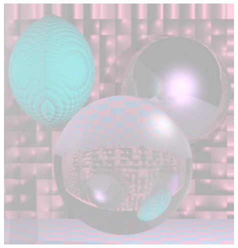
CS G140
Graduate Computer Graphics

Prof. Harriet Fell

Spring 2009

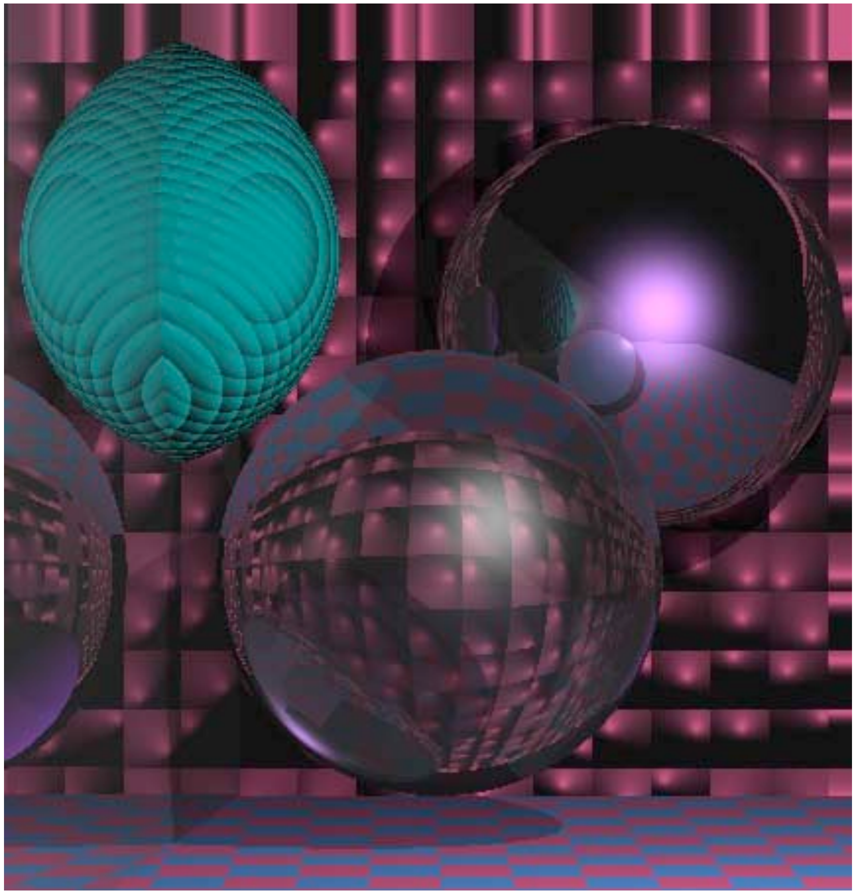
Lecture 1 - January 7, 2009

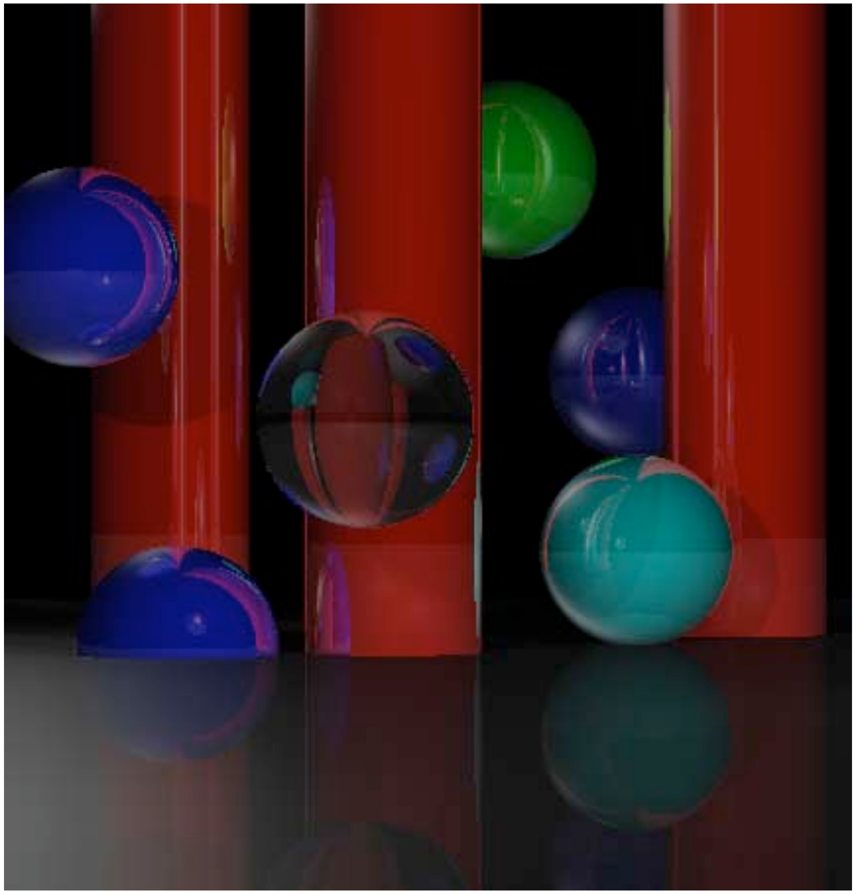


Course Overview - Topics

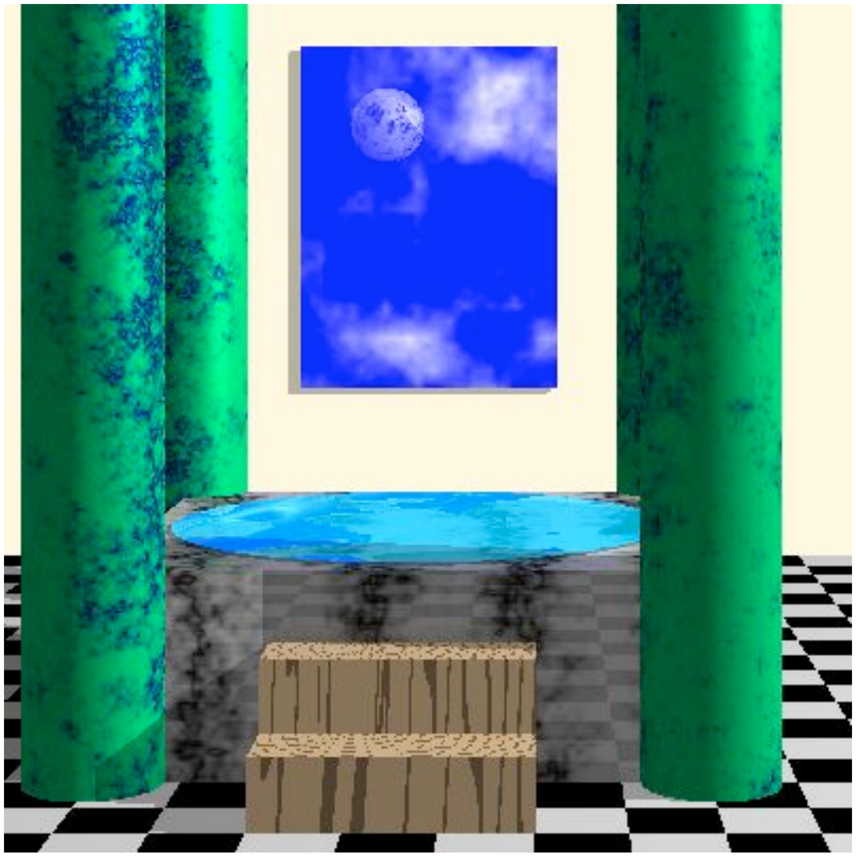
- Emphasis on rendering realistic images.
- Fundamentals of 2- and 3- dimensional computer graphics
  - 2-dimensional algorithms for drawing lines and curves, anti-aliasing, filling, and clipping
  - Using ray-tracing to render 3-dimensional scenes
    - composed of spheres, polygons, quadric surfaces, and bi-cubic surfaces
  - Techniques for adding texture to surfaces using texture and bump maps, noise, and turbulence
- Other topics as time permits

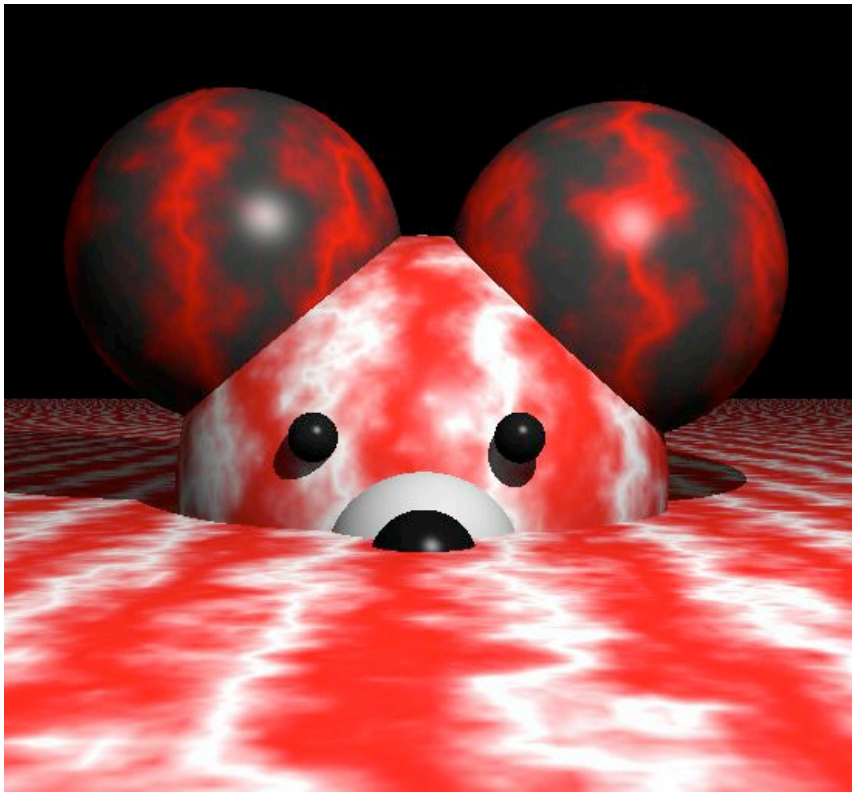
Sample Images





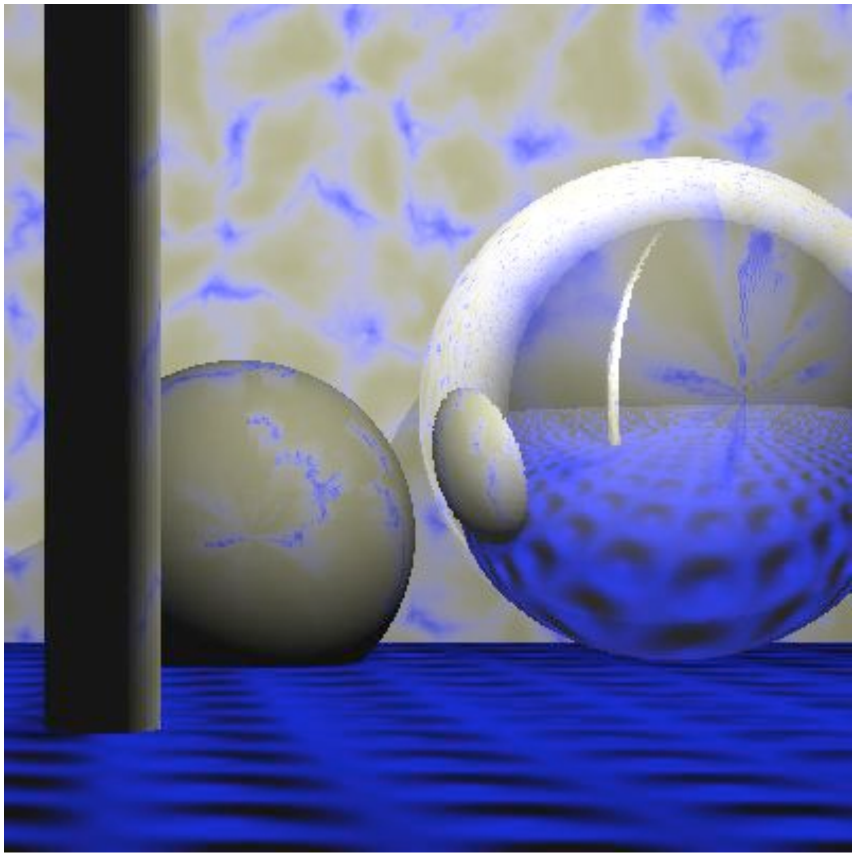
Sample Images

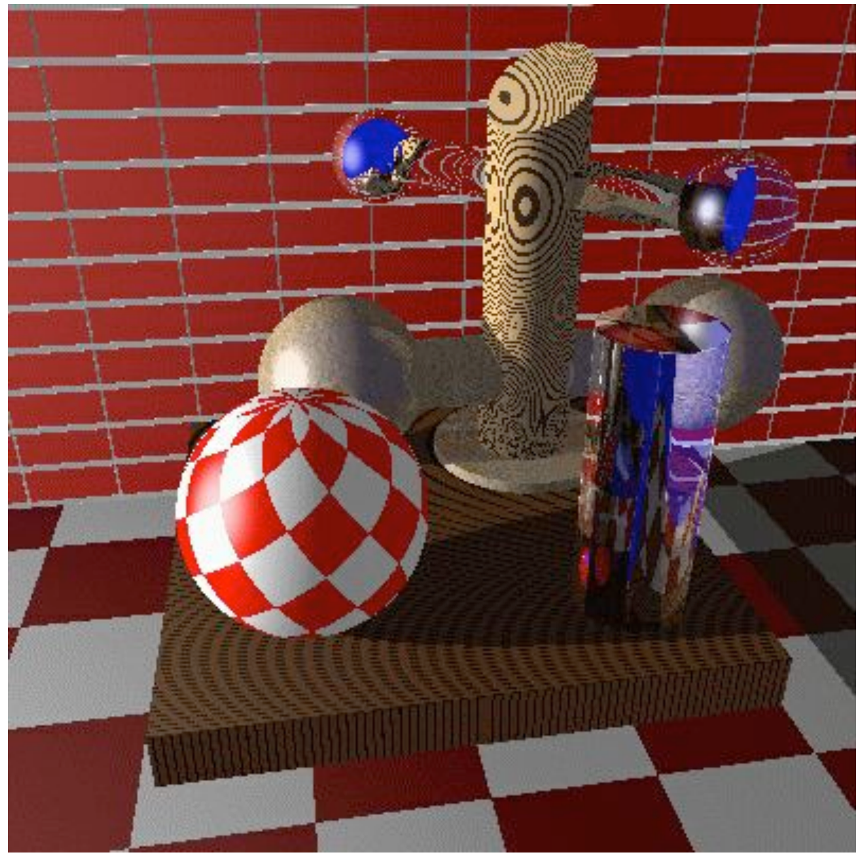




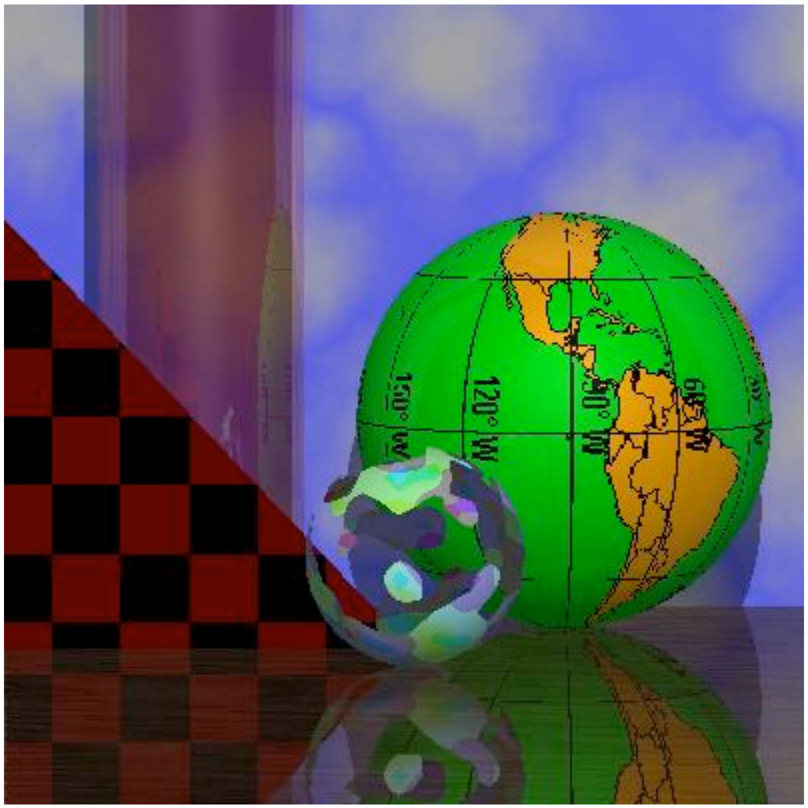
Sample Images

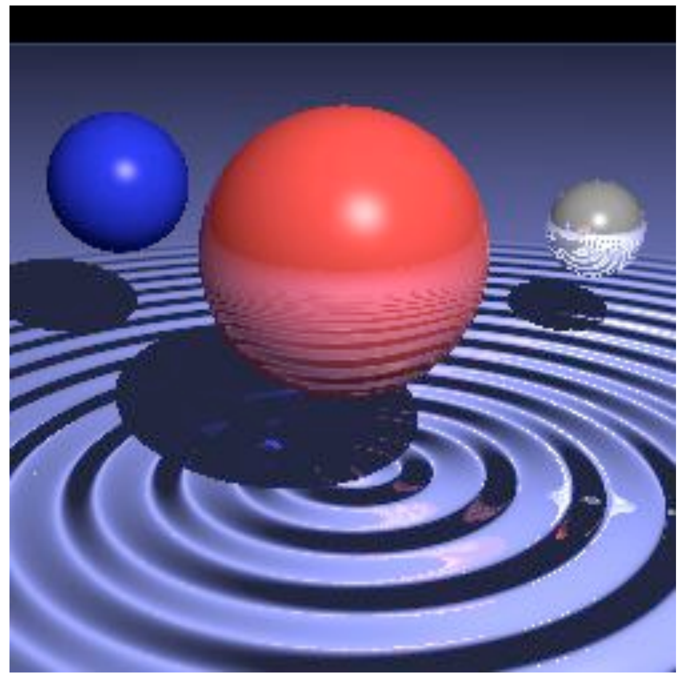
Sample Images

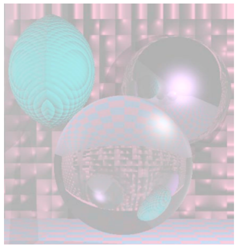




Sample Images



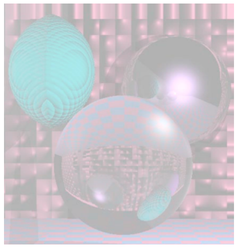




Course Overview - Organization

- **Texts:**
  - Peter Shirley, *et al. Fundamentals of Computer Graphics, 2nd Edition*, A K Peters, 2005
  - Alan Watt, *3D Computer Graphics, 3rd Edition* , Addison Wesley, 1999.

- **Grading**
  - Assignment 0: 10%
  - Assignment 1: 15%
  - Assignment 2: 15%
  - Assignment 3: 10%
  - Assignment 4: 10%
  - Exam: 25%
  - Project and Presentation: 15%



Early History

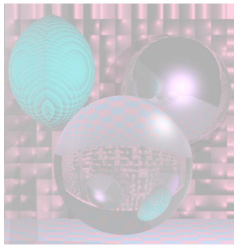
- http://accad.osu.edu/~waynec/history/timeline.html
- http://sophia.javeriana.edu.co/~ochavarr/computer_graphics_history/historia/
- 1801 Joseph-Marie Jacquard invented an automatic loom using punched cards to control patterns in the fabrics. The introduction of these looms caused the riots against the replacement of people by machines.
- 1941 First U.S. regular TV broadcast,
  1st TV commercial (for Bulova watches)
- 1948 Transistors
- 1949 Williams tube (CRT storage tube)

Jacquard Loom

From Wikipedia.org

Early TV


NBC
HORIZONTAL RESOLUTION
WNBT
NEW YORK
FREQ. IN MEGACYCLES
© NBC 1947



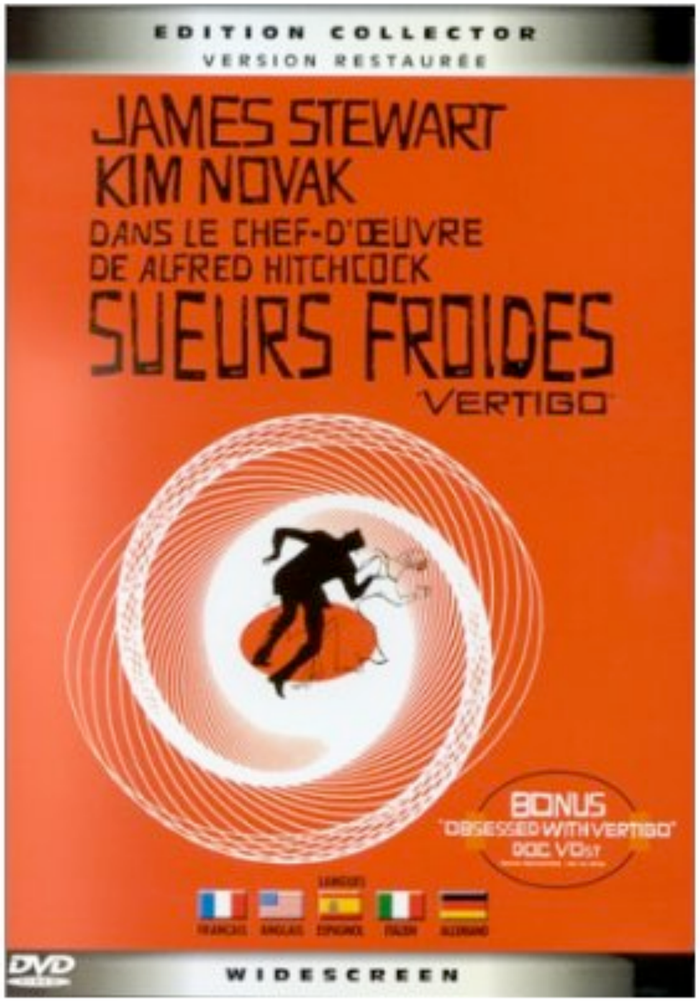
History – the 50s

- 1951 Graphics display, Whirlwind computer
- 1954 color TV
- 1955 Light Pen, SAGE- Lincoln Lab
- 1958 Graphics Console, TX-1 MIT
- 1958 John Whitney Sr. uses analog computer to make art

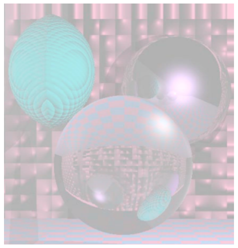
1951 Graphics display, Whirlwind computer

SAGE

John Whitney Sr. 1958 CG



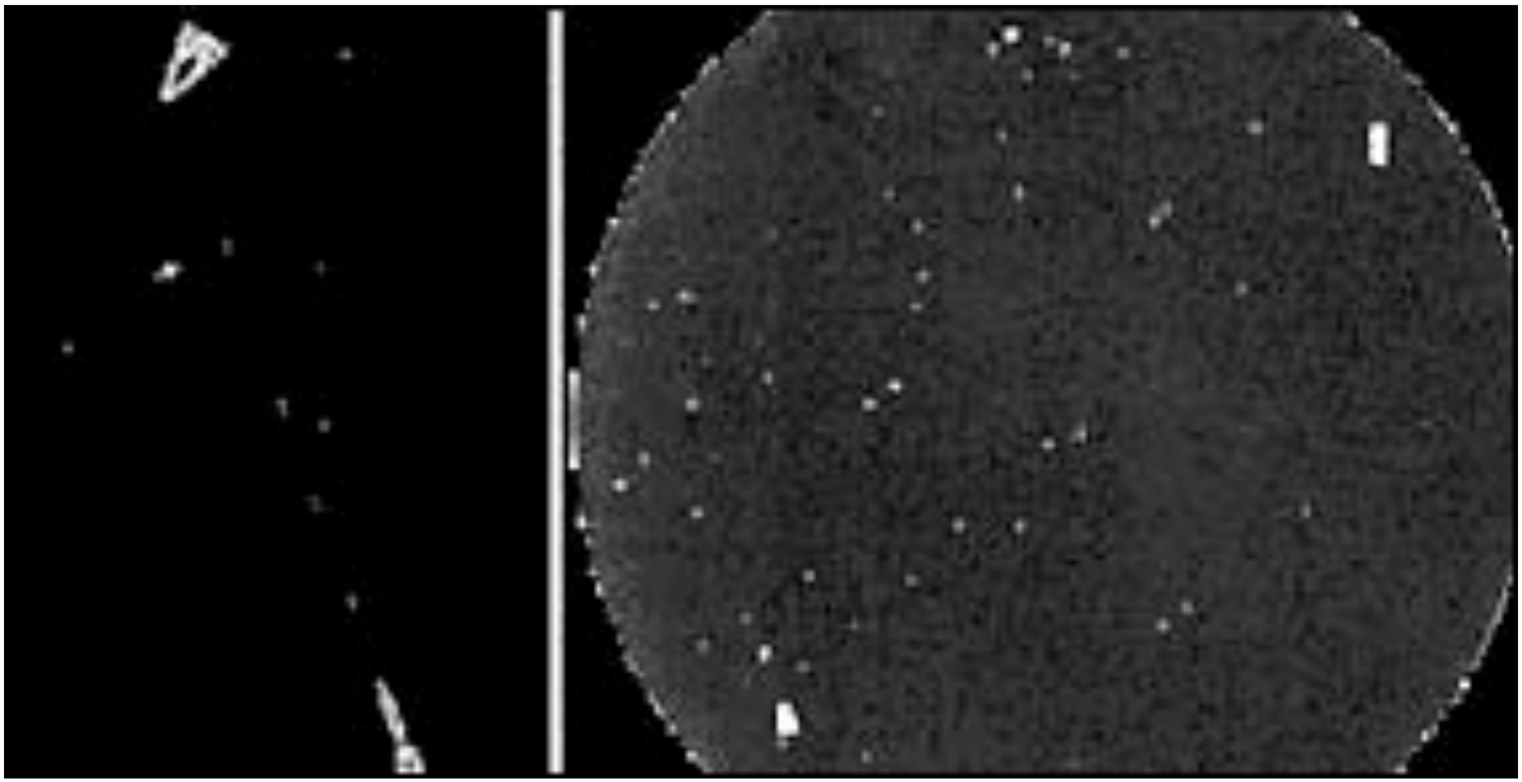
[Vertigo Start Titles]



History - the 60s

- 1961 Spacewars, 1st video game, Steve Russell, MIT for PDP-1
- **1963 Sketchpad, Ivan Sutherland**, MIT
- 1963 Mouse invented, Doug Englebart, SRI
- 1963 Roberts hidden line algorithm, MIT
- 1965 Bresenham Algorithm for plotting lines, IBM
- 1966 Odyssey, home video game, Ralph Baer,
  - Sanders Assoc, is 1st consumer CG product
- 1967 Full-color, real-time, interactive
  flight simulator for NASA - Rod Rougelet, GE

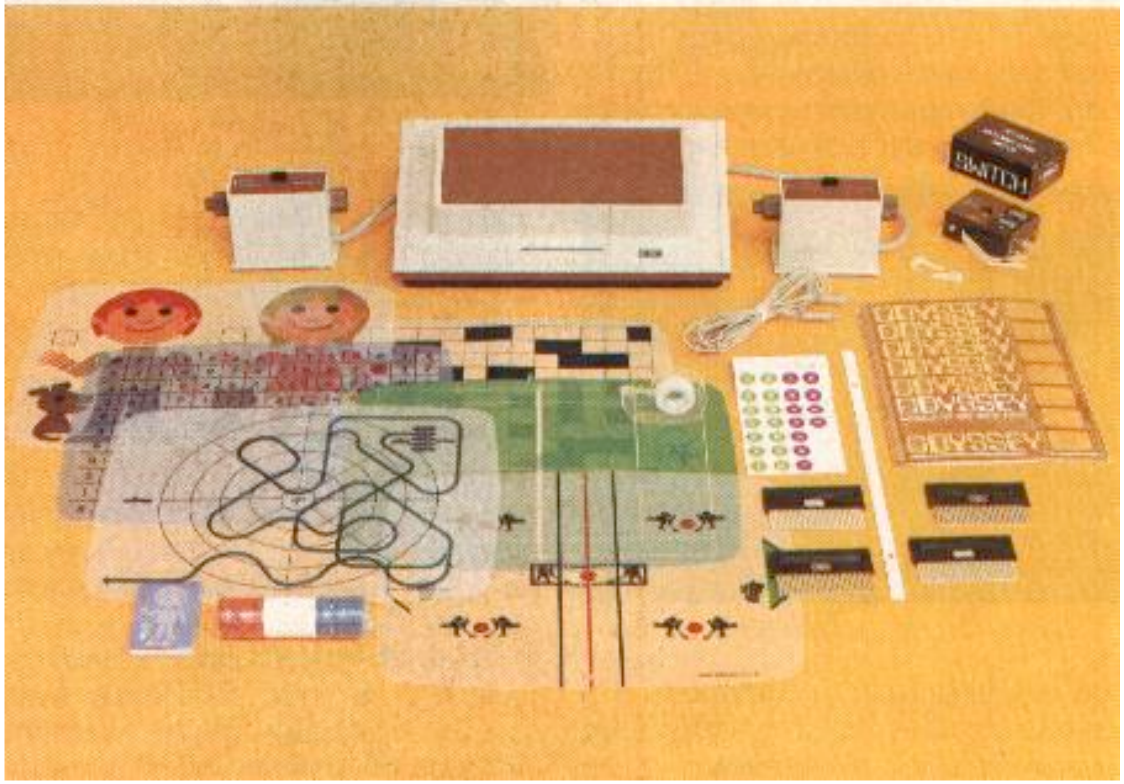
Spacewars



Ivan Sutherland & Sketchpad System on TX-2 at MIT(1963)

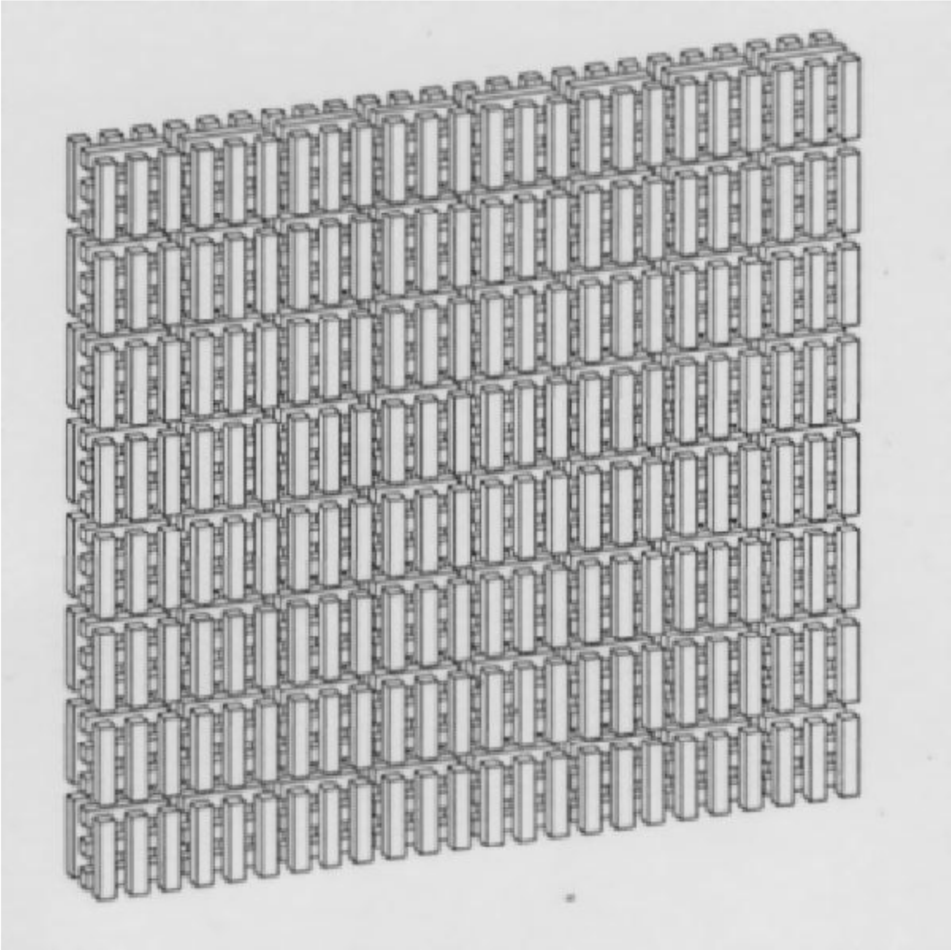
Odyssey

The very first home videogame, Odyssey, used Laner-created transparent overlays in lieu of computer-generated graphics.

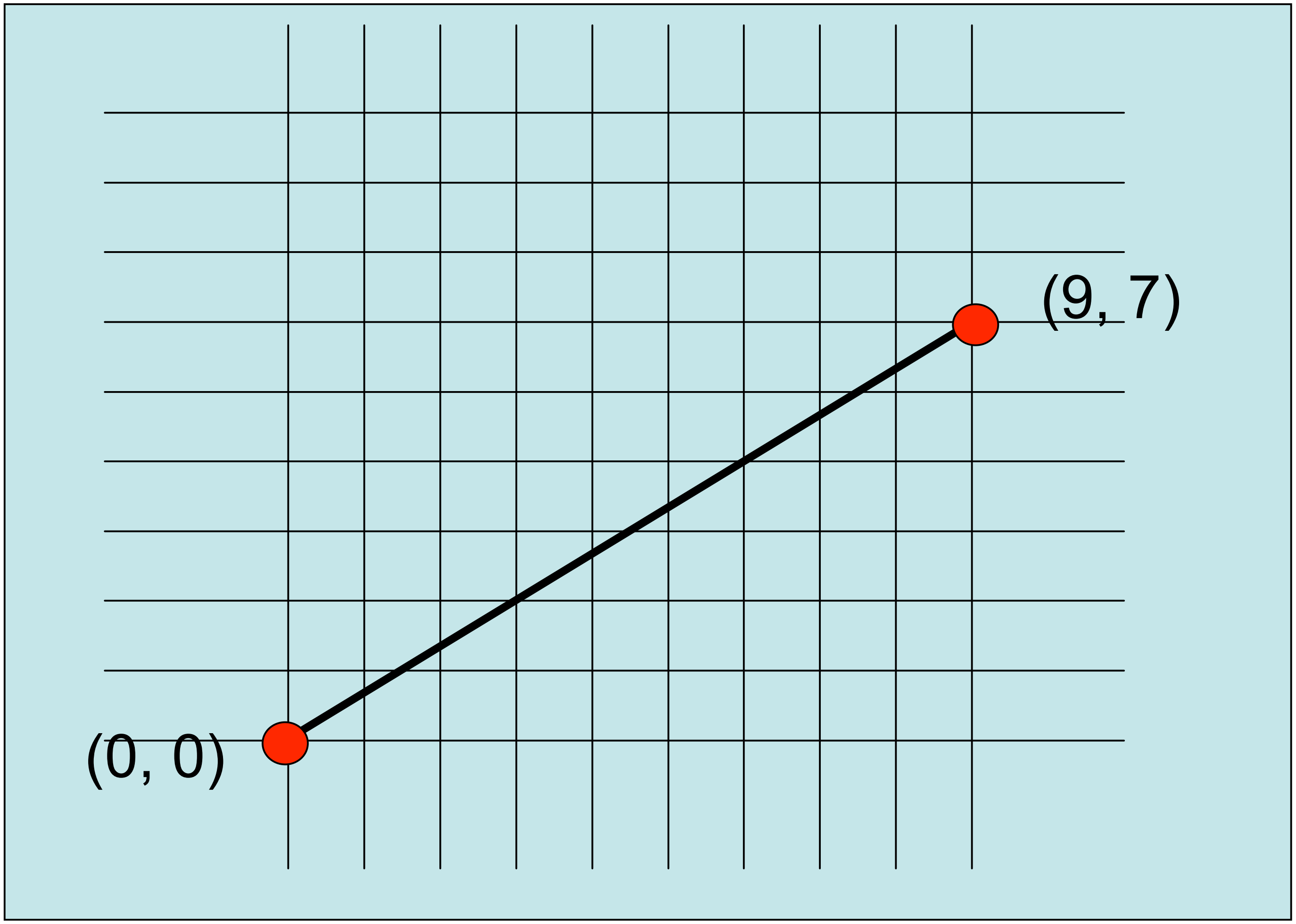


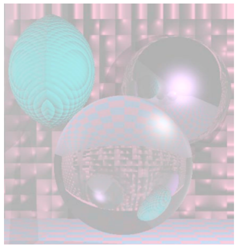
http://www.geekcomix.com/vgh/first/odyssey.shtml

Roberts Hidden Line Algorithm Block scene (576 blocks)



Bresenham Line Algorithm

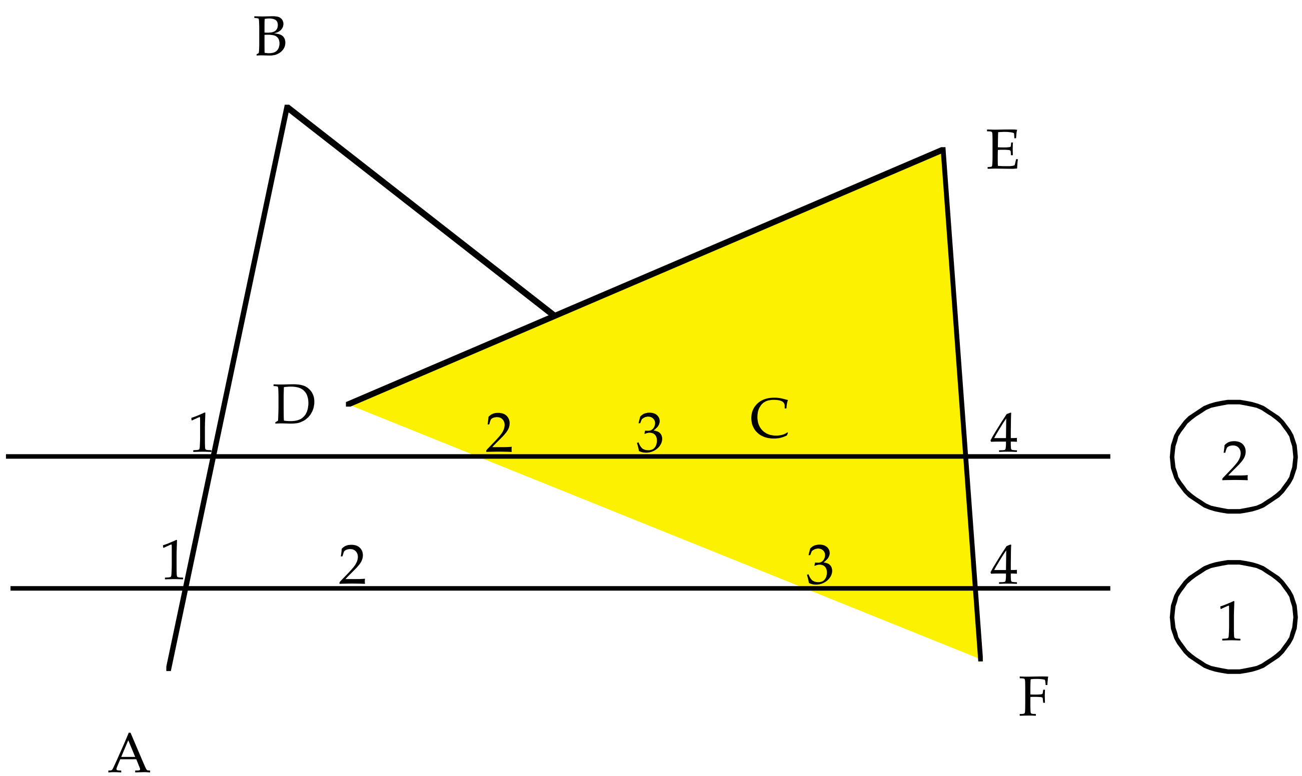




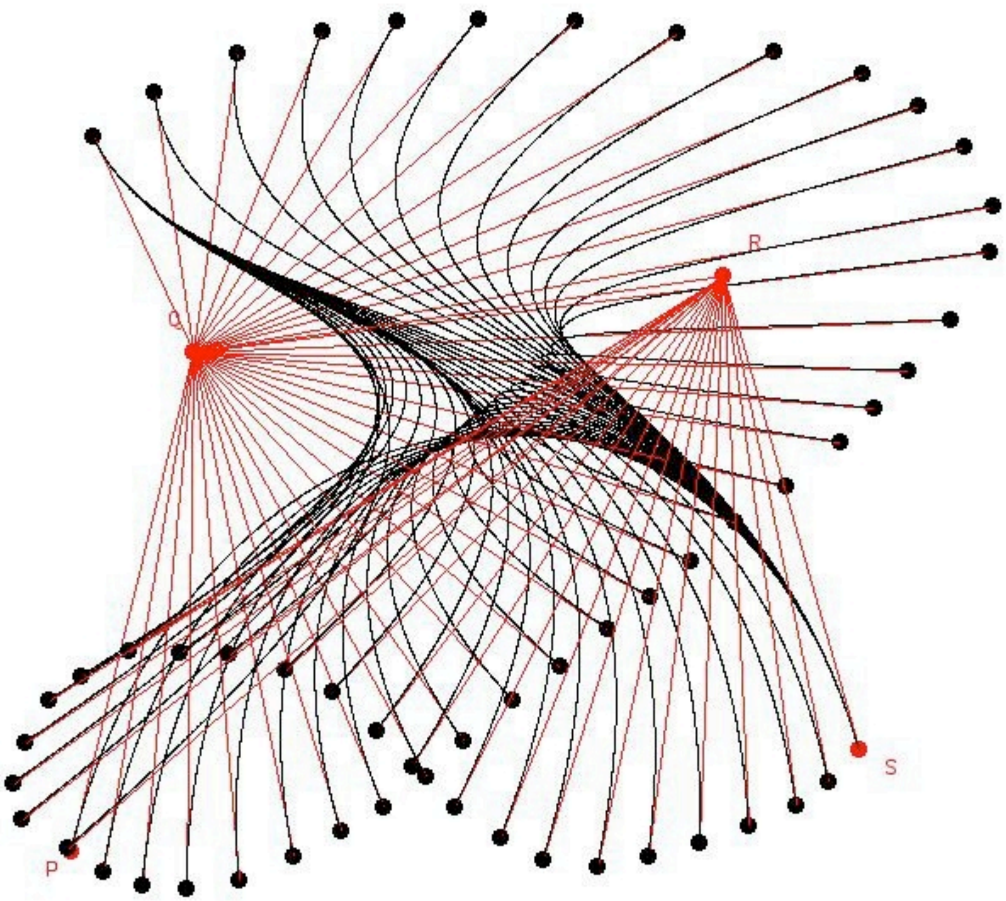
History – the 70s

- **1970s Utah dominated - algorithm development**
- 1970 Watkins algorithm for visible surfaces
- 1970 Bezier free-form curve representation
- 1971 Gouraud shading
- 1973 Principles of Interactive Computer Graphics (Newman and Sproull)
- 1974 Addressable cursor in a graphics display terminal - DEC VT52
- 1974 z-buffer developed by Ed Catmull (Univ of Utah)
- 1975 Phong shading
- 1975 Fractals - Benoit Mandelbrot (IBM)
- 1978 Bump mapping, Blinn
- 1979 George Lucas starts Lucasfilm
  - with Ed Catmull, Ralph Guggenheim, and Alvy Ray Smith

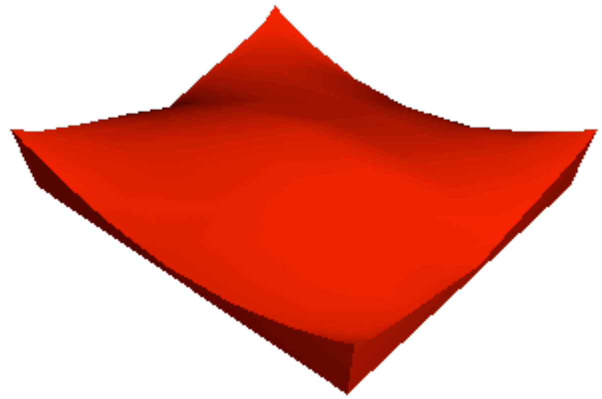
Watkins Scan-Line Algorithm



Bezier Curves

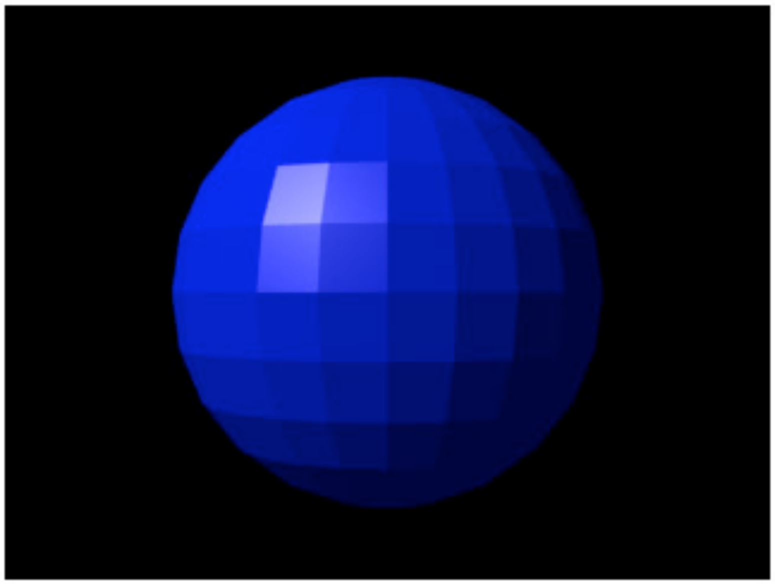


Gouraud Shading

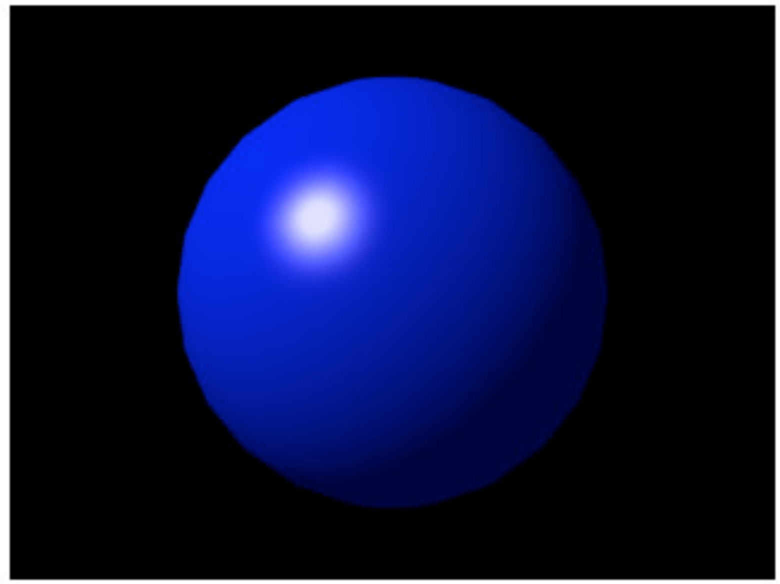


http://freespace.virgin.net/hugo.elias/graphics/x_polygo.htm

Phong Shading

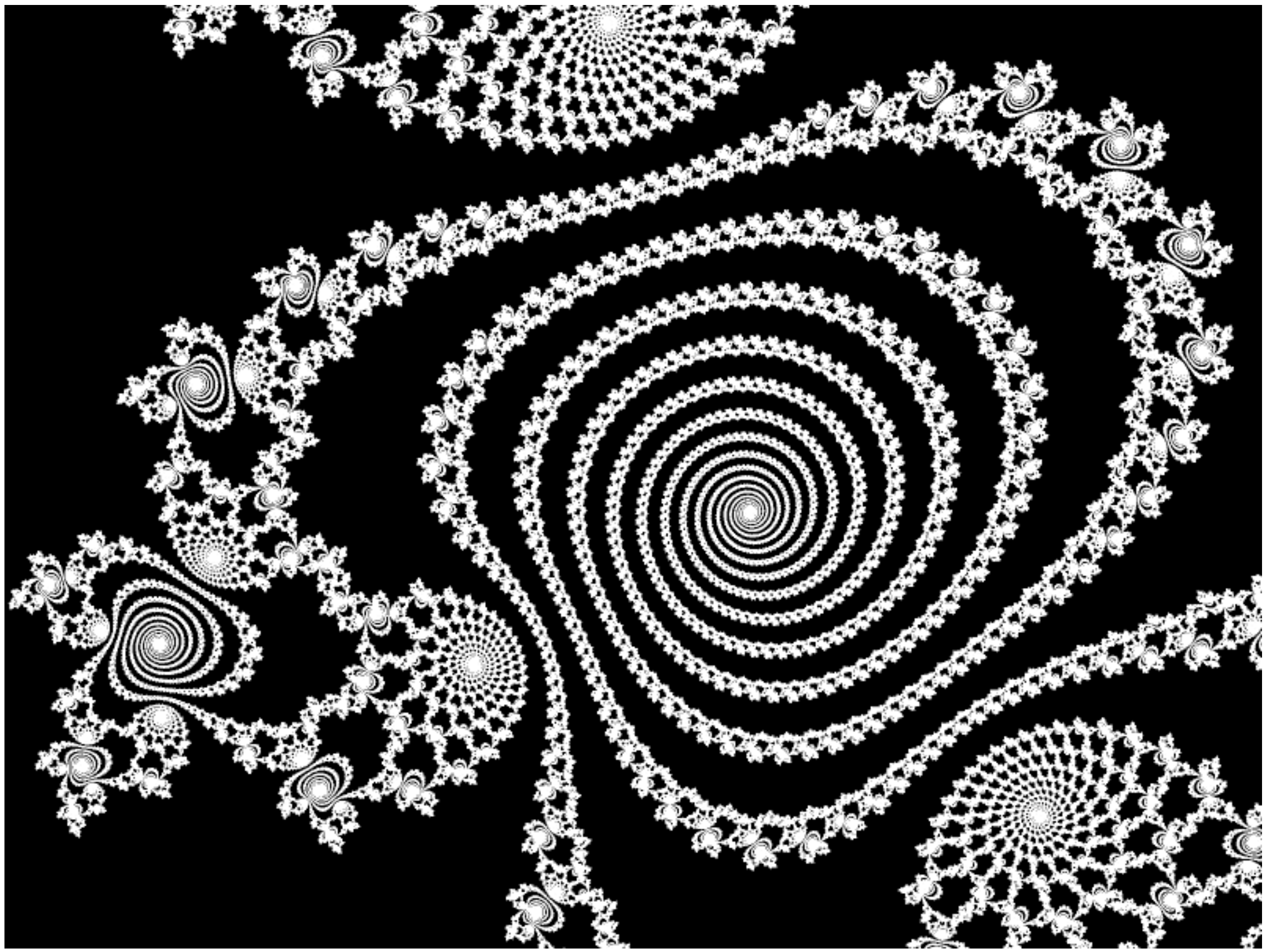


FLAT SHADING

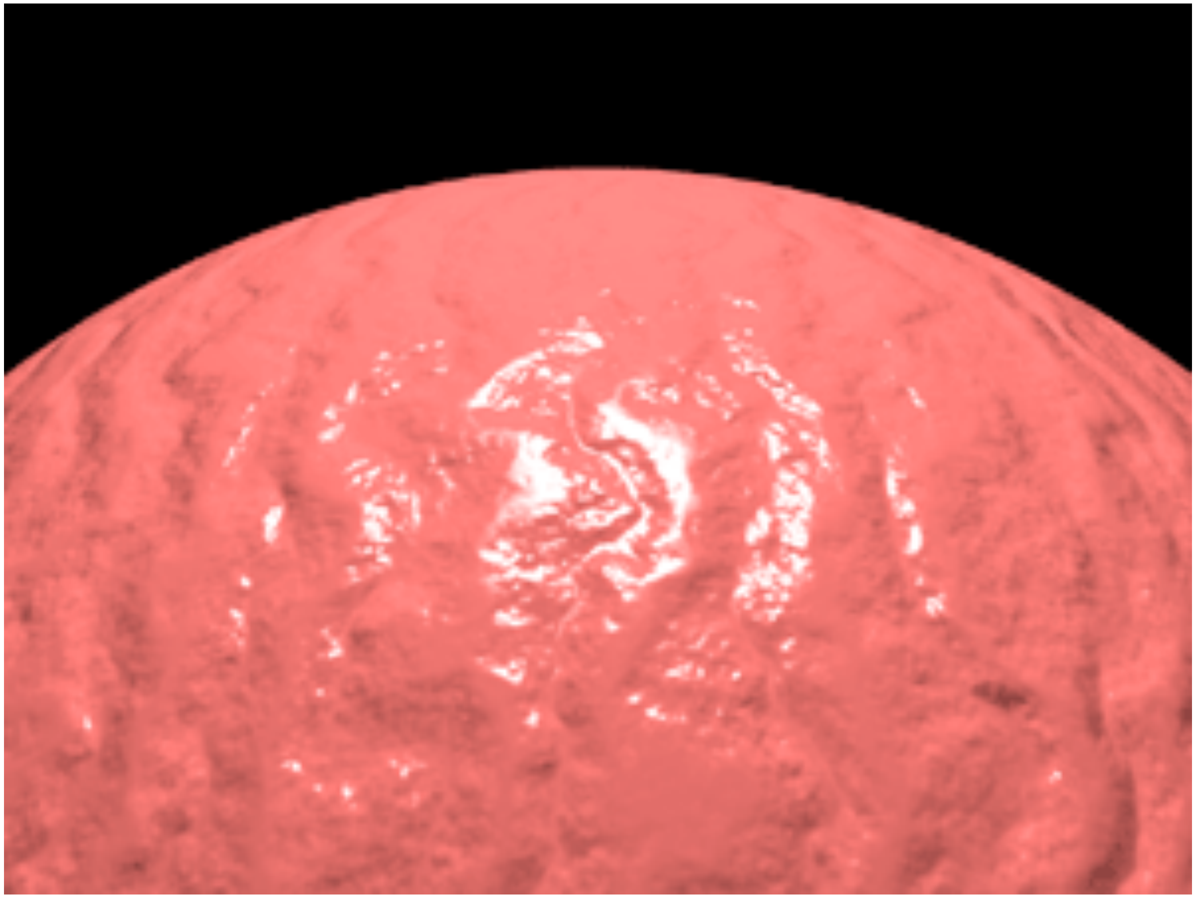


PHONG SHADING

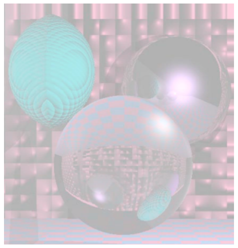
Fractals



Bump Map



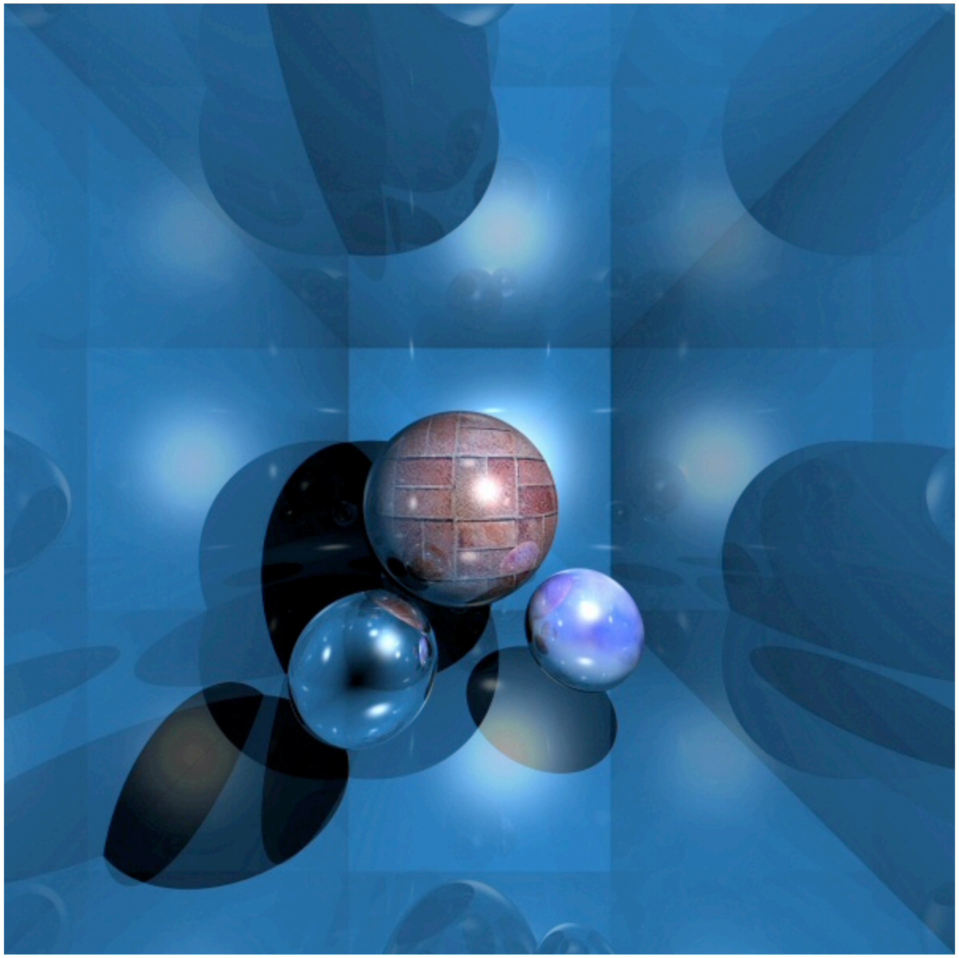
Bump Maps in PovRay



History - the 80s

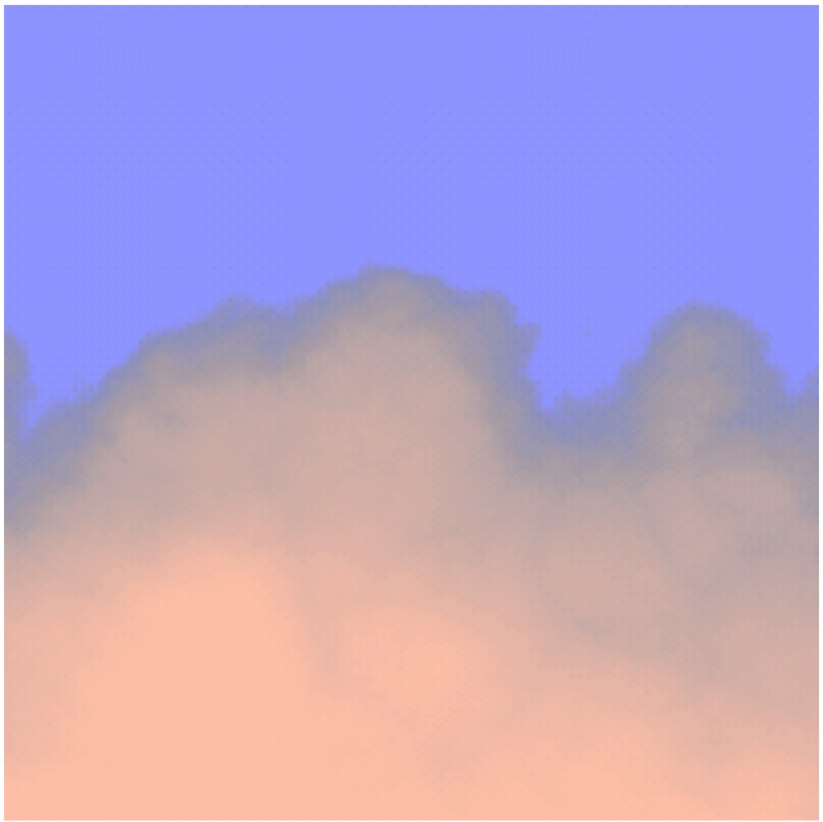
- **1980s Cheaper machines, memory - quest for realsim**
- 1980 Ray Tracing, Turner Whitted, Bell Labs
- 1981 IBM introduces the first IBM PC (16 bit 8088 chip)
- 1982 Data Glove, Atari
- 1984 Macintosh computer
  - introduced with Clio award winning commercial during Super Bowl
- 1985 Perlin Noise
- 1986 GIF format (CompuServe)
- 1988 *Who Framed Roger Rabbit* live action & animation

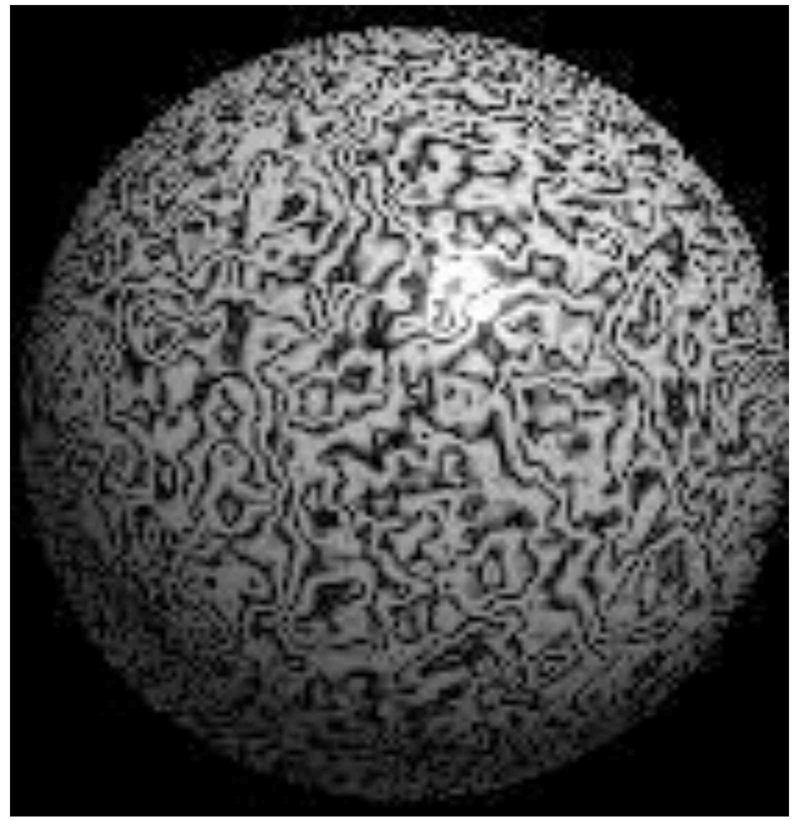
Whitted Ray-Tracing



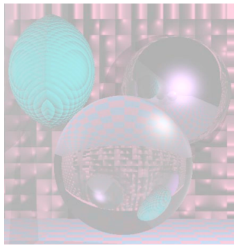
http://en.wikipedia.org/wiki/Ray_tracing

Perlin Noise





Who Framed Roger Rabbit

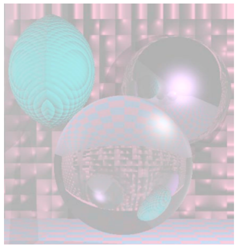


History- the 90s

- **1990s Visualization, Multimedia, the Net**
- 1991 JPEG/MPEG
- 1993 Myst, Cyan
- 1994 U.S. Patent to Pixar
  - for creating, manipulating and displaying images
- 1995 Toy Story, Pixar
- 1995 Internet 2 unveiled
- 1997 DVD technology unveiled
- 1998 XML standard
- 1999 deaths

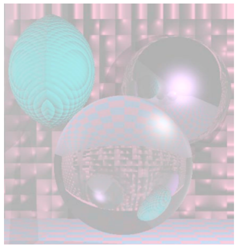
Myst

Toy Story



Recent History

- **2000s Virtual Reality, Animation Reality**
- 2001 Significant Movies
  - Final Fantasy, Square)
  - Monsters Inc, Pixar
  - Harry Potter, A.I., Lord of the Rings, Shrek, PDI
  - The Mummy, ILM
  - Tomb Raider, Cinesite
  - Jurassic Park III, Pearl Harbor,ILM
  - Planet of the Apes, Asylum
- 2001 Microsoft xBox and Nintendo Gamecube
- 2001, 2002, 2003 Lord of the Rings

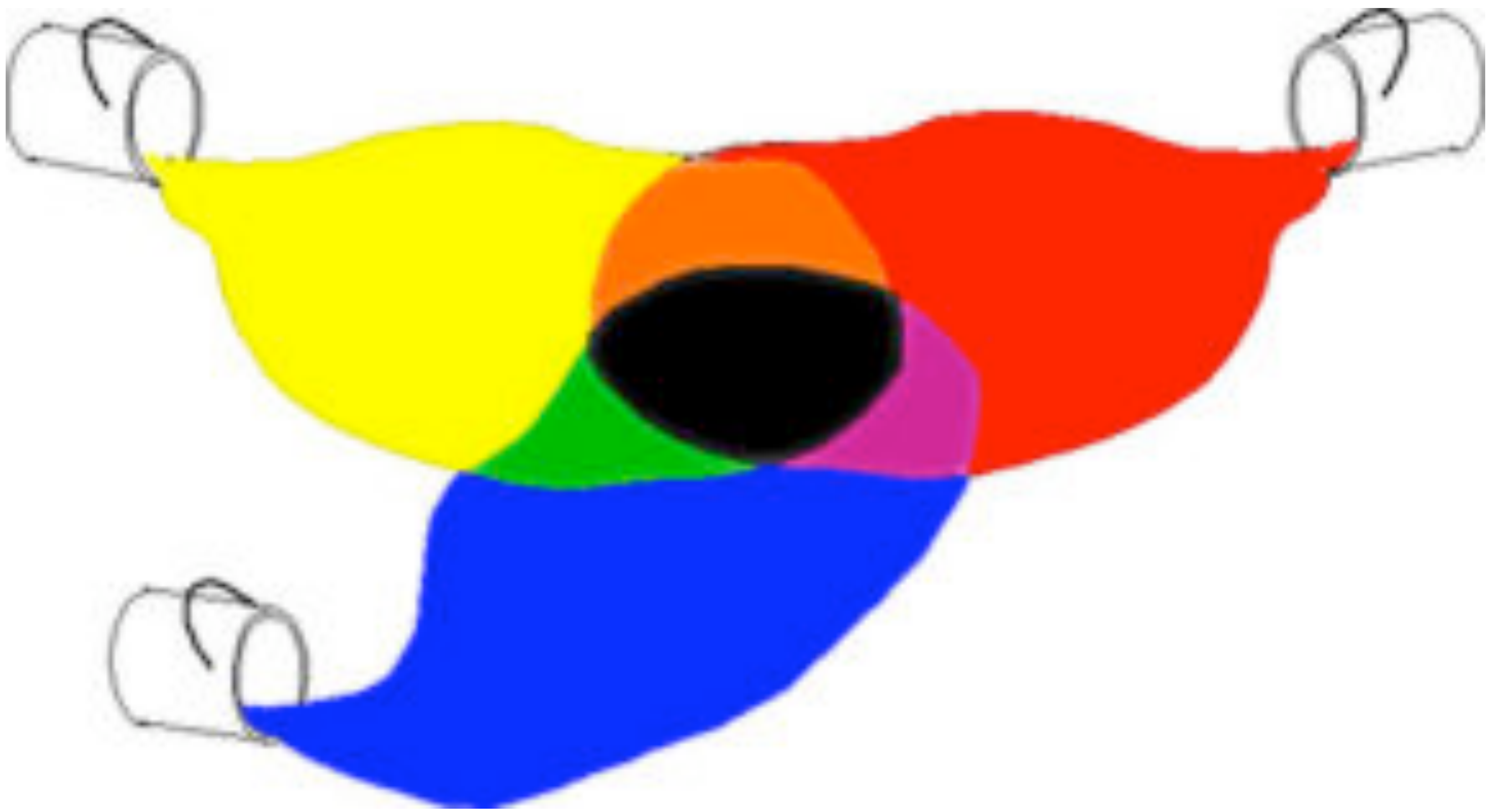


from Lord of the Rings

- Motion Capture Technology
  - Andy Serkis "played" Gollum by providing his voice and movements on set, as well as performing within a motion capture suit.
- SKIN
  - Christoper Hery, Ken McGaugh and Joe Letteri received a 2003 Academy Award, Scientific or Technical for implementing the BSSRDF (Bidirectional Surface Scattering Reflection Distribution Function) technique used for Gollum's skin in a production environment. Henrik Wann Jensen, Stephen Robert Marschner, and Pat Hanrahan, who developed BSSRDF, won another the same year.
- MASSIVE
  - a computer program developed by WETA to create automatic battle sequences rather than individually animate every soldier. Stephen Regelous developed the system in 1996, originally to create crowd scenes in *King Kong*.

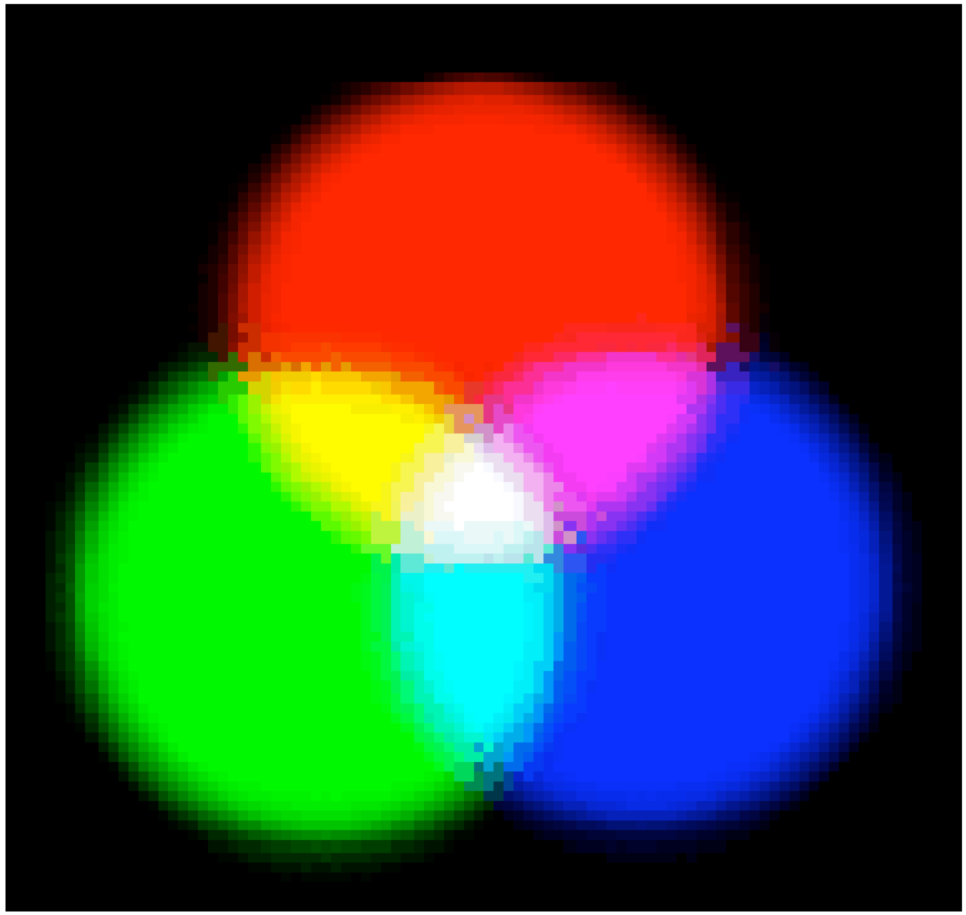
Time for a Break

Color

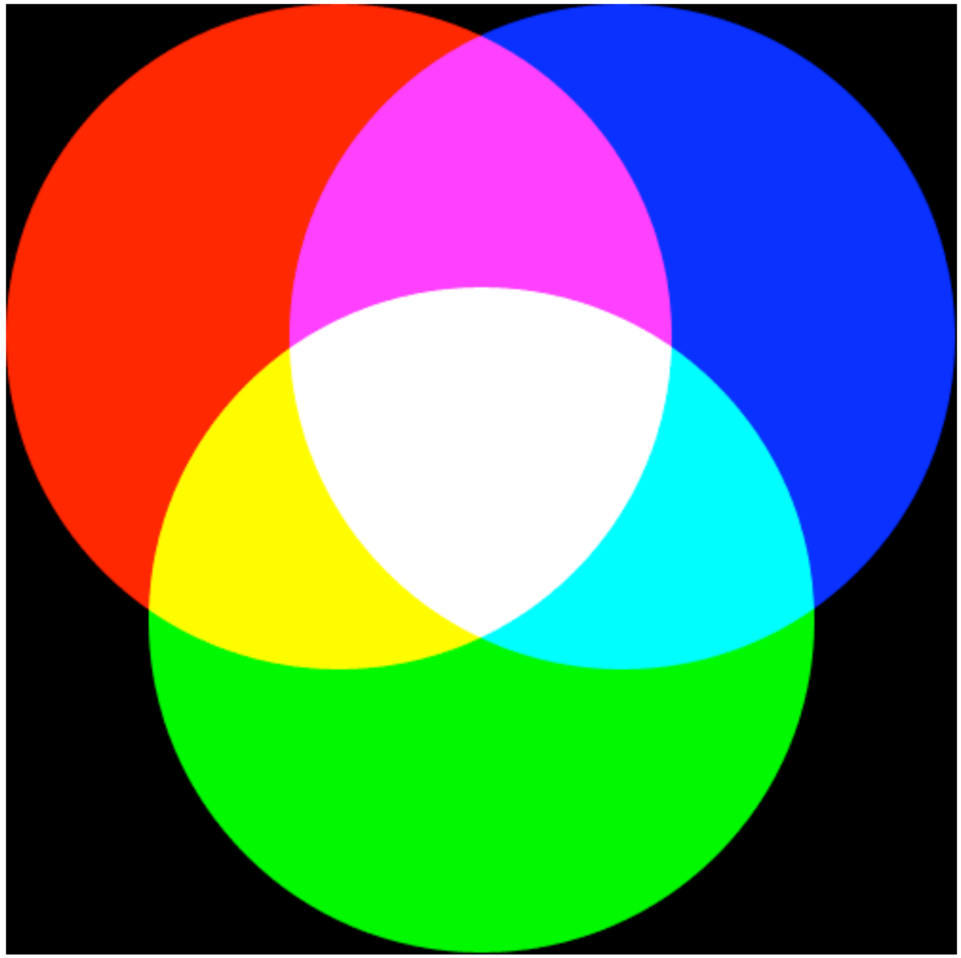


www.thestagecrew.com

Red, Green, and Blue Light



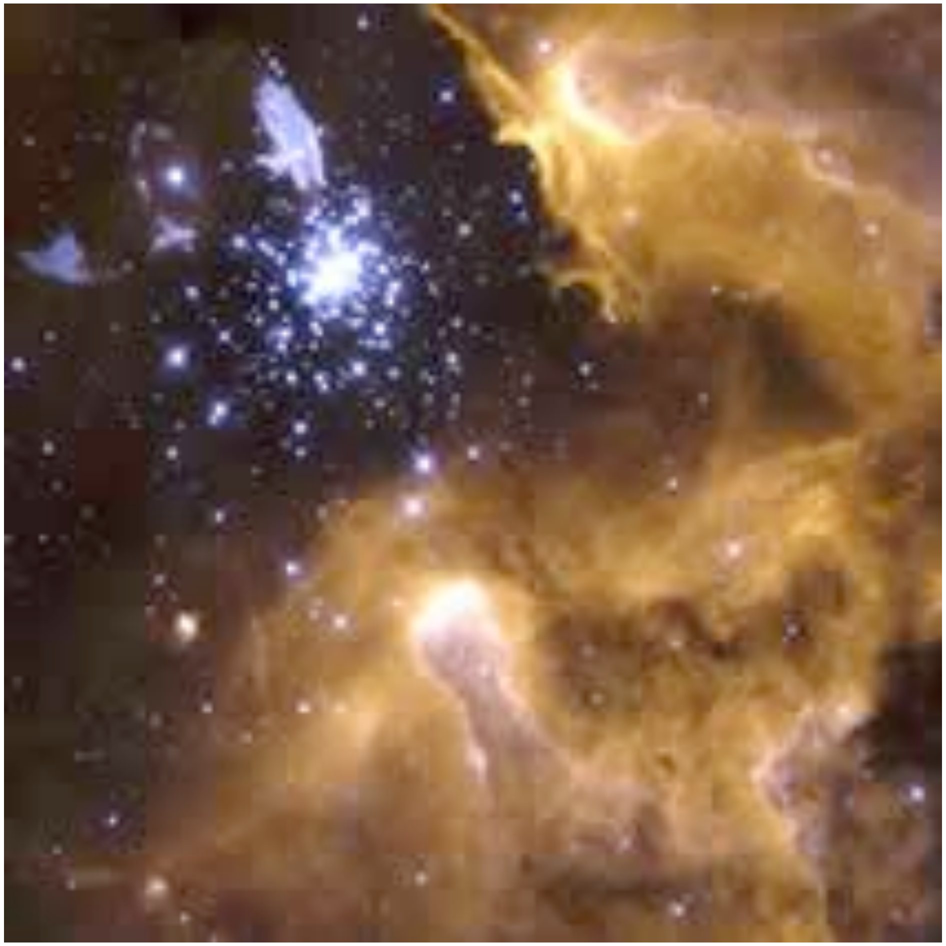
Adding R, G, and B Values



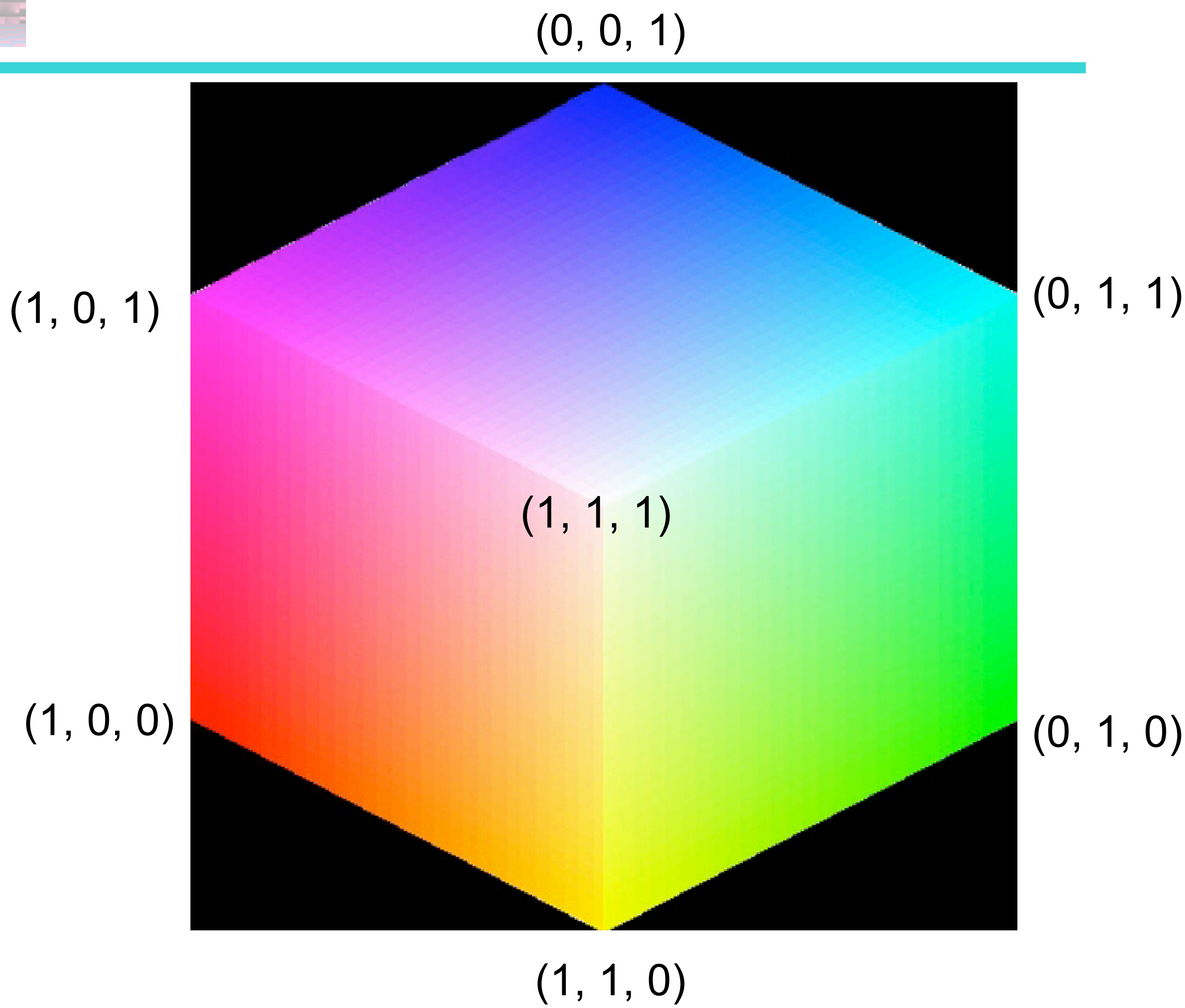
http://en.wikipedia.org/wiki/RGB

From the Hubble

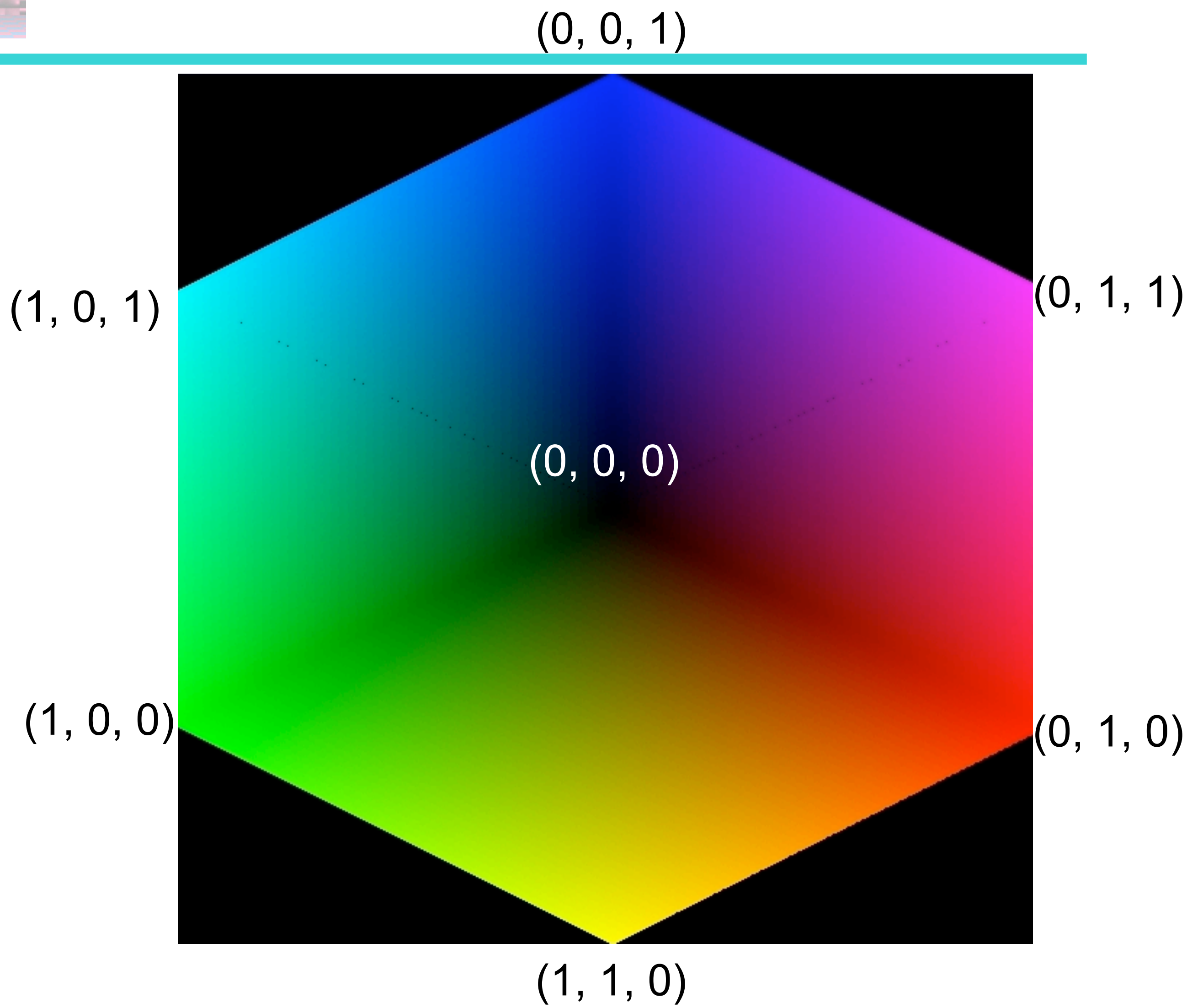
Hubble Site Link

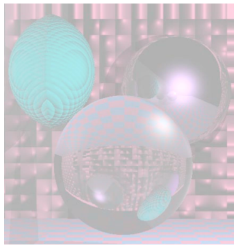


RGB Color Cube

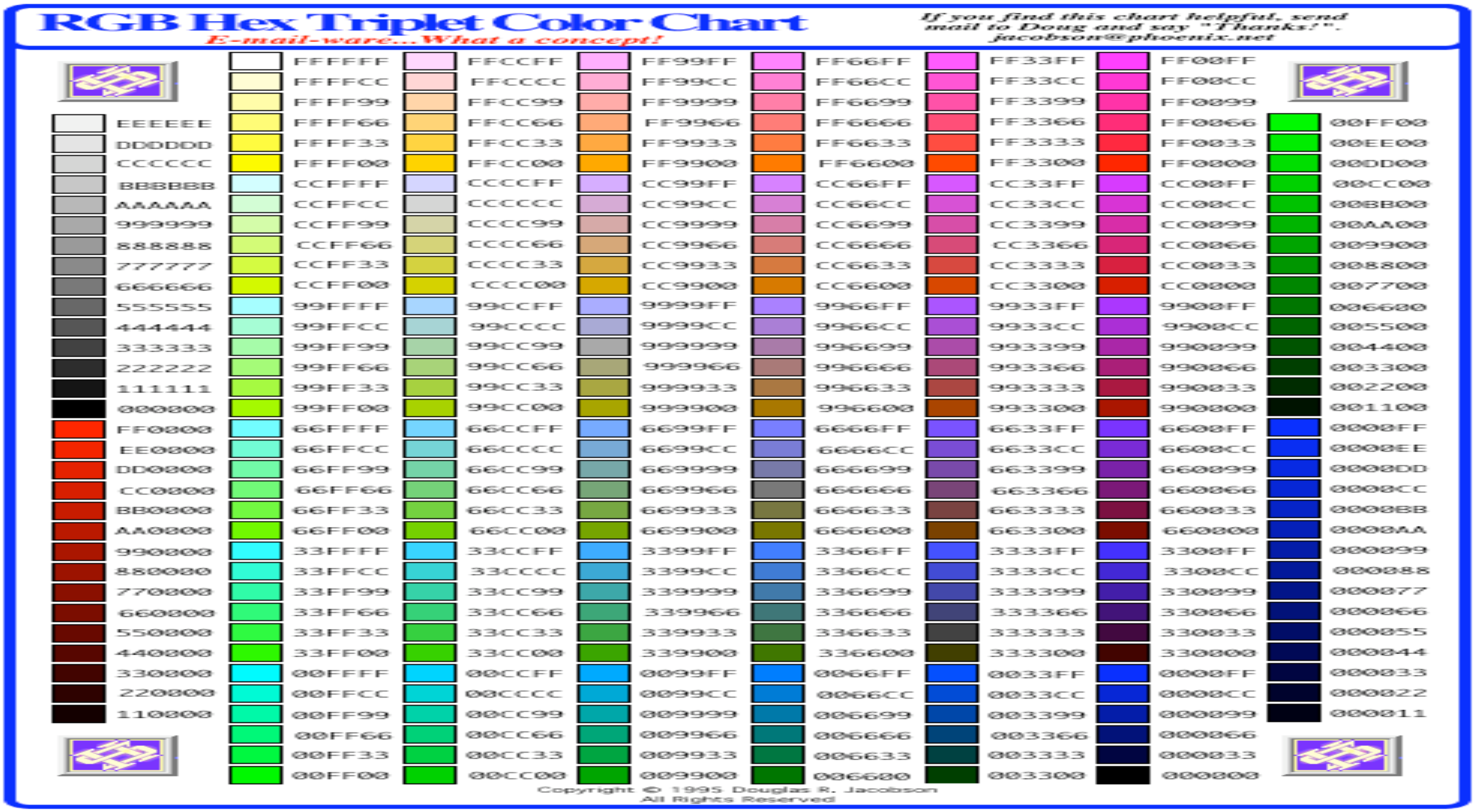


RGB Color Cube The Dark Side

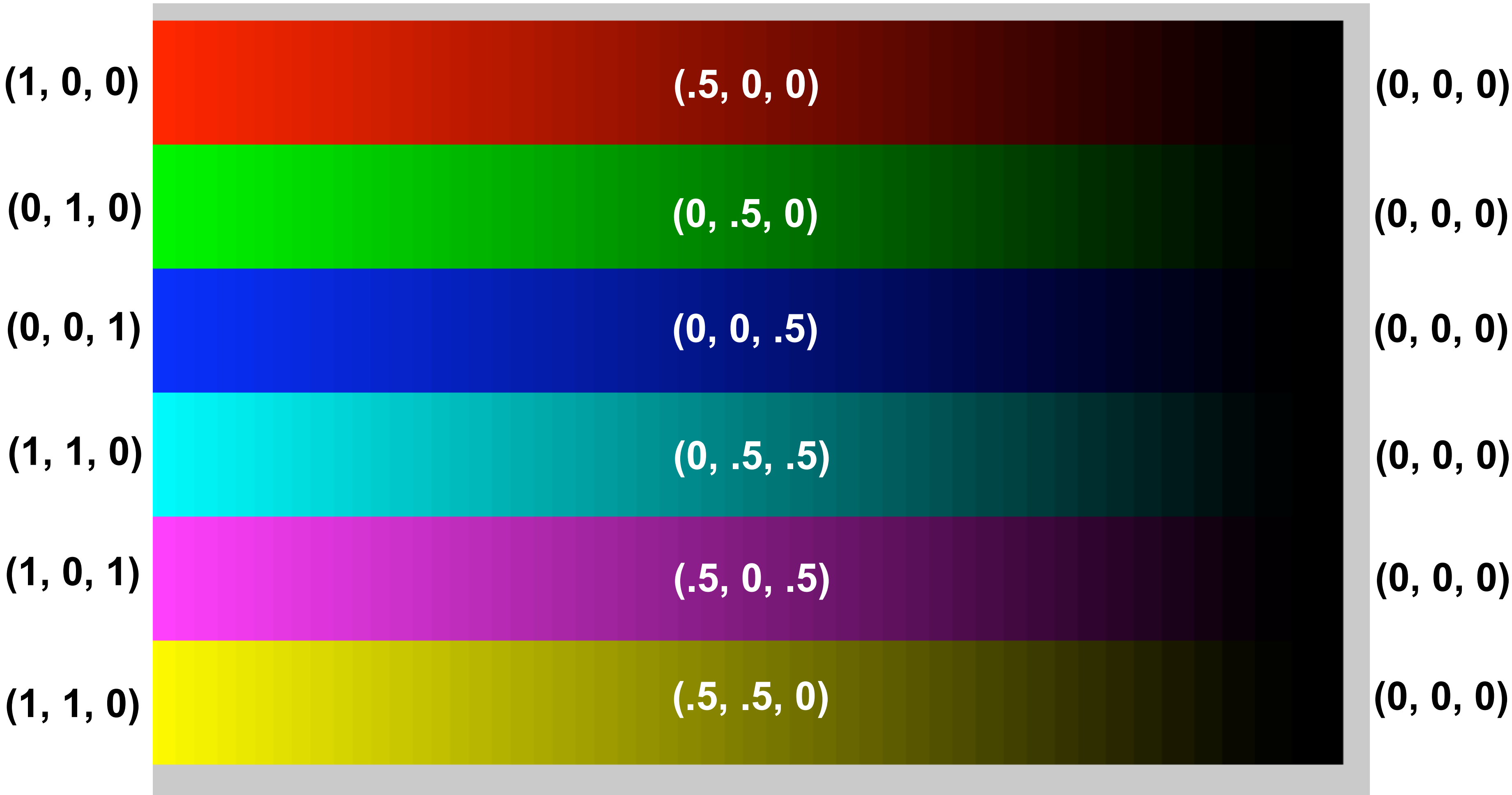




Doug Jacobson's RGB Hex Triplet Color Chart

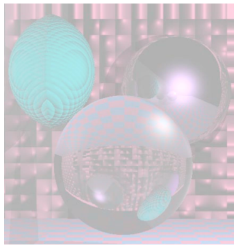


Making Colors Darker



Getting Darker, Left to Right

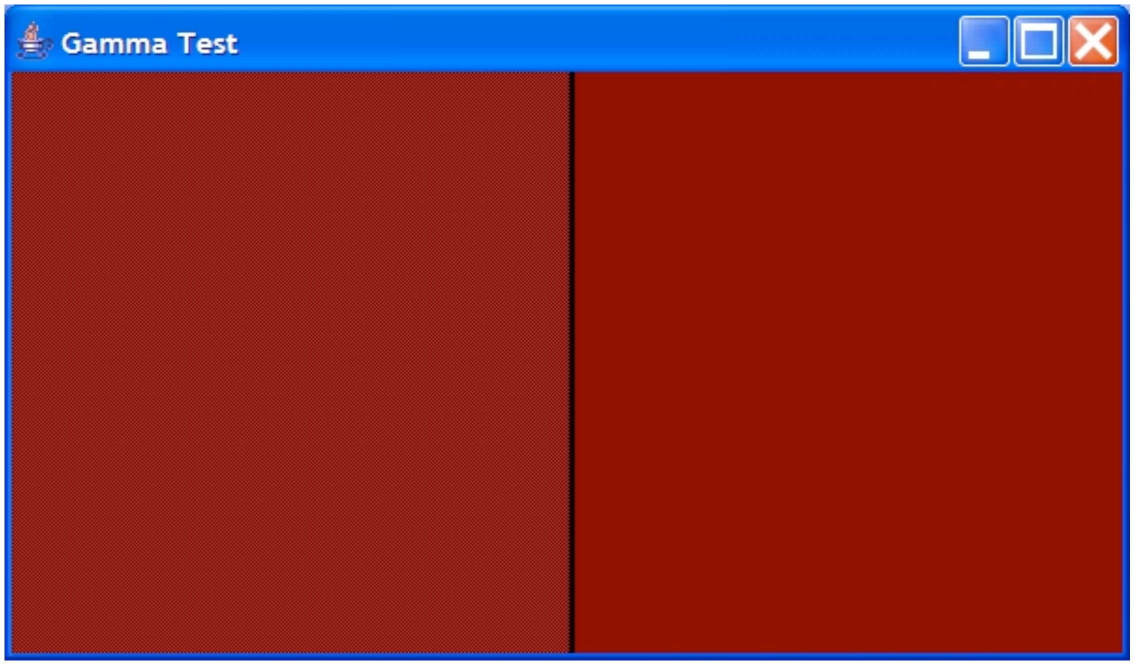
```
for (int b = 255; b >= 0; b--){
    c = new Color(b, 0, 0); g.setPaint(c);
    g.fillRect(800+3*(255-b), 50, 3, 150);
    c = new Color(0, b, 0); g.setPaint(c);
    g.fillRect(800+3*(255-b), 200, 3, 150);
    c = new Color(0, 0, b); g.setPaint(c);
    g.fillRect(800+3*(255-b), 350, 3, 150);
    c = new Color(0, b, b); g.setPaint(c);
    g.fillRect(800+3*(255-b), 500, 3, 150);
    c = new Color(b, 0, b); g.setPaint(c);
    g.fillRect(800+3*(255-b), 650, 3, 150);
    c = new Color(b, b, 0); g.setPaint(c);
    g.fillRect(800+3*(255-b), 800, 3, 150);
}
```



Gamma Correction

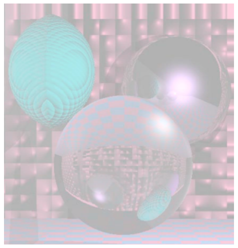
- Generally, the displayed intensity is not linear in the input ($0 \leq a \leq 1$).
- dispIntensity = (maxIntensity)a^{γ}
- To find γ
  - Find a that gives you .5 intensity
  - Solve $.5 = a^{\gamma}$
  - $\gamma = \dfrac{\ln(.5)}{\ln(a)}$

Gamma Correction

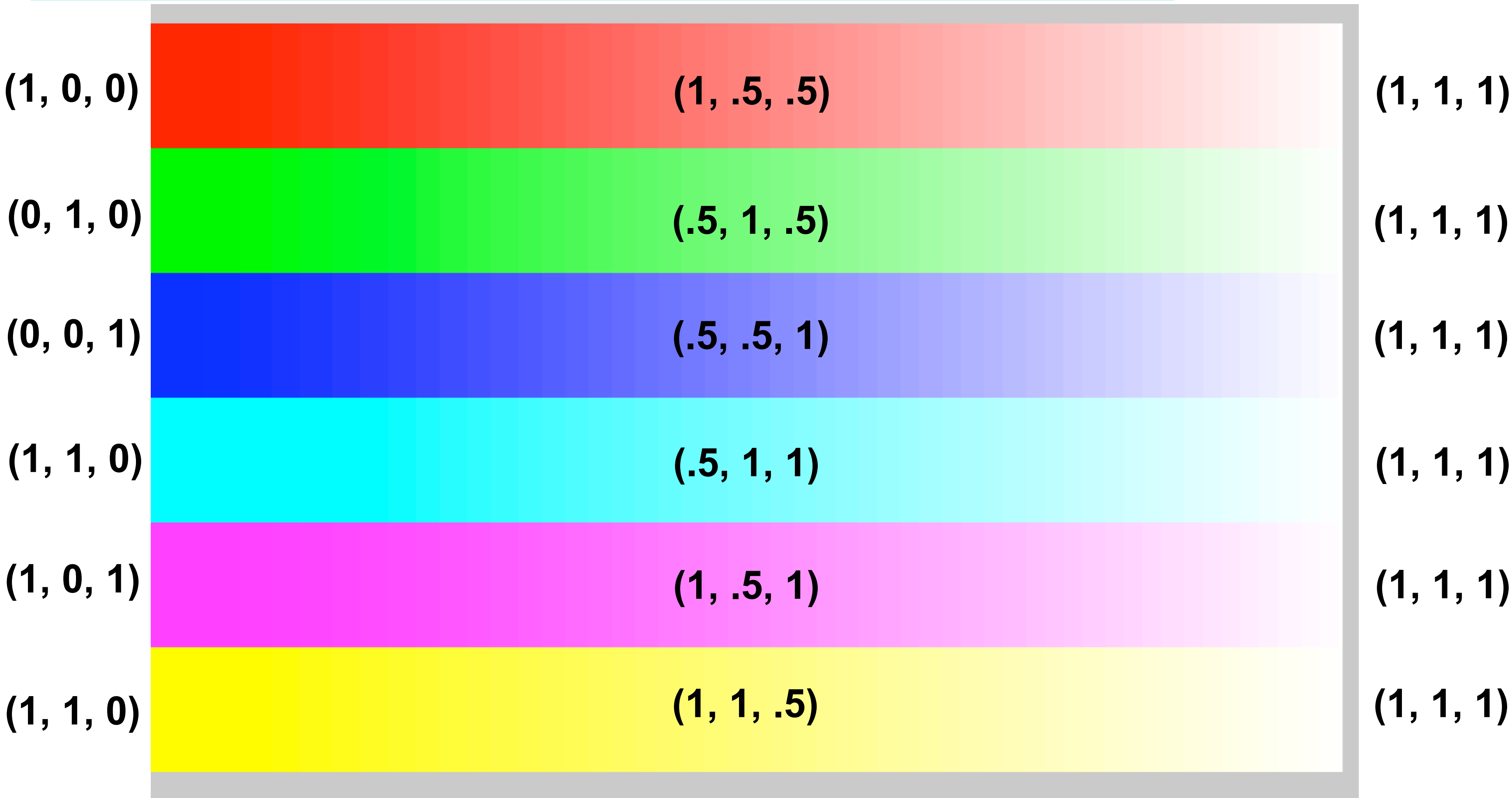


half black half red (127, 0, 0)

- Gamma

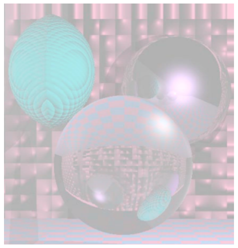


Making Pale Colors



Getting Paler, Left to Right

```
for (int w = 0; w < 256; w++){
        c = new Color(255, w, w); g.setPaint(c);
        g.fillRect(3*w, 50, 3, 150);
        c = new Color(w, 255, w); g.setPaint(c);
        g.fillRect(3*w, 200, 3, 150);
        c = new Color(w, w, 255); g.setPaint(c);
        g.fillRect(3*w, 350, 3, 150);
        c = new Color(w, 255, 255); g.setPaint(c);
        g.fillRect(3*w, 500, 3, 150);
        c = new Color(255,w, 255); g.setPaint(c);
        g.fillRect(3*w, 650, 3, 150);
        c = new Color(255, 255, w); g.setPaint(c);
        g.fillRect(3*w, 800, 3, 150);
}
```



Portable Pixmap Format (ppm)

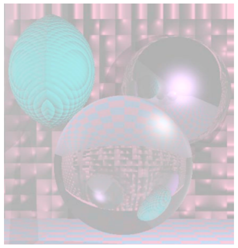
A "magic number" for identifying the file type.

- A ppm file's magic number is the two characters "P3".
- Whitespace (blanks, TABs, CRs, LFs).
- A width, formatted as ASCII characters in decimal.
- Whitespace.
- A height, again in ASCII decimal.
- Whitespace.
- The maximum color value again in ASCII decimal.
- Whitespace.
- Width * height pixels, each 3 values between 0 and maximum value.
  - start at top-left corner; proceed in normal English reading order
  - three values for each pixel for red, green, and blue, resp.
  - 0 means color is off; maximum value means color is maxxed out
  - characters from "#" to end-of-line are ignored (comments)
  - no line should be longer than 70 characters

ppm Example

```
P3
# feep.ppm
4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```

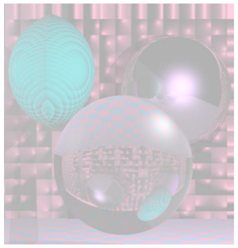
```
private void saveImage() {
    String outFileName = “my.ppm";
    File outFile = new File(outFileName);
    int clrR, clrG, clrB;
    try {
        PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(outFile)));
        out.println("P3");
        out.print(Integer.toString(xmax-xmin+1)); System.out.println(xmax-xmin+1);
        out.print(" ");
        out.println(Integer.toString(ymax-ymin+1)); System.out.println(ymax-ymin+1);
        out.println("255");
        for (int y = ymin; y <= ymax; y++){
            for (int x = xmin; x <= xmax; x++) {
                // compute clrR, clrG, clrB
                out.print(" "); out.print(clrR);
                out.print(" "); out.print(clrG);
                out.print(" "); out.println(clrB);
            }
        }
        out.close();
    } catch (IOException e) {
        System.out.println(e.toString());
    }
}
```



Math Basics

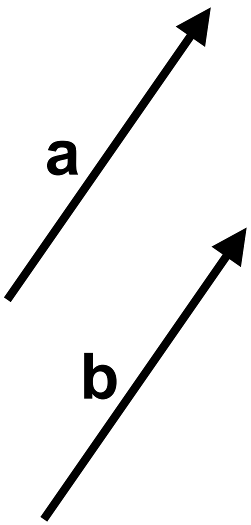
(All Readings from Shirley)

- Sets and Mappings – 2.1
- Quadratic Equations – 2.2
- Trigonometry – 2.3
- Vectors – 2.4
- 2D Parametric Curves – 2.6
- 3D Parametric Curves – 2.8
- Linear Interpolation – 2.10
- Triangles – 2.11



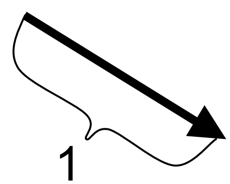
Vectors

- A *vector* describes a length and a direction.



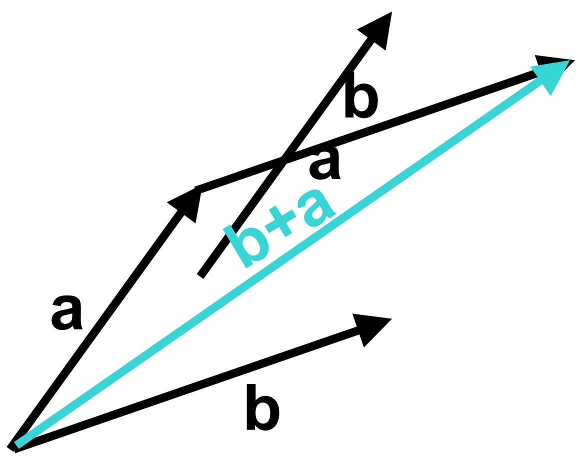
a = b

- a zero length vector

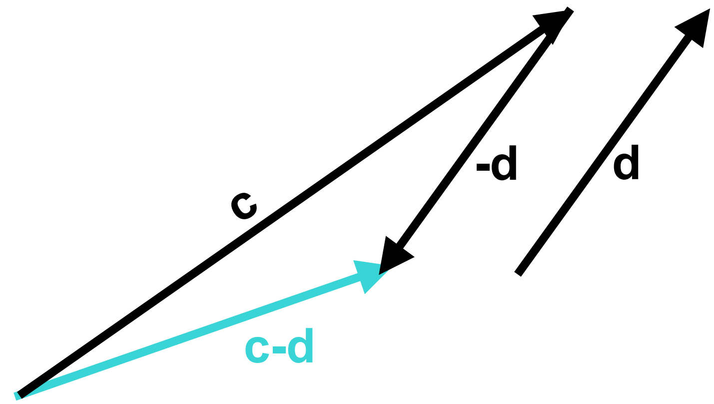


a unit vector

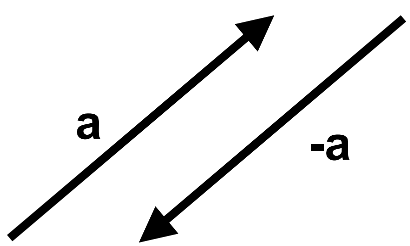
Vector Operations

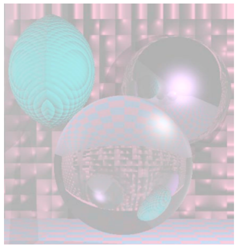


Vector Sum



Vector Difference





Cartesian Coordinates

- Any two non-zero, non-parallel 2D vectors form a *2D basis*.
- Any 2D vector can be written uniquely as a linear combination of two 2D basis vectors.
- **x** and **y** (or **i** and **j**) denote unit vectors parallel to the *x-axis* and *y-axis.*
- **x** and **y** form an *orthonormal* 2D basis.

$$a = x_a\mathbf{x} + y_a\mathbf{y}$$

$$a = (x_a, y_a) \quad \text{or} \quad a = \begin{bmatrix} x_a \\ y_a \end{bmatrix}$$

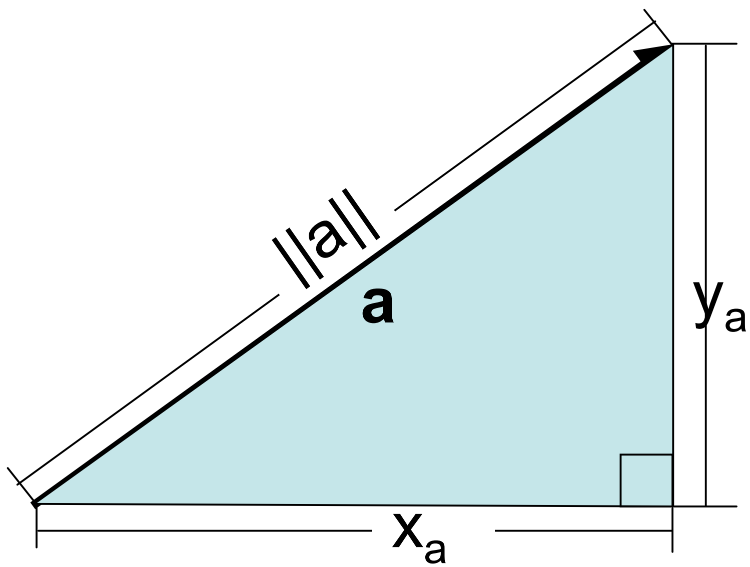
$$\text{or } a = (a_x, a_y)$$

- **x, y** and **z** form an *orthonormal* 3D basis.

Vector Length

Vector **a** =(x_a, y_a)

$$Length(\mathbf{a}) = Norm(\mathbf{a}) = \|\mathbf{a}\| = \sqrt{x_a^2 + y_a^2}$$



Dot Product

Dot Product

$\mathbf{a} = (x_a, y_a)$ $\mathbf{b} = (x_b, y_b)$

$\mathbf{a} \cdot \mathbf{b} = x_a x_b + y_a y_b$

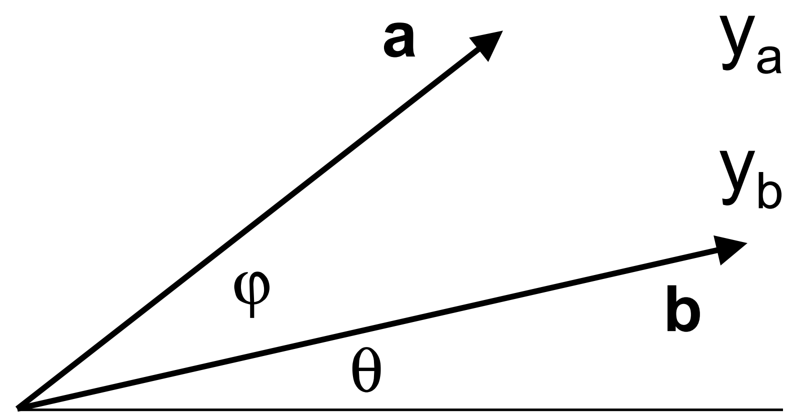
$\mathbf{a} \cdot \mathbf{b} = ||\mathbf{a}|| \cdot ||\mathbf{b}|| \cos(\varphi)$

$x_a = ||\mathbf{a}|| \cos(\theta + \varphi)$

$x_b = ||\mathbf{b}|| \cos(\theta)$

$y_a = ||\mathbf{a}|| \sin(\theta + \varphi)$

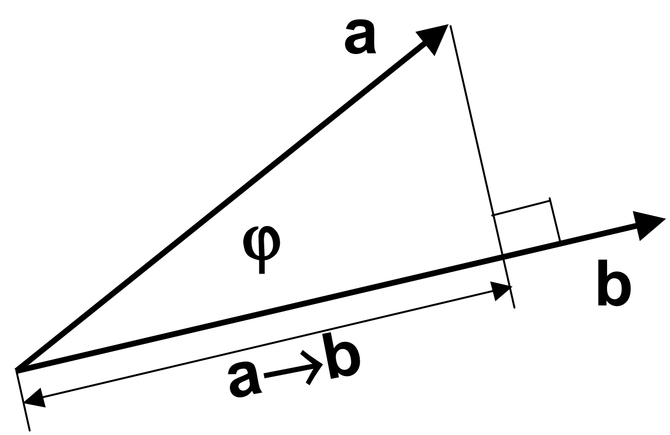
$y_b = ||\mathbf{b}|| \sin(\theta)$



Projection

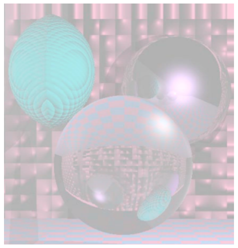
a =(x_a, y_a) **b** =(x_b, y_b)

a·**b** = ||**a**||·||**b**||cos(φ)



The length of the projection of **a** onto **b** is given by

$$a \to b = \|a\| \cos(\varphi) = \frac{a \bullet b}{\|b\|}$$



3D Vectors

This all holds for 3D vectors too.

$\mathbf{a} = (x_a, y_a, z_a)$ $\mathbf{b} = (x_b, y_b, z_b)$

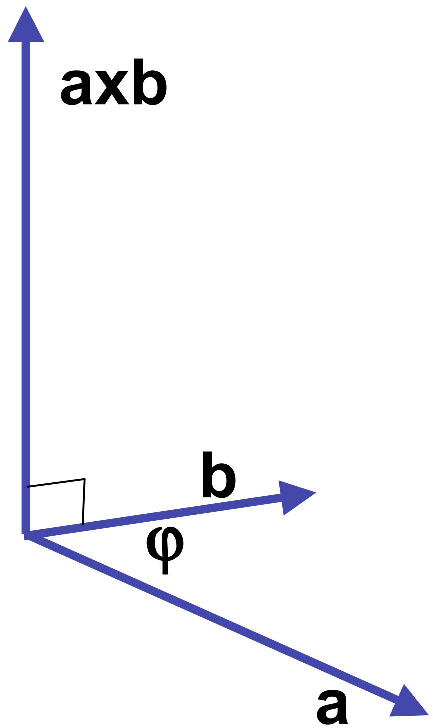
$$Length(\mathbf{a}) = Norm(\mathbf{a}) = \|\mathbf{a}\| = \sqrt{x_a^2 + y_a^2 + z_a^2}$$

$$\mathbf{a} \cdot \mathbf{b} = x_a x_b + y_a y_b + z_a z_b$$

$$\mathbf{a} \cdot \mathbf{b} = ||\mathbf{a}|| \cdot ||\mathbf{b}|| \cos(\varphi)$$

$$\mathbf{a} \rightarrow \mathbf{b} = \|\mathbf{a}\| \cos(\varphi) = \frac{\mathbf{a} \bullet \mathbf{b}}{\|\mathbf{b}\|}$$

Vector Cross Product



$$\|\mathbf{a} \times \mathbf{b}\| = \|\mathbf{a}\| \, \|\mathbf{b}\| \sin \varphi$$

axb is perpendicular to **a** and **b**.

Use the right hand rule to determine the direction of **axb.**

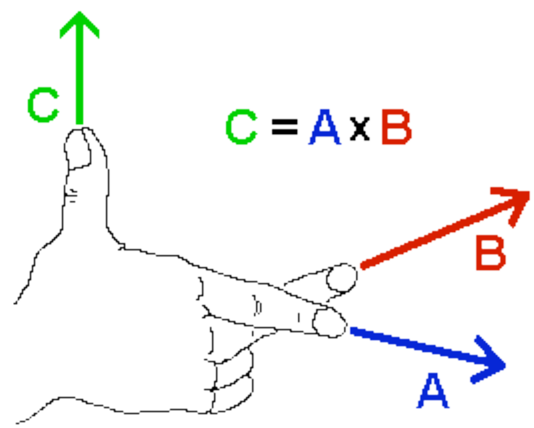
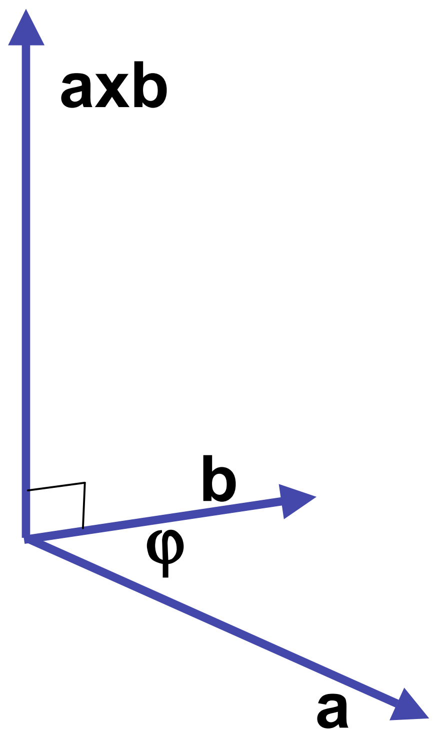
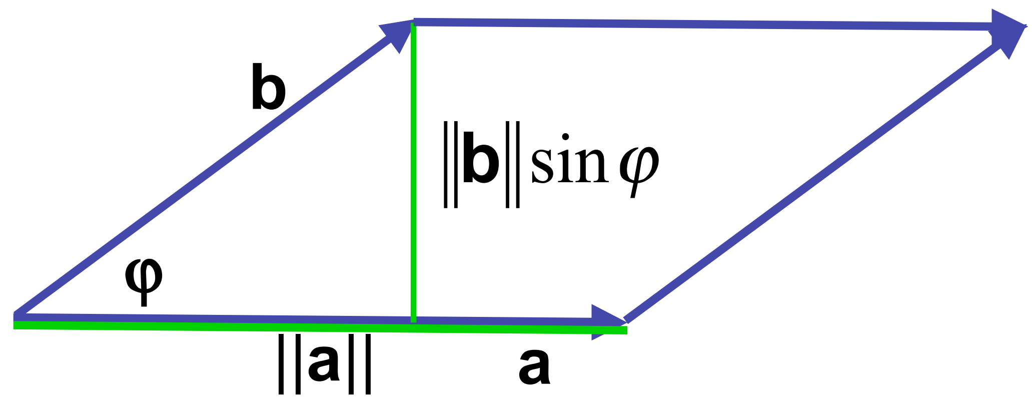


Image from www.physics.udel.edu

Cross Product and Area



$$\|\mathbf{a} \times \mathbf{b}\| = \|\mathbf{a}\| \, \|\mathbf{b}\| \sin\varphi$$

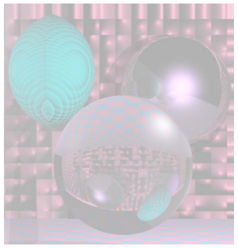


||**a**||x||**b**|| = area of the parallelogram.

Computing the Cross Product

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} i & j & k \\ \mathrm{a}_x & \mathrm{a}_y & \mathrm{a}_z \\ \mathrm{b}_x & \mathrm{b}_y & \mathrm{b}_z \end{vmatrix}$$

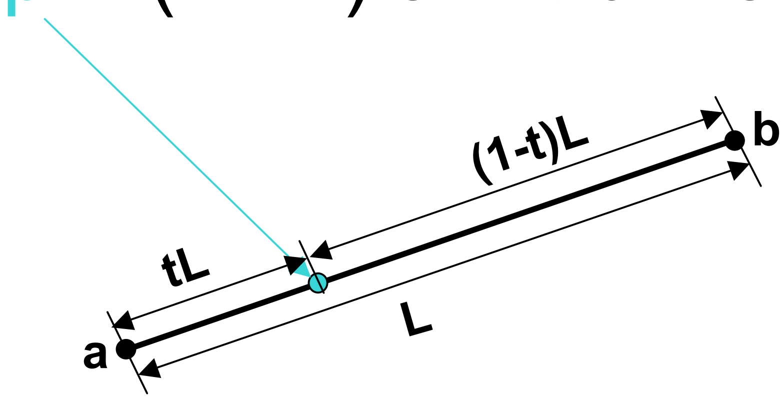
$$= \left(\mathrm{a}_y \mathrm{b}_z - \mathrm{a}_z \mathrm{b}_y\right) i + \left(\mathrm{a}_z \mathrm{b}_x - \mathrm{a}_x \mathrm{b}_z\right) j + \left(\mathrm{a}_x \mathrm{b}_y - \mathrm{a}_y \mathrm{b}_x\right) k$$



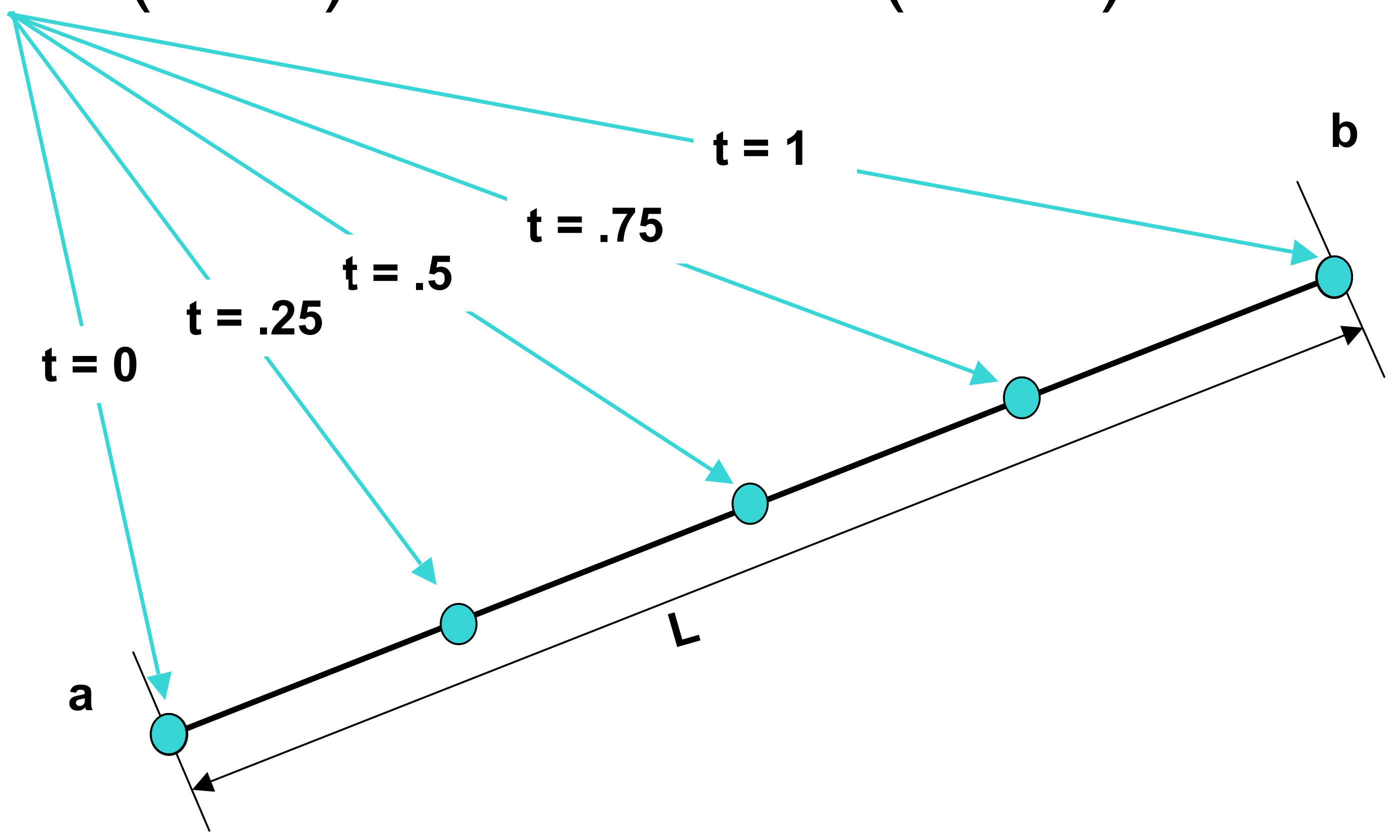
Linear Interpolation

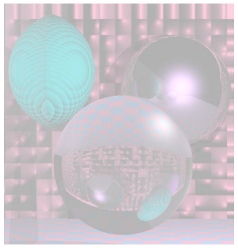
- **LERP**: /lerp/, vi.,n.
  - Quasi-acronym for Linear Interpolation, used as a verb or noun for the operation. “Bresenham's algorithm lerps incrementally between the two endpoints of the line.”

$$p = (1 – t)\ a + t\ b = a + t(b – a)$$

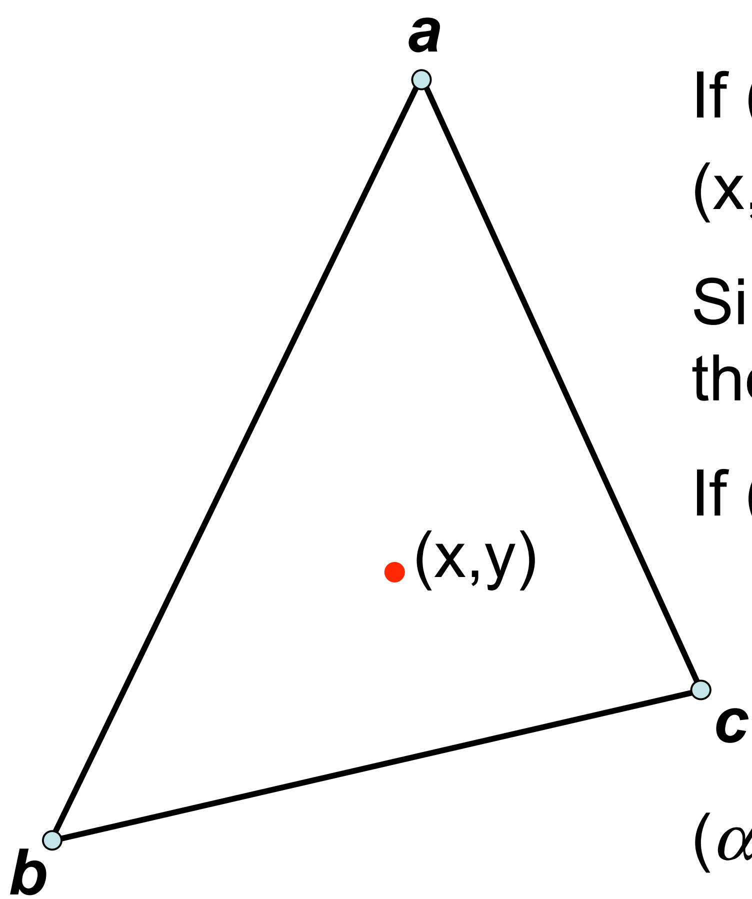


$$p = (1 - t)\, a + t\, b = a + t(b - a)$$





Triangles



If (x, y) is on the edge ab,

(x, y) = (1 – t) a + t b = a + t(b – a).

Similar formulas hold for points on the other edges.

If (x, y) is in the triangle:

$$(x, y) = \alpha\, a + \beta\, b + \gamma\, c$$

$$\alpha + \beta + \gamma = 1$$

(α, β, γ) are the

Barycentric coordinates of (x, y).

Triangles

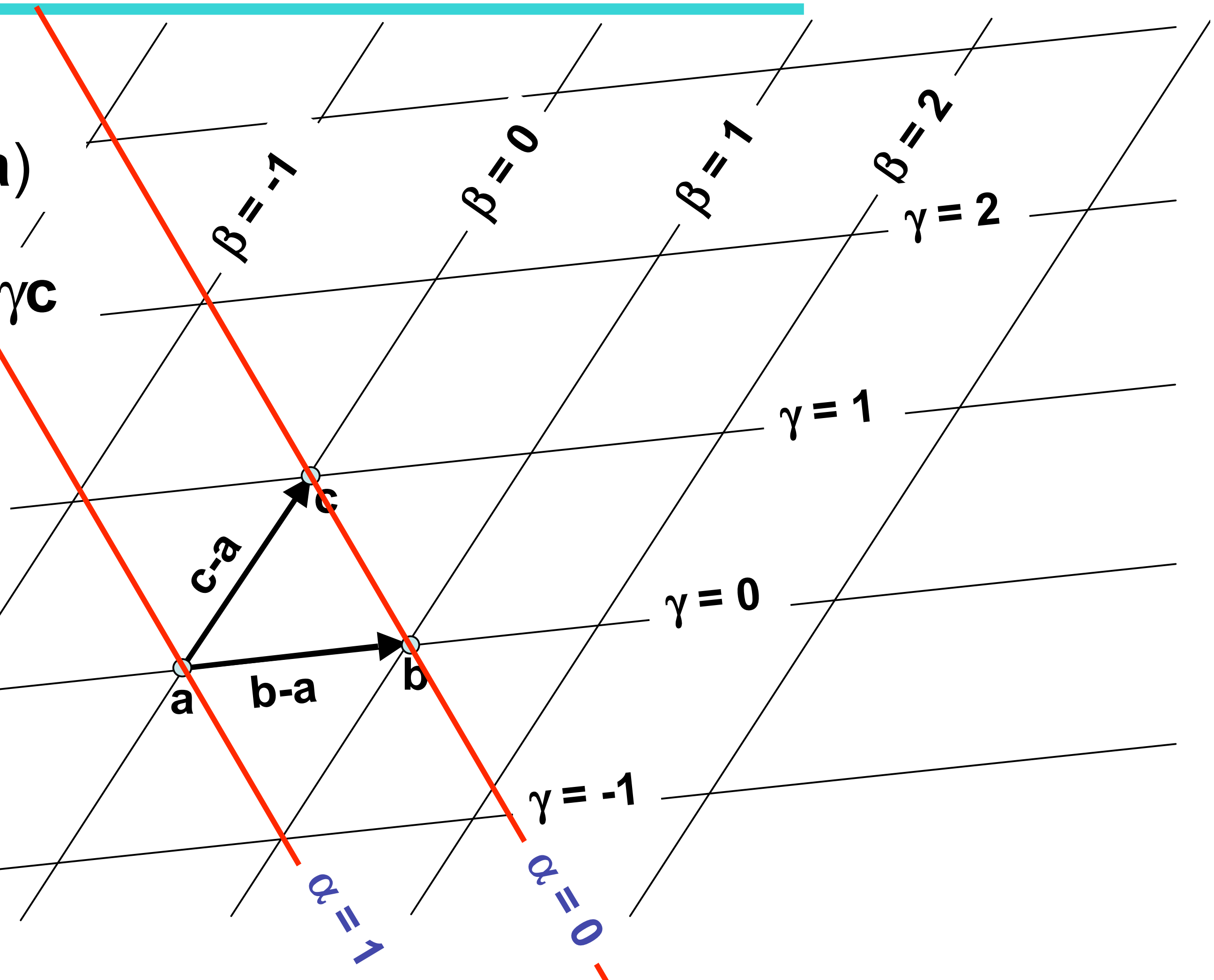
$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b}-\mathbf{a}) + \gamma(\mathbf{c}-\mathbf{a})$$

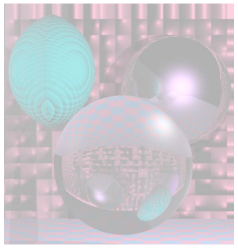
$$\mathbf{p} = (1-\beta-\gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$

$$\alpha = 1-\beta-\gamma$$

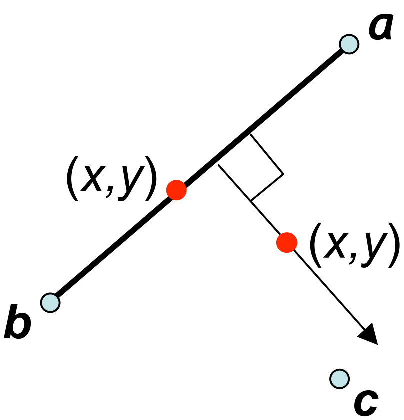
$$\mathbf{p} = \mathbf{p}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$

Barycentric coordinates





Computing Barycentric Coordinates



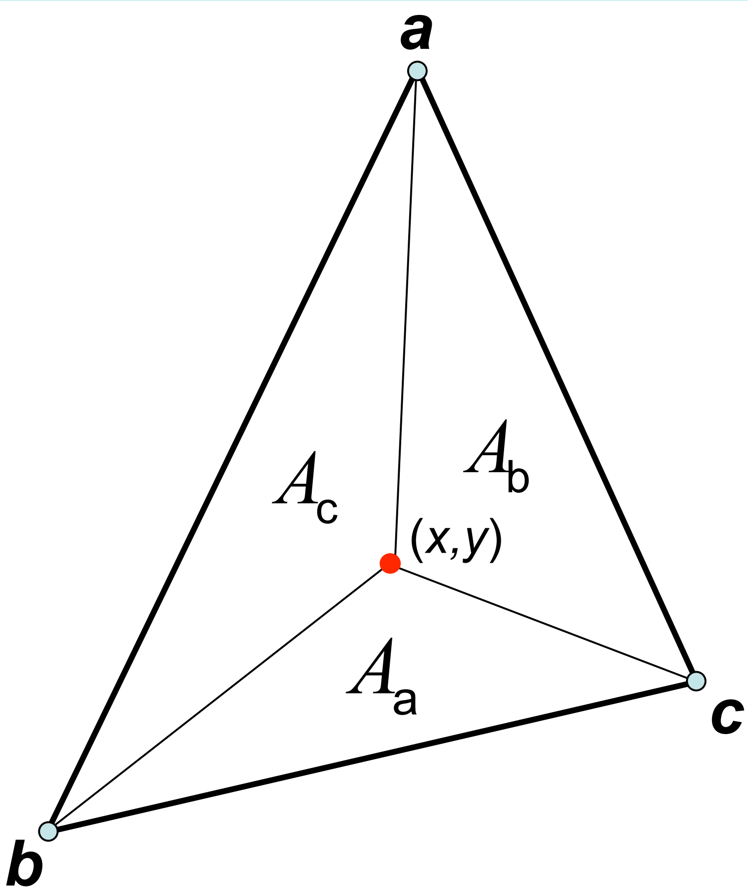
$$\frac{y - y_a}{x - x_a} = \frac{y_b - y_a}{x_b - x_a}$$

$$(y - y_a)(x_b - x_a) - (y_b - y_a)(x - x_a) = 0$$

$$f_{ab}(x, y) = (y - y_a)(x_b - x_a) - (y_b - y_a)(x - x_a)$$

$$\gamma = \frac{f_{ab}(x, y)}{f_{ab}(x_c, y_c)}$$

Barycentric Coordinates as Areas



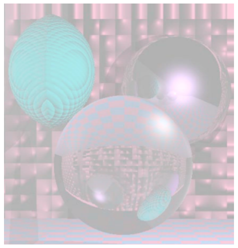
$$\alpha = A_a / A$$

$$\beta = A_b / A$$

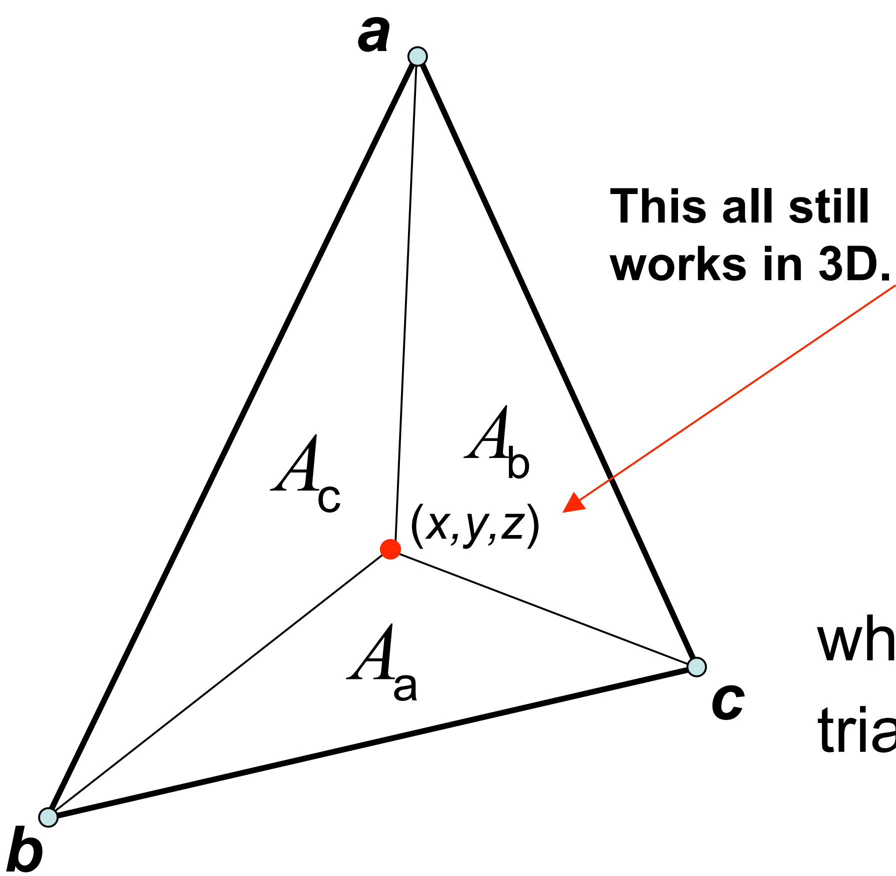
$$\gamma = A_c / A$$

where A is the area of the triangle.

$$\alpha + \beta + \gamma = 1$$



3D Triangles



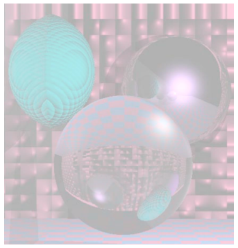
$$\alpha = A_a / A$$

$$\beta = A_b / A$$

$$\gamma = A_c / A$$

where *A* is the area of the triangle.

$$\alpha + \beta + \gamma = 1$$



Assignment 0

- You will choose a programming platform for the quarter and familiarize yourself with RGB color and the ppm format. In part, this assignment is to ensure that you have a method of submitting you work so that I can:
  - read the code
  - compile (or interpret) the code
  - run the code to produce a file in ppm format.
- Sample Program
- You will write your own 3D vector tools (e.g. as a JAVA class) that you will use for your later programming assignments.