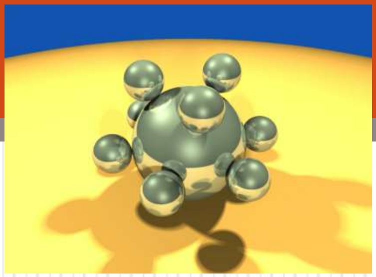


Introduction to Shading Models

Visual Realism Requirements

- Light Sources
- Materials (e.g., plastic, metal)
- Shading Models
- Textures
- Reflections
- Shadows

Rendering Objects

- we want to make the objects look visually interesting, realistic, or both.

- Develop methods of **rendering** for the objects of interest.

- Rendering: ***computes*** **how each pixel of a picture should look using different shading models.**

Rendering Objects (2)

- Much of rendering is based on different shading models,
  - describes how light from light sources interacts with objects in a scene.
- It is impractical to simulate all of the physical principles of light, scattering and reflection.
- A number of approximate models have been invented that do a good job and produce various levels of realism.

Shading Models: Introduction

- A shading model dictates how light is scattered or reflected from a surface.
- Simple shading models focuses on achromatic light
  - It has brightness but no color
  - Only shade of gray
  - Described by single intensity value.
- Graphics uses two types of light sources
  - Ambient light - doesn't come directly from a source, but through windows or scattered by the air, comes equally from all directions.
  - Point-source light comes from a single point.

Shading Models: Introduction (2)

- When light hits an object,
  - some light is absorbed (and turns into heat),
  - some is reflected,
  - some may penetrate the interior (e.g., of a clear glass object).
- If all the light is absorbed, the object appears black and is called a blackbody.
- If all the light is transmitted, the object is visible only through reflection

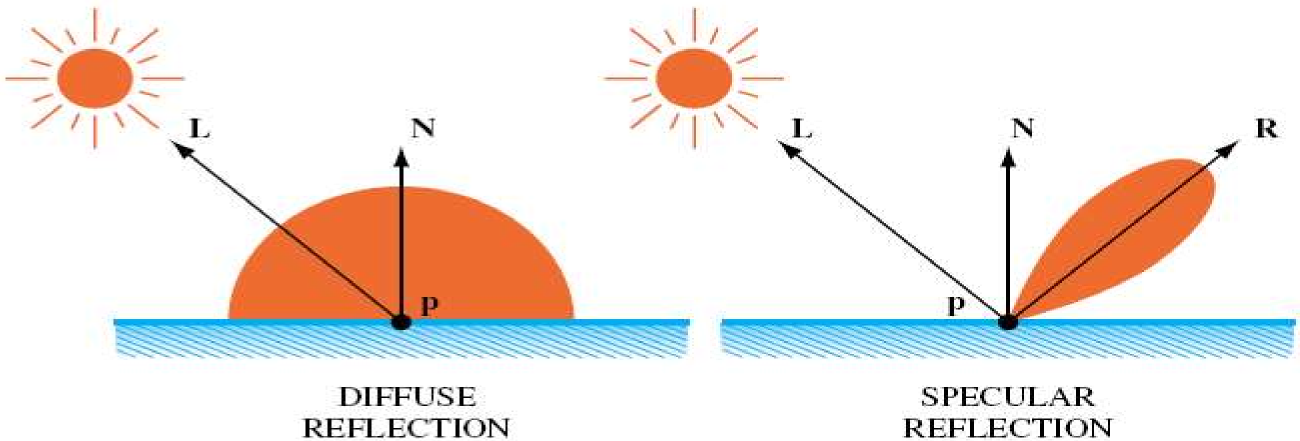
Shading Models: Introduction (3)

- When light is reflected from an object, some of the reflected light reaches our eyes, and we see the object.

- The amount of light that reaches the eye depends on the
  - Orientation of the surface
  - Light sources
  - Observer

- There are two types of reflection of incident light:
  - **Diffuse Scattering**
  - **Specular Reflections**

Shading Models: Introduction (4)



- **Diffuse Scattering**:
  - some of the incident light slightly penetrates the surface
  - re-radiated uniformly in all directions.
  - The light takes on some fraction of the color of the surface.
- **Specular reflection**:
  - more mirror-like and highly directional.
  - Incident light does not penetrate.
  - Light is reflected directly from the object's outer surface, giving rise to highlights of approximately the same color as the source.
  - The surface looks shiny.

Shading Models: Introduction (5)

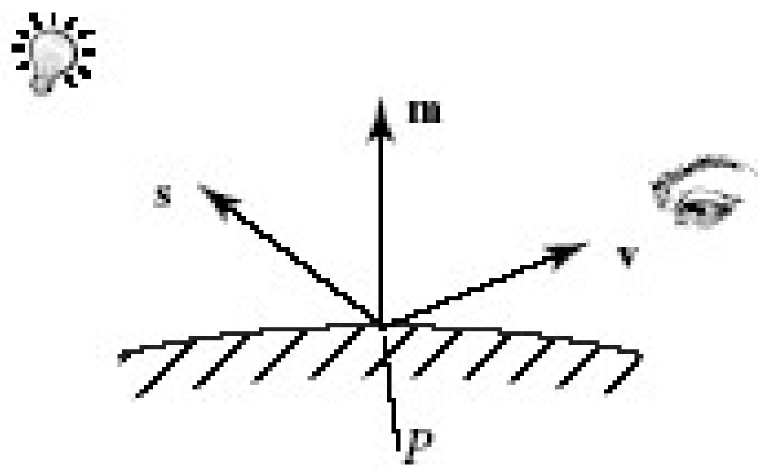
- In the simplest model, specular reflected light has the same color as the incident light. This tends to make the material look like plastic.
- In a more complex model, the color of the specular light varies over the highlight, providing a better approximation to the shininess of metal surfaces.
- Most surfaces produce some combination of diffuse and specular reflection, depending on surface characteristics such as roughness and type of material.
- The total light reflected from the surface in a certain direction is the sum of the diffuse component and the specular component.

Reflected Light Model

- Finding Reflected Light: a model
  - Model is not completely physically correct, but it provides fast and relatively good results on the screen.
  - Intensity of a light is related to its brightness. We will use I_s for intensity, where s is R or G or B.

Calculating Reflected Light

- To compute reflected light at point P, we need 3 vectors:
  - normal **m** to the surface at P
  - vectors **s** from P to the source
  - **v** from P to the eye.
  - the angles between these three vectors form the basis for computing light intensities



Ambient Light

- Our desire for a simple reflection model leaves us with far from perfect renderings of a scene.
  - E.g., shadows appear to be unrealistically deep and harsh.

- To soften these shadows, we can add a third light component called ambient light.

- This light arrives by multiple reflections from various objects in the surroundings and from light sources that populate the environment, such as light coming through a window, fluorescent lamps, etc.

- We assume a uniform background glow called **ambient light** exists in the environment.

Calculating Ambient Light

- The source is assigned an intensity, I_a.
- Each face in the model is assigned a value for its **ambient reflection coefficient**, ρ_a (often this is the same as the diffuse reflection coefficient, ρ_d),
- The term $I_a \rho_a$ is simply added to whatever diffuse and specular light is reaching the eye from each point *P* on that face.
- I_a and ρ_a are usually arrived at experimentally, by trying various values and seeing what looks best.

Calculating Diffuse Light

- A fraction of incident light is reradiated in all directions

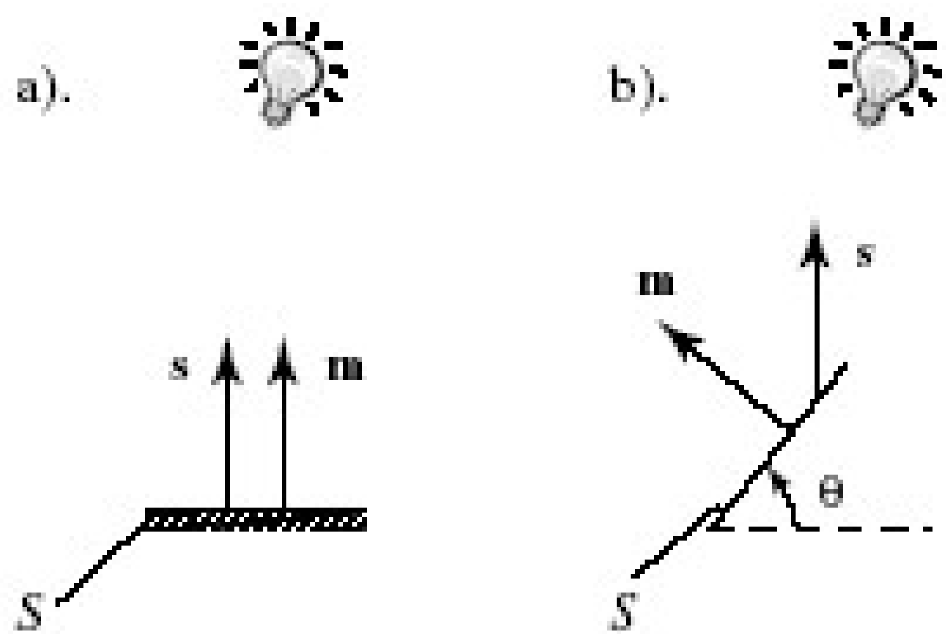
- Diffuse scattering is assumed to be independent of the direction from the point, *P*, to the location of the viewer's eye. (omindirectional scattering)

- Because the scattering is uniform in all directions, the orientation of the face *F* relative to the eye is not significant,
  - I_d is independent of the angle between **m** and **v** (unless $\mathbf{v} \cdot \mathbf{m} < 0$, making $I_d = 0$.)

- The amount of light that illuminates the face *does* depend on the orientation of the face relative to the point source:
  - the amount of light is proportional to the area of the face that it sees: the area *subtended* by a face.

Calculating Diffuse Light (2)

- The relationship between brightness and surface orientation is called as Lambert's law.
- Left : I_s (normal vector m is aligned with s)
- Right: $I_s \cos\theta$ (face is turned partially away from light source)



Calculating Diffuse Light (3)

- For θ near 0°, brightness varies only slightly with angle, because the cosine changes slowly there.
- As θ approaches 90°, the brightness falls rapidly to 0.

- We know $\cos\theta = (\mathbf{s} \cdot \mathbf{m})/(|\mathbf{s}|\,|\mathbf{m}|)$.

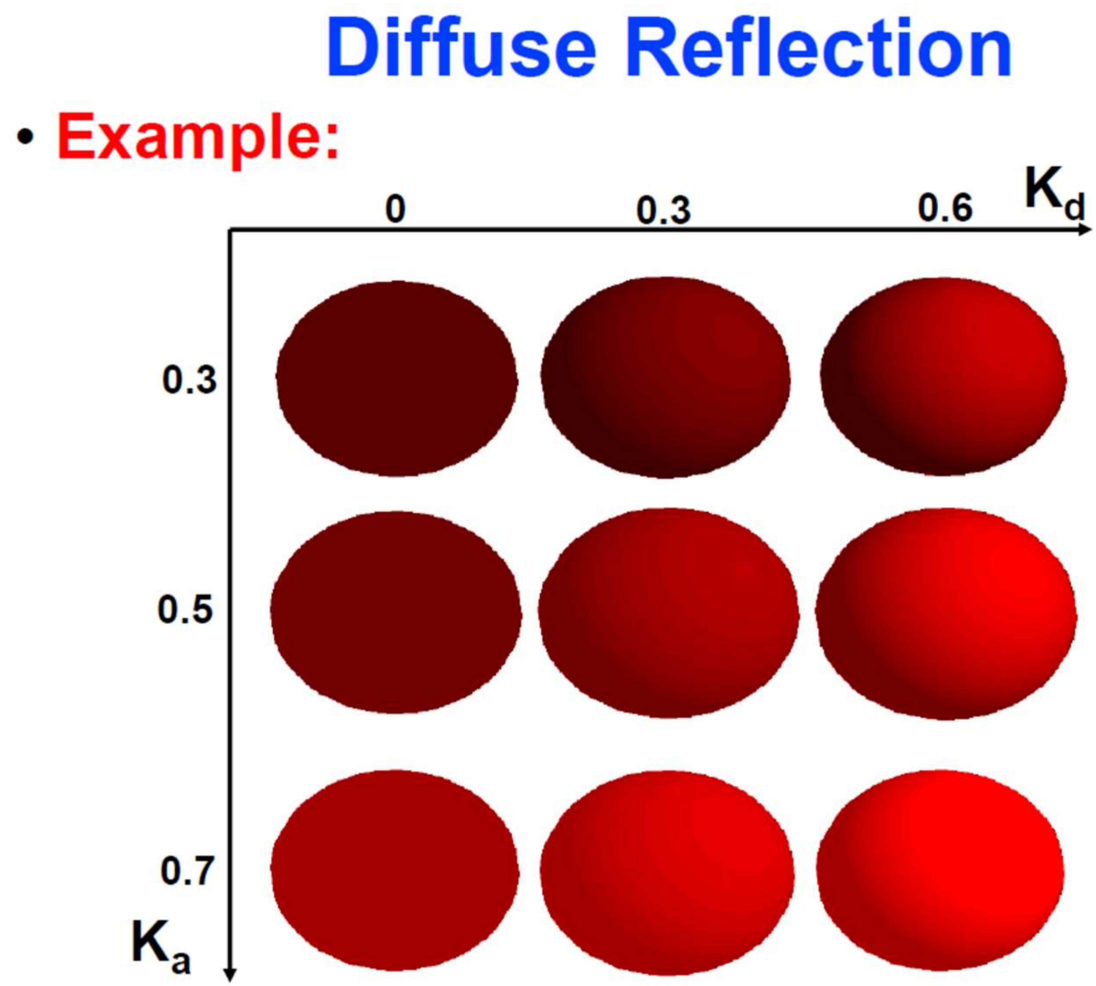
- $I_d = I_s\, \rho_d\, (\mathbf{s} \cdot \mathbf{m}) / (|\mathbf{s}|\,|\mathbf{m}|)$
  - I_d – Intensity of the reradiated light that reaches eye.
  - I_s is the intensity of the source.
  - ρ_d is the diffuse reflection coefficient and depends on the material the object is made of.

Calculating Diffuse Light (4)

- If facet is aimed away from the eye this dot product is negative and we want I_d to evaluate to zero
- **If $\mathbf{s} \cdot \mathbf{m}$** < 0 we want $I_d = 0$.
- So to take all cases into account, we use

$$\boxed{I_d = I_s \rho_d \max\left[(\mathbf{s} \cdot \mathbf{m}) / (|\mathbf{s}|\,|\mathbf{m}|),\ 0\right]}$$

Example: Spheres Illuminated with Diffuse Light.

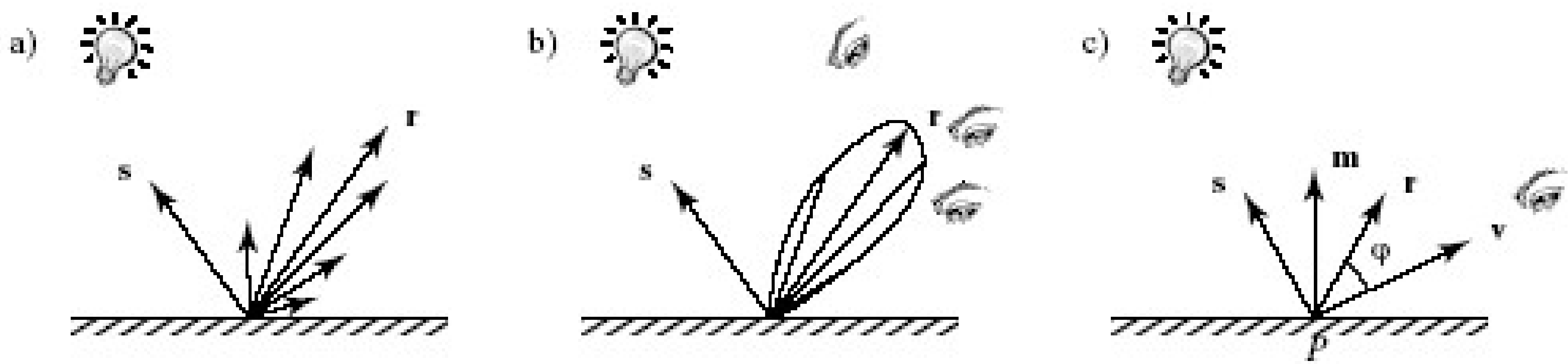


Calculating the Specular Component

- Real objects do not scatter light uniformly in all directions; a specular component is added to the shading model.
- Specular reflection causes highlights, which can add significantly to realism of a picture when objects are shiny.
- A simple model for specular light was developed by Phong. It is easy to apply.
  - The highlights generated by the Phong model give an object a plastic-like or glass-like appearance.
  - The Phong model is less successful with objects that are supposed to have a shiny metallic surface,
  - In this model, the amount of light reflected is the greatest in the direction of perfect mirror reflection,r where the angle of incidence equals the angle of reflection.

Calculating the Specular Component (2)

- Most of the light reflects at equal angles from the (smooth and/or shiny) surface, along direction **r,** the reflected direction.



Calculating the Specular Component (2)

- The direction r of perfect reflection depends on both s and normal vector m
  - compute $\mathbf{r} = -\mathbf{s} + 2\,\mathbf{m}\,(\mathbf{s}\cdot\mathbf{m})/(|\mathbf{m}|^2)$ (mirror reflection direction).
- For surfaces that are not mirrors, the amount of reflected light decreases as the angle φ between **r** and **v** increases.
- For a simplified model, we say the intensity decreases as $\cos^f \varphi$,
  - where f (amount of falloff) is chosen experimentally between 1 and 200.

Calculating the Specular Component (3)

- $\cos \varphi = \mathbf{r} \cdot \mathbf{v} / (|\mathbf{r}|\,|\mathbf{v}|)$

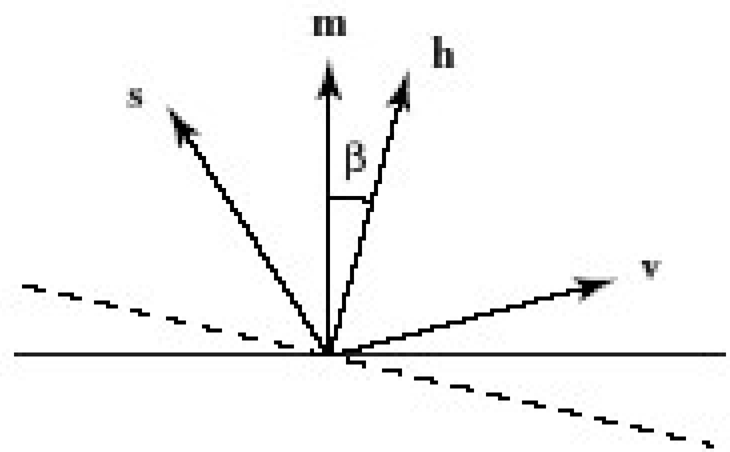
- $I_{sp} = I_s \, \rho_s \, (\mathbf{r} \cdot \mathbf{v} / (|\mathbf{r}|\,|\mathbf{v}|))^f$.
  - ρ_s is the specular reflection coefficient, which depends on the material.

- If $\mathbf{r} \cdot \mathbf{v} < 0$, there is no reflected specular light, the set $I_{sp} = 0$

Specular Component, $I_{sp} = I_s \, \rho_s \, \max[(\mathbf{r} \cdot \mathbf{v} / (|\mathbf{r}|\,|\mathbf{v}|))^f, 0]$

Speeding up Calculations for Specular Light

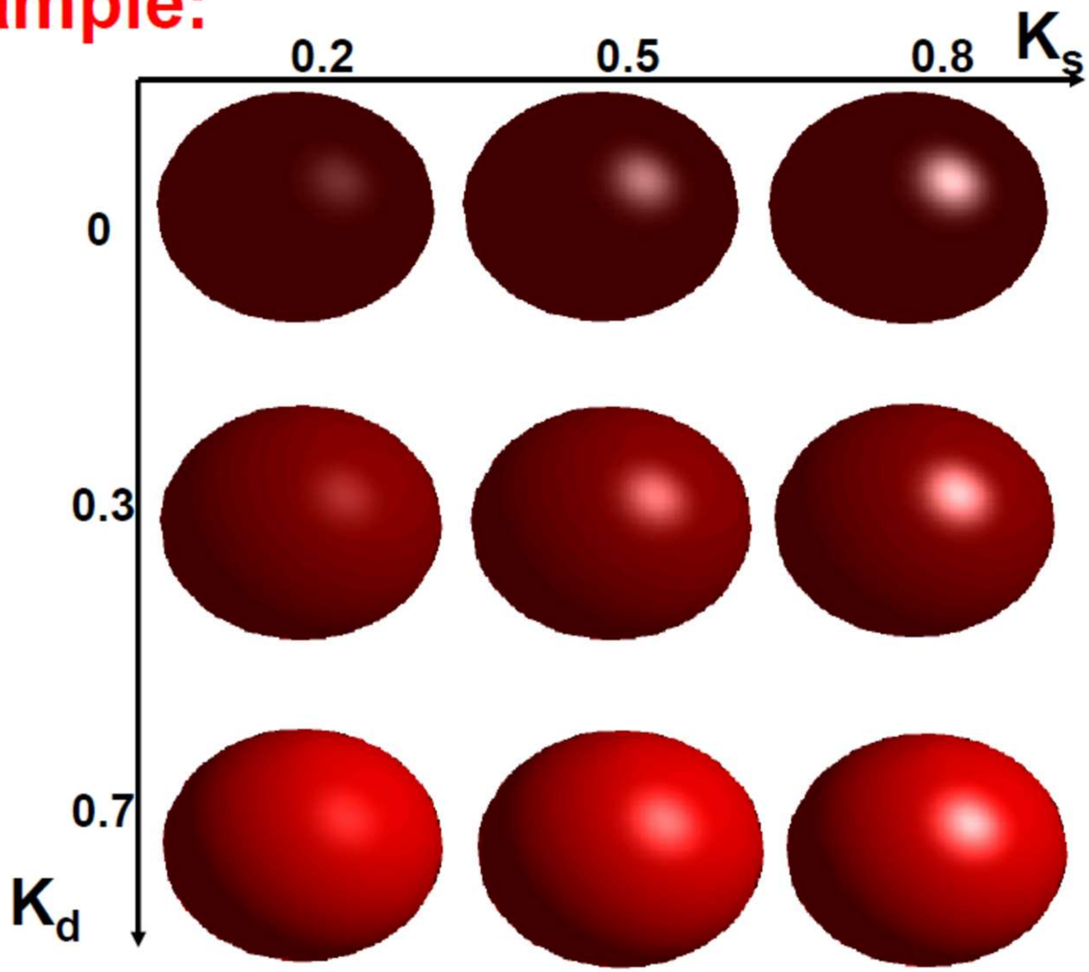
- Find the halfway vector $\mathbf{h} = \mathbf{s} + \mathbf{v}$.
- Then the angle β between $\mathbf{h}$ and $\mathbf{m}$ approximately measures the falloff intensity.
- To take care of errors, we use a different f value, and write



$$I_{sp} = I_s \, \rho_s \max[(\mathbf{h} \cdot \mathbf{m} / (|\mathbf{h}| \, |\mathbf{m}|))^f, 0]$$

Specular Reflection

- **Example:**



Combining Light Contributions and Adding Color

- I=amibent+diffuse+specular

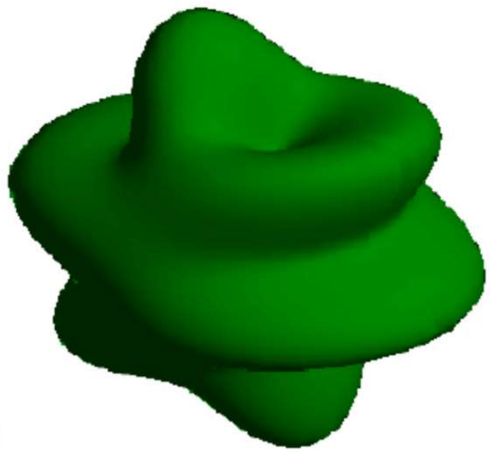
- **$I = I_a\, \rho_a + I_s\, \rho_d$ lambert $+ I_s\, \rho_s$ x phong^f**
  - **Lambert = max[(s·m)/(|s||m|), 0]**
  - **Phong = max[(h·m/(|h||m|), 0]**

- To add color, we use 3 separate total intensities one each for Red, Green, and Blue, which combine to give any desired color of light.
- We say the light sources have three types of color:
- **$I_r = I_{ar}\, \rho_{ar} + I_{sr}\, \rho_{dr}$ lambert $+ I_{sr}\, \rho_{sr}$ x phong^f (similarly for I_g, I_d)**
  - ambient = (I_{ar}, I_{ag}, I_{ab})
  - diffuse = (I_{dr}, I_{dg}, I_{db})
  - specular = $(I_{spr}, I_{spg}, I_{spb})$.

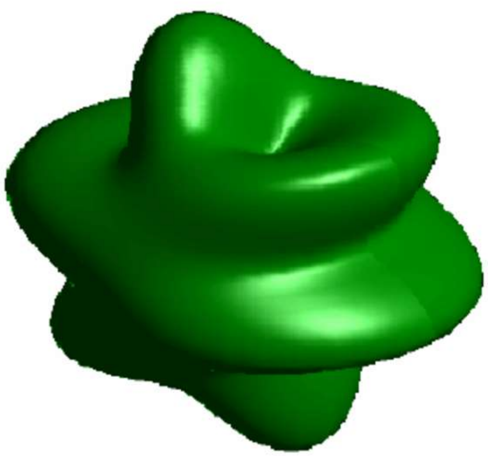
Light model : Simple to Complex

- **Example:**

Ambient Illumination



Ambient + Diffuse



Ambient + Diffuse + Specular

Ambient

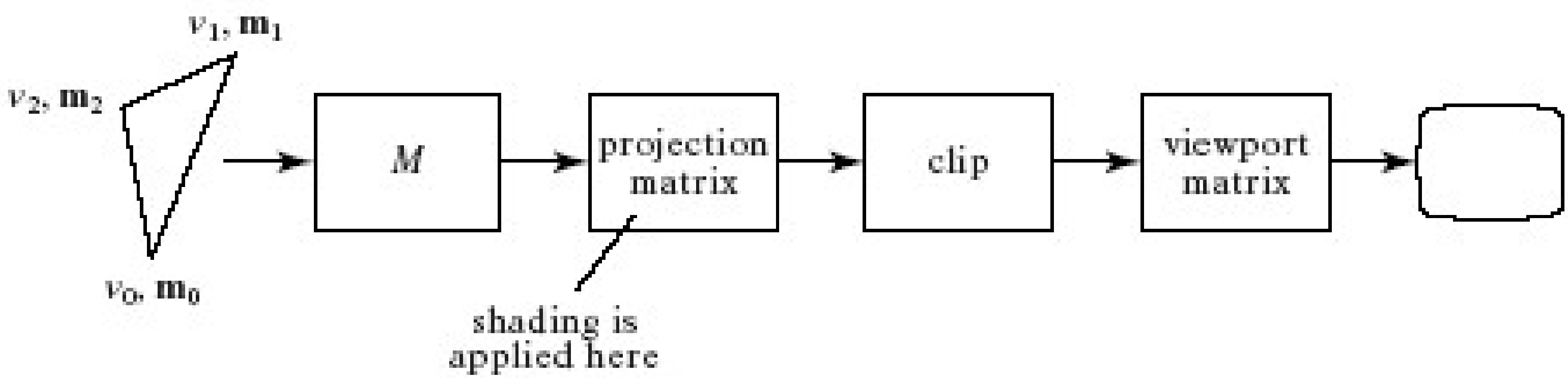
Ambient + Diffuse

Ambient + Diffuse + Specular

Shading and Graphics Pipeline

Shading and the Graphics Pipeline

- Shading is applied to a vertex at the point in the pipeline where the projection matrix is applied.
- We specify a normal and a position for each vertex.



Shading and the Graphics Pipeline (2)

- glNormal3f (norm[i].x, norm[i].y, norm[i].z) specifies a normal for each vertex that follows it.
- The modelview matrix M transforms both vertices and normals (**m**), the latter by $M^{-T}\mathbf{m}$.
- M^{-T} is the transpose of the inverse matrix M.
- The positions of lights are also transformed.
- OpenGL allows to specify various light sources and their locations.

Shading and the Graphics Pipeline (3)

- Then a color is applied to each vertex, the perspective transformation is applied, and clipping is done.
- Clipping may create new vertices which need to have colors attached, usually by linear interpolation of initial vertex colors.
- Suppose color at v_0 (r_0, g_0, b_0) and v_1 (r_1, g_1, b_1):
- If the new point a is 40% of the way from v_0 to v_1, the color associated with a is a blend of 60% of (r_0, g_0, b_0) and 40% of (r_1, g_1, b_1):
- $color\ at\ point\ a = (lerp(r_0, r_1, 0.4), lerp(g_0, g_1, 0.4), lerp(b_0, b_1, 0.4))$

Shading and the Graphics Pipeline (4)

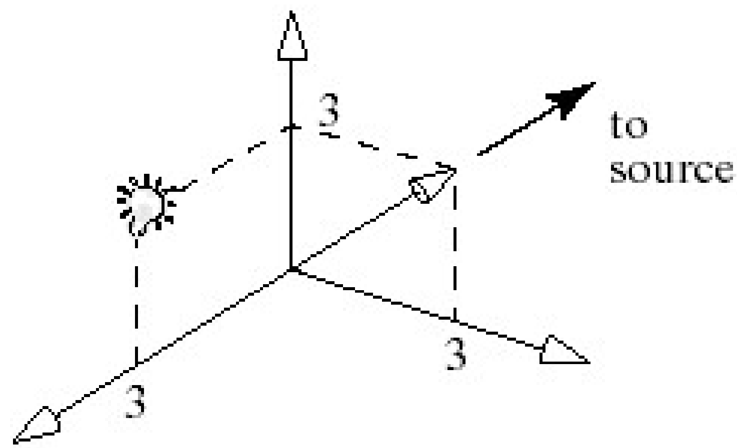
- The vertices are finally passed through the viewport transformation where they are mapped into screen coordinates (along with pseudodepth, which now varies between 0 and 1).
- The quadrilateral is then rendered (with hidden surface removal).

Creating and Using Light Sources in Open-GL

- OpenGL allows to define up to eight sources
- Light sources are through number in [0, 7]: GL_LIGHT_0, GL_LIGHT_1, etc.
  - Each light has a position specified in homogeneous coordinates using a GLfloat array named litepos, for example,
    - GLfloat litePos[]={3.0,6.0,5.0,1.0}
- The light is created using
  - glLightfv (GL_LIGHT_0, GL_POSITION, litePos);
- If the position is a vector (4th component = 0), the source is infinitely remote (like the sun).

Point and Vector Light Locations

- The figure shows a local source at (0, 3, 3, 1) and a remote source “located” along vector (3, 3, 0, 0).
- Infinitely remote light sources are often called “**directional**”.
- There are computational advantages to use directional light sources.
- since direction **s** in the calculations of diffuse and specular reflections is *constant* for all vertices in the scene.
- But directional light sources are not always the correct choice.
- some visual effects are properly achieved only when a light source is close to an object.



Creating and Using Light Sources in OpenGL(2

- Arrays are defined to hold the colors emitted by light sources and are passed to glLightfv
- The light color is specified by a 4-component array [R, G, B, A] of GLfloat, named (e.g.) amb0.
- The A (alpha: used to blend two colors on the screen) value can be set to 1.0 for now.
  - Glfloat amb0[]={0.2,0.4,0.6,1.0}; similar for diff0[],spec0[];
- The light color is specified by

```
glLightfv (GL_LIGHT_0, GL_AMBIENT, amb0);
```

Similar statements specify GL_DIFFUSE and GL_SPECULAR.

Creating and Using Light Sources in OpenGL(3

Default values:

- For all sources: default ambient = (0,0,0,1) – dimmest possible: black.
- For light source LIGHT0:
- default diffuse = (1,1,1,1) – brightest possible : white.
- Default specular = (1,1,1,1) – brightest possible: white.
- For all other light sources, diffuse and specular values have default black

Creating and Using Light Sources in OpenGL (4)

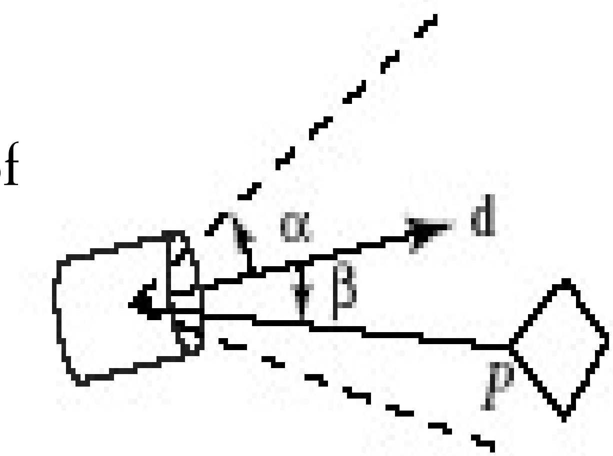
- Lights do not work unless you turn them on.
  - In your main program, add the statements
    - **glEnable (GL_LIGHTING);**
    - **glEnable (GL_LIGHT_0);**
  - If you are using other lights, you will need to enable them also.
- To turn off a light,
    - **glDisable (GL_LIGHT_0);**
- To turn them all off,
    - **glDisable (GL_LIGHTING);**

Creating an Entire Light

```
GLfloat amb0[ ] = {0.2, 0.4, 0.6, 1.0};
   // define some colors
GLfloat diff0[ ] = {0.8, 0.9, 0.5, 1.0};
GLfloat spec0[ ] = { 1.0, 0.8, 1.0, 1.0};
glLightfv(GL_LIGHT0, GL_AMBIENT, amb0);
   // attach them to LIGHT0
glLightfv(GL_LIGHT0, GL_DIFFUSE, diff0);
glLightfv(GL_LIGHT0, GL_SPECULAR, spec0);
```

Creating and Using Spotlights in Open-GL

- Light sources are point sources emit light uniformly in all directions.
- OpenGL allows to make into spotlights
- A spotlight emits light only in a restricted set of directions;
- The spotlight is aimed in **direction d**, with a cut-off angle alpha.



- There is no light outside the cone. Inside the cone,

$$I = I_s(\cos\beta)^{\varepsilon}, \text{ where}$$

β is the angle between **d** and a line from the source to *P* and

ε is chosen by the user to give the desired falloff of light with angle.

$(I_s(\cos\beta)^{\varepsilon})$ - attenuation of light when reaching P.

Creating and Using Spotlights in OpenGL (2)

- To create the spotlight, create a GLfloat array for **d**.
- Default values are **d** = {0, 0,-1}, $\alpha = 180^o$, $\varepsilon = 0$: a point source.
- Spotlight parameters can be set by adding the statements
  - glLightf (GL_LIGHT_0, GL_SPOT_CUTOFF, 45.0); (45.0 is α in degrees)
  - glLightf (GL_LIGHT_0, GL_SPOT_EXPONENT, 4.0); (4.0 is ε)
  - GLfloat d[ ]={2.0,1.0,-4.0};
  - glLightfv (GL_LIGHT_0, GL_SPOT_DIRECTION, d);
  - glLightf->to set single value, glLightfv->to set a vector

Attenuation of Light with Distance

- OpenGL also allows you to specify how rapidly light diminishes with distance from a source.
- OpenGL attenuates the strength of a positional light source by the following attenuation factor:

$$atten = \frac{1}{k_c + k_l D + k_q D^2}$$

- where k_c, k_l, and k_q are coefficients and D is the distance between the light's position and the vertex in question.
- The expression helps to model constant, linear and quadratic (inverse square law) dependence on distance from a source.

Attenuation of Light with Distance (2)

- These parameters are controlled by function calls:
- glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 2.0);
- and similarly for GL_LINEAR_ATTENUATION, and GL_QUADRATIC_ATTENUATION.
- The default values are $k_c = 1$, $k_l = 0$, and $k_q = 0$ (no attenuation). Which eliminate any attenuation

Changing the OpenGL Light Model

- **3 parameters** to be set that specify general rules for applying the lighting model.
- **The color of global ambient light**:
  - We can establish a gobal ambient light source in a scene that is independent of any sorce :

    GLfloat amb[ ] = {0.2, 0.3, 0.1, 1.0};

    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, amb);
- **Is the viewpoint local or remote?**
  - OpenGL computes specular reflections using the "halfway vector" **h** = **s** + **v** .
  - The true directions **s** and **v** are normally different at each vertex in a mesh.
  - If light source is directional s is constant. But v still varies from vertex to vertex
  - To use the true value of **v** for each vertex, execute

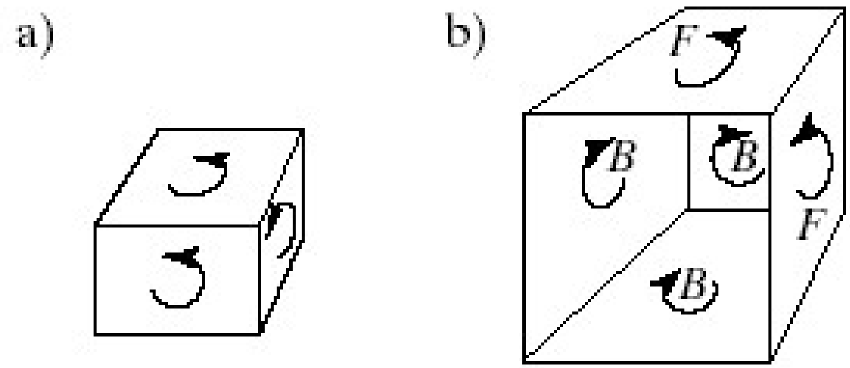
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);

Changing the OpenGL Light Model (2)

- **Are both sides of a polygon shaded properly?**
  - Each polygonal face in a model has two sides. When modeling, we tend to think of them as the "inside" and "outside" surfaces.
  - The convention is to list the vertices of a face in counter-clockwise (CCW) order as seen from outside the object.
- OpenGL has no notion of inside and outside. It can only distinguish between "front faces" and "back faces".
- A face is a front face if its vertices are listed in counter-clockwise (CCW) order as seen by the eye.

Changing the OpenGL Light Model (3)

- For a space-enclosing object (a) all visible faces are front faces; OpenGL draws them properly with the correct shading.
- If a box has a face removed (b), OpenGL does not shade back faces properly. To force OpenGL to shade back faces, use:
- glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);



Lightning :Light Source

- Control light position and direction
  - OpenGL treats the position and direction of light source just as it treats the position of geometric primitives
  - MODELVIEW transformation is applied.
- Three types of control
  - A light position that remains fixed
  - A light that moves around the stationary object
  - A light that moves along the view point

Controlling the Light's Position (Light stationary:)

- **glModelMatrixMode(GL_MODELVIEW)**
- **glLoadIdentity();**

- modeling and viewing here
  - GLfloat position[]={3.0,6.0,5.0,1.0}
  - glLightfv(GL_LIGHT0, GL_POSITION, position)

Moving Light Sources in OpenGL

- To move a light source independently of the camera:
  - set its position array,
  - clear the color and depth buffers,
  - set up the modelview matrix to use for everything except the light source and push it
  - move the light source and set its position
  - pop the matrix
  - set up the camera, and draw the objects.

Code: Independent Motion of Light

```
void display()
{  GLfloat position[ ] = {2, 1, 3, 1}; //initial light position
                        // clear color and depth buffers
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glPushMatrix();
           glRotated(…);   // move the light
           glTranslated(…);
           glLightfv(GL_LIGHT0, GL_POSITION, position);
   glPopMatrix();

gluLookAt(…); // set the camera position
<.. draw the object ..>
glutSwapBuffers(); }
```

Controlling the Light's Position
Move light with viewpoint:

key: specify light position in eye coordinates before viewing transf.

```
GLfloat position[] = {0, 0, 0, 1}
 glModelMatrixMode(GL_MODELVIEW)
glLoadIdentity();
glLightfv(GL_LIGHT0, GL_POSITION, position)
gluLookAt( …)
    draw object()
```

COLOR MODELS

OverView

- Color model
- Visible light Spectrum
- Color terminology
- Energy Spectrum
- Additive & Subtractive Mixing
- CIE standard
- RGB color model
- CMY color model (also, CMYK)
- HSV color model
- HLS color model

COLOR MODELS

- A color model is a method for explaining the properties or behavior of color within some particular context.
  - Mathematical model in which a color is represented as numbers.
  - Forms a 3D coordinate system and each point represents a color
- No single color model explains all aspects of color, so different models are used to describe the different perceived characteristics of color

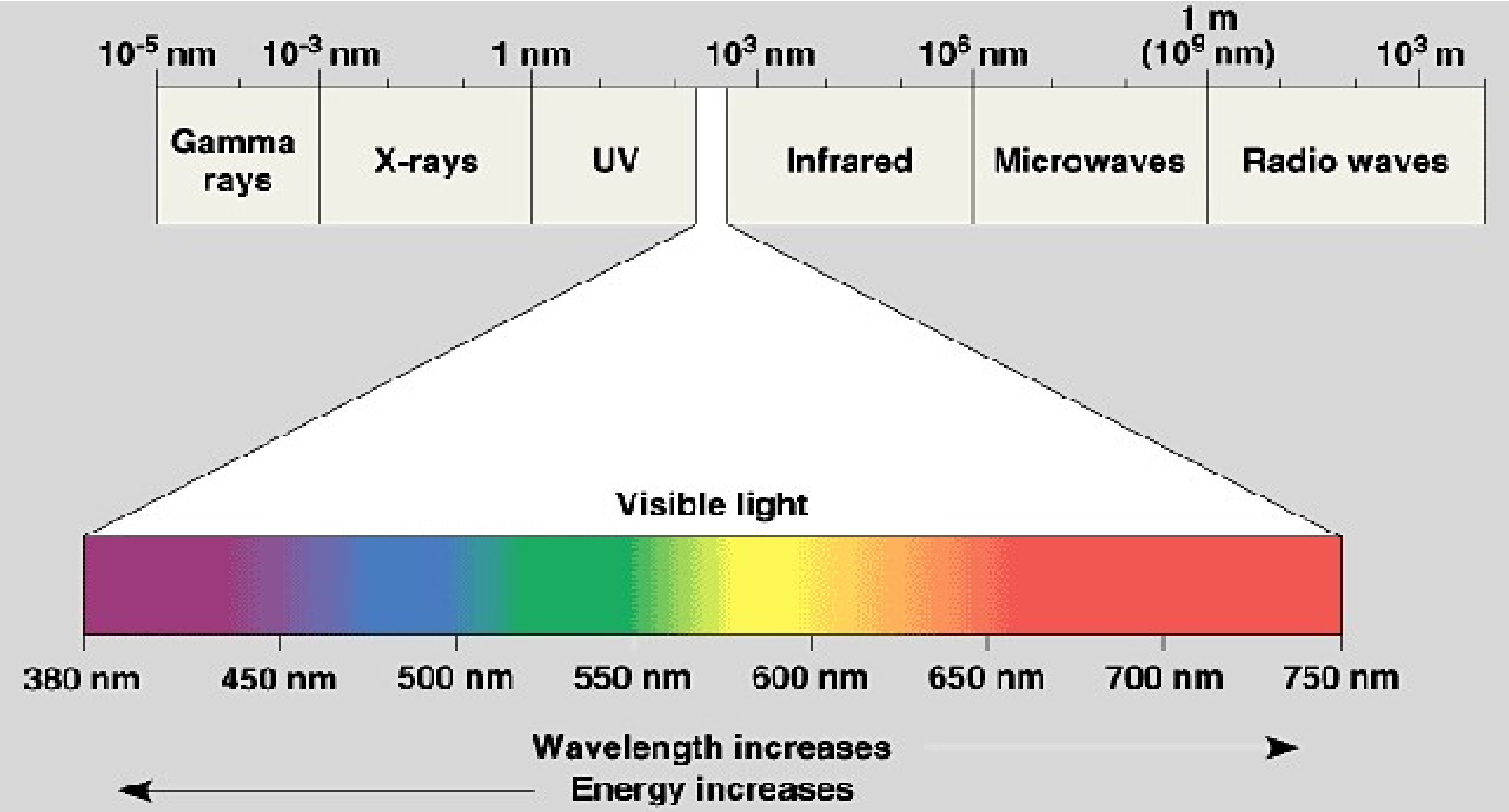
Color

- Light is a narrow frequency band within the electromagnetic spectrum (380-750 nm).
- Each frequency value within the visible band corresponds to a distinct color.
- At low frequency end is red color and highest frequency is violet color.
- The various colors are described in terms of either frequency f or wavelength λ of electromagnetic wave.
- The colors that we see in the world around us are generally not pure colors consisting of a single wavelength.

Color

- The combination of frequencies present in the reflected light determines what we perceive as the color of the object.
- Rather, color sensation results from the *dominant wavelength* of the light reflecting off or emanating from an object.
- The dominant frequency is called as HUE .

Color



1 m
10^{-5} nm
10^{-3} nm
1 nm
10^3 nm
10^6 nm
(10^9 nm)
10^3 m
Gamma rays
X-rays
UV
Infrared
Microwaves
Radio waves
Visible light
380 nm
450 nm
500 nm
550 nm
600 nm
650 nm
700 nm
750 nm
Wavelength increases
Energy increases

Color Terminology

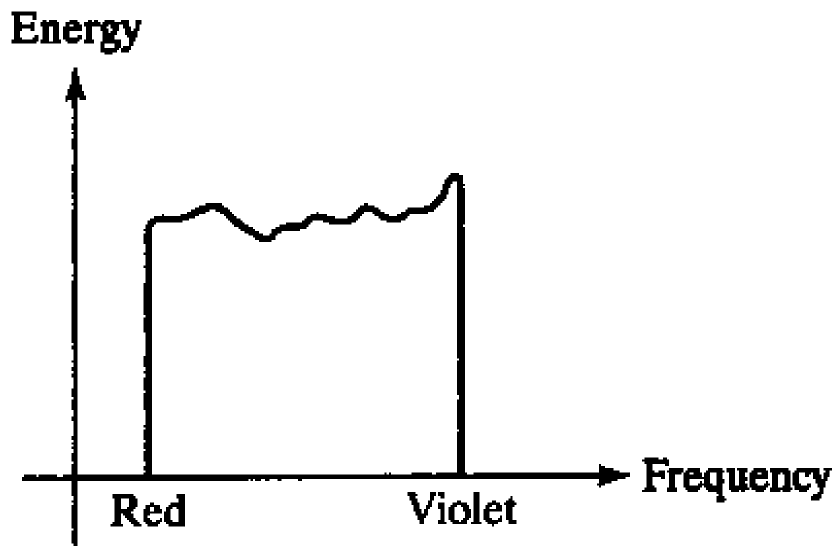
- **Hue** – the **dominant frequency** is called hue or simply color
- **Monochromatic color** – a color that is created from only one wavelength.
- **Brightness or Luminance** : perceived intensity of light.
- **Purity or saturation** : Describes how "pure" the color of light appears. *Saturation* is a matter of how much white light is added in. The less white light, the more saturated the color.(how strong a color is)
- ***Lightness*** is how much black is in the color.
- **Chromaticity**: Refers to two properties of color characteristics purity and dominant frequency
- Hue and saturation are elements of *chrominance*. Lightness is a matter of *luminance*.

Color Terminology

- **Additive color systems** – based on adding colored light (as in computer monitors). A combination of all colors gives white.
- **Subtractive color systems** – based on adding pigments (as in printing). A combination of all colors gives black.

Physical properties of light

- Energy emitted by a white-light source has a distribution over the visible frequencies as shown
- The distribution showing the relation between energy and wavelength (or frequency) is called *energy spectrum.*
- *Each frequency component from red to violet contributes more or less equally to total energy.*
- *The color of source are descibred as white.*

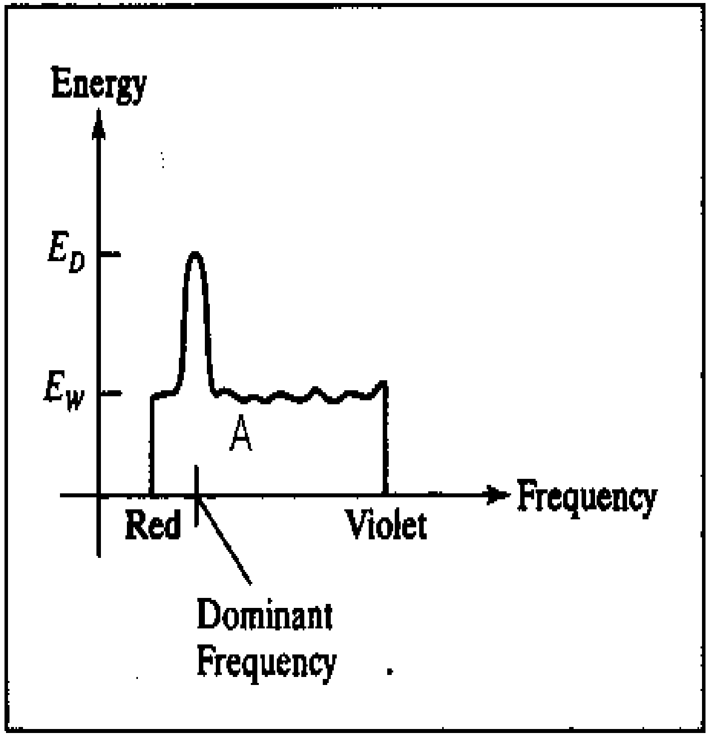


Physical properties of light

- *The light has color corresponding to the dominant frequency*

This distribution may indicate:

1) a <u>dominant wavelength</u> (or frequency) which is the color of the light (*hue*), E_D
2) Contributions from the other frequencies produces white light of energy density E_w
3) <u>brightness</u> (luminance), intensity of the light (*value*), is the area *A* under curve.
4) <u>purity</u> (*saturation*), E_D - E_W



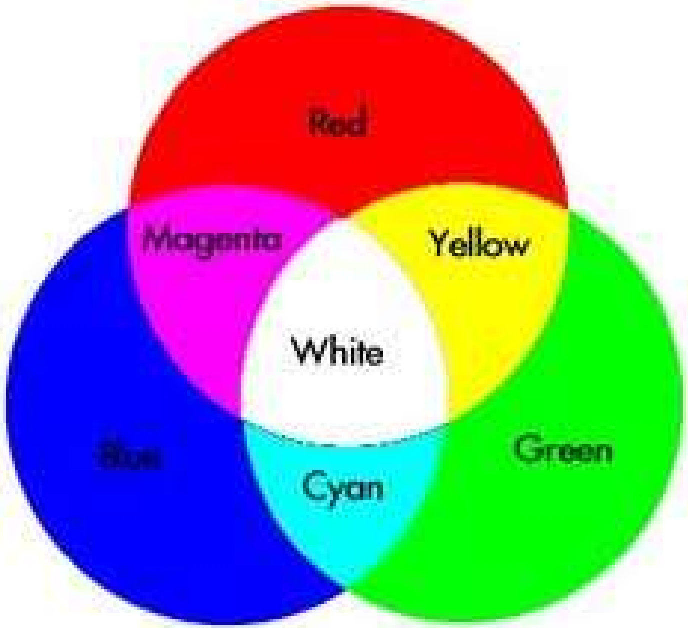
Energy spectrum for a light source with a dominant frequency near the red color

Color definitions

- Two different color light sources with suitable chosen intensities can be used to produce other range of colors.
- ***Complementary colors*** - two colors combine to produce white light.
  - Eg: red and cyan, green and magenta, blue and yellow
- **Color Gamut:**color models used to describe combination of light, in terms of hue, use three colors to obtain wide range of colors.
- ***Primary colors*** - (two or) three colors used for describing other colors
- Two main principles for mixing colors:
  - *Additive mixing & Subtractive mixing*

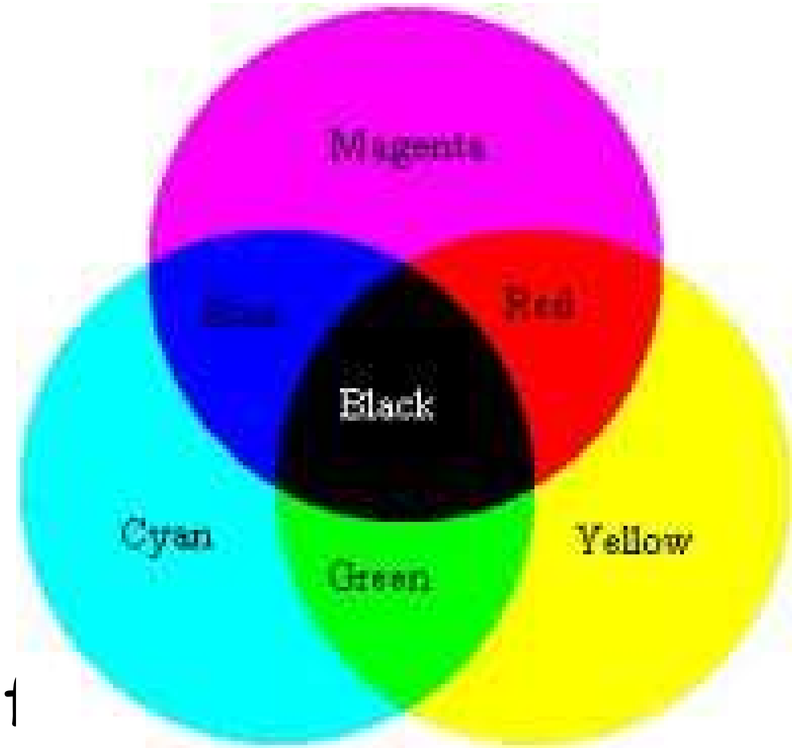
Additive mixing

- This system express a color D, as the sum of certain amount of primaries.
- Overlapping gives yellow, cyan, magenta and white
- Typical technique used on color displays

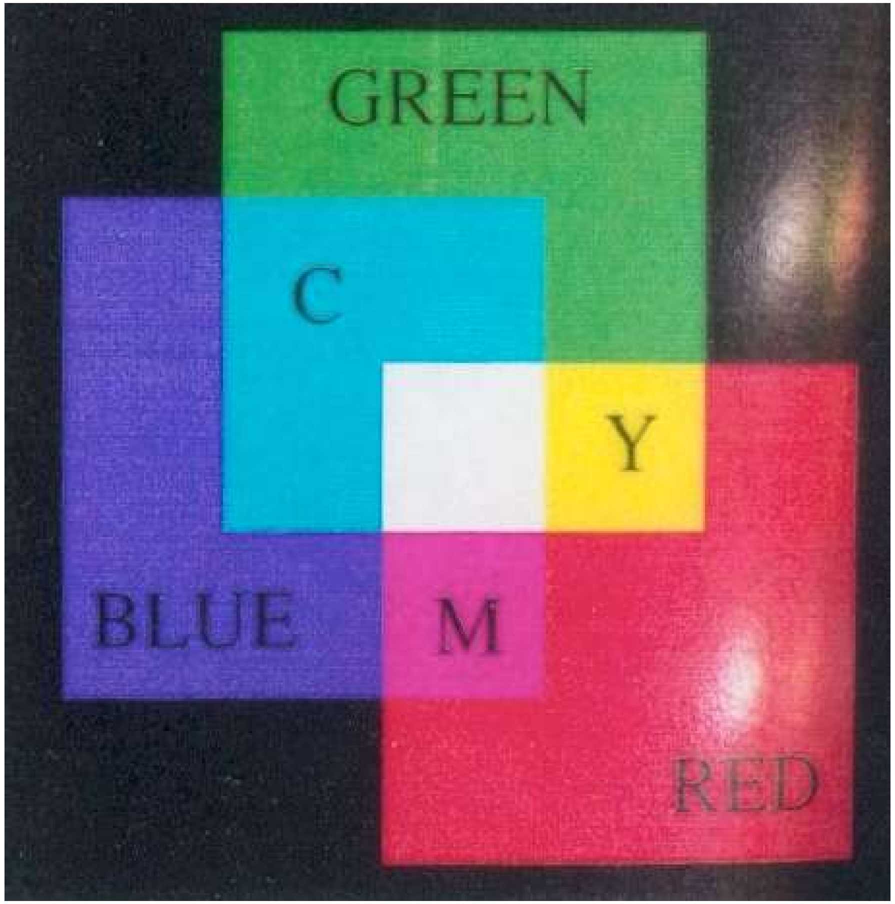


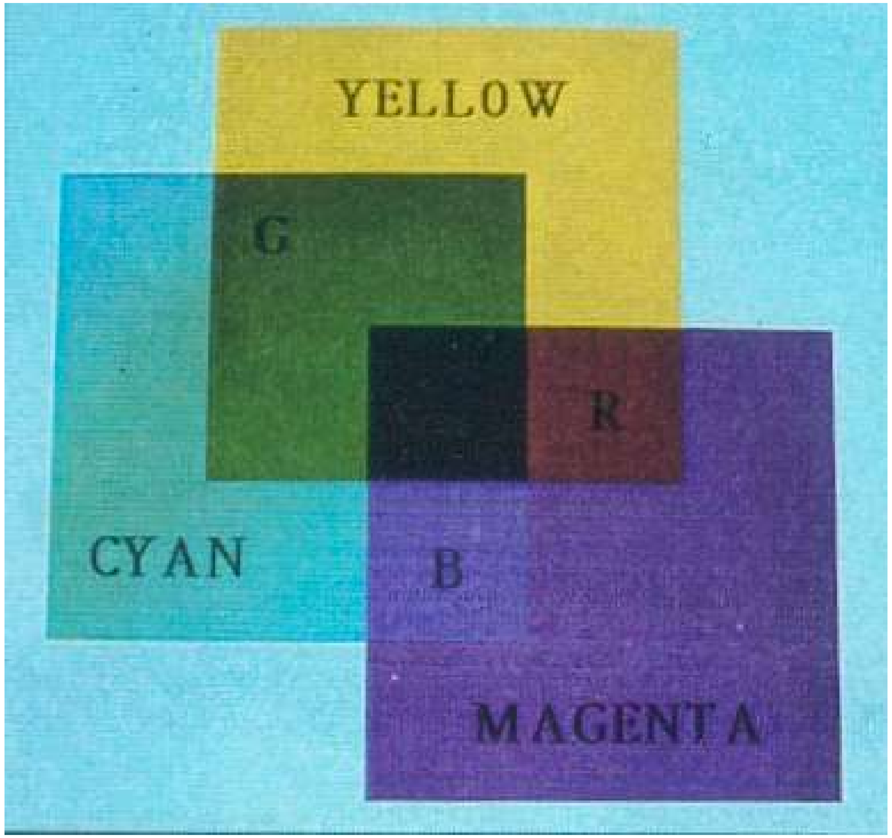
Subtractive mixing

- Color pigments are mixed directly in some liquid, e.g. Ink
- Primary colors: **cyan, magenta and yellow**, i.e. CMY
- The typical technique in printers/plotters
- This system expresses a color, D by means of three tuple in which each of the three values specifies how much of certain color to remove from white in order to produce D.



Additive/subtractive mixing





Overview of color models

The human eye can perceive about 382000(!) different colors

Some kind of classification system is necessary; all models use three coordinates as a basis:

1) CIE standard (XYZ model)
2) RGB color model
3) CMY color model (also, CMYK)
4) YIQ color model
5) HSV color model
6) HLS color model

CIE Color Primaries

- The CIE (International Commission on Illumination) color primaries is referred as X, Y, and Z .
- X, Y, and Z are "artificial primaries,"(imaginary) not visible colors like R, G, and B.
  - Just a hypothetical model; to make it machine independent
- These primaries can be combined in various proportions to produce all the colors the human eye can see.
- In the CIE color model ,a color **C** is given by

  C = X1***X** + Y1***Y** + Z1***Z**

  XYZ – Vectors in color space X1Y1Z1 – amt of standard primaries needed to match C

CIE Color Model on the X+Y+Z = 1 Plane

- If we want to consider each component as a percentage of the total amount of light, we can "normalize" the values:

$$x = \frac{X}{X+Y+Z} \qquad (1)$$

$$y = \frac{Y}{X+Y+Z} \qquad (2)$$

$$z = \frac{Z}{X+Y+Z} \qquad (3)$$

Note: X + Y + Z is the luminance
Also note that x + y + z = 1

CIE Color Model on the X+Y+Z = 1 Plane

- x, y represent chromaticity values and depend on hue and purity.
- If we specify colors only with x and y values, we cannot obtain the amounts X, Y, and Z
- So, for complete description of any color, we need x, y & Y.

$$(X, Y, Z) = \left(\frac{xY}{y}, Y, \frac{(1 - x - y)Y}{y} \right).$$

(1) *can be written as* $X = x\ (X + Y + Z)$

from (2) we know that, X+Y+Z = Y/y

Therefore, X = x (Y/y)

Similarly for Z

CIE Chromaticity Diagram

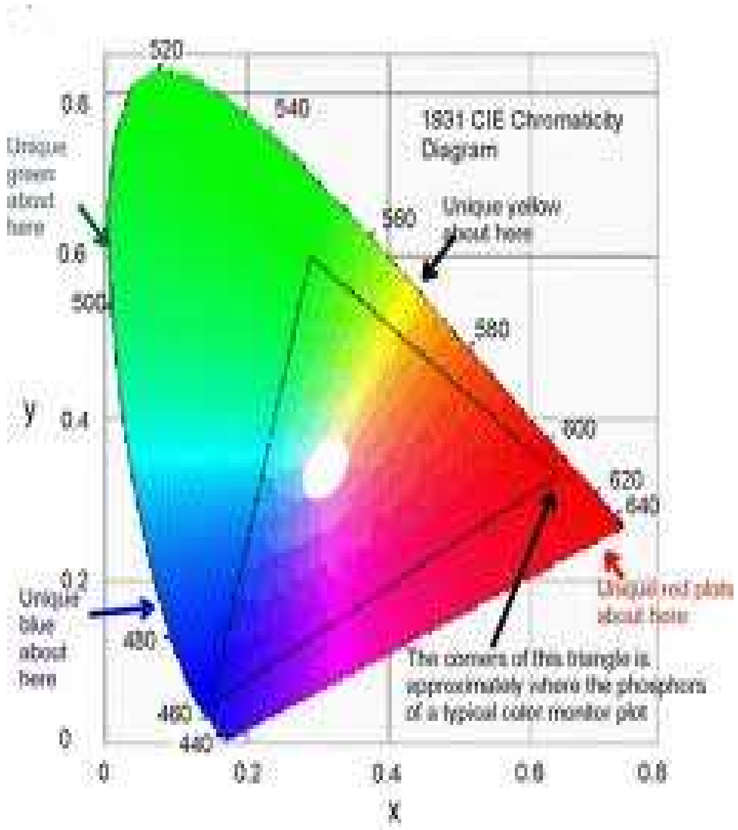
Plotting x vs. y for colors in the visible spectrum ,we obtain tongue shaped curve called *CIE Chromaticity diagram*

.Points along the curve are the 'pure' colors in the spectrum.

• The Line Joining red to violet is called *purple line*, and is not a part of the spectrum

• Interior points specify all the visible color combinations.

• The dot corresponds to white light position.



Chromaticity Diagram

- The CIE Chromaticity diagram is found useful in the following situations
  - Comparing *color gamuts* of different set of primaries
  - Identifying *complementary colors*
  - Determining *dominant wavelength* & *purity* of different colors

Dominant Wavelength on CIE Color Diagram

- Color Gamut are represented in the chromaticity diagram as straight line segments or polygons.
- To determine the dominant wavelength of a color C1, draw a line between C through C1 to intersect the spectral curve at Cs. The dominant wavelength is at Cs.
- The purity is given by the ratio of distance of C to C1 and distance of C to Cs.
- The closer C1 is to the perimeter, the more saturated the color.
- Dominant wavelength of C2 is Csp(compliment of Cp) – 'coz Cp is on the purple line which is not a part of visible spectrum

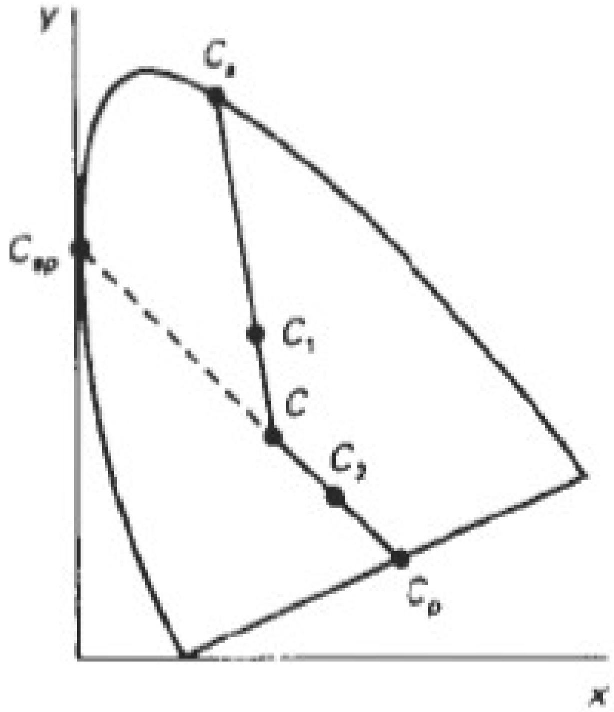
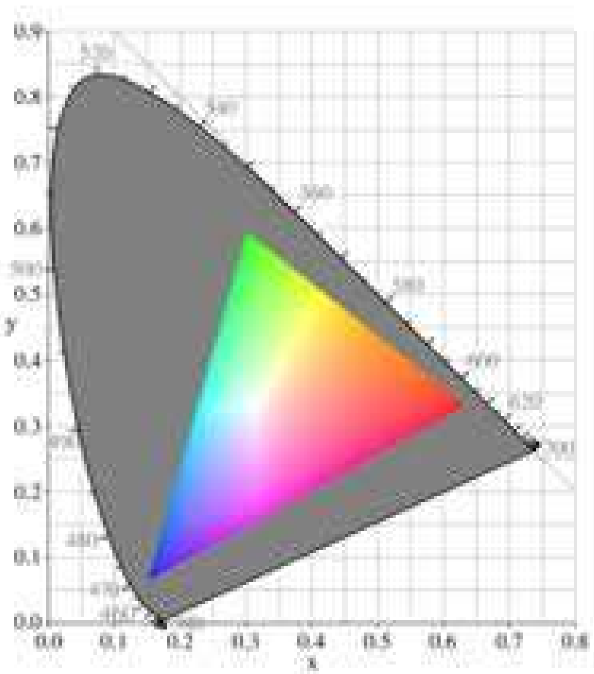
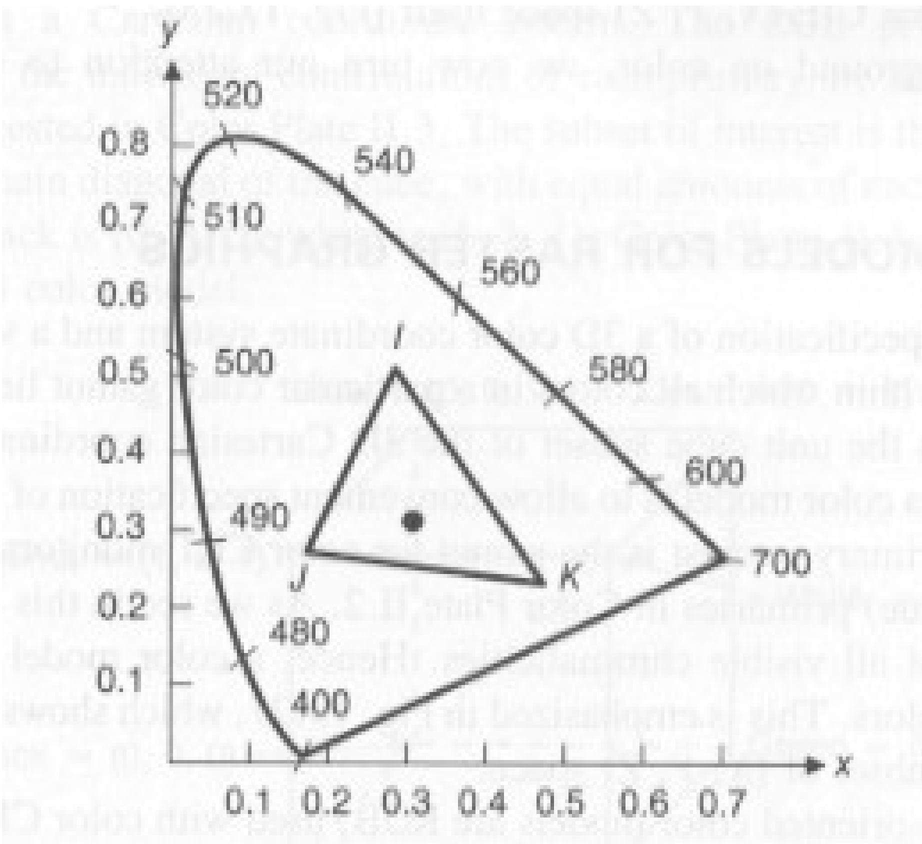


Figure 15-10
Determining dominant wavelength and purity with the chromaticity diagram.

Color Gamuts Represented on CIE Diagram

- All colors on the line IJ can be created by additively mixing colors I and J; all colors in the triangle IJK can be created by mixing colors I, J, and K.





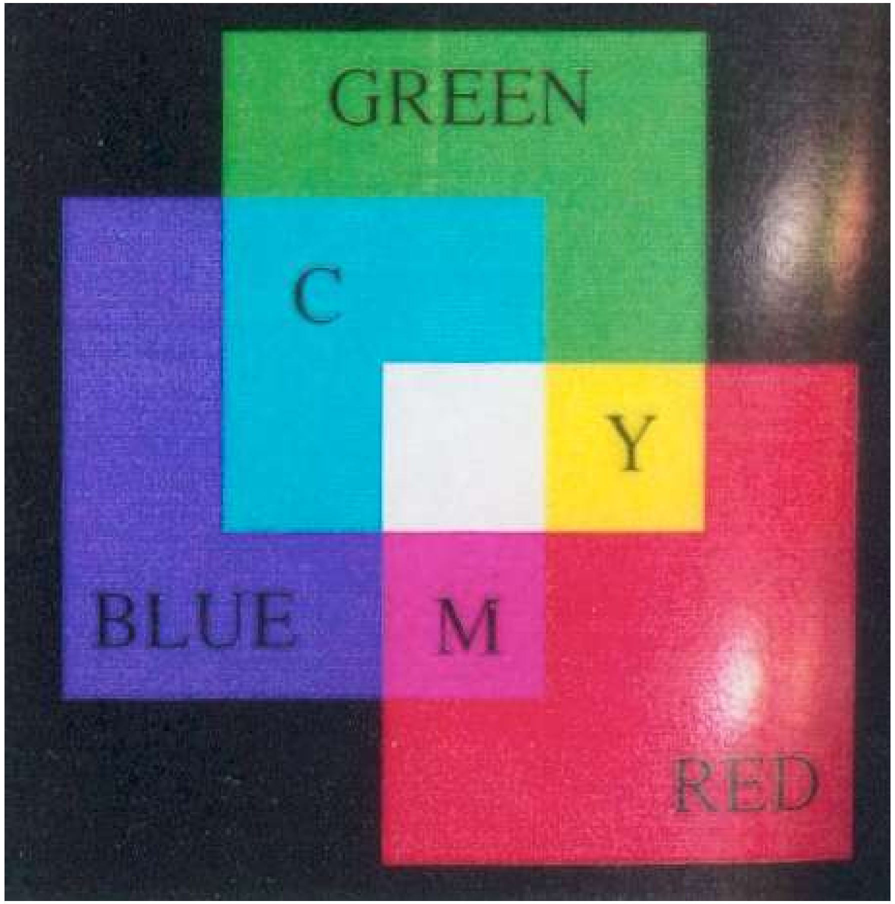
Color Concepts

- **An artist creates a color painting by mixing color pigments with white and black pigments**

- **Shades**: Pure color + Black pigment
- **Tints** : White Pigment + original color
- **Tones** : original color + Black +White pigments

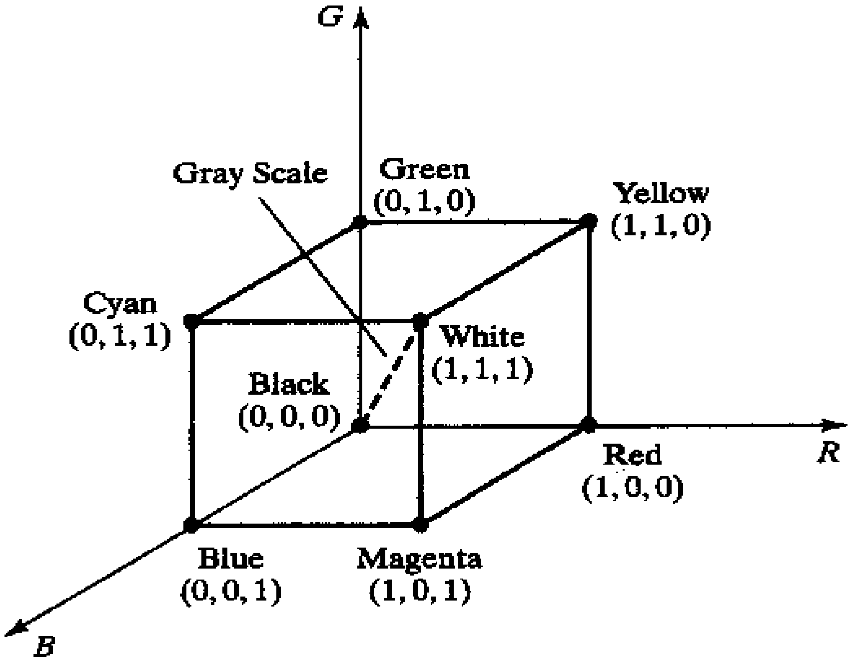
RGB model

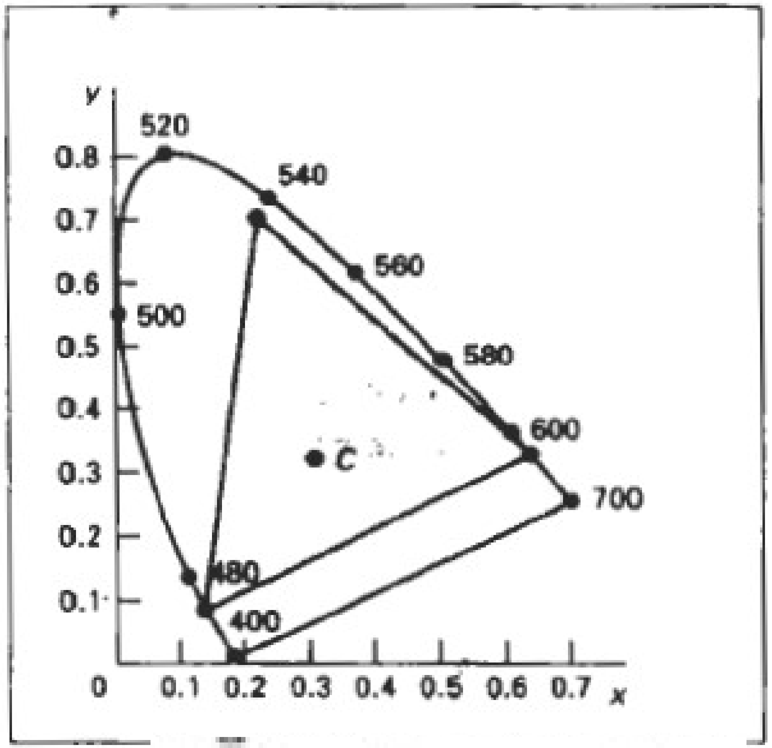
- All colors are generated from the three primaries R,G,B.
- various colors are obtained by changing the amount of each primary
- Additive mixing (r,g,b), $0 \le r,g,b \le 1$ referred to RGB color model.



RGB Model

- The *RGB unit cube defined with R,G,B axes. Each color point within the cube is given as (R,G,B)*
- *A color is expressed as C=R**R**+G**G**+B**B***
- Orgin=>black, 1,1,1=>white, .5,.5,.5=>gray
- Vertices of cube axes=> primary colors
- Other vertices=> complementary colors





RGB color gamut.

YIQ Colour Model

- Whereas an RGB monitor requires separate signals for the red, green, and blue components of an image, a television monitor uses a single composite signal
- The NTSC colour model for forming the composite video signal is the YIQ model. Same as XYZ model and used in television.
- Y represents the luminance IQ represents the hue and purity.
- A combination of red, blue and green are chosen for Y parameter to yield standard luminosity curve
- Since Y represents the luminance information black-white monitors use only the Y signal.
- I contains orange-cyan hue info (flesh-tone) Q contains green-magenta hue information.

YIQ Colour Model

- Conversion of RGB values to YIQ values

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

Calculated using the chromaticity coordinates of the RGB phosphor

- Conversion of YIQ values to RGB values can be done with the inverse matrix transformation

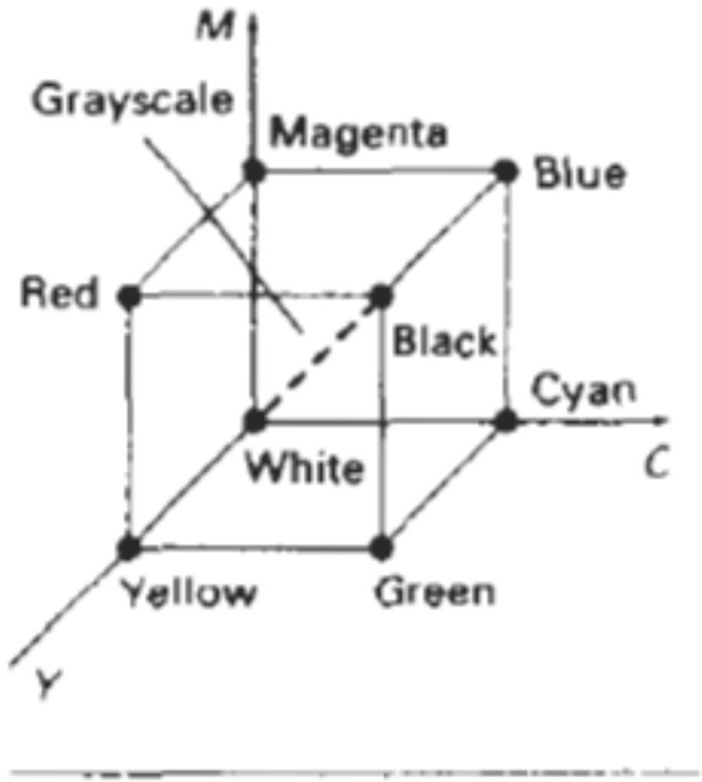
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.620 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.108 & 1.705 \end{bmatrix} \cdot \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

CMYK model

- CMYK is primarily a printing color model.
- Cyan, magenta, and yellow are called the subtractive primaries.
- The hard-copy devices produces color pictures by coating a paper with color pigment.
- Human eye - See the colors by reflected light which is a subtractive process.

CMYK Model

- Cyan, magenta, yellow, and black
- Cyan is white light with red taken out.
  C = G + B = W – R
- cyan can be formed by adding green and blue light. Therefore, when white light is reflected from cyan-colored ink, the reflected light must have no red component. That is red light is absorbed, or subtracted



CMYK Model

- Magenta is white light with green taken out.
  M = R + B = W - G
- Yellow is white light with blue taken out.
  Y = R + G = W – B
- 1,1,1 => black (since all components of incident light are subtracted)
- Orgin=>white
- CMY model generates a color point with a collection of four ink dots, like a RGB monitor uses a collection of three phosphor dots.
- Cyan, magenta, yellow and black dots (black dot coz cyan+magenta+yellow=dark gray instead of black)

CMYK vs. RGB

- RGB tO CMY conversion:

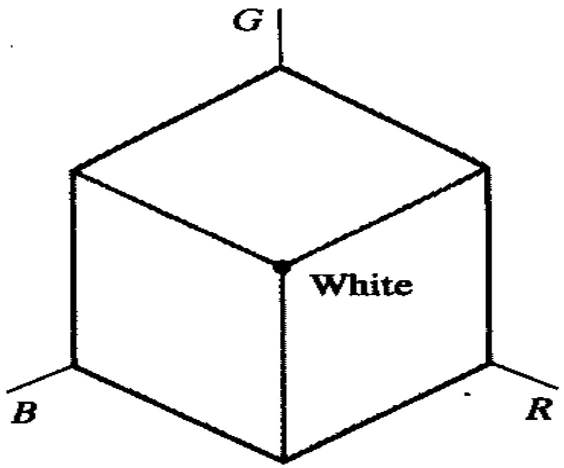
$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix}.$$

CMY to RGB conversion can be done with matrix transformation

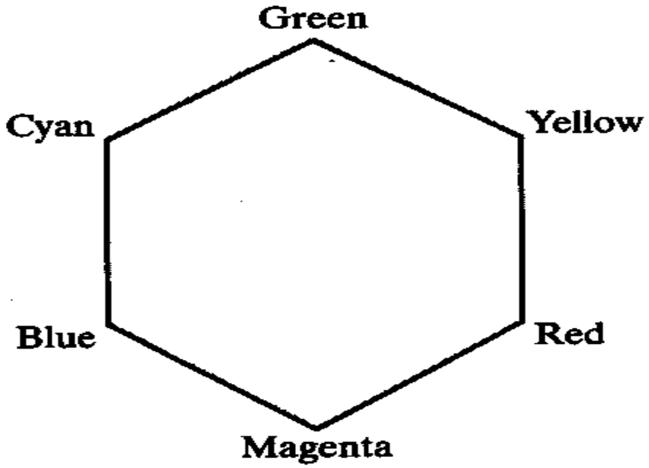
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

HSV model

- Uses color descriptions that are more intuitive to user. User selects a spectral color and then decides a shade, tint and tone
- HSV stands for Hue-Saturation-Value
- described by a hexcone derived from the RGB cube



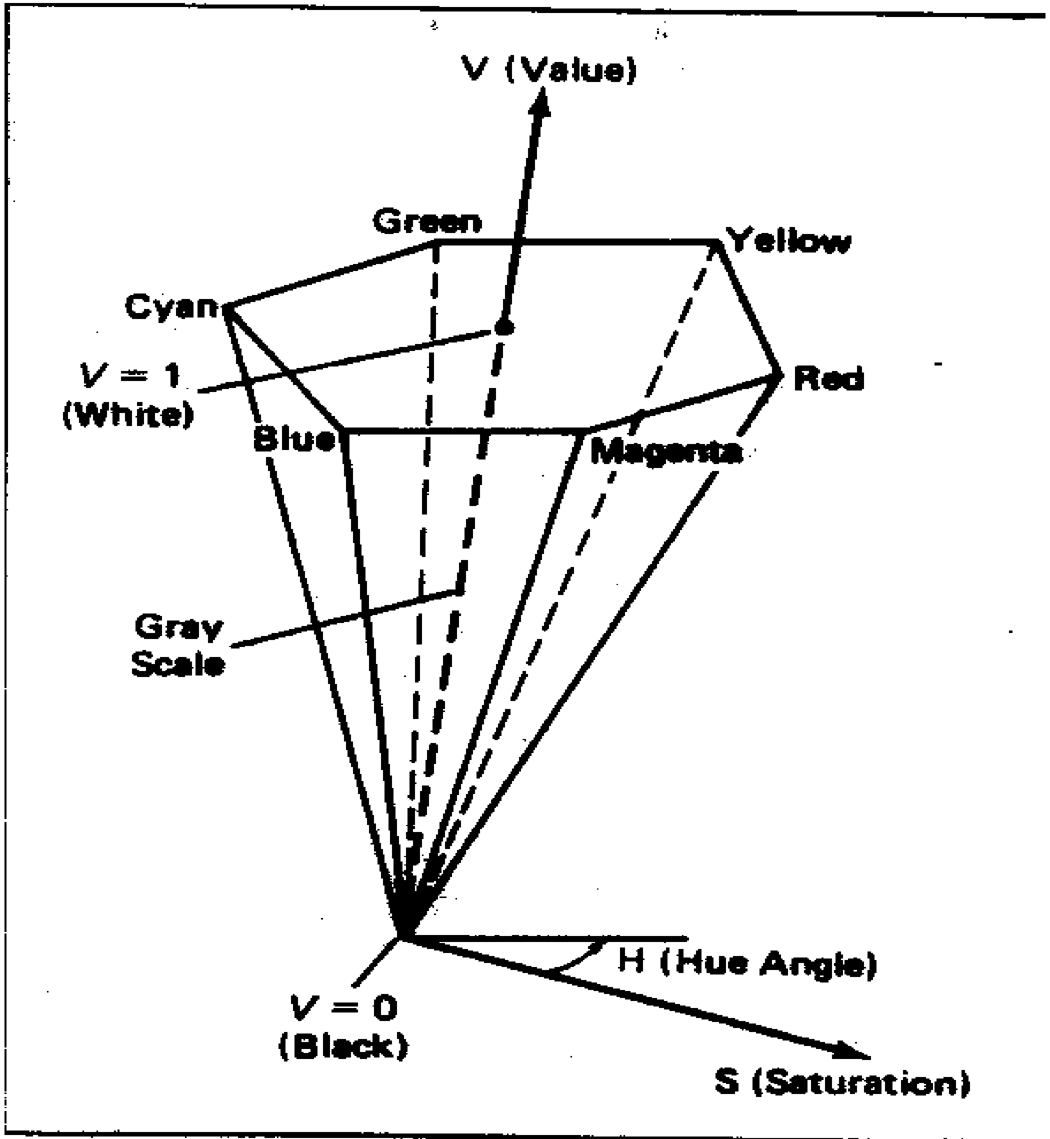
RGB Color Cube
(a)



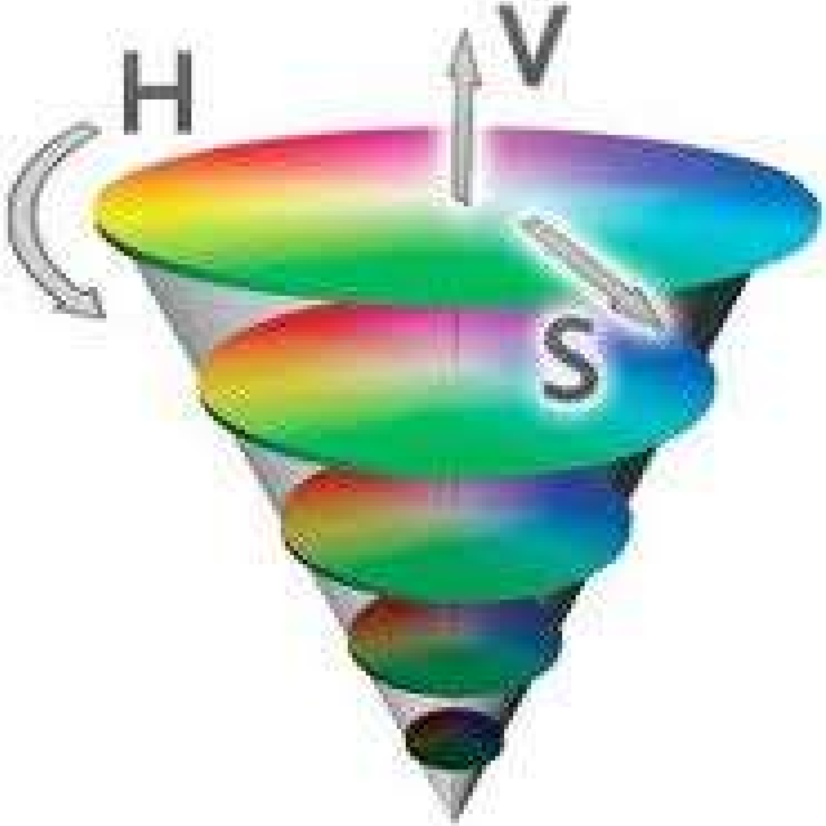
Color Hexagon
(b)

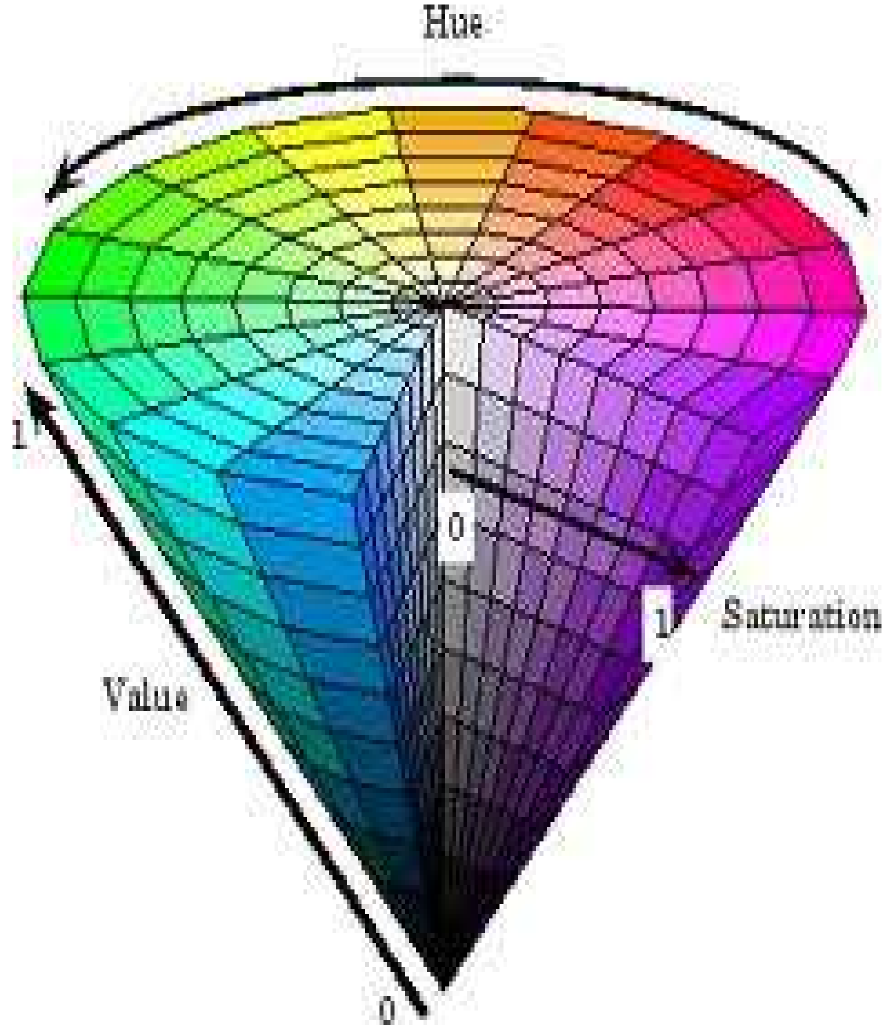
HSV model

- Hue (0-360°); "the color",
- Saturation (0-1); "the amount of white"
- Value (0-1); "the amount of black"
- Top of HSV hex cone is projection seen by looking along principal diagonal of RGB color



HSV Color Model





HSV color model

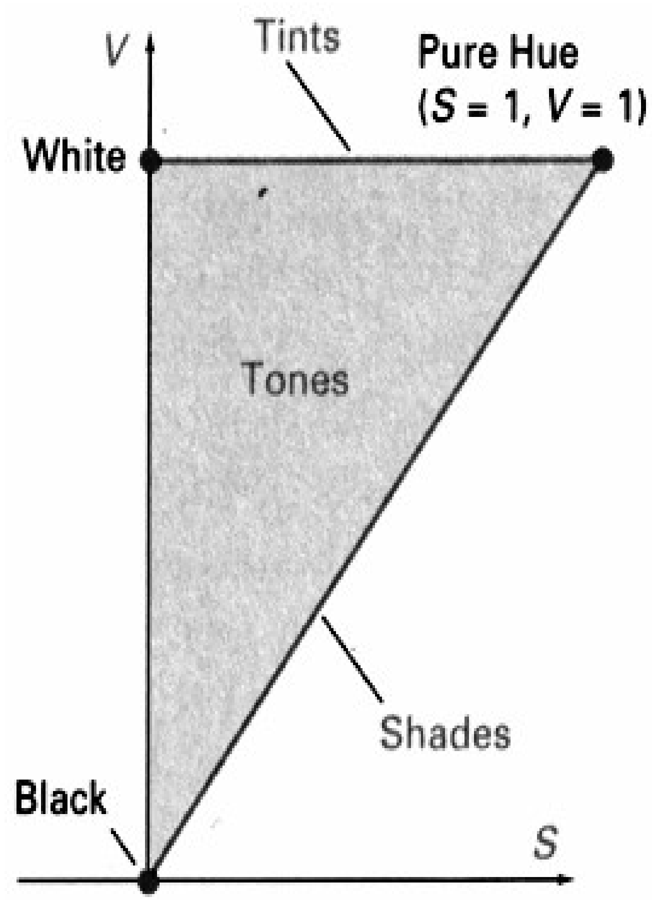
- Hue: ranges from 0° at red through 360°
- Vertices of the hexagon are separated by 60° intervals-Y at 60°, G at 120° etc
- Complementary colors 180° opposite
- Saturation S ranges from 0 to 1 – ratio of purity of a selected hue to its maximum purity at S=1.
- Value V varies from 0 at apex(black) to 1 at top(white).
- At
  - V=1 and S=1, pure hues
  - V=1 and S=0, white
  - V=0 and S=0 black

HSV Color model

- To get Dark Blue:
  - H=240, say V= 0.4 and S= 1
  - Adding black decreases V while S is constant
- To get Light Blue:
  - H=240, V=1 and say S=0.3
  - Adding white decreases S while V is constant

HSV Color Definition

- Cross section of the HSV hex cone showing regions for shades, tints, and tones.
- Shades: S=1 $0 \leq V \leq 1$
- Tints: V=1 $0 \leq S \leq 1$

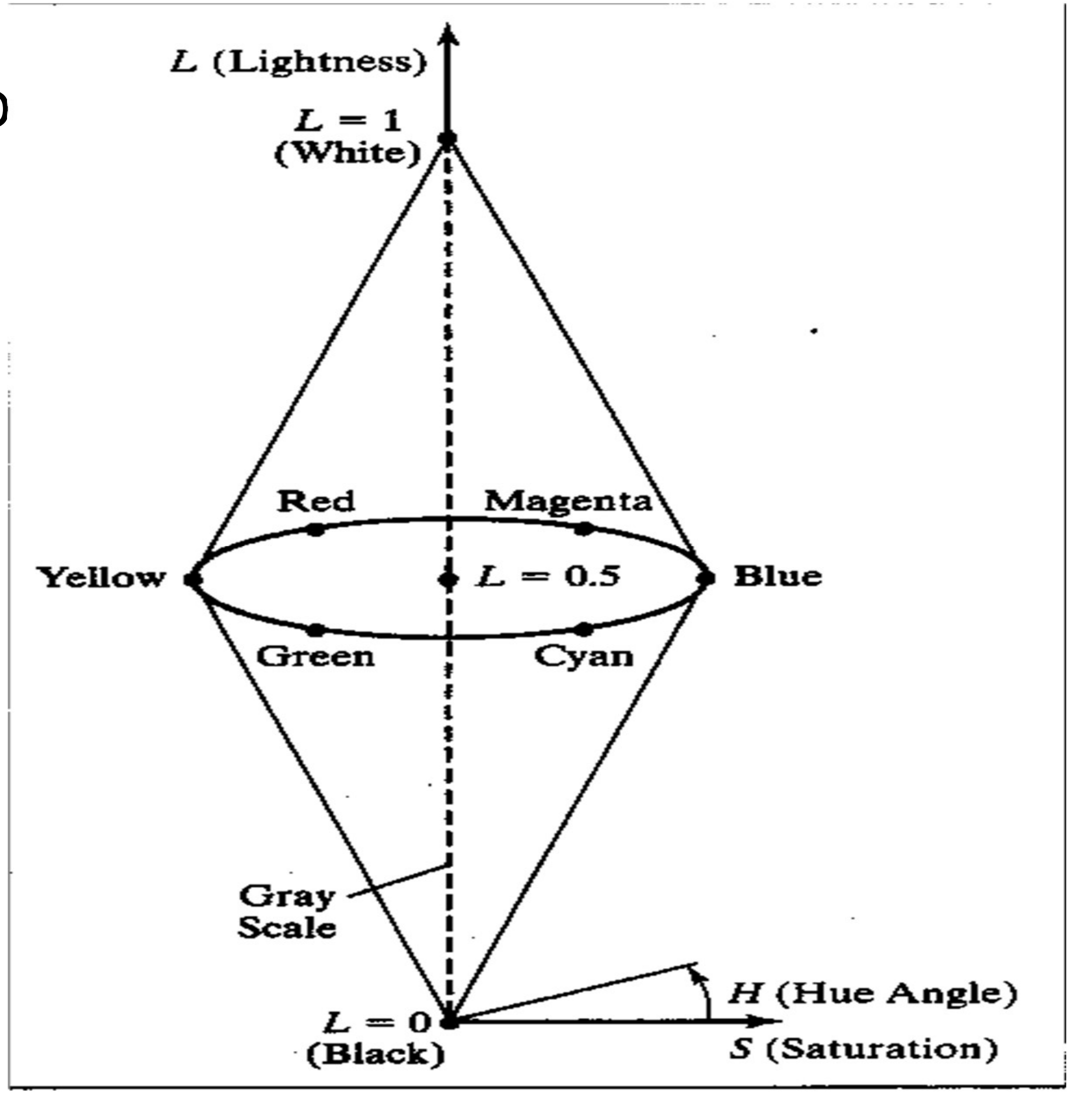


HLS model

Another model similar to HSV

L stands for *Lightness*

- color components:
- hue (H)∈ [0°, 360°]
- lightness (L)∈ [0, 1]
- saturation (S)∈ [0, 1]



Color Models Summary

- **CIE-XYZ: standard color description**
- **RGB: for monitors**
- **CMY, CMYK: for printers**
- **HSV, HLS: for user interfaces**
- **YIQ: for television (NTSC)** (Y=luminance, I=R-Y, Q=B-Y)

Animations

Animations

- **Computer animation generally refers to any time sequence of visual changes in a scene.**
- Computer generated animation could display time variations in
  - object size, color, transparency or surface texture.
- Computer animations can be generated by changing
  - camera parameters, such as position, orientation and focal length.
- Computer animations are produced by changing
  - lighting effects or other parameters
  - procedures associated with illumination and rendering.

Design of Animation Sequences

- Animation sequence is designed with the following steps.
  - Storyboard Layout
  - Object Definitions
  - Key-Frame specifications
  - Generation of in-between frames.

1.Storyboard Layout

• Outline of the action.

• Defines the set of basic events that are to take place.

• Story board consists of rough sketches or basic ideas for motion.

• Storyboard is divided into scene segments.

• Animators and Mentors decide which segments each animator will work on.

• Segments are reviewed and revised.

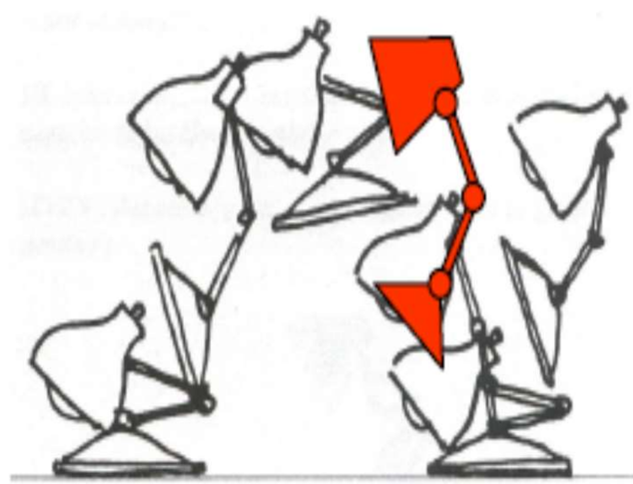
• Dialog is created based on storyboard and segments.

2.Object Definition

- Object definition is given for each participant in the action.
- In simple manual systems, the objects can be simply the artist drawings
- In computer-generated animations, models are used
- Objects can be defined with basic shapes such as polygons or splines.
- The associated movements along with the shape are also specified.
- Examples of models:
  - a "flying logo" in a TV ad
  - a walking stick-man

3.Key Frame Specifications

- Define character poses at specific time steps called “keyframes”
- Some key frames are chosen at extreme or characteristic positions in the action.
- Others are spaced so that the time interval between key frames is not too large.
- More key frames are specified for intricate motions than for simple and slowly varying motions.



4.In-Between Frames

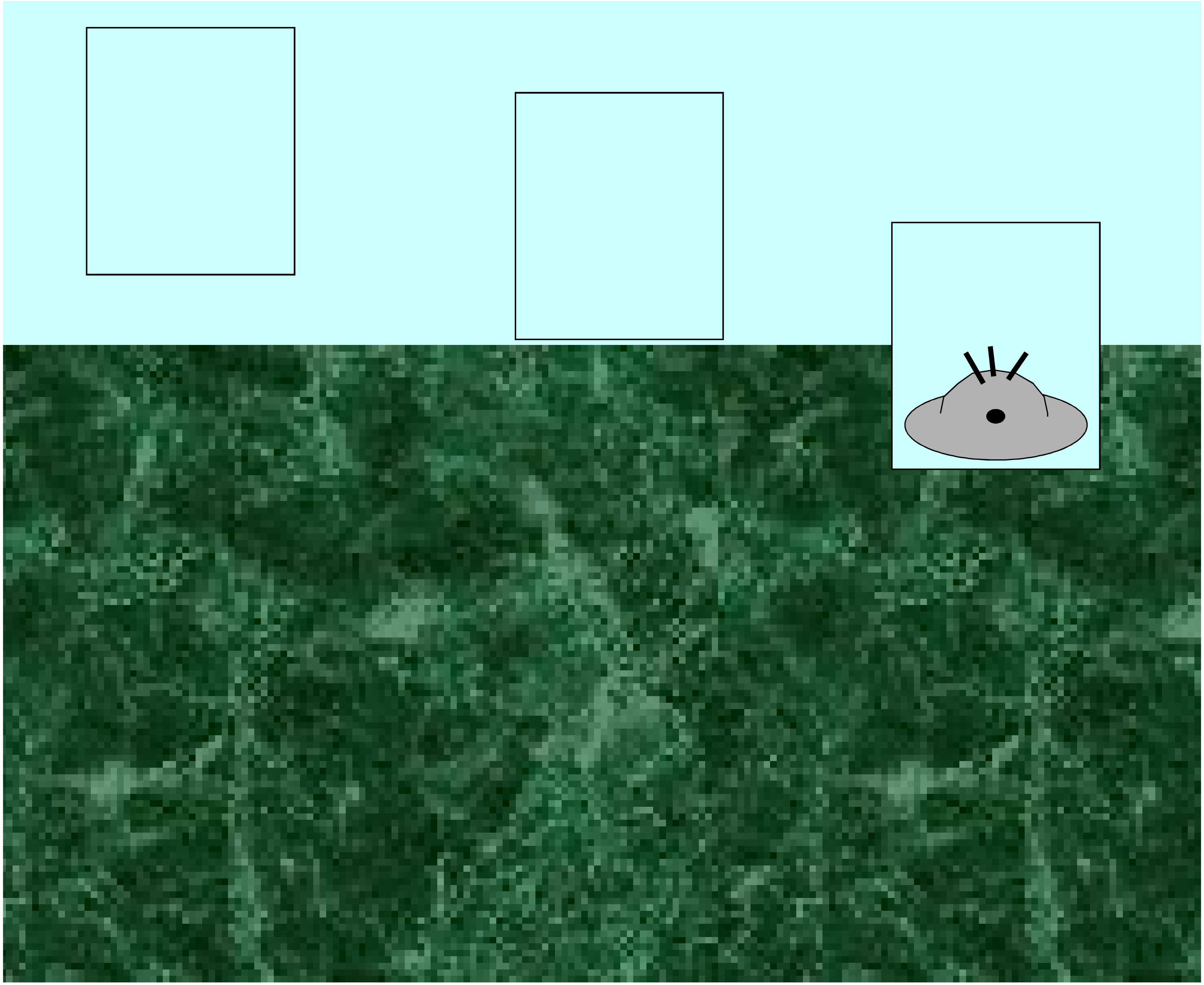
- They are intermediate frames between the key frames.
- Interpolate variables describing keyframes to determine poses for character "in-between".
- Interpolation is either be linear interpolation, spline interpolation or cubic spline interpolation.
- The process of generating in-betweens is called in-betweening.
- The number of in-betweens needed is determined by the media to be used to display the animation.
  - Film requires 24 frames per second and graphics terminals are refreshed at a rate of 30 to 60 frames per second.
- Time intervals for motion are setup so that there are from 3 to 5 in-betweens for each pair of key frames.
- Depending upon the speed specified for the motion, some key frames can be duplicated.

Animation Systems

- **Raster animation systems : U**se a sequence of raster operations to produce real-time animation of 2-D or 3-D objects.
- **Key-frame systems:** Designed to generate the in-betweens from the user specified key frames using interpolation.
- **Parameterized systems :**
  - Allow object-motion characteristics to be specified as part of the object definitions.
  - These characteristics are controlled by adjustable parameters: degrees of freedom, motion limitations, and allowable shape changes.
- **Scripting systems:** object specifications and animation sequences to be defined with user-input script

Raster Animation Systems

- Real time animations are generated for limited applications using raster operations.
- Animation done using raster operations or colour-table transformations.
- Raster based animation frames are made up of individual pixels. These pixels each contain information about the colour and brightness of that particular spot on the image
- Color Table Transformations:
  - Predefine the objects at successive positions along the motion-path and set the successive block of pixel values to color-table entries.
  - Set the pixels at the first position of the object to "on" values and set the pixels at other positions to the background color.
  - Animation is accomplished by changing the color-table values so that the object is "on" at successive positions of the motion path as the preceding position is set to background intensity



Key-frame systems

- From the specified two or more key frames, the key-frame systems generate sets of in-betweens.
- Motion paths, can be
  - given with a kinematical description as a set of spline curves.
  - physically based by specifying the forces acting on the objects to be animated.
- Given the animation paths, we can interpolate the position of individual objects between any two times.
- With the application of complex object transformations, the shapes of objects may change over time. Eg: clothes, facial features etc.,

Key-frame systems

- If all surfaces are described with polygon meshes, the number of edges per polygon can change from one frame to the next.
- Consequently, the total number of line segments can be different in different frames.

Morphing

- Morphing, a shortened form of metamorphosis is a transformation of an object from one form to another.
- Morphing methods can be applied to any motion or transition involving a change in shape and thus they yield evolving shapes.
- Uses linear interpolation for generating the in-betweens.
- Object shapes are described by polygons.
- Given two key-frames for an object transformation,
  - adjust the object specification in one of the frames such that the number of polygon edges (or the number of vertices) in two frames is the same.

Morphing

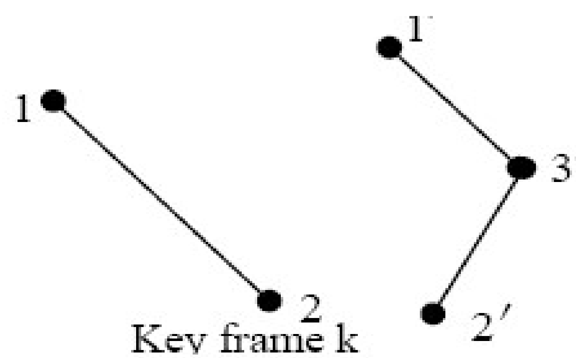


Fig. 1

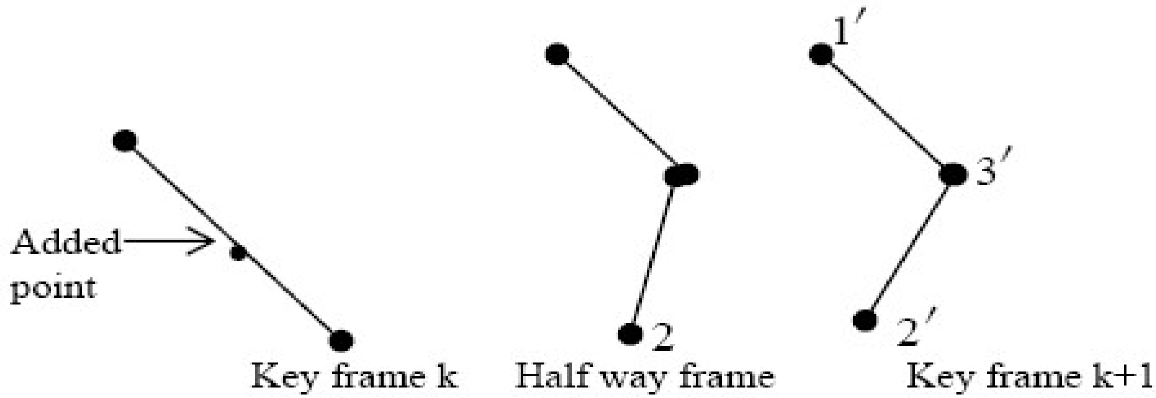


Fig. 2 Linear interpolation

Morphing

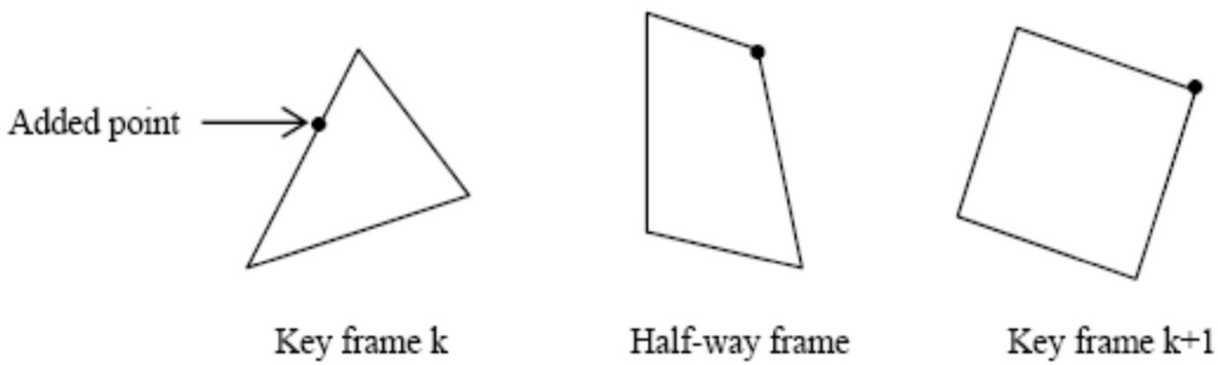


Fig.3
Linear interpolation for transforming a triangle into quadrilateral

Morphing

- We can equalize either edge count or vertex count

Equalizing Edge count:

- Let L_k and L_{k+1} denote the number of line segments in two consecutive frames. We define,

$$L_{\max} = \max(L_k, L_{k+1}) \qquad L_{\min} = \min(L_k, L_{k+1})$$

$$N_e = L_{\max} \bmod L_{\min} \qquad N_s = \operatorname{int}\left(\frac{L_{\max}}{L_{\min}}\right)$$

- Preprocessing is accomplished by

1) Dividing N_e edges of $\text{keyframe}_{\min}$ into N_s+1 sections
2) Dividing the remaining lines of $\text{keyframe}_{\min}$ into N_s sections

Eg: $L_k = 15$ and $L_{k+1} = 11$ we would divide 4 lines of keyframe_{k+1} into 2 sections each. The remaining lines of keyframe_{k+1} are left intact.

Morphing

- **Equalizing the vertex count:** let parameters V_k and V_{k+1} denote the vertices

$$V_{max} = \max(V_k, V_{k+1}) \qquad V_{min} = \min(V_k, V_{k+1})$$

$$N_{ls} = (V_{max} - 1) \bmod (V_{min} - 1) \qquad N_p = \mathrm{int}\left(\frac{V_{max} - 1}{V_{min} - 1}\right)$$

- Preprocessing using vertex count is performed by

  1. Adding N_p points to N_{ls} line sections of $keyframe_{min}$
  2. Adding N_p-1 points to the remaining edges of $keyframe_{min}$

- For the triangle to quadrilateral examples, $V_k = 3$ and $V_{k+1} = 4$.
- Both N_{ls} and N_p are 1 from Eqns.
- we would add one point to one edge of $keyframe_{k.}$
- No points would be added to the remaining lines of $keyframe_{k+1}$.

Simulating Accelerations

- we use time interpolation to specify the animation paths between key frames and produce realistic displays of different speed changes.
- Curve fitting techniques used to specify the animation paths between keyframes.
- Given the vertex positions at the key frames, we can fit the positions with linear or nonlinear paths.
- But to simulate accelerations, we need to adjust the time spacing for the in-betweens.
- First, we consider constant speed (zero acceleration) using equal time-interval spacing for the in-betweens

Simulating Accelerations

- Let there be n in-betweens for key frames at times t1 and t2
- We now divide the time interval between key frames into (n+1) sub intervals, yielding an in-between spacing of

$$\Delta t = \frac{t_2 - t_1}{n+1}$$

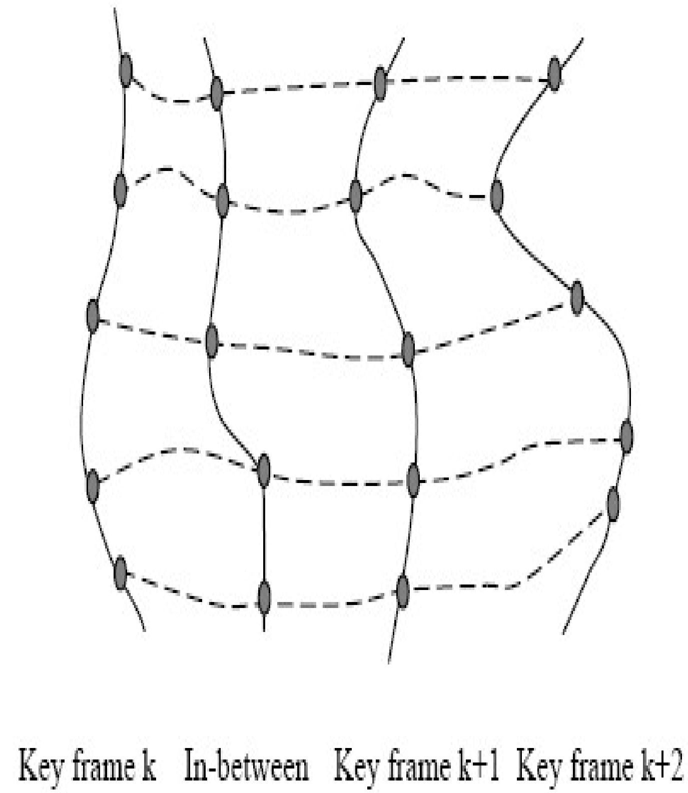


Fig. 4: Fitting key frame vertex positions with non linear splines

Simulating Accelerations

- We can calculate the time for any in-between by the interpolation as

$$t_{Bj} = t_1 + j\,\Delta t, \qquad j = 1, \ldots\ldots\ldots\ldots, n \quad \ldots(6)$$

- Then, determine the values for coordinate positions, color, and other physical parameters.

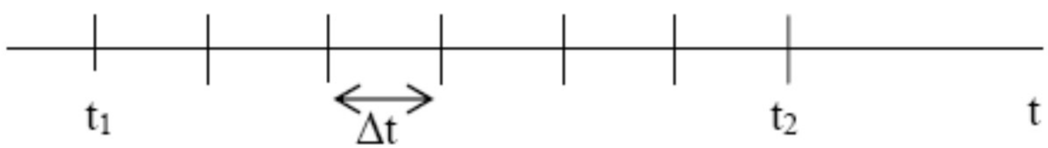


Fig. 5 :
In-between positions for motion at constant speed

Simulating Accelerations

- To produce realistic displays of speed changes particularly at the beginning and at the end of a motion sequence, Non zero accelerations are used
- Model the start-up and slow-down portions of an animation path with spline or trigonometric functions.
- To model an increasing speed (positive acceleration), the time spacing between frames has to be increased so that greater changes in position occur as the object moves faster.
- We can obtain an increasing interval size with the function

$$1-\cos\theta \qquad\qquad 0 < \theta < \pi/2$$

- The time for the jth in-between can be calculated from the above function as

$$t_{bj} = t_1 + \Delta t \left[1 - \cos\frac{j\pi}{2(n+1)} \right], \qquad j = 1,2,\ldots,n$$

Simulating Accelerations

- We can model decreasing speed (deceleration) with $\sin\theta$, using the angle in the range $0 < \theta < \pi/2$.
- The time spacing of an in-between in this case is defined as

$$t_{bj} = t_1 + \Delta t \sin \frac{j\pi}{2(n+1)} \qquad j = 1, 2, \ldots, n$$

- Can model a combination of increasing-deceasing speed by first increasing the in-between time spacing, then decreasing this spacing.
- A function to accomplish these time changes is

$$\frac{1}{2}(1 - \cos\theta) \qquad 0 < \theta < \pi/2$$

•The time for the j-th in-between is now calculated as:

$$t_{bj} = t_1 + \Delta t \left\{ \frac{1 - \cos[j\pi/(n+1)]}{2} \right\} \qquad j = 1, 2, \ldots, n$$

Motion Specifications

- Several ways in which the motions of objects can be specified in an animation system.
- Define motion directly or in a more abstract or general approach.

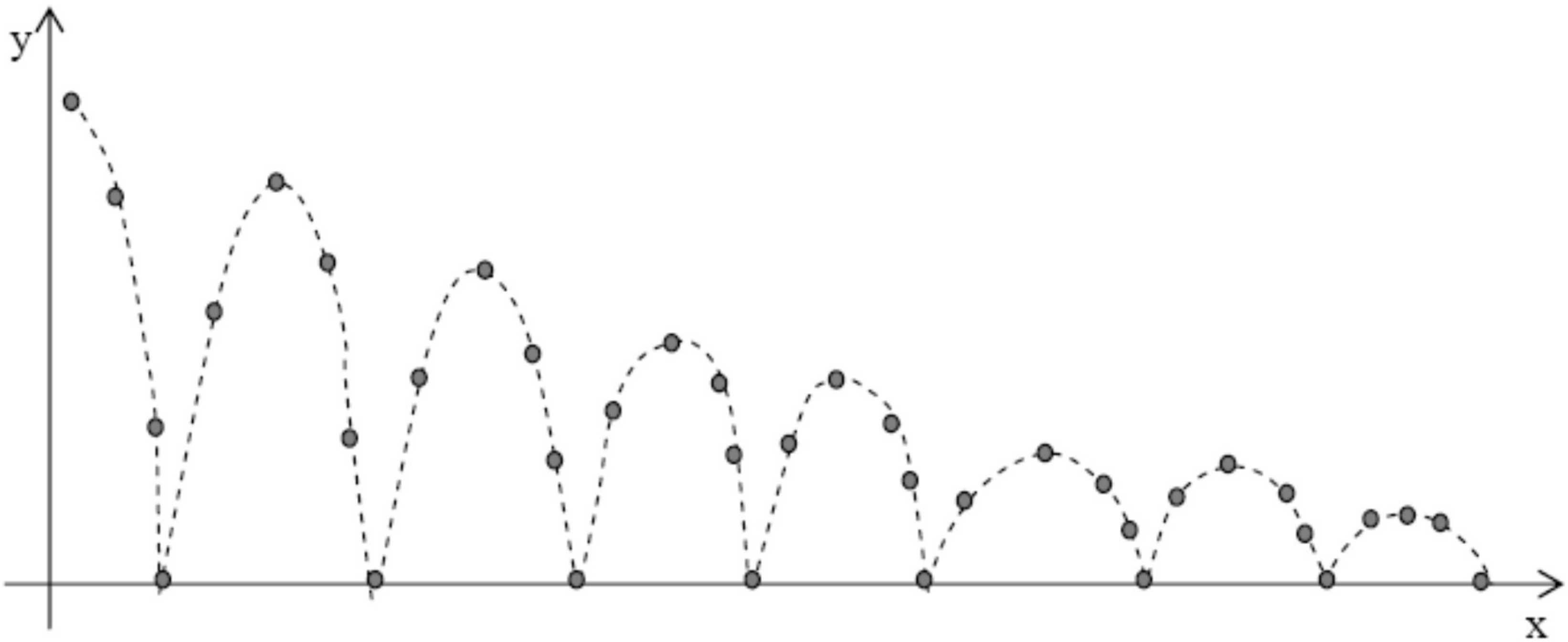
1) Direct motion specification
2) Goal Directed systems
3) Kinematics and Dynamics.

Direct motion specification

- The most straightforward method of defining a motion sequence is the direct specification of motion parameters.
- The specification consists of rotation angles and translation vectors.
- Then geometric transformation matrices are applied to transform coordinate positions.
- Alternatively, use an approximation equation to specify certain kinds of motions.
- Approximate the path of bouncing ball with a damped, rectified sine curve

$$y(x) = A|\sin (wx+\theta_0)|e^{-kx}$$

A- initial amplitude, w –angular frequence, θ_0 – phase angle, k- damping constant



–y(x) = A|sin (wx+θ0)|e-kx

Constraint based and Goal directed system

- Specify the motions that are to take place in general terms that abstractly describe the actions.
- Called as goal directed ‘coz they determine the specific motion parameters given the goals of animation.
- Ex: specify - want an object to “walk” to run” to a particular destination.
- Input directives are then interpreted in terms of component motions that will accomplish the specified task.
- Human motions, can be defined as a hierarchical structure of sub motions for the torso, limbs etc.,

Kinematics and Dynamics

- We can also construct animation sequences using either kinematics, which refers to positions, velocities and acceleration of points without reference to forces that cause motion.
- For constant velocity we infer the motions by giving initial position and velocity vector for each object.
- Eg: If velocity is specified as (3,0,-4) km/sec then
  - Direction – straight line path
  - Speed (magnitude) is 5 km/sec
- If acceleration is also specified, speed-ups slowdowns and curved motion paths can be generated.
- **Inverse Kinematics:** we specify the initial and final positions of objects at specified times and the motion parameters are computed by the system.

Kinematics and Dynamics

- **Dynamics:** Requires specification of forces that produce the velocities and accelerations.
- **Physically based modeling:** Descriptions of object behavior under the influence of forces.
- **Eg of forces:** Gravitational, electro magnetic, friction and other mechanical forces.
- Object motions are obtained from force equations.
- Eg: Newton's second law , F= ma.
- **Inverse Dynamics:** Obtain forces given the initial and final positions of objects and type of motion.