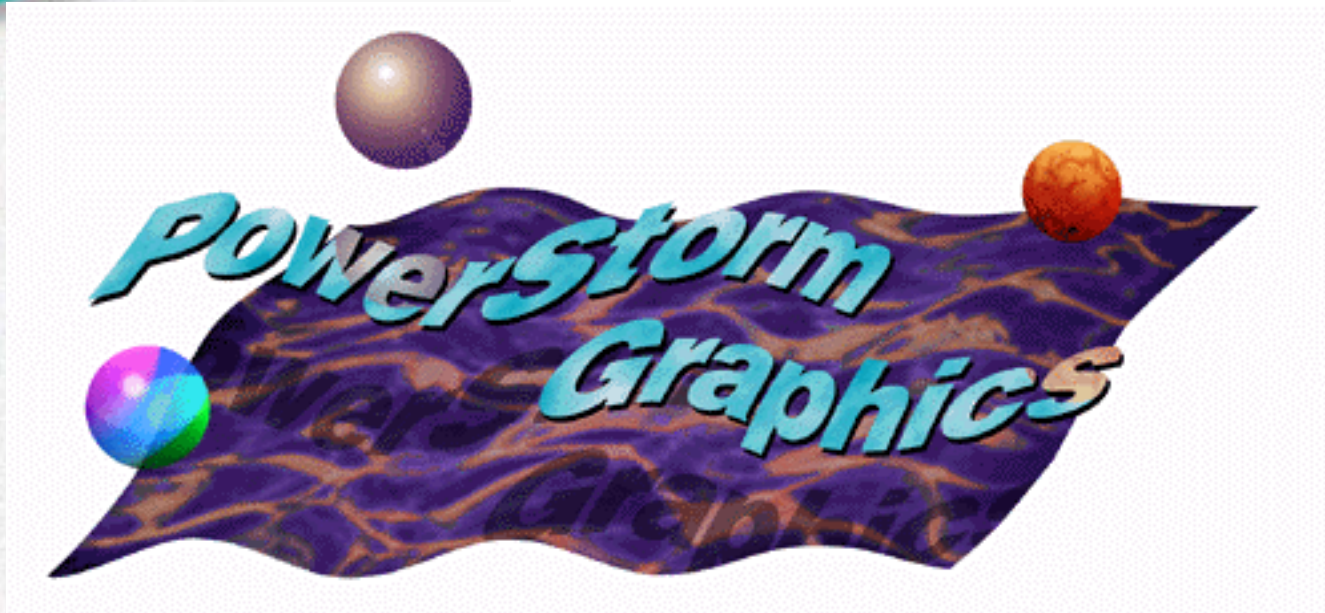


digital™



“Graphics 201”

Russell Doty
Graphics Product Manager



This Session

- Provides a complete foundation in 3D graphics
- Begins with fundamental concepts and provides theoretical basis for graphics
- Goes into considerable technical depth!
- After this session:
 - You should understand *how* 3D graphics works
 - You should be able to explain Digital's products
 - You should be strong competitively

I Have Good News and Bad News

- **The Bad News:**
 - We will go into enough depth to include mathematical equations and theoretical physics!
- **The Good News:**
 - I've left out the integrations and derivations

Subjects

- Introduction
- Ray Tracing
- Texturing
- Graphics Pipeline
- Performance Optimizations
- Color
- Aliasing and Anti-Aliasing
- PowerStorm Graphics
- Road Warrior (Megatek)
- Short Subjects

Kp vt q f w e vkq p

General Productivity Graphics

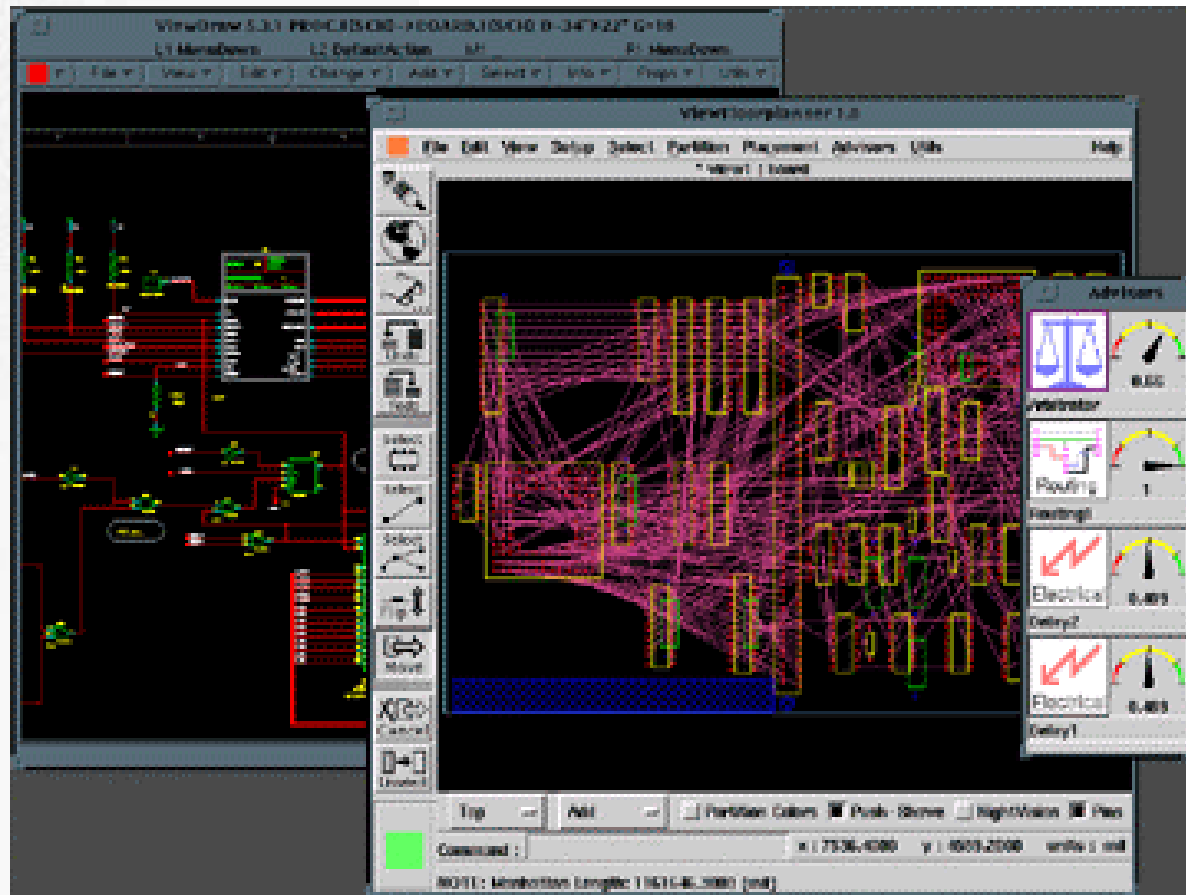
The screenshot displays a Windows desktop with four overlapping application windows:

- Microsoft Access:** Shows a 'Main Switchboard' for 'CatEyes Board Mana' with options like 'Enter/Update Board Data', 'Reports', and 'Exit Application'.
- Microsoft Word - PowerStorm_Whitepaper.doc:** Displays a document with the 'digital' logo and the title 'PowerStorm Graphics for Alpha Workstations'. The text below reads: 'White paper: A full family of competitive solutions for markets world'.
- Microsoft Project - CATLAUNCH.MPP:** Shows a Gantt chart with a task list table. The table includes columns for Task Name, Duration, and Start date.
- Microsoft PowerPoint - [PWRGRAFX.PPT]:** Shows a slide with the 'PowerStorm Graphics' logo.

Task Name	Duration	Start
1 Announcement Criteria	30d	Mon 5/20/94
2 Develop Criteria	1w	Mon 5/20/94
3 Initial Status Review	0d	Mon 6/17/94
4 Final Status Review	1d	Fri 6/28/94
5 Content Development	10d	Mon 5/13/94
6 Product Descriptions	10d	Mon 5/13/94
7 Key Applications	10d	Mon 5/13/94
8 Key Competitive Comparisons	5d	Mon 5/20/94
9 Target Markets	5d	Mon 5/20/94
10 Value Proposition	5d	Mon 5/13/94

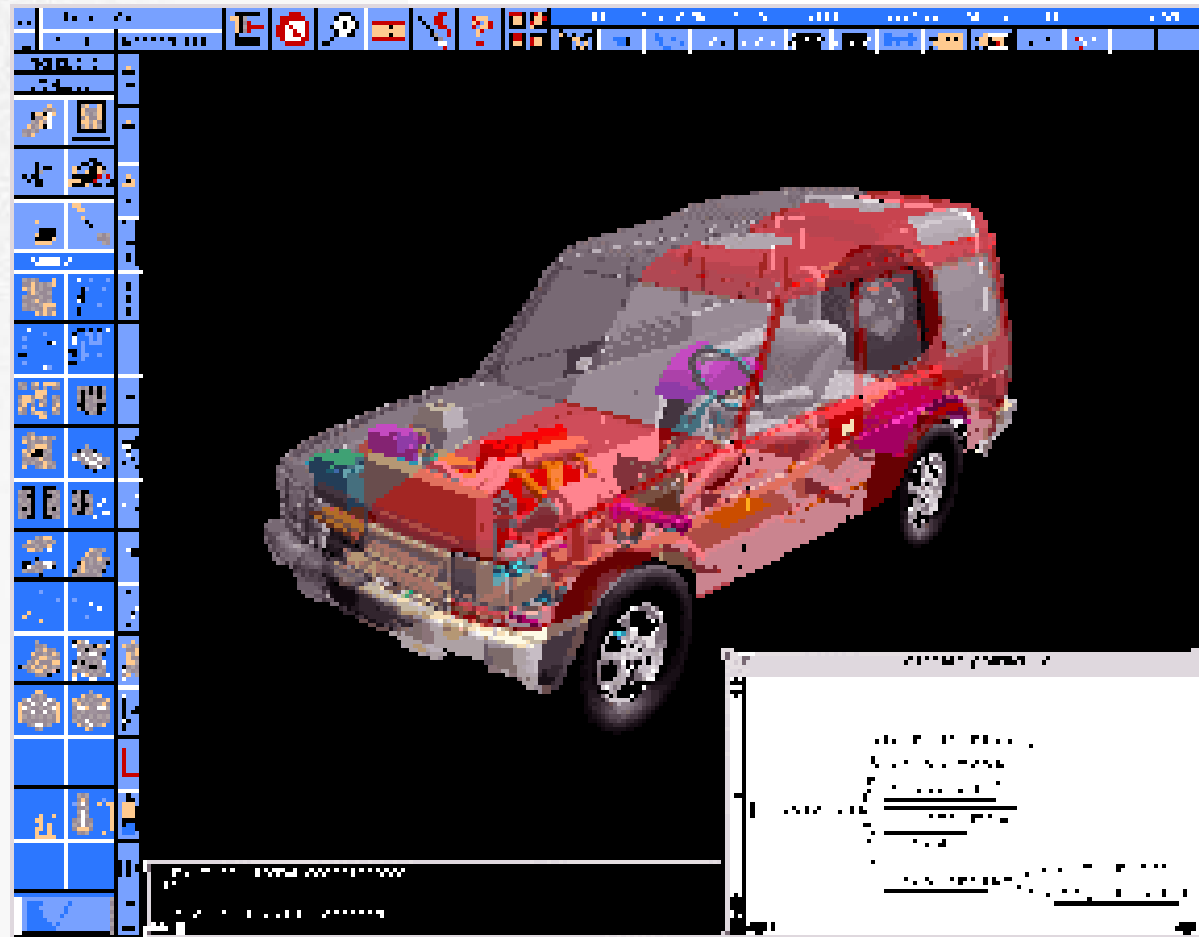
digital™

2D: Electrical CAD



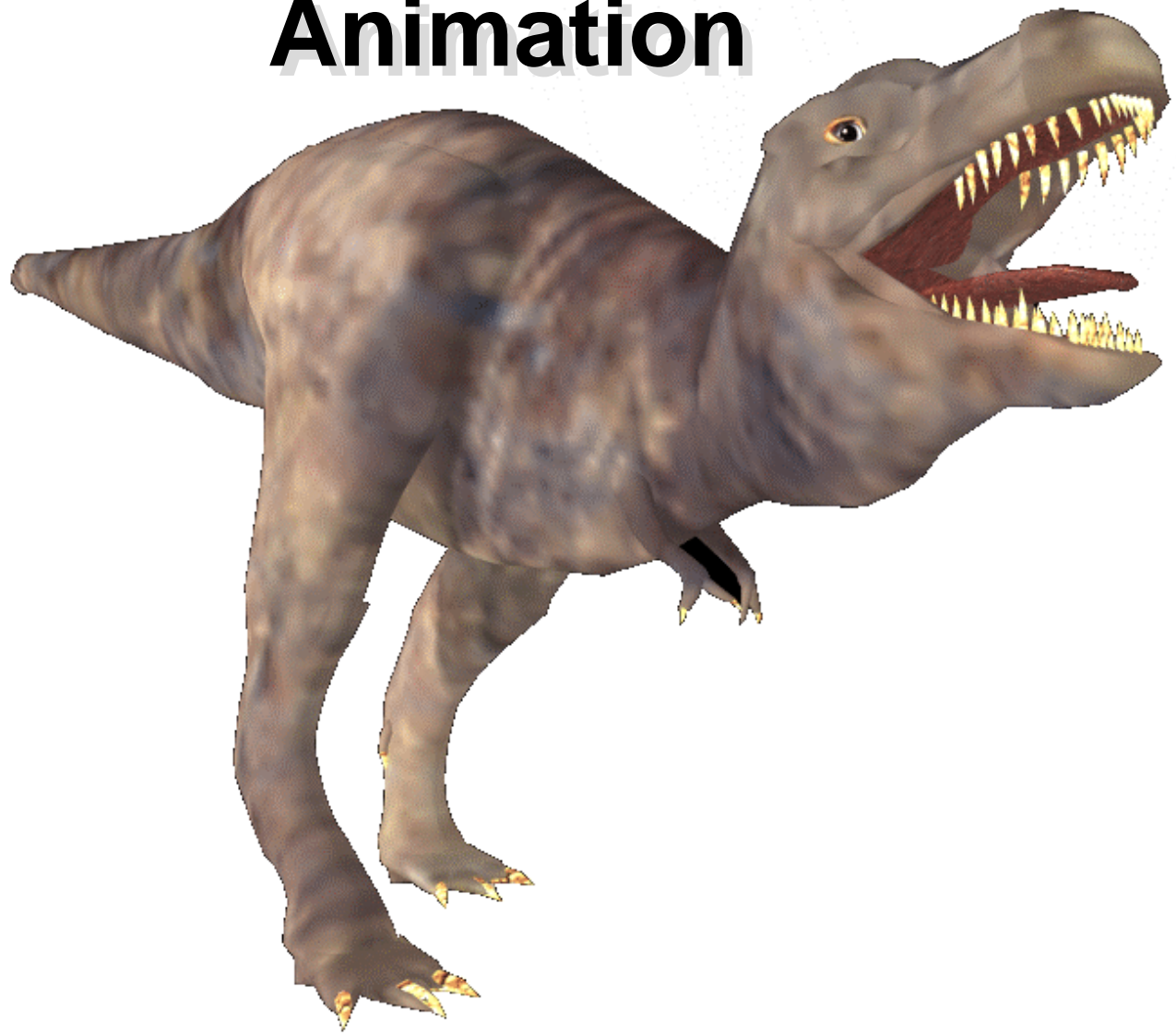
digital™

3D CAD



digital™

Animation



Graphics Training

digital™

T c { V t c e k p i



The Problem

- **Calculate the color and brightness of a pixel, based on the reflection of light from a 3D part (represented by a computer model)**
 - **Optionally include advanced effects**

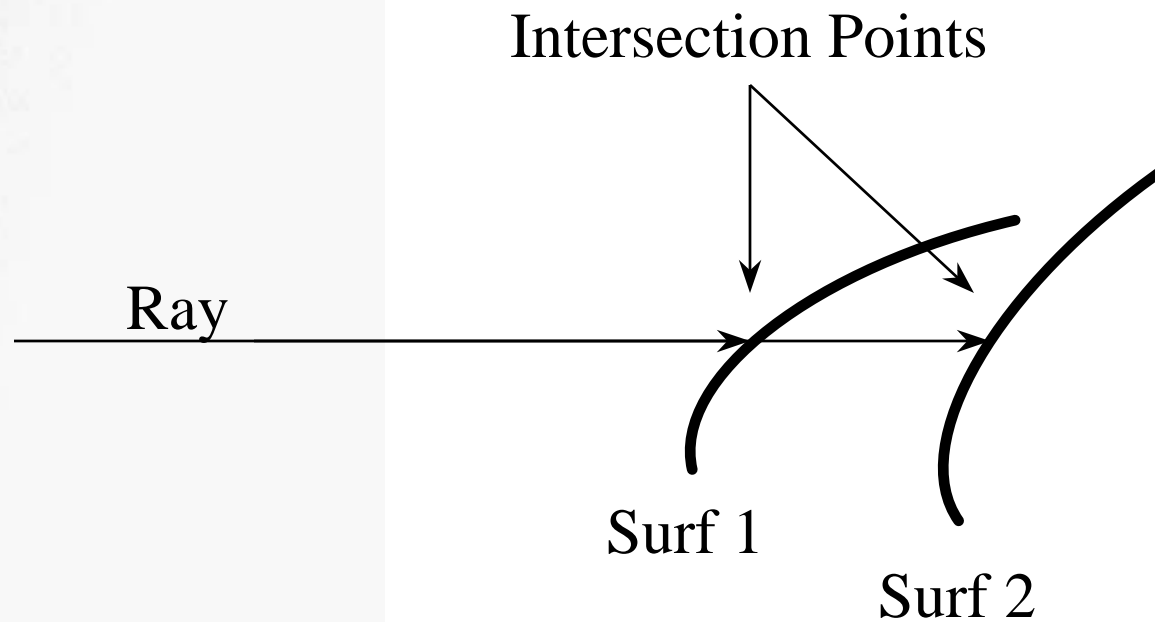
Ray Tracing

- The earliest method of generating shaded images of 3D models
- Advanced variations still used to produce the most sophisticated images and effects
- Hardware accelerated 3D graphics uses concepts *similar to ray tracing*

Theory of Ray Tracing

- **Raster images based on Pixels, which have area**
- **The value of an individual pixel can be approximated by calculating the value for a mathematical point at the center of the pixel and extrapolating to entire pixel**
- **This reduces the problem from a double integration to a point calculation (and produces aliasing, which will be examined later)**

Ray Casting



Ambient Light

$$I = I_a k_a$$

I = Intensity (of light)

I_a = Intensity of ambient light

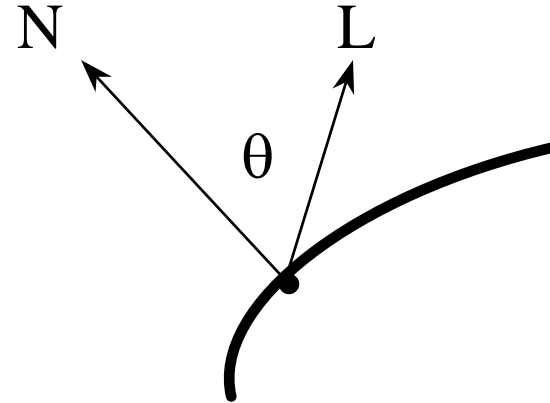
k_a = Ambient reflection coefficient

Diffuse Reflection

- **Models reflection of light from non-shiny surfaces**
 - eg: chalk
- **Variables:**
 - Light direction and intensity
 - Surface reflectivity coefficient
 - Surface normal
- **Brightness independent of viewing angle**
- **Smooth fall-off in brightness**

Diffuse Reflection

$$I = I_p K_d \cos \theta$$



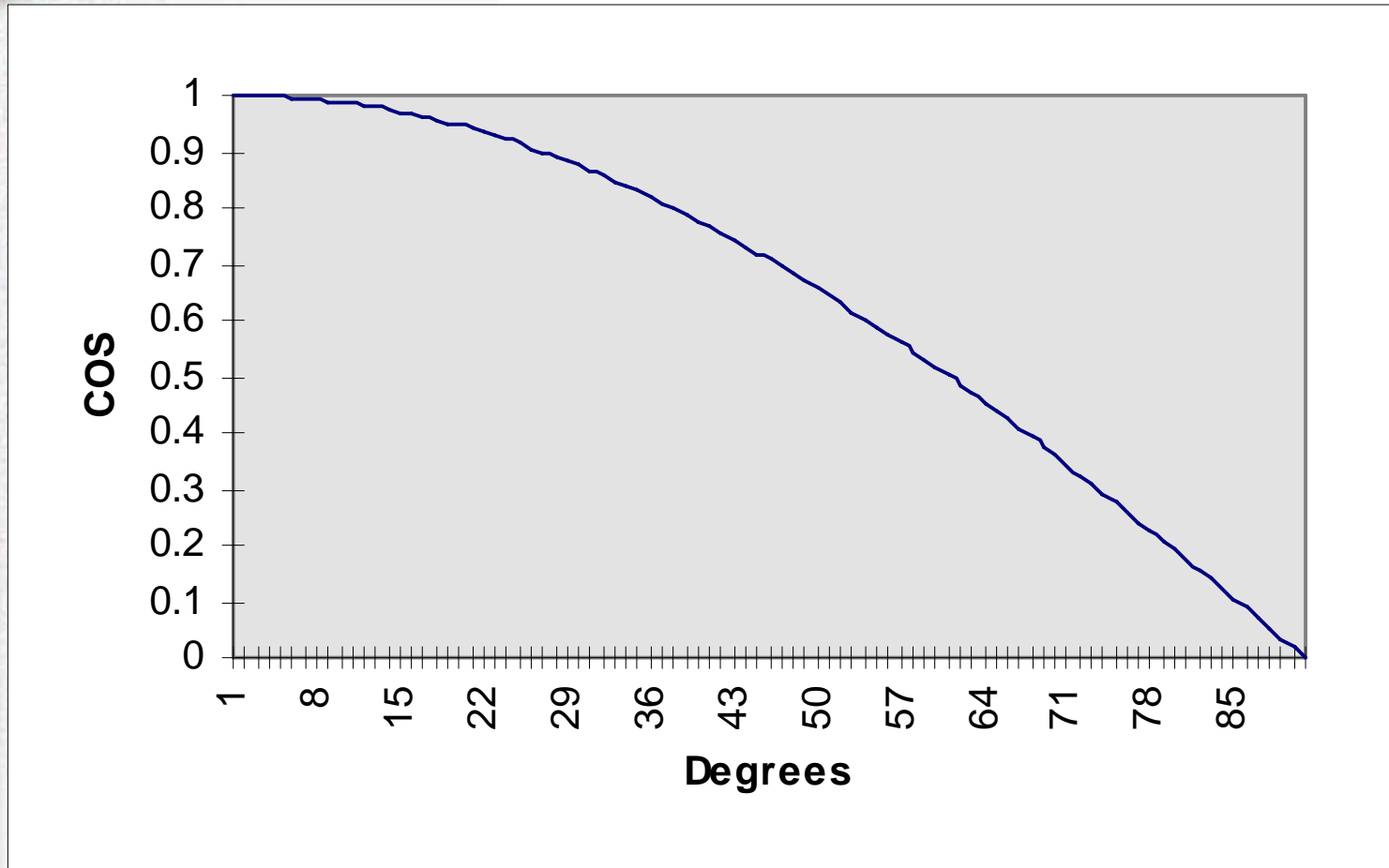
I = Intensity

I_p = Point Light Intensity (source)

K_d = Diffuse Reflection Coefficient

θ = Angle between surface normal
and light source

Lambertian Shading



Graph of $Y = \cos(\theta)$

Graphics Training

Diffuse Reflection for Colored Surface

$$I_r = I_p K_{dr} \cos \theta$$

$$I_g = I_p K_{dg} \cos \theta$$

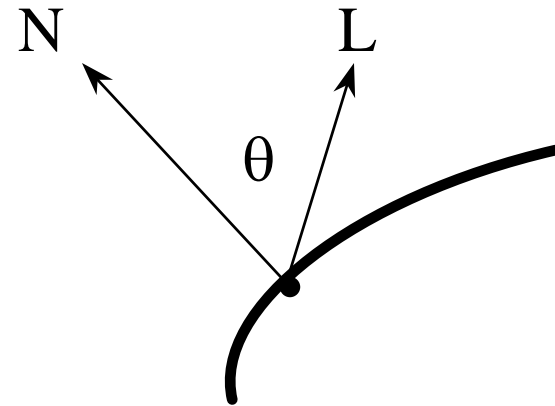
$$I_b = I_p K_{db} \cos \theta$$

I = Intensity

I_p = Point Light Intensity (source)

$K_{d(rgb)}$ = Diffuse Reflection Coefficient for each color component

θ = Angle between surface normal and light source

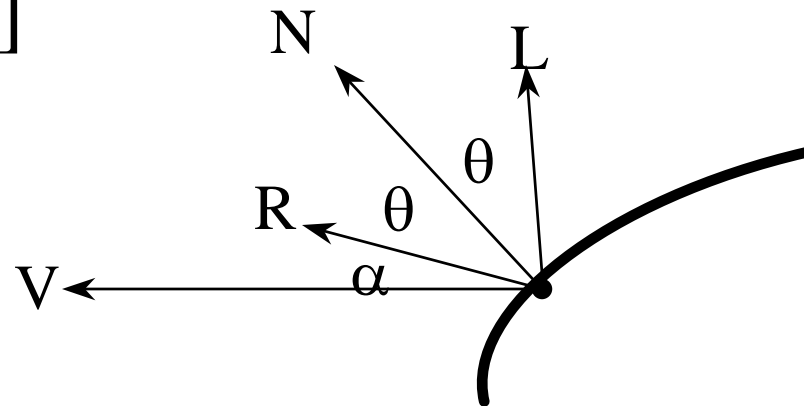


Specular Reflection

- **Models reflection from shiny surfaces**
 - eg: plastic, metal, *waxed* cars
- **Produces intense highlights**
 - Adjustable specular coefficient
- **Highly sensitive to light direction, surface normal, and viewing direction**
- **Commonly used in conjunction with diffuse reflection**

Specular Reflection

$$I = I_p k_s \cos^n \alpha]$$



I = Intensity

I_p = Point Light Intensity (source)

K_s = Specular Reflection Coefficient

n = Specular reflection factor

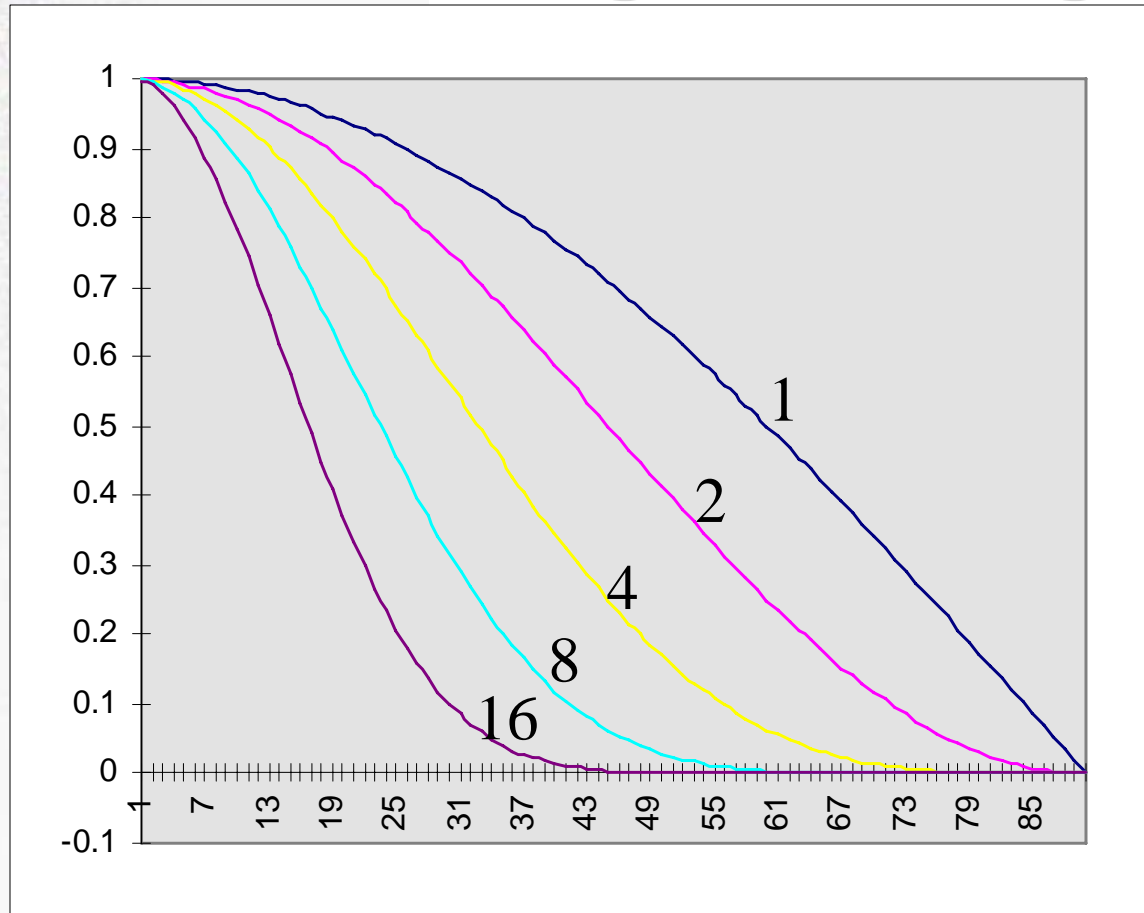
θ = Angle between surface normal
and light source

L = Light Vector

R = Reflection Vector

α = Angle between View Vector and Reflection Vector

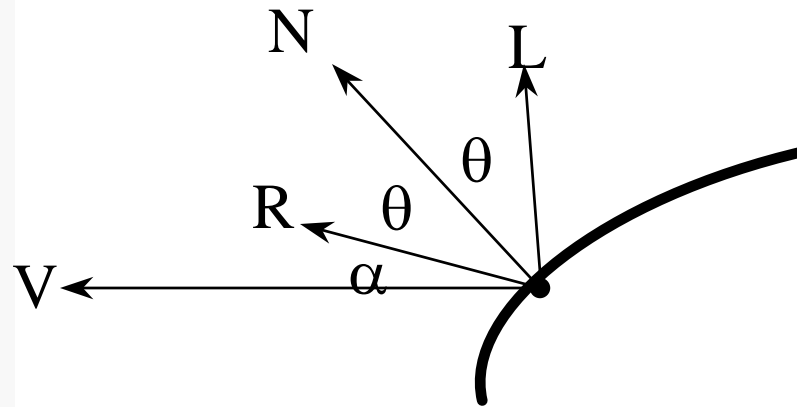
Phong Shading



Graph of $Y = \cos^n(x)$ for $n = 1, 2, 4, 8, 16$

Phong Shading

$$I = I_{\alpha\lambda} k_{\alpha} O_{\delta\lambda} + f_{\text{att}} I_{p\lambda} [K_d O_{d\lambda} \cos \theta + W(\theta) \cos^n \alpha]$$



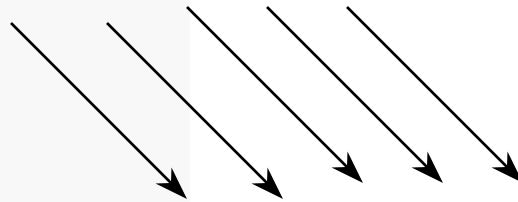
Includes ambient, diffuse and specular terms

Lights

- **Directional**
- **Point**
- **Spot**
- **Colored**
- **Multiple**
- **Area**
- **Attenuation**

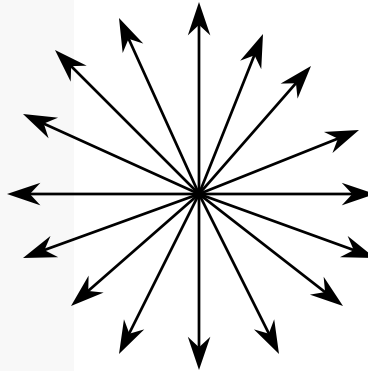
Directional Light

- **Single direction vector -- all light vectors are parallel**



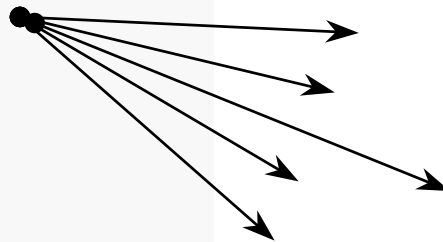
Point Light

- **Emitted from a point -- light vectors in all directions**



Spot Light

- Emitted from a point
- Has *direction* and *cut-off angle*
- May have *drop-off rate*



Performance Notes

- **Directional lights fastest**
- **Point lights slower**
- **Spot lights much slower**
–especially with fall-off
- **Two-sided lighting slow**

Colored Lights

- R, G and B components
- Separate calculations for each component

Example: Color Calculations for Diffuse Reflection

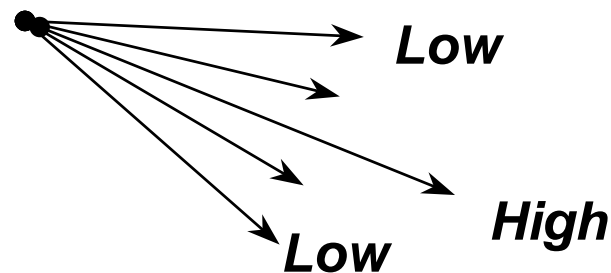
$$I_r = I_{pr} k_{dr} \cos\theta$$
$$I_g = I_{pg} k_{dg} \cos\theta$$
$$I_b = I_{pb} k_{db} \cos\theta$$

Multiple Lights

- **Calculate the Intensity for each light individually**
 - **Can include ambient, diffuse and specular components**
- **Sum the contribution of all components**
- **May need to clamp to maximum displayable intensity**

Attenuation

- Reduction in light intensity with increasing distance from light source
- Can provide visual clues of depth
- Most common use is fall-off rate with spot lights



Area Lights

- **Light is emitted from an extended area
eg: a fluorescent light**
- **Difficult to model and calculate**
- **Normally only used in advanced
rendering systems such as SoftImage**

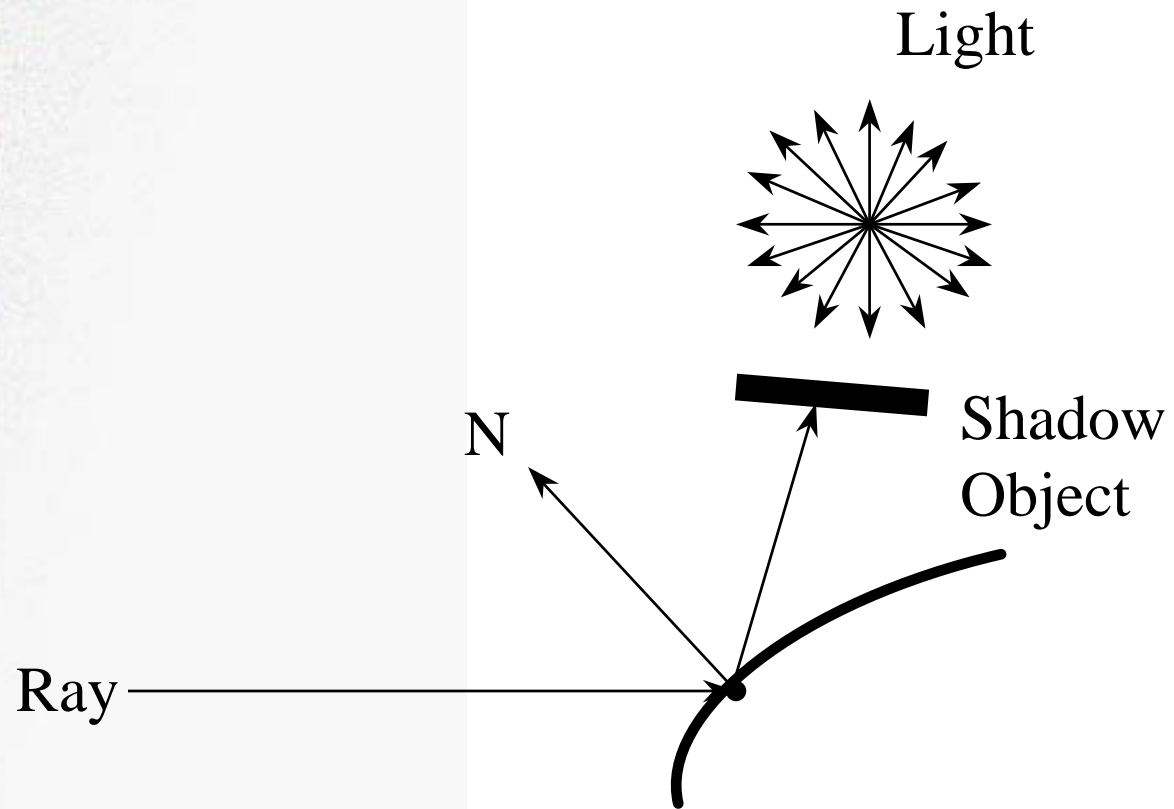
Extended Ray Casting

- **Shadows**
- **Reflections**
- **Transparency**
- **Refraction**

Shadow Calculations

- Calculate intersection, then calculate vector from intersection point to each light
- See if the light vector intersects anything -- if so, then in shadow

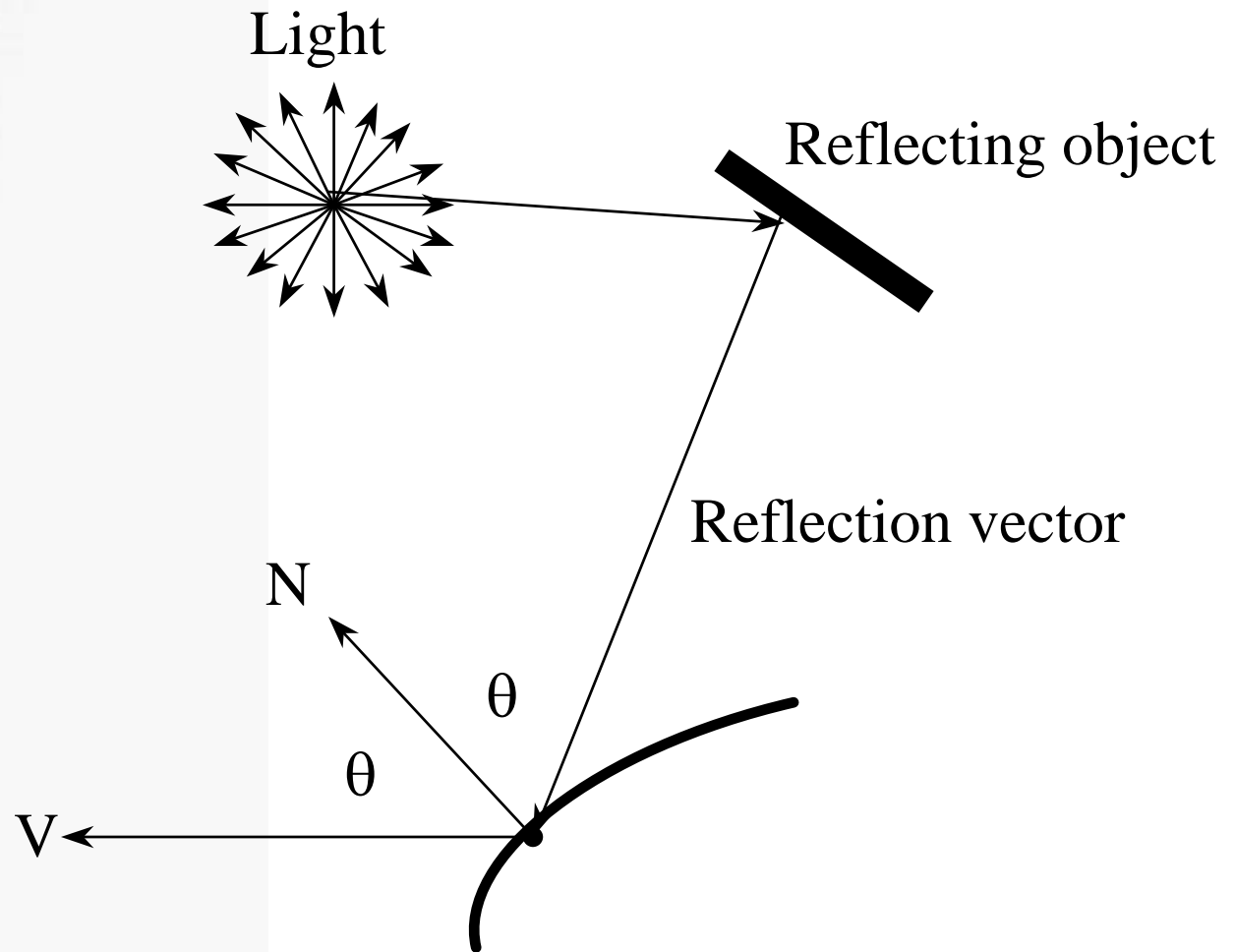
Shadows



Reflection Calculations

- **Adds illumination from light reflecting off of other objects**
- **May include diffuse and specular components**
- **May be extended to include textures in reflections**

Reflection

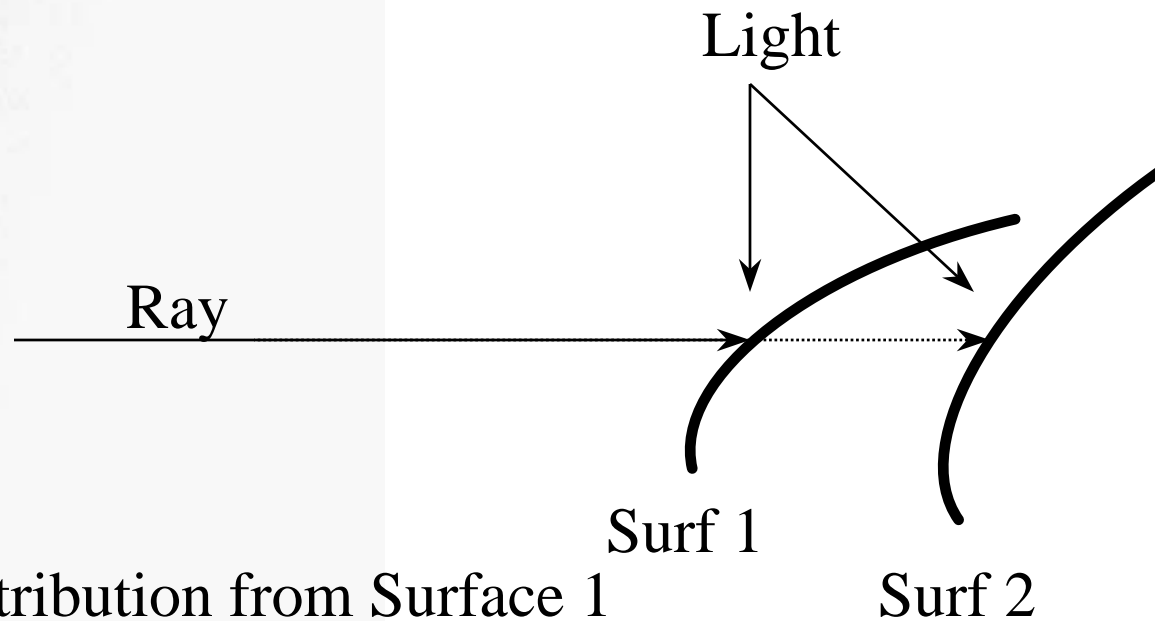


Transparency

- May be able to *partially* see through an object
- Opacity (transparency) defined by *alpha coefficient*
 - the degree to which you can see *through* an object
 - varies from *totally opaque* to *totally transparent*

Alpha Blending

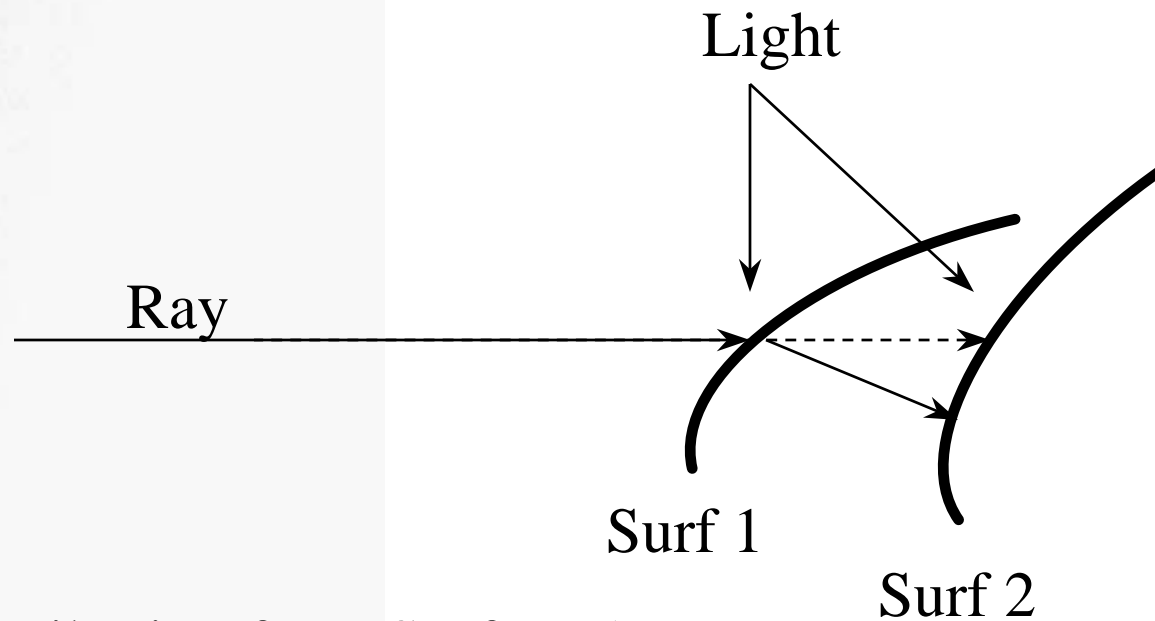
$$I = AI_p k_1 \alpha_1 \cos \theta_1 + (1-A)I_p k_2 \alpha_2 \cos \theta_2$$



Contribution from Surface 1
and surface 2

Refraction

Note: requires alpha blending



Contribution from Surface 1
and surface 2

Physically Based Lighting

- **Based on physics of radiation**
- **Radiant Intensity**
 - **Flux radiated into a unit solid angle in a particular direction**
- **Microfacets**
- **Area summation**

Material Properties

- **Attributes assigned to a surface or object**
 - **Color**
 - **Reflectivity**
 - **Ambient, specular and diffuse**
 - **Texture**
 - **Bumps**
 - **Transparency**
 - **Refraction**

Radiosity

- **Models real-world illumination**
- **Models *multiple inter-reflections* of light inside a model**
- **Provides much more realistic scenes**
- **Based on physics of radiation**
 - **Each area of a surface absorbs and emits quanta of energy**
 - **Radiation model can be processed iteratively**

Ray digital™ Tracing





digital™

RRQ

- **Recap**
- **Review**
- **Questions**

V g z v w t k p i

Texture Agenda

- **Texture Maps**
 - MIPmaps
- **Texture Mapping Operation**
 - User Defined
 - Projections
 - Environment (Reflection) Mapping
- **Bump Mapping**

Texture Maps

- A 2D array of *texels*
 - Texel: *texture element*
- Contains color (and *alpha*) values
 - represents an *image*
- Mapped to a range of 0.0 to 1.0
 - Indexed to either parameter or texel locations
- May be scanned, generated, or painted
- 2D texture maps are applied to 3D surfaces

Texture Mapping

- **Maps 2D texture coordinates to 3D surface locations (algorithmic mapping)**
 - **Maps ST texture coordinates to UV surface coordinates**
- **Many types of mapping possible**
 - **Projection, cylindrical, spherical, environment or reflection, application defined**
- **Mappings can be defined for small portions of a surface**

Texturing

- **Determining the specific texture ST value that corresponds to a surface UV value and a screen XY location**
- **Texture extrapolation: Tile/Repeat and Clamp**
- **Texture Modes:**
 - **Modulate: combines surface and texture (most common)**
 - **Decal: replaces surface with texture**
 - **Blend: commonly used for alpha maps or shadow maps**

Textures

- Typically square (implementation issue)
- Typically power of 2 (64, 128, 256, 512) (implementation issue)
- May be RGB or RGBA
- May be 16 bit or 32 bit
- In hardware, typically stored in *texture memory*

MIPmapping

- ***Multum in parvo*** -- “many things in a small place”
- **Texels approximately match pixel size**
- **Done by creating multiple copies of a texture map, each 1/2 the size of the previous**
 - eg: 512, 256, 128, 64, 32, 16, 8, 4
 - 1.5X increase texture memory
 - Improves visual quality

Texturing Modes

- **Point Sample**
- **Point Sample against MIPmap**
- **BiLinear Interpolation**
- **BiLinear Interpolation against MIPmap**
- **TriLinear Interpolation**

Point Sample

- **Single ray down center of pixel**
 - Floating point texture coordinates
 - Use the texel that the ray hits
- **Point Sample against MIPmap**
 - Use the MIP level where a projected texel is approximately the size of a pixel

Bilinear Interpolation

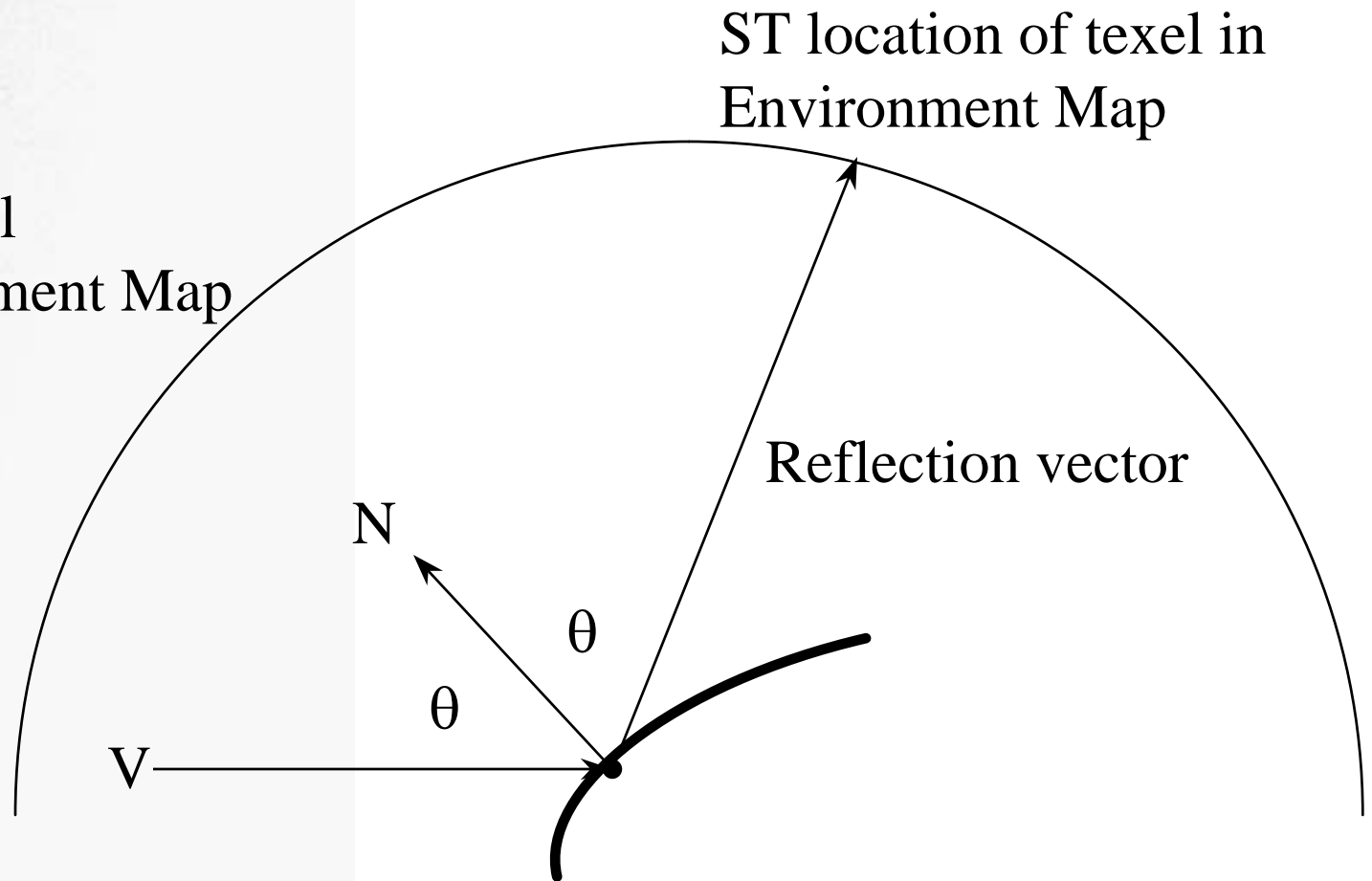
- **Single ray down center of pixel**
 - Floating point texture coordinates
 - Take the *4 texels* nearest the intersection point
 - Average these 4 texels
- **Against MIPmap**
 - Use the MIP level where a projected texel is approximately the size of a pixel

Trilinear Interpolation

- **Single ray down center of pixel**
- **Take the *4 texels* nearest the intersection point**
- **Take the *4 texels* nearest the intersection on the *next (coarser) MIP level***
- **Average the 8 texels**

Environment Mapping

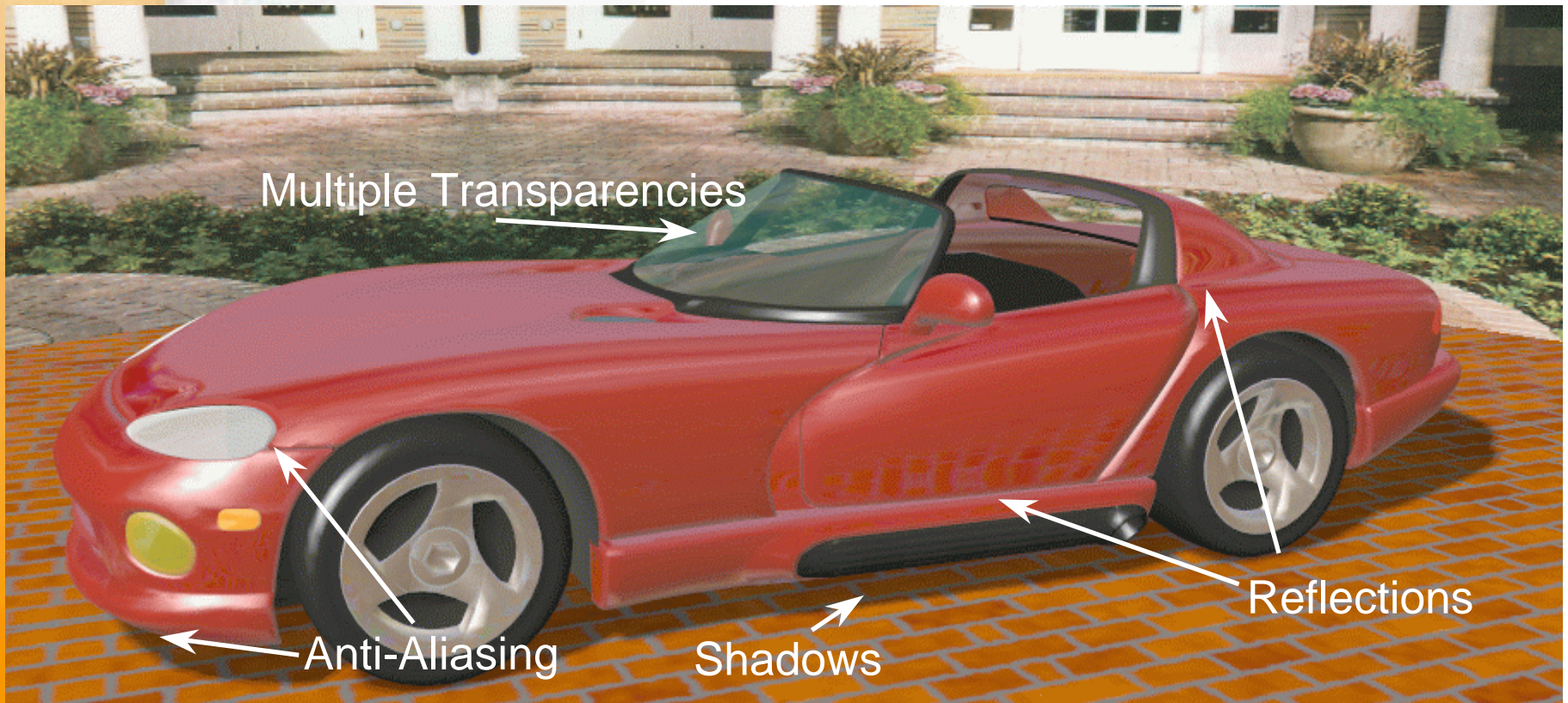
Spherical
Environment Map



Bump Mapping

- **Texture Maps vary RGBA**
- **Bump Maps vary the *surface normal***
 - **Changes the lighting calculations**
 - **Produces *apparent* changes in shading**
 - **Models bulges and dimples**
 - **eg: dimples in golf ball, waves or ripples in water**
 - **No actual change in surface**

Texture and Highlights



A decorative graphic on the left side of the slide. It features a blue and cyan trapezoidal shape at the top left, containing three spheres: a purple one, a multi-colored one, and a red one. Below this is a vertical orange bar. The background is a light gray gradient.

digital™

RRQ

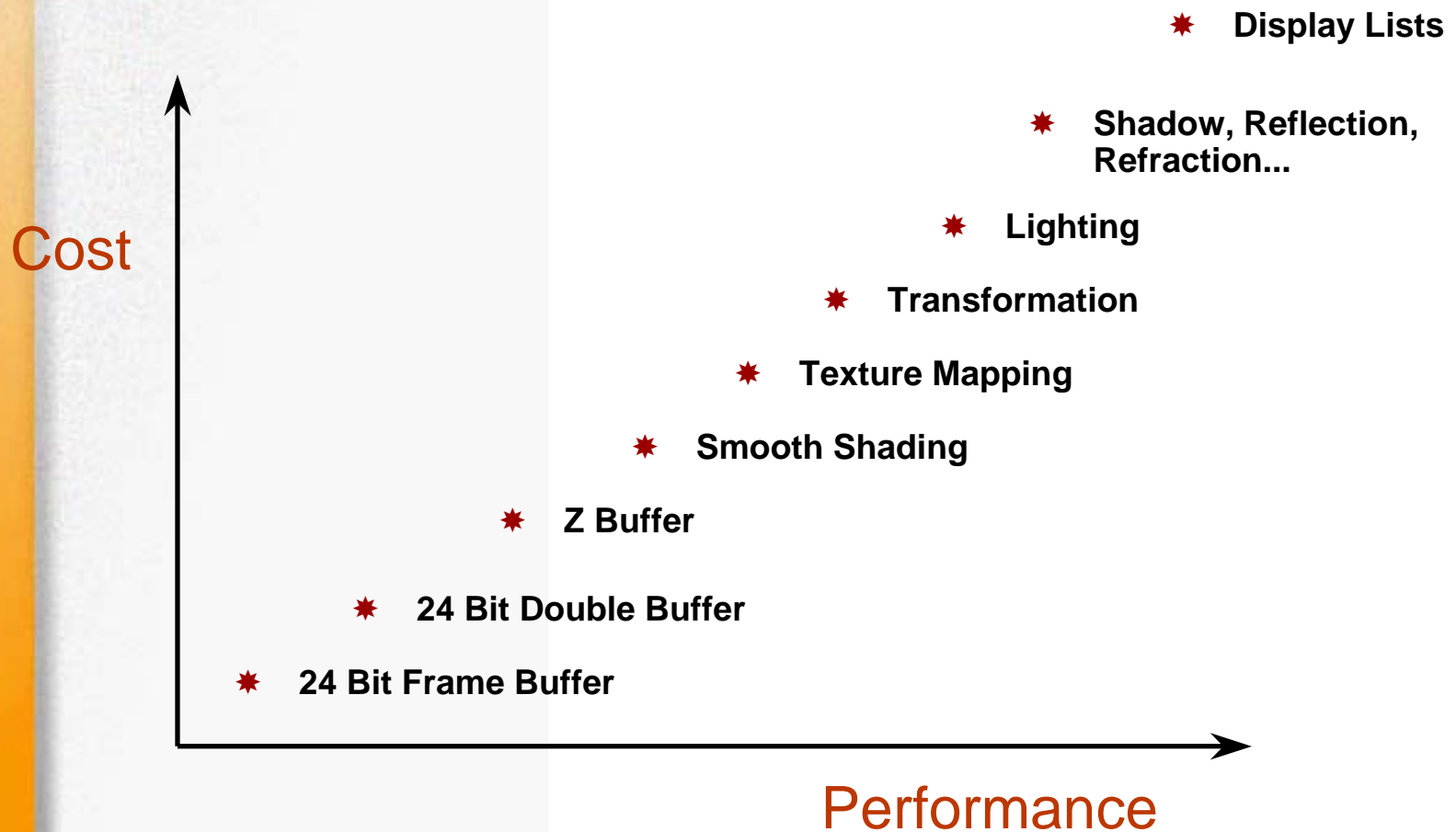
- **Recap**
- **Review**
- **Questions**

I t c r j k e u
R k r g n k p g

What's the Problem?

- **Displaying pixels on the monitor requires:**
 - Traversing application data structures and generating display data structures
 - Model level transformations
 - Viewing transformations
 - Lighting
 - Texturing
 - Display
- **Some steps are computationally intensive**
- **Some are simple, but repeated -- pixel level operations**
- **Must be done very fast to achieve interactive graphics performance**

Cost/Performance Tradeoffs



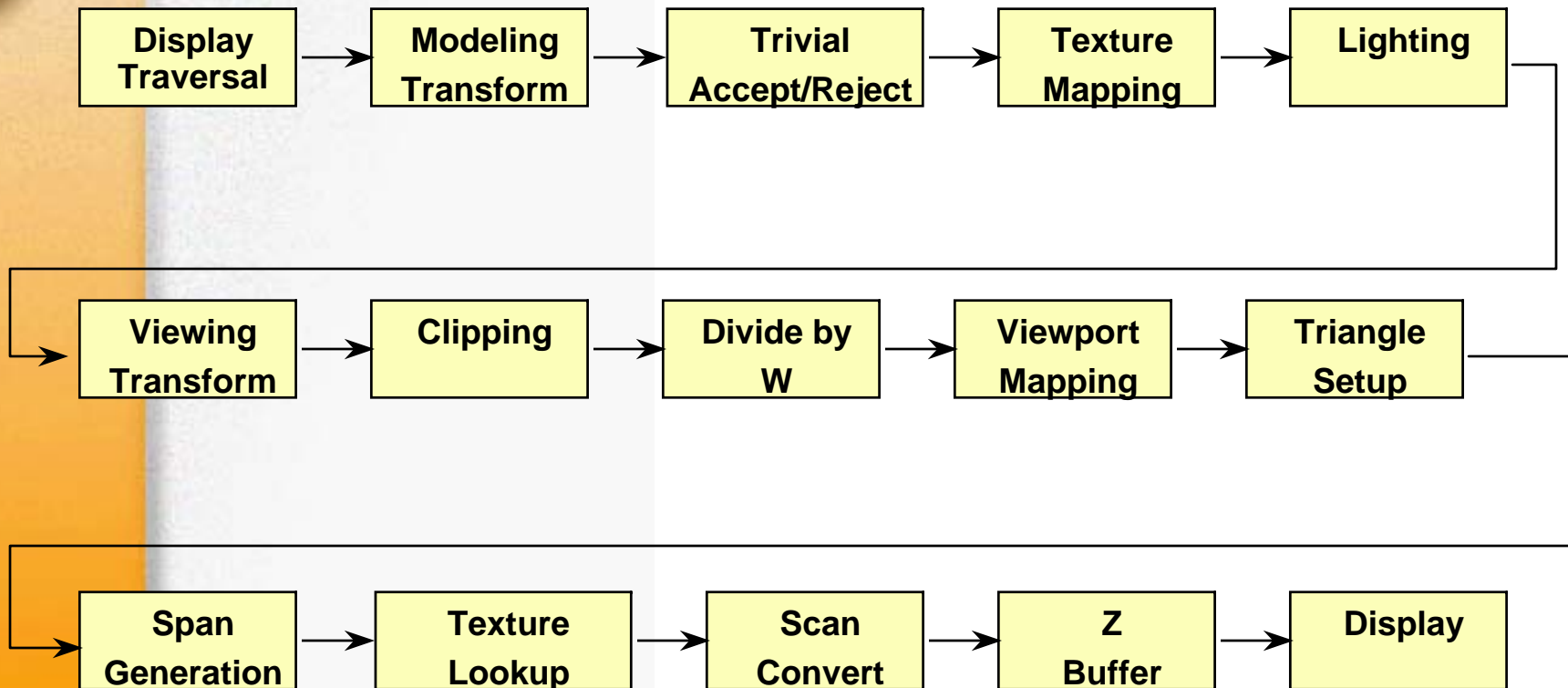
Surface Tessellation

- **Graphics *Hardware* requires flat polygons**
- ***Triangles* are the best choice**
- **Curved surfaces (spheres, cones, toroids, sculptured surfaces) must be approximated by a set of flat polygons**
- **This approximation is called *tessellation of the surface***

Fast Shading

- Ray tracing techniques can be applied at the vertices of a *polygon*
 - Polygon to screen, not screen to polygon
 - Surface normals calculated during tessellation
- The results can be *interpolated* across the triangle

Graphics Pipeline



R g t h q t o c p e g
V w p k p i

Performance Note

- **Host processor limiting factor today with most applications!**
 - Especially with 4D50T & 4D60T
 - 4D40T well balanced with Alpha
- **Sharing processor between application and graphics effective, but impacts graphics performance**
- **For best graphics performance, use the *fastest Alpha processor!***

The “Laws” of Graphics Performance

- Do as little work as possible
- Discard as much as possible
- Draw as little as possible
- Simplify wherever possible
- Combine as many operations as you can

Display Data Structures

- Many applications have internal data forms that must be processed for display
 - eg: surface tessellation
- It is possible to do this “on the fly”
- In general, much faster to create, save and use *intermediate display data*
 - Improves performance
 - Leverages hardware acceleration
 - Increases memory consumption

Transformations

- Many transformations may occur between *object local coordinates* and *screen coordinates*
- Transformations can be *concatenated*
 - Not critical for small objects
 - Important for large objects
- Some applications redundantly reset and recalculate transformations
 - Watch for this!

Scene Database Structure

- Applications can structure data to:
 - provide hierarchical structure
 - maximize locality of reference and efficiency
 - optimize database traversal
 - optimize graphics operations
 - Provide additional information:
 - Pre-processed transformations
 - Bounding boxes
 - Display lists and LOD

Level of Detail

- **Objects may contain more detail than can be shown on screen**
- **Simpler representations of object may appear same on screen, but *draw faster***
- **Example of object:**
 - High Detail: 1,000 polygons
 - Medium Detail: 100 polygons
 - Low Detail: 16 polygons

Trivial Accept/Reject

- **Discard objects behind the view point**
 - Discard polygons behind the view point
- **Discard objects entirely outside the view port**
 - Discard polygons entirely outside the view port
- **Bounding boxes useful**

Backface Culling

- **Discard polygons facing away from view point**
 - can be determined from surface normal
- **Cuts number of displayed polygons in half**
- **Only use with *solids* (closed objects)**
 - Otherwise prone to visual errors

Triangle Strips

- A triangle requires 3 vertices to define
- A triangle strip uses 2 vertices from previous triangle, and creates a new triangle with the addition of a *single vertex*
- Works best with long strips
- A performance win even with short strips (5-8 triangles)
- Use wherever possible!

OpenGL Display Lists

- A set of OpenGL calls between OpenGL *begin* and *end* statements
- Allows OpenGL to optimize graphics
- Use wherever possible!
- Best usage:
 - Use multiple small/medium display lists
 - Separate *State Changes* and *Geometry*
 - Minimize per vertex operations

Lighting

- **Use as few lights as possible (but as many as necessary)**
- **Directional lights the fastest**
- **Point lights are slower**
- **Spot lights (especially with falloff) are slowest**
- **Gouraud shading faster than Phong**
- **Only use 2-sided lighting where necessary**

Texturing

- **Point Sampling fastest, Trilinear looks best**
- **Bilinear against a MIPmap a good compromise (4DT optimized for Trilinear)**
- **Make sure textures *in use* fit into texture memory**
 - **Use MIPmapping and control maximum map size**
- **Use *Repeat Mode*; avoid *Clamp Mode***

Perspective Projection

- **Involves additional calculations, including a divide operation**
 - Divide slow on EV5
 - Much faster on EV6
- **Use where necessary**
- **Don't use where not needed**
 - See if orthographic projection can be used in benchmarks

Performance Tools

- **NT Performance Monitor**
 - **100% CPU: CPU bound**
 - **<100% CPU: I/O bound or Graphics bound**
- **“Dump on Floor” (Unix only today)**
 - **Bypass graphics card**
 - **Bypass entire graphics pipeline**
 - **Determines conclusively if graphics issue**

Performance Example

- **Flight simulator: 10,000,000 poly database**
- **5,000,000 polys behind view; discard**
- **3,000,000 polys more than 10km away; discard**
- **1,000,000 polys outside viewport; discard**
- **500,000 polys in objects less than 100 pixels, use minimum LOD to reduce to 5,000 (buildings & vehicles)**
- **400,000 polys in objects less than 2,500 pixels, use medium LOD to reduce to 40,000 polys**
- **142,500 drawable polys**
- **Backface culling reduces to 70,000 polys**
- **Smooth motion on 2M tri/sec graphics**

A decorative graphic on the left side of the slide. It features a blue and cyan trapezoidal shape at the top left, containing three spheres: a purple one, a multi-colored one, and a red one. Below this is a vertical orange bar. The background of the slide is white with a subtle texture.

digital™

RRQ

- **Recap**
- **Review**
- **Questions**



digital™

E q nq t

Reasons for Color

- **Human visual system highly attuned to color**
- **Color conveys more information than monochrome**
- **Color necessary for realism**

What is Color?

- **Visible Light: “Electromagnetic radiation with a wavelength of 400-700nm”**
- **Color: Specific wavelengths of electromagnetic radiation:**
 - Red: 650 nm
 - Green: 550 nm
 - Blue: 450 nm

The Human Eye

- **Two visual receptors: Rods and Cones**
- **Cones sensitive to color**
 - Three types, sensitive to Red, Green, and Blue
 - Narrow Angle of Vision
 - Requires high intensity light
- **Rods**
 - Color insensitive: “black and white”
 - Very light sensitive
 - Broad angle of vision, very motion sensitive

Human Eye: Cones

- High color sensitivity
- Excellent detail resolution
- Over-emphasis of transitions; built in *“edge detection”*
- Non-linear brightness response -- sensitive to ratios of intensity, not actual intensity
 - Change in brightness of 0.1-0.11 appears the same as change from 0.9-0.99

Reproduction of Color

- **Key concept: mix primary colors to reproduce all colors**
- **Two basic categories:**
 - Additive (monitors)
 - Subtractive (print; normal viewing)
- **“Color Spaces” used to define colors**

Additive Color

- Builds color and brightness by *adding* Red, Green, and Blue light
- Common examples: Monitors and TVs
 - Each pixel has a red pixel, a blue pixel, and a green pixel

Monitors

- **Monitors commonly use 8 bits (0-255 steps) for each color**
 - **Close to the number of color steps the monitor can reproduce**
 - **Close to the number of color steps the human eye can perceive**
 - **Specialized applications may require more steps (eg: medical imaging, multi-spectral satellite data)**

“True Color”

- **Uses 8-bits each for R, G, and B
–24 bits total**
- **Displays 16.7 M colors**
- **Good match for monitors and human eye**
- **Allows applications to work directly with colors**
- **Works extremely well with *interpolation***

Alternative True Color

- **“True Color” not restricted to 8 bits/pixel**
- **32 bit/pixel**
 - 8/8/8/8 RGBA (common for textures)
- **36 bit/pixel (used by SGI Infinite Reality)**
 - 12 bit/component RGB
- **16 bit/pixel**
 - 5/5/5 or 5/5/6 (RGB)
 - 4/4/4/4 or 5/5/5/1 (RGBA)
 - Saves memory and bandwidth
 - Looks worse

Indexed Color

- Builds a table of colors (color palette)
 - Commonly 8 bit, may be 15 bit or 16 bit
- Applications use pointer to color

	Red	Green	Blue
1	0	0	0
2	255	255	255
3	158	223	17
...			
255	96	128	96

Subtractive Color

- ***Subtracts colors*** from reflected light
- **Common examples:** printed material, photographs, everything you see!
- **Uses *complements*** to additive colors:
Cyan, Yellow, Magenta (CYM)
- **Common printing practice adds black (K)**
–*CYMK color*

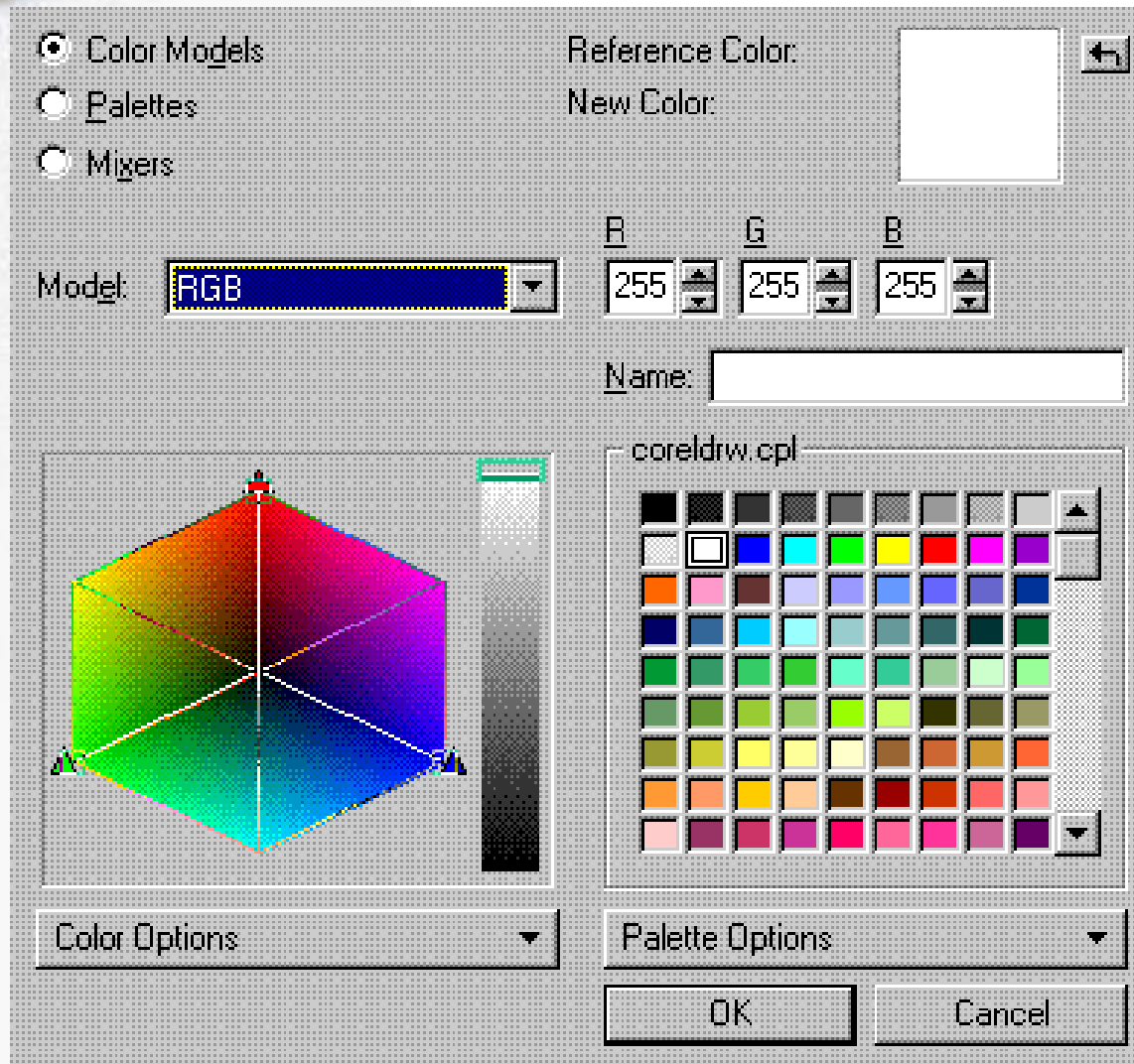
Color Separations

- **Primary colors represented independently**
- **RGB: separate bit planes & color guns**
- **CYMK: separate printing plates for each color**
- **Alignment is critical!**
- **Each color plane can be manipulated separately, as a monochrome image**

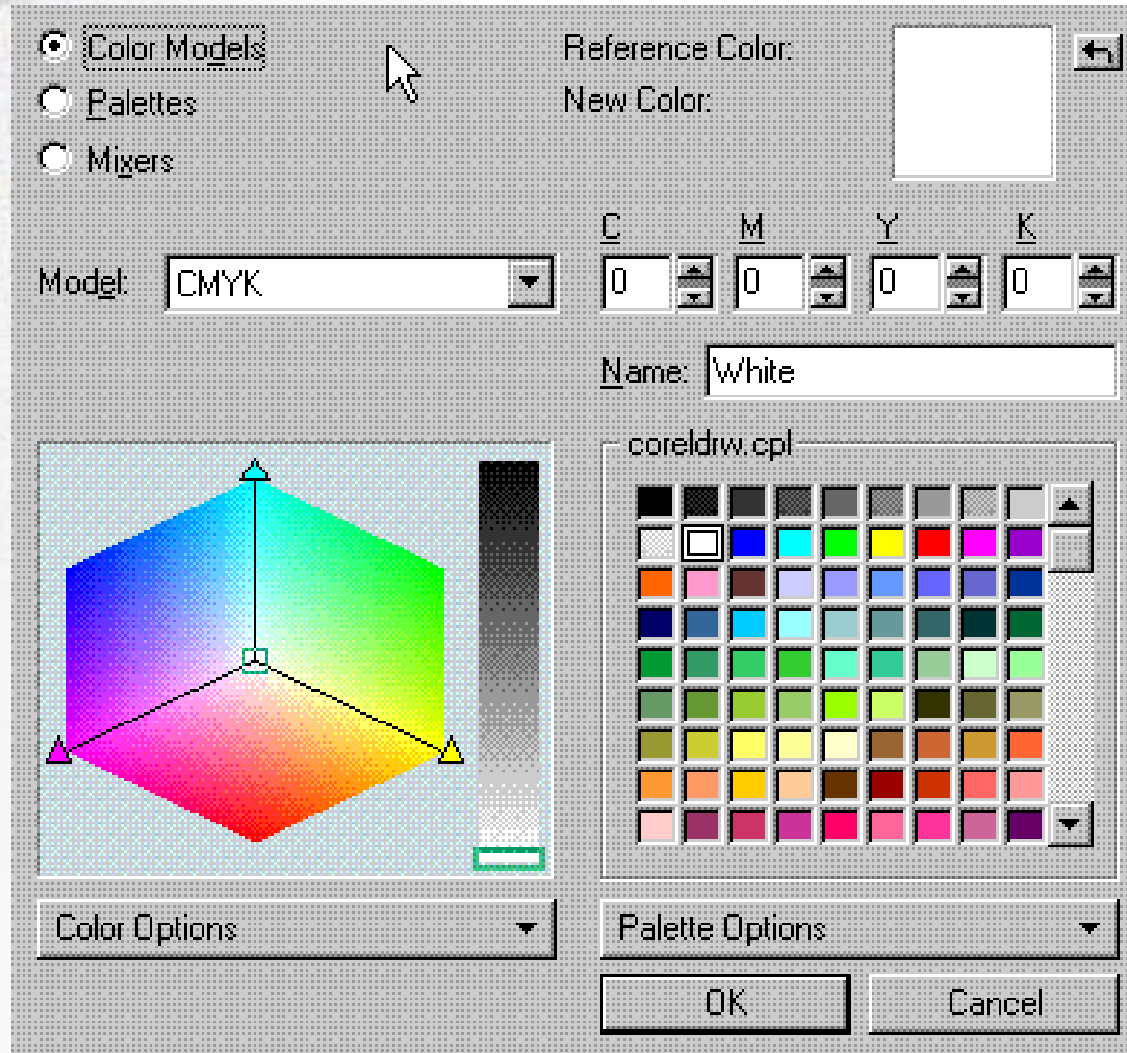
Color Spaces

- **Used to produce, reproduce, and manage colors**
- **Can produce the same color in each color space**
 - **Can mathematically convert between color spaces**
- **Color spaces developed for specific applications**

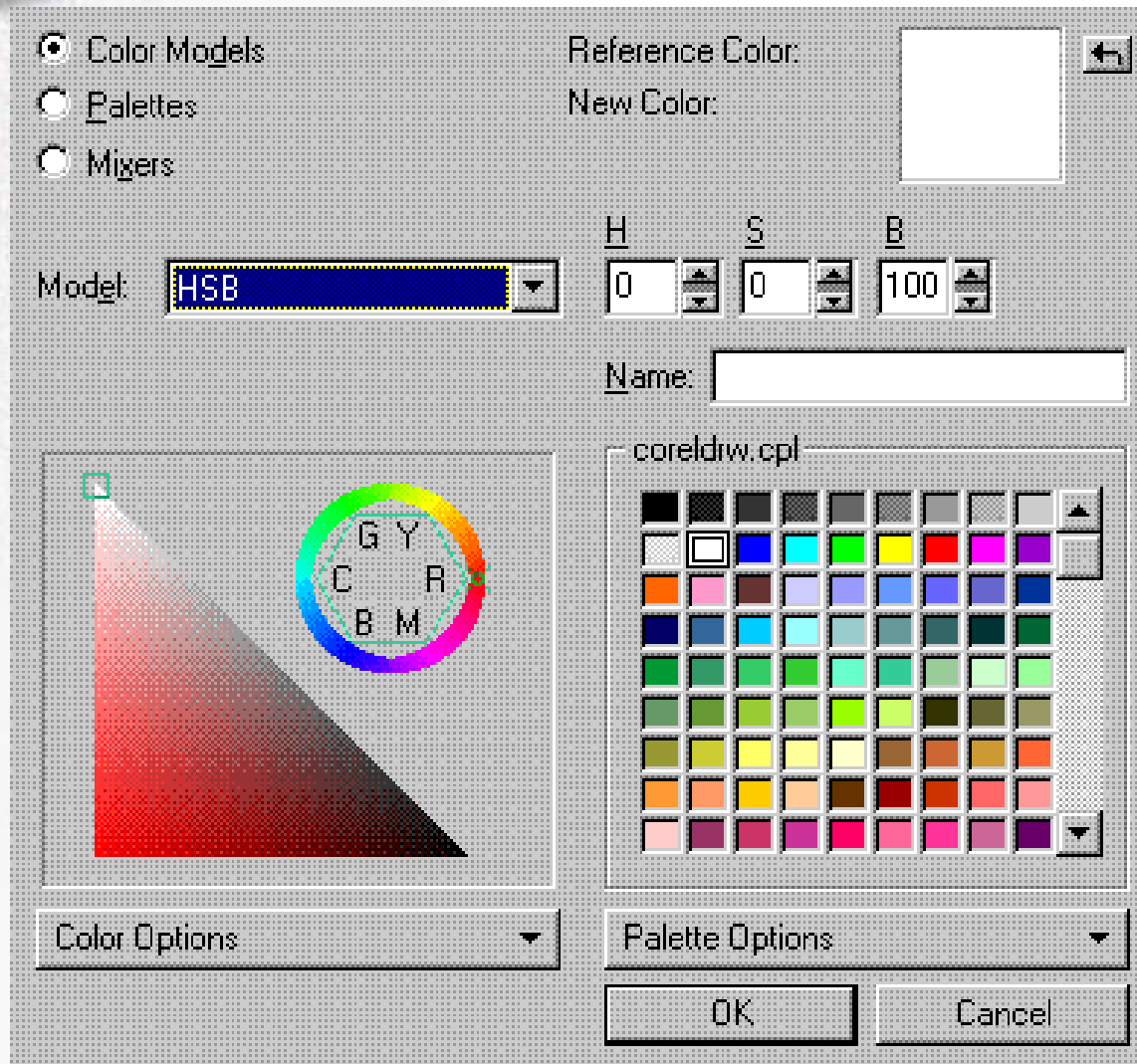
RGB Color



CMYK Color



HSB Color



Real World Notes 1:

- **Individual monitors reproduce colors differently**
 - **[207, 104, 12] will appear as different color on different brands of monitors**
 - **Individual monitors have different response curves**
- **Monitors may need to be “color tuned”**
 - **Gamma correction**
 - **Color correction**

Real World Notes 2:

- **Color matching is extremely difficult!**
 - Across technologies (RGB to CMYK)
 - Across output devices (color inkjet, color laser, web print presses)
 - Across inks and papers
 - Across day to day variables
- **Many techniques used by print professionals**
- **Remains a “Black Art”**



digital™

RRQ

- **Recap**
- **Review**
- **Questions**

C nkc ukp i

cpf

C p vk / C nkc ukp i

Aliasing

- ***Visible artifacts*** resulting from display of smooth data on coarse pixel grid
- Results when model data has greater detail than screen resolution
- Results are ***“Jaggies,” “Sparkles” and “Jerkiness”***
- Aliasing is visually disturbing
- Human eye has built-in edge enhancement

Vector Aliasing

- A mathematical line has *no width*
- Pixels have *area*
- How to map lines to pixels?
- Vector anti-aliasing:
 - determine contribution of line to *nearby* pixels
 - assign an intermediate value to these pixels

Polygon Aliasing

- Aliasing of polygon interiors
 - Shaded
 - Textured
- Aliasing of polygon edges
- *Temporal* aliasing

Polygon Interiors: Shading

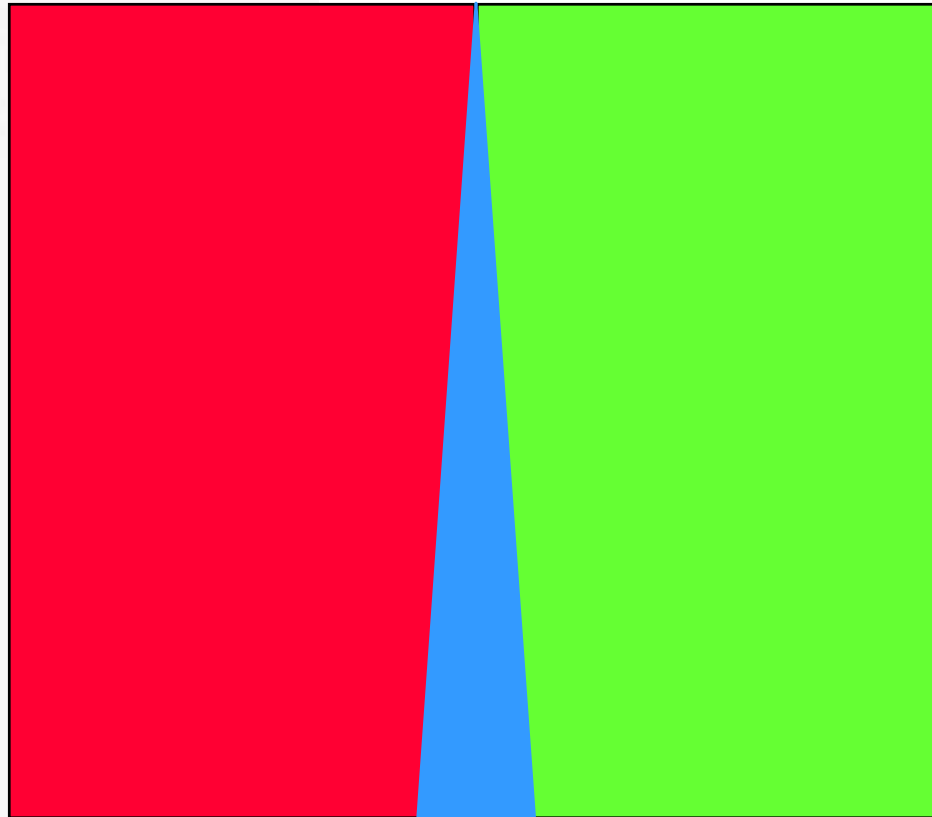
- **Smooth (Gouraud) shading is an interpolated technique, and does not have inherent aliasing problems**

Polygon Interiors: Texturing

- Point sampling has aliasing problems
 - Shows as “sparkles” or “crawling” of textures in motion
- Bilinear and Trilinear interpolation provide *anti-aliasing* for textures

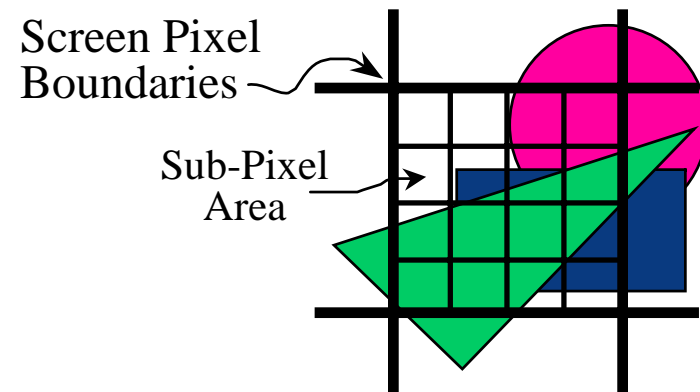
Polygon Edges

- What color is this pixel?



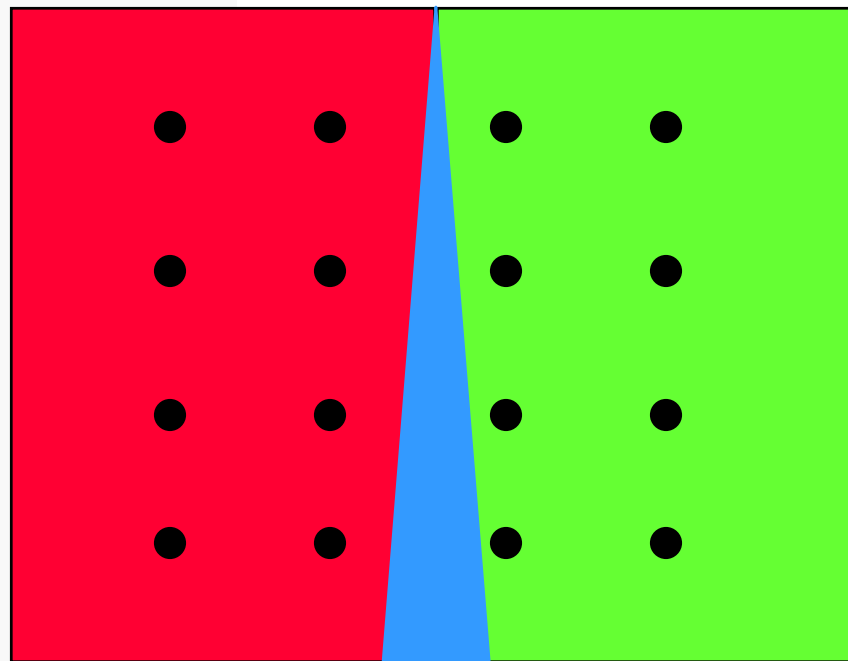
Polygon Anti-aliasing

- Multiple polygons can contribute to a single pixel -- using only one polygon per pixel produces aliasing
- To avoid aliasing, determine what each polygon contributes to the pixel, and include all contributions



Supersampling

- Instead of one sample per pixel, take multiple samples per pixel (typically 4-16) and sum them



Supersampling

- Effectively a higher resolution *virtual display* mapped to the lower resolution *physical display*
- Requires graphics processing and frame buffer storage of large virtual display
- Each pixel requires
 - 8 bytes/sample (RGBA + Z)
 - Roughly 12 bytes/pixel
 - Roughly 1120 bits/pixel!

Supersampling Issues

- **Requires 16X greater graphics processing**
- **Major performance hit!**
- **Uses massive amounts of frame buffer memory**
- **Still prone to aliasing**
- **Doesn't handle multiple transparency**
- **For software (not hardware), anti-aliases reflections, shadows, etc.**

HiFive

- Polygon interiors anti-aliased by texture interpolation
- At polygon edge, it is easy to calculate *area of polygon that overlaps a pixel*
- For *edge pixels*, build a *Pixel Link List* that has an ordered list of intersecting poly edges, their area, their color and alpha values, and their depth

HiFive vs Supersampling

- HiFive requires less graphics processing than supersampling
- HiFive uses less frame buffer memory
- HiFive produces higher visual quality images
- HiFive handles multiple transparencies

Clear Digital Advantage!

Temporal Aliasing

- Movement between frames produces “jerky” display
- Higher frame rate minimizes effect
- Advanced systems (eg: SoftImage) use *motion blur* to interpolate between frames
- Primarily software implementations

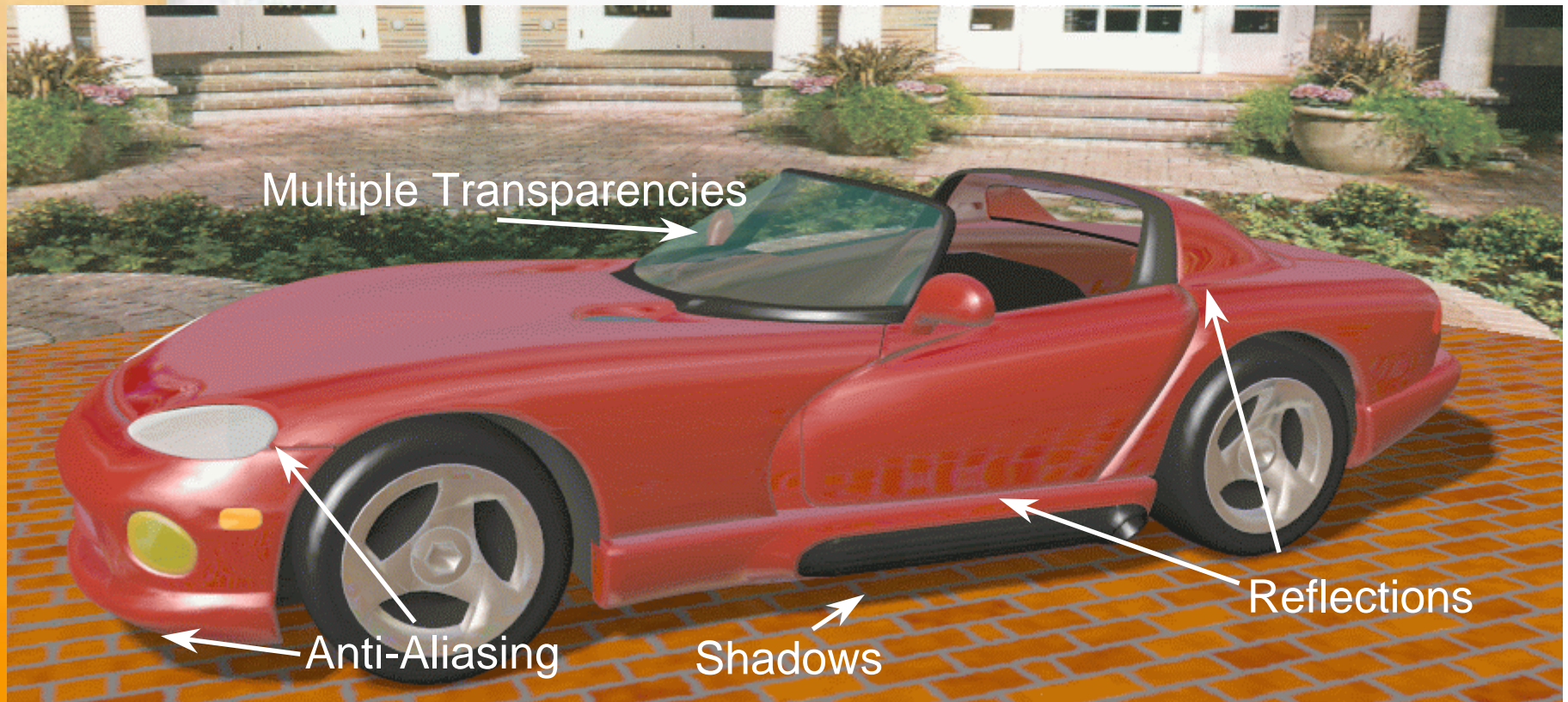
digital™

Without HiFiVE Technology



digital™

With HiFiVE Technology





digital™

RRQ

- **Recap**
- **Review**
- **Questions**

***LEADING THE NEXT
GENERATION OF COMPUTING !***

digital™

Digital Equipment Corporation
© 1997