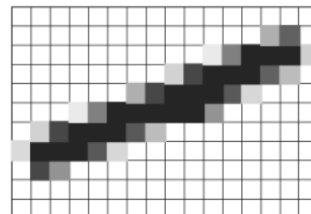
Line two pixels wide, aliased

## Anti-aliasing

- **Anti-aliasing** is a technique for reducing the jaggedness of lines or edges caused by aliasing
- The idea is to color a pixel with a shade of its designated color in proportion to the amount of the pixel that is covered by the line or edge



a line two pixels wide



Line two pixels wide, anti-aliased

## Aliasing of bitmap & vector graph

- Vector graphics suffer less from aliasing problems than do bitmap images in that vector graphics images can be resized without loss of resolution
- **Upsampling**
  - The image is later enlarged by increasing the number of pixels



Bitmap

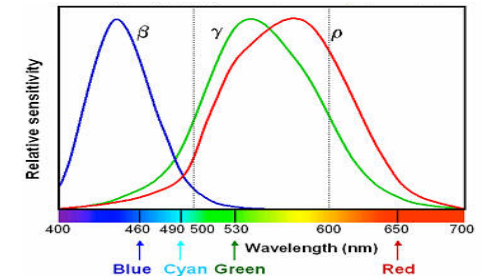
Vector Graph

## Aliasing of bitmap & vector graph

- Aliasing in rendering can occur in both bitmap and vector graphics, but vector graphics has an advantage over bitmap images with respect to aliasing
- Vector graphics and bitmap imaging each has advantages.
  - Vector graphics imaging is suitable for pictures that have solid colors and well-defined edges and shapes
  - Bitmap imaging is suitable for continuous tone pictures like photographs

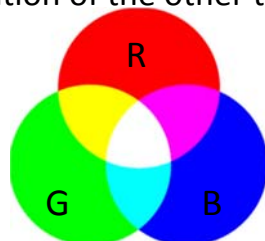
## Color

- Color is composed of electromagnetic waves
  - The wavelength of visible color fall between approximately 370 and 780 nanometers.
- Three Characteristics of colors
  - Hue (essential color): *dominant wavelength*
  - Saturation (color purity)
  - Luminance (lightness)
- Color model
  - RGB color model
  - CMY color model
  - HSV color model
  - HLS color model



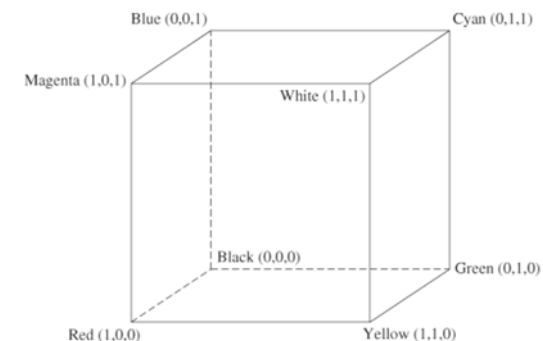
## RGB color mode

- Additive color mixing
- Combinations of three primary colors. No one of them can be created as a combination of the other two
  - Red
  - Green
  - Blue
- $C = rR + gG + bB$ 
  - $R, G, B$ : constant values based on the wavelengths
  - $r, g, b$ : the relative amounts. The values  $r, g, b$  are referred to as the values of the **RGB color components** (color channels)



## RGB color cube

- R, G, and B correspond to three axes in 3D space
- Normalize the relative amount of R, G and B in a color. Each value varies between 0 and 1.





## RGB color to grayscale

- The conversion of RGB color to grayscale

- In RGB color (R, G, B)
- In grayscale (L, L, L)

$$L = 0.30R + 0.59G + 0.11B$$

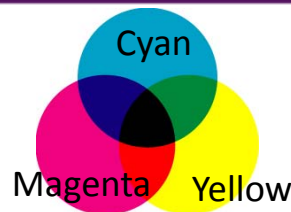
- Since all three color components are equal in a gray pixel, only one of the three values needs to be stored. Thus a 24-bit RGB pixel can be stored as an 8-bit grayscale pixel.

## RGB color model – advantage & disadvantage

- Advantage
  - The human eye perceives color
  - The computer monitor can be engineered to display color
- Disadvantage
  - Exist visible colors that can't be represented with positive value for each of the red, green and blue components
- It is necessary to “subtract out” some of the red, green, or blue in the combined beams to match the pure color

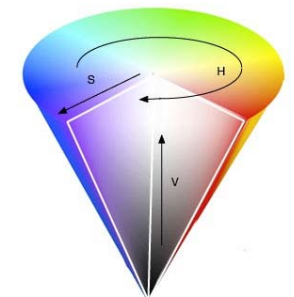
## CMY color model

- Subtractive color mixing
- Commonly used in printing
- It has three primaries
  - Cyan, Magenta, Yellow
- Conversion between RGB and CMY color models
  - $C = 1 - R$ ,  $M = 1 - G$ ,  $Y = 1 - B$
- CMYK model
  - In CMY color mode, the maximum amount of three primary should combine to black, but in fact producing a dark brown
  - Add a pure black -> K
  - $K = \min(C, M, Y)$ ,  $C_{\text{new}} = C - K$ ,  $M_{\text{new}} = M - K$ ,  $Y_{\text{new}} = Y - K$



## HSV color model

- Represent color by three terms
  - **Hue**, the color type (such as red, blue, or yellow)
    - ✓ Ranges from 0-360 (but normalized to 0-100% in some applications)
  - **Saturation**, the "vibrancy" of the color
    - ✓ Ranges from 0-100%
    - ✓ The lower the saturation of a color, the more "grayness"
  - **Value**, the brightness of the color
    - ✓ Ranges from 0-100%
- Also called HSB(hue, saturation and brightness)



### ALGORITHM 2.3 - RGB TO HSV

```

/* Input: r, g, and b, each real numbers in the range [0 . . . 1].
Output: h, a real number in the range of [0 . . . 360),
except if s = 0, in which case h is undefined.
s and v are real numbers in the range of [0 . . . 1].*/
{
    max = maximum(r,g,b)
    min = minimum(r,g,b)
    v = max
    if max ≠ 0 then s = (max - min)/max
    else s = 0
    if s = 0 then h = undefined
    else {
        diff = max - min
        if r == max then h = (g - b) / diff
        else if g == max then h = 2 + (b - r) / diff
        else if b == max then h = 4 + (r - g) / diff
        h = h * 60
        if h < 0 then h = h + 360
    }
}

```

## HLS color model

### Similar with HSV color model

- **Hue:**  
same as HSV
- **Saturation:**  
same as HSV
- **Lightness:**  
lightness varies from 0 at the black tip to 1 at the white tip of the double cones

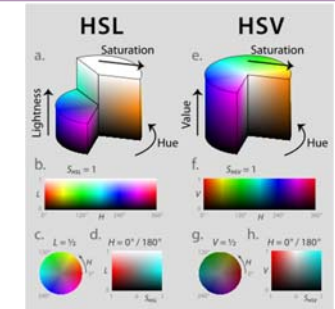


Fig. 21a. Color photograph

b. HSL/HSV hue of each color shifted by -30°

Source :  
[http://en.wikipedia.org/wiki/HSL\\_and\\_HSV](http://en.wikipedia.org/wiki/HSL_and_HSV)

## YIQ color model

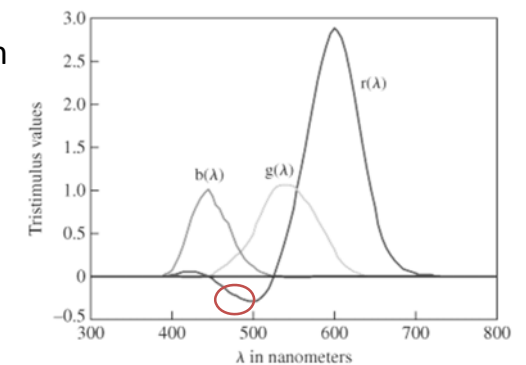
- It captures the luminance information in one value and puts the color (chrominance) information in the other two values
  - Luminance component: Y
  - Chrominance components: I, Q
- Convert RGB to YIQ

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- Translate YIQ to RGB => Invert the matrix
- YIQ color model is used in U.S. commercial television broadcasting
- Advantage in color/black and white broadcasting

## CIE color model

- Create a color space in which all other color models could be compared
  - A standard color model that represents all visible colors was called **CIE XYZ**
- Color matching function
  - X-axis: the wavelength,  $\lambda$ , ranging through the colors of the visible spectrum
  - Y-axis: the relative amounts of red, green, and blue light energy



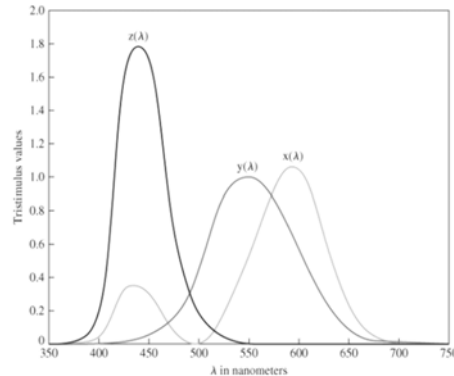
$$C(\lambda) = r(\lambda)R + g(\lambda)G + b(\lambda)B$$

## CIE color model

- CIE XYZ

- No three visible primary colors that can be combined in positive amounts to create all colors
- Use three “virtual” primaries—called X, Y, and Z—are purely theoretical rather than physical entities.
- Linear transform of the previous space

$$C(\lambda) = x(\lambda)X + y(\lambda)Y + z(\lambda)Z$$

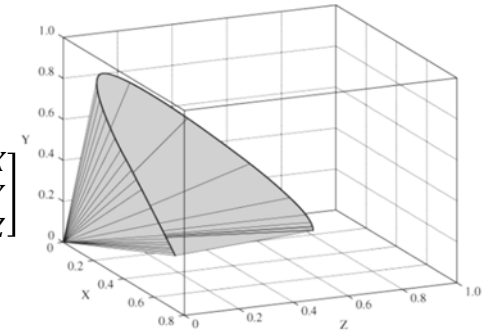


## CIE color model

- Can think of X, Y, Z as coordinates
- Linear transform from typical RGB or LMS

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.41 & 0.36 & 0.18 \\ 0.21 & 0.72 & 0.07 \\ 0.02 & 0.12 & 0.95 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.24 & -1.54 & -0.5 \\ -0.97 & 1.88 & 0.04 \\ 0.06 & -0.2 & 1.06 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

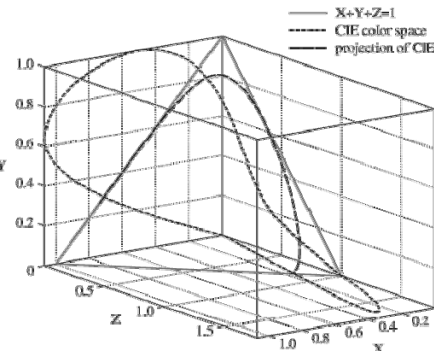


## Chromaticity and CIE color space

- Chromaticity values

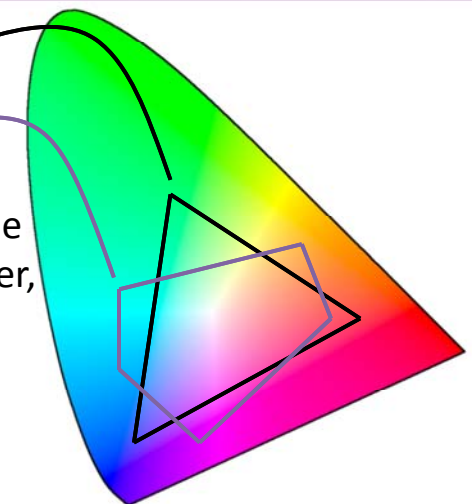
$$x'(\lambda) = \frac{x(\lambda)}{x(\lambda) + y(\lambda) + z(\lambda)}, y'(\lambda) = \frac{y(\lambda)}{x(\lambda) + y(\lambda) + z(\lambda)}, z'(\lambda) = \frac{z(\lambda)}{x(\lambda) + y(\lambda) + z(\lambda)}$$

- Visible color spectrum projected onto the  $X + Y + Z = 1$  plane



## CIE color model

- RGB color space
- CMYK color space
- The gamut for RGB color is larger than the CMYK gamut. However, neither color space is entirely contained within the other.



## CIE L\*a\*b\*, CIE L\*U\*V\*, and Perceptual Uniformity

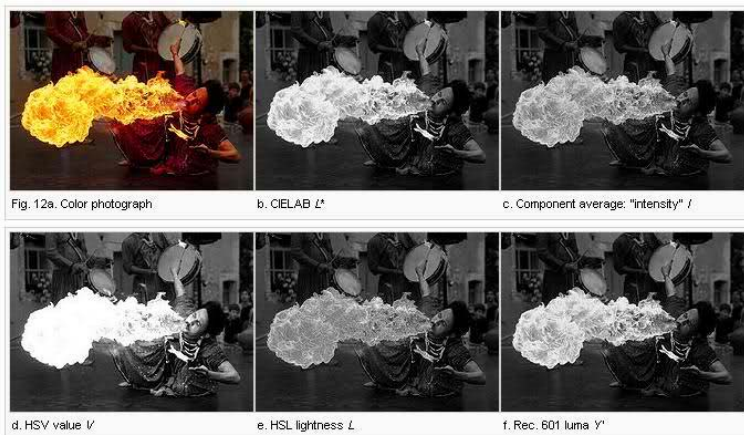
- The CIE XYZ model has three main advantages:
  - device-independent
  - provides a way to represent all visible colors
  - representation is based upon spectrophotometric measurements of color.
- A remaining disadvantage of the CIE XYZ model is that it is not perceptually uniform.
- In a **perceptually uniform color space**, the distance between two points is directly proportional to the perceived difference between the two colors.

## CIE L\*a\*b\*, CIE L\*U\*V\*

- CIE L\*a\*b\* is a subtractive color model in which the L\* axis gives brightness values varying from 0 to 100, the a axis moves from red (positive values) to green (negative values), and the b axis moves from yellow (positive values) to blue (negative values).
- CIE L\*U\*V is an additive color model that was similarly constructed to achieve perceptual uniformity, but that was less convenient in practical usage.
- Because the CIE L\*a\*b\* model can represent all visible colors, it is generally used as the intermediate color model in conversions between other color models.

## Image in different color models

- Source : [http://en.wikipedia.org/wiki/HSL\\_and\\_HSV](http://en.wikipedia.org/wiki/HSL_and_HSV)

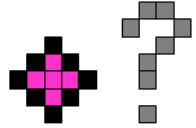



## Vector Graphics

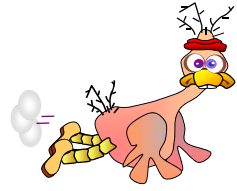





- **Vector graphics** is an image file format suitable for pictures with areas of solid, clearly separated colors—cartoon images, logos, and the like.
- Instead of being painted pixel by pixel, a vector graphic image is drawn object by object in terms of each object's geometric shape.
- Many file formats for vector graphics—.fh, .ai, .wmf, .eps, etc.—they contain the parameters to mathematical formulas defining how shapes are drawn.
  - A line can be specified by its endpoints, a square by the length of a side, a rectangle by the length of two sides, a circle by radius



## Bitmaps vs. Vector graphics

|                 | Pixel   | Vector  |
|-----------------|---|---|
| Example         |  |  |
| Grow and shrink | Bad   | Good  |
| Speed           | Fast to create.<br>Very hard to edit!   | Slow to create.<br>Much faster to edit.   |
| Applications    | Paint<br>Photoshop  | Power Point<br>Illustrator  |

## Bitmaps vs. Vector graphics

|                 | Original  | Shrink  | Grow  |
|-----------------|---|---|---|
| Vector graphics |  |  |  |
| Pixel           |  |  |  |

## Specifying curves

- Explicit form of a function**  $y = f(x)$   
provide a prescription for determining the *output* value of the function  $y$  in terms of the *input* value  $x$ 
  - E.g.  $y = \sqrt{r^2 - x^2}$
- Implicit form of the function**  $f(x, y) = 0$   
the value of  $y$  is obtained from  $x$  by *solving* an equation of the form
  - E.g.  $x^2 + y^2 - r^2 = 0$

## Parametric Curve

- Parametric representation of a function  $p(t) = (x(t), y(t))$ 

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$

$$P(t) = [x(t) \ y(t)] = [t^3 \ t^2 \ t \ 1] \begin{bmatrix} a_x & a_y \\ b_x & b_y \\ c_x & c_y \\ d_x & d_y \end{bmatrix} \quad 0 \leq t \leq 1$$

$$P = T * C$$
- Curve-generating algorithms can be divided into two main categories: interpolation algorithms and approximation algorithms.



## Bézier curves

- A Bézier curve is a parametric curve described by polynomials based on any number of control points
  - Degree of polynomials = number of points – 1
  - A Bézier curve passes through its first and last control points, but, in general, no others.

## Bézier curve

- A Bézier curve is defined by a cubic polynomial equation
- $P(t) = T * M * G$
- $T = [t^3 \ t^2 \ t \ 1]$ ,  $M$  is called the **basis matrix**.  $G$  is called the **geometry matrix**.

$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad G = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

- The blending functions are given by  $T * M$ ,  
 $P(t) = (T * M) * G = (1 - t)^3 p_0 + 3t(1 - t)^2 p_1 + 3t^2(1 - t) p_2 + t^3 p_3$

## Blending Function

- Bézier curves can be described in terms of the blending functions

$$P(t) = \sum_{k=0}^n p_k \text{blending}_{k,n}(t) \text{ for } 0 \leq t \leq 1$$

- In the blending functions  $\text{blending}_{k,n}(t)$ ,  $n$  is the degree of the polynomial and  $k$  refers to the “weight” for the  $k$ th term in the polynomial.

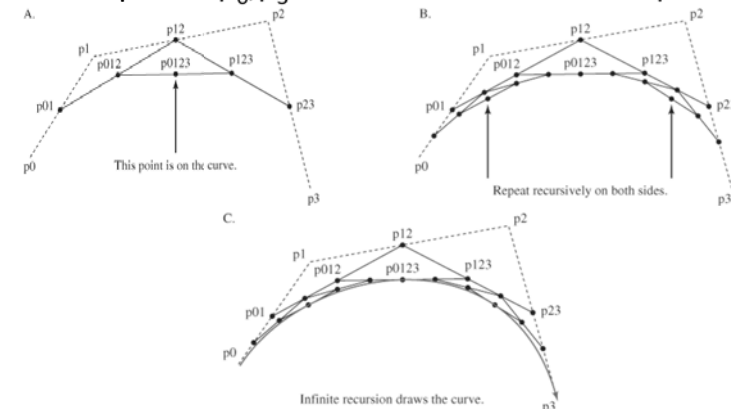
$$\text{blending}_{k,n}(t) = C(n, k) t^k (1 - t)^{n-k}$$

$$C(n, k) = \frac{n!}{k!(n - k)!}$$

- For cubic polynomials,  $\text{blending}_{0,3} = (1 - t)^3$ ,  $\text{blending}_{1,3} = 3t(1 - t)^2$   
 $\text{blending}_{2,3} = 3t^2(1 - t)$ ,  $\text{blending}_{3,3} = t^3$

## de Casteljau algorithm

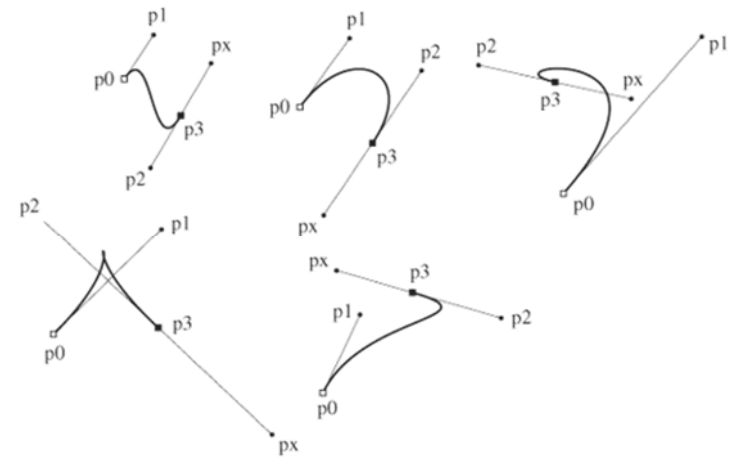
- For constructing a Bézier curve recursively
- Two endpoints  $p_0, p_3$  and two interior control points  $p_1, p_2$ .



## Bézier curves – advantage & disadvantages

- Advantage
  - Very simple
- Disadvantages
  - Expensive to evaluate the curve at many points
  - No easy way of knowing how fine to sample points, and maybe sampling rate must be different along curve
  - No easy way to adapt. In particular, it is hard to measure the deviation of a line segment from the exact curve

## Examples of Bézier curves



## Procedural modeling

- You create a digital image by writing a computer program based on some **mathematical** computation or unique type of **algorithm**
  - Procedural models are often defined by a small set of data that describes the overall properties of the model. For example, a tree might be defined by some branching properties and leaf shapes
  - The actual model is constructed by a procedure that often makes use of randomness to add variety. In this way, a single tree pattern can be used to model an entire forest

## Procedural modeling – fractal

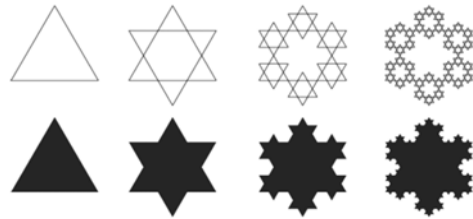
- **fractal** - algorithmic art
  - Fractal : a graphical image characterized by a recursively repeating structure
  - Fractals exist in natural phenomena.
  - If you look at the image at the macro-level—the entire structure—you'll see a certain geometric pattern, and then when you zoom in on a smaller scale, you'll see that same pattern again

Natural fractal structure



## Procedural modeling – fractal

- fractal is created with a recursive program that draws the same shape down to some base level.
- Koch's snowflake, a recursively defined fractal



- Sierpinski's gasket



## Procedural modeling – mandelbrot fractal

- A beautiful and complex geometry emerges from the nature of the computation
- The Mandelbrot fractal computation is based on a simple iterative equation
  - $f(z) = z^2 + c$
  - $z$  and  $c$  are complex numbers.  $c = c_r + c_i i$ ,  $z = z_r + z_i i$
  - The output of the  $i$ th computation is the input  $z$  to the  $i + 1$ st computation

## Procedural modeling – mandelbrot fractal

Goal: determine the color of each pixel

### 1) Initial

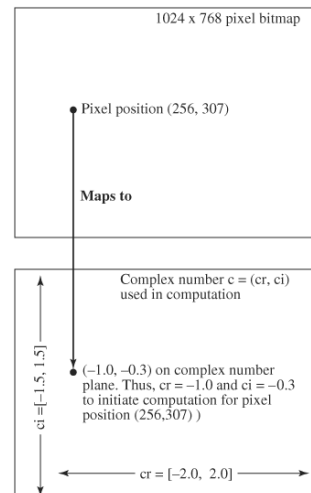
- Complex number  $c \leftrightarrow$  pixel position
- Complex number  $z = 0$

### 2) $f(z) = z^2 + c$ iteratively

The output of the  $i$ th computation is the input  $z$  to the  $i + 1$ st computation

### 3) The color of pixel

- Black:** the computation has not revealed itself as being unbounded
- Color:** related to how many iterations were performed



### ALGORITHM 2.5 – Mandelbrot Fractal

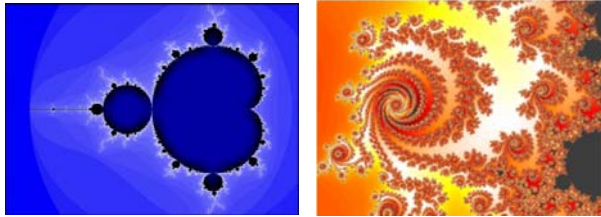
/\*Input: Horizontal and vertical ranges of the complex number plane.

Output: A bitmap for a fractal.\*/

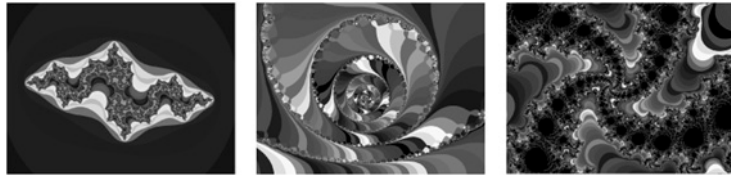
```
{
/*constant MAX is the maximum number of iterations*/
for each pixel in the bitmap {
    map pixel coordinates (x,y) to complex number plane coordinates (cr, ci)
    num_iterations = 0
    z = 0
    while num_iterations < MAX and not_unbounded* {
        z = z^2 + c
        num_iterations = num_iterations + 1
    }
    /*map_color(x,y) uses the programmer's chosen color map to determine the color of
    each pixel based on how many iterations are done before the computation is found to
    be unbounded*/
    if num_iterations == MAX then color(x,y) = BLACK
    else color(x,y) = map_color(num_iterations)
}
}
```

## Examples of mandelbrot fractal

- Mandelbrot fractals

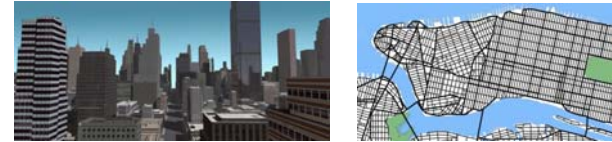


- **Julia fractal**, similar with Mandelbrot fractals, but has different  $c$



## Procedural modeling - example

- Procedural Modeling Of Cities
  - Procedural Modeling of Cities [Parish and Müller, 2001]



- Procedural Modeling of Buildings [Haegler et al. 2006]

