

A photograph of a living room. A woman in a beige top and purple skirt stands in the center, looking down at a small object in her hands. To her right, another person is seated in a blue armchair, facing away from the camera. The room features a large window on the right side, offering a view of a harbor with a bridge and city buildings. A wicker basket sits on the floor in the foreground, and a television is visible on the left. The text "CSC 305 Midterm Review" is overlaid in white across the middle of the image.

# CSC 305 Midterm Review

# There are no new technical slides in this deck

Topics we have gone through:

<b>The Geometry of Ray Tracing</b>	<b>Shading and Texturing</b>	<b>Intro to Raster Graphics</b>
Vector Arithmetics, Ray Equation	Color Theory, RGB Color, Perception based color spaces	Translate, Rotate and Scale with Matrices, Homogeneous Coordinates, Matrices Concatenation
Ray-Sphere Intersection	Phong-Blinn Reflection Model (Ambient + Diffuse + Specular)	
Ray-Triangle Intersection, Barycentric Coordinates	Texture Mapping, UV Coordinates	

The University of Victoria  
Department of Computer Science  
**CSC 305 Midterm Exam February 2019**

\* \* \*

**Time allowed: 60 minutes, during the lecture time of February 14, 2019.**

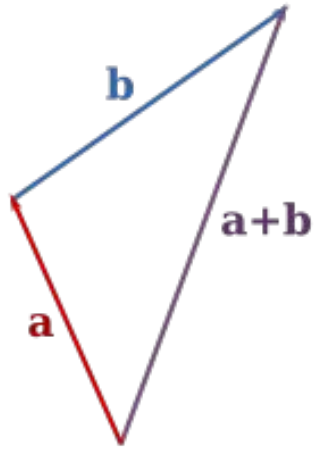
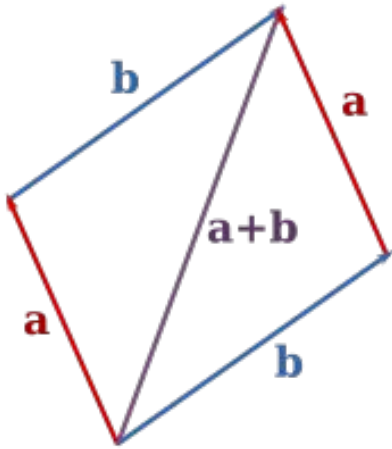
**Closed book; notes are not permitted. Non-graphical calculators are permitted. The grade of this midterm is marked out of 80 points in total, and this exam contributes to 10% of your final course grade.**

**Clearly show all your steps of derivation and calculation. Partial marks will be given based on the correctness of mathematical procedures when there are arithmetics mistakes.**

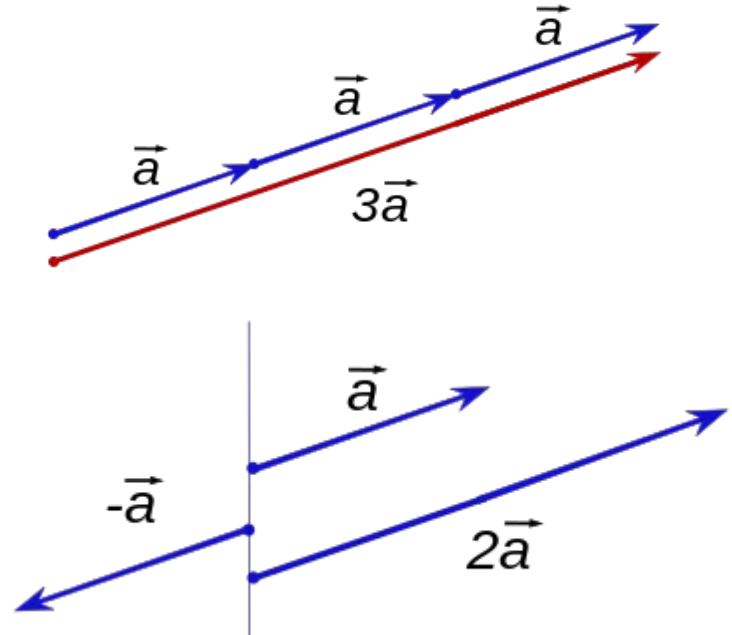
**Bring your student card - we will have sign-in sheet.**

# Geometry Vectors

- Additions and scalar multiplications already work with geometry (without specifying if its a pair or triplet of some sort of numbers).

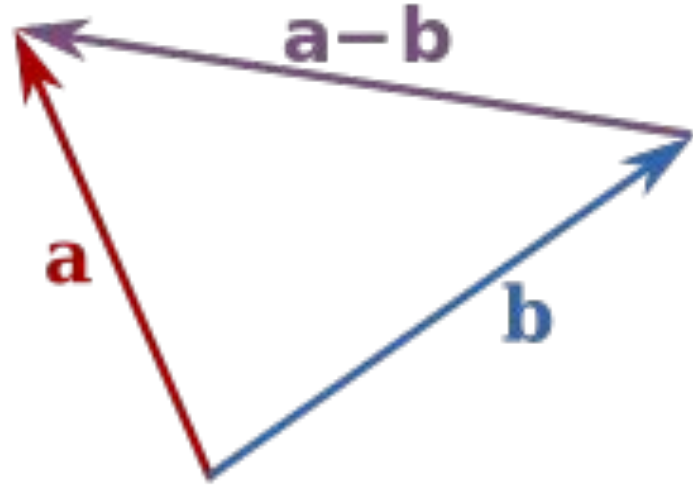
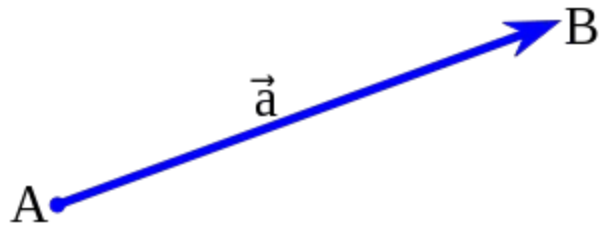


wiki



# Geometric Vector

One specific operation: What is the vector pointing from location A to location B?



# Length and Dot (inner) product

- With the 3D Cartesian coordinates, we **define length** as

$$|a| = \sqrt{x_a^2 + y_a^2 + z_a^2}$$

(this is because we **defined** the length of the orthonormal basis to be 1)

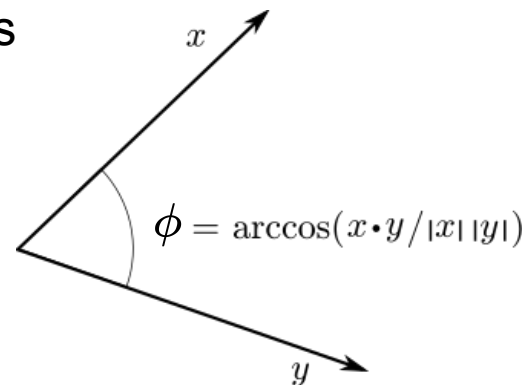
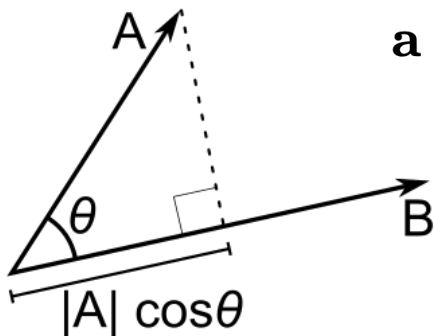
- With the 3D Cartesian coordinate, we can define a dot (inner) product that

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

Because of the **orthonormal** nature of the Cartesian basis.

- We further have that the **projection length** of  $\mathbf{a}$  on  $\mathbf{b}$  is

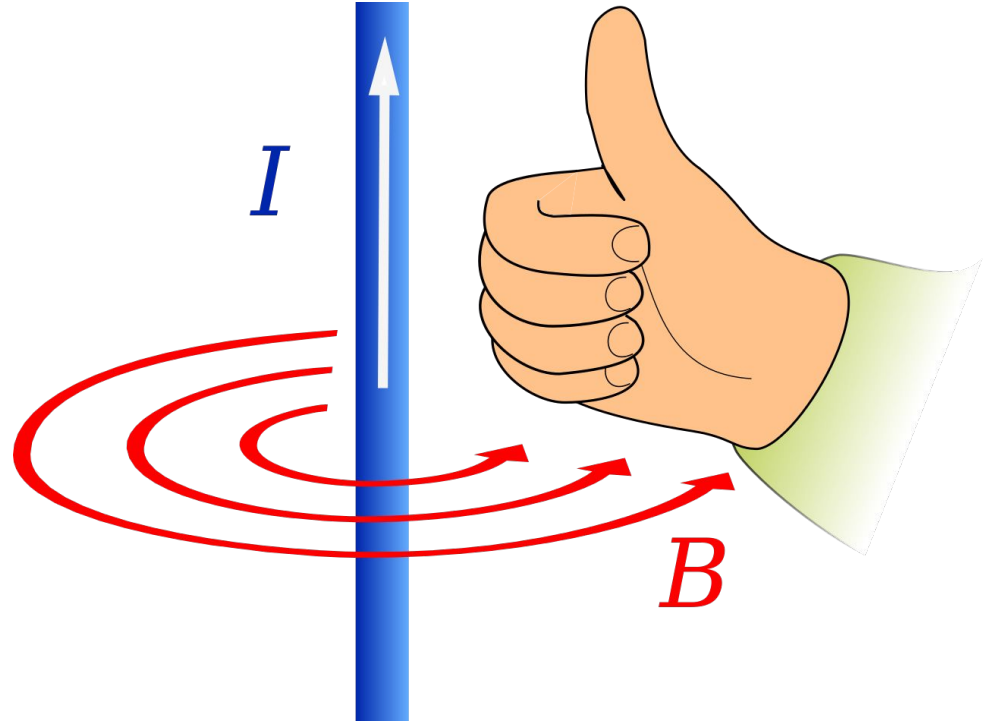
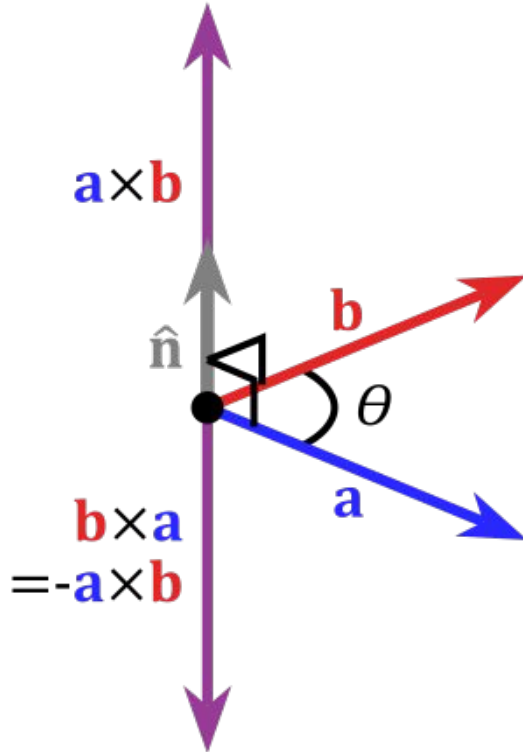
$$\mathbf{a} \rightarrow \mathbf{b} = |\mathbf{a}| \cos \phi = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{b}|}$$



# Cross (Outer, Exterior, Vector) product

- ONLY in 3 dimensional space, for any pair of vectors  $\mathbf{a}$  and  $\mathbf{b}$ , there exist a third vector  $\mathbf{a} \times \mathbf{b}$ , such that,
  - $|\mathbf{a} \times \mathbf{b}| = |\mathbf{a}||\mathbf{b}| \sin \phi$  (Length requirement)
  - $\mathbf{a} \times \mathbf{b}$  is perpendicular (orthogonal) to both  $\mathbf{a}$  and  $\mathbf{b}$  (Direction requirement)
- In other dimensionalities, give a pair of vectors, in general there is no third vector that conforms to these two requirements.
- $\mathbf{a} \times \mathbf{b} = -\mathbf{b} \times \mathbf{a}$
- In Cartesian coordinates,
$$\mathbf{a} \times \mathbf{b} = (y_a z_b - z_a y_b, z_a x_b - x_a z_b, x_a y_b - y_a x_b)$$

# Right handed



wiki, Ampere's right hand rule



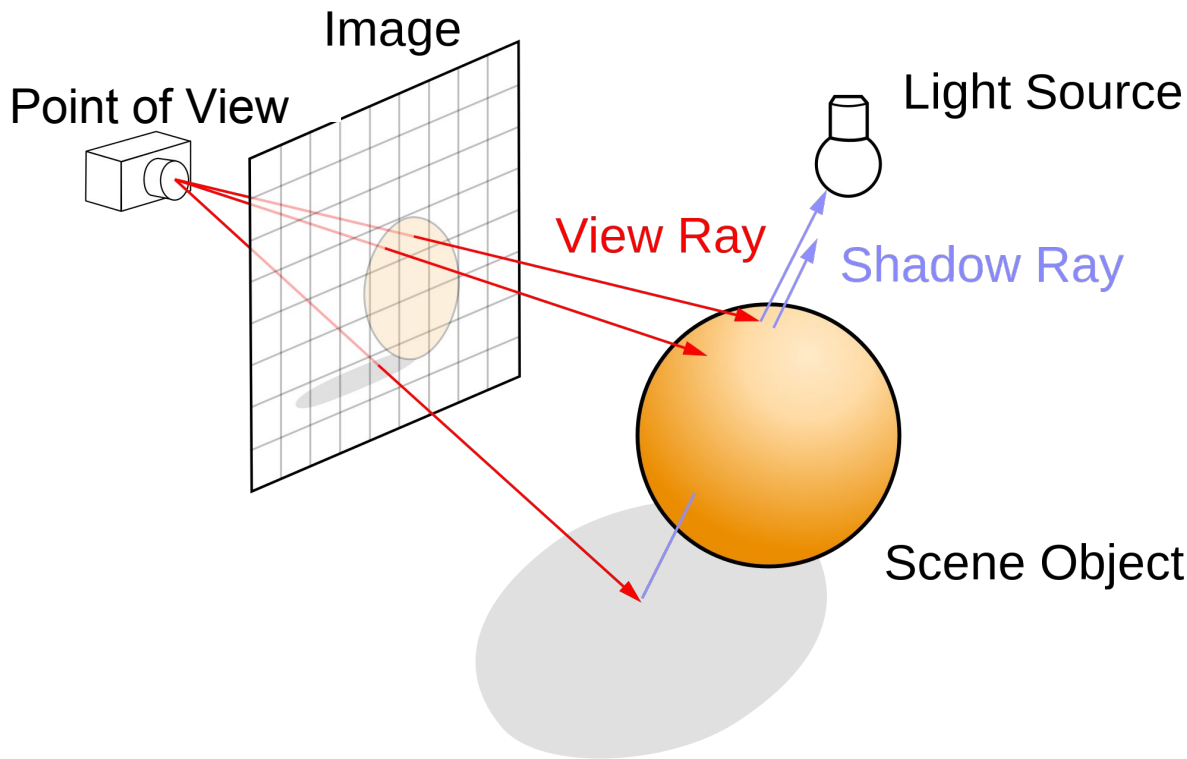
# The Geometry of a Ray

We now know the locations of the pixels.

To create rays that go through these pixels we also need the **origin** point where all ray starts.

## Ray Equation

$$\vec{p} = \vec{o} + t\vec{d}$$



# Ray Equation - vector form

$$\vec{p} = \vec{o} + t\vec{d}$$

- $\vec{o}$  is the origin of ray (Point of View, 3D vector)
- $\vec{d}$  is the direction of ray (Normalized vector, has length 1)
- $\vec{p}$  is the current point
- $t$  is the distance of current point to the origin of the ray (scalar)
- The entire Ray Equation is in **World Space**

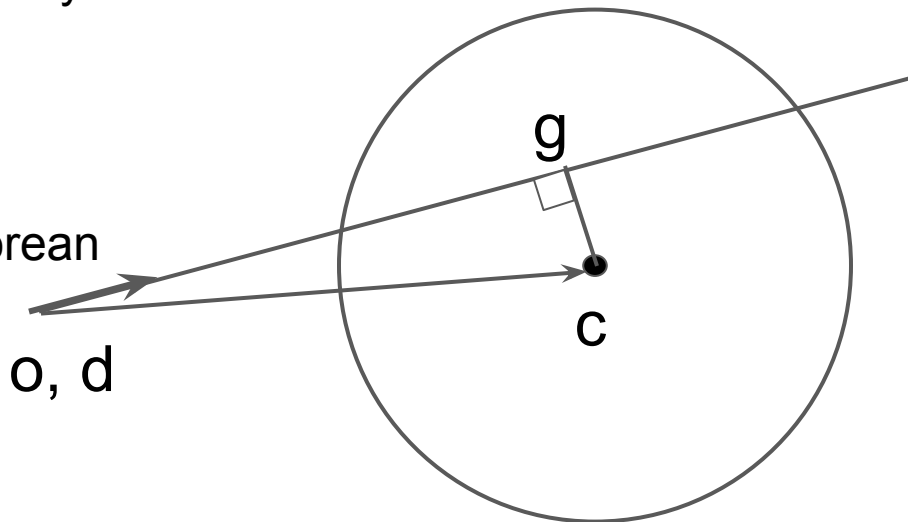
# Ray sphere intersection - a vector arithmetics approach

1. Calculate  $|\mathbf{cg}|$ , the distance between (1) the projection of the center  $\mathbf{c}$  on the ray  $\mathbf{g}$ , and (2) the center  $\mathbf{c}$  itself:

$$\mathbf{oc} = \mathbf{c} - \mathbf{o};$$

$$|\mathbf{og}| = \text{dot}(\mathbf{oc}, \mathbf{d}); \text{ // } \mathbf{d} \text{ is normalized}$$

$$|\mathbf{cg}| = \text{sqrt}(|\mathbf{oc}|^2 - |\mathbf{og}|^2); \text{ //Pythagorean}$$



# Ray sphere intersection - a vector arithmetics approach

2. Calculate the position of **p**:

$|\mathbf{pg}| = \sqrt{r^2 - |\mathbf{cg}|^2}$ ; //Pythagorean

$|\mathbf{op}| = |\mathbf{og}| - |\mathbf{pg}|$ ;

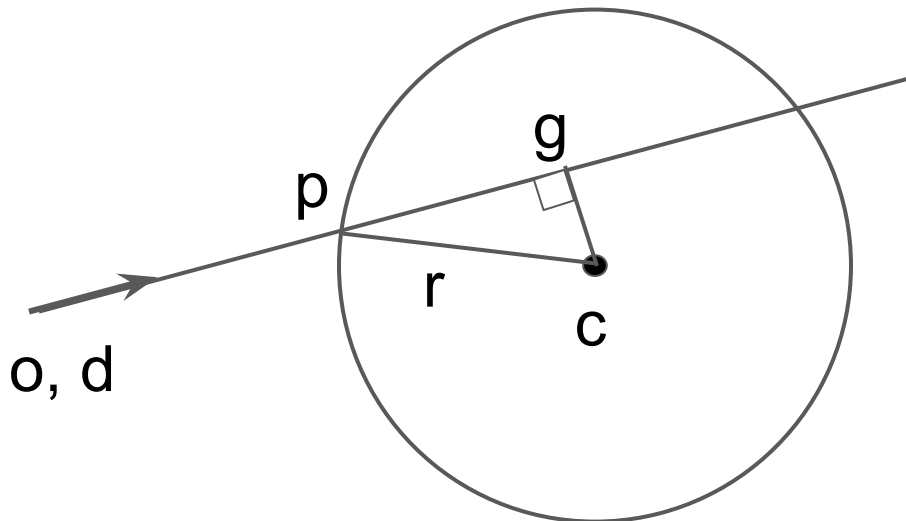
$\mathbf{p} = \mathbf{o} + |\mathbf{op}|\mathbf{d}$ ; //d is normalized

How do we know no-intersection?

Triangle **pgc** does not form:  $r < |\mathbf{cg}|$

One intersection point?

$r == |\mathbf{cg}|$



# Reading assignment :

## Textbook 4.4 Algebraic method

### 2.3 RAY-SPHERE INTERSECTION

A sphere with center  $\mathbf{c} = (c_x, c_y, c_z)$  and radius  $R$  can be represented by the implicit equation

$$(x - c_x)^2 + (y - c_y)^2 + (z - c_z)^2 - R^2 = 0.$$

We can write this same equation in vector form:

$$(\mathbf{p} - \mathbf{c}) \cdot (\mathbf{p} - \mathbf{c}) - R^2 = 0.$$

Again, any point  $\mathbf{p}$  that satisfies this equation is on the sphere. If we plug points on the ray  $\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$  into this equation we can solve for the values of  $t$  on the ray that yield points on the sphere:

$$(\mathbf{o} + t\mathbf{d} - \mathbf{c}) \cdot (\mathbf{o} + t\mathbf{d} - \mathbf{c}) - R^2 = 0.$$

# Calculate ray-triangle intersection with barycentric coordinates - A vector arithmetics approach

Roadmap:

1. Determine where is the intersection of the ray with the plane of the triangle.
  - a. Maybe there is no intersection - ray parallel to the plane of the triangle.
  - b. Maybe there is an intersection but it is backwards, behind the PoV:  $t < 0$
2. Determine the barycentric coordinate of the intersection
3. Check if  $\alpha, \beta, \gamma \in [0, 1]$  holds. If not, the intersection is outside of the triangle.
4. We have the intersection point  $\vec{p} = \alpha\vec{A} + \beta\vec{B} + \gamma\vec{C}$

# Vector Arithmetics Solution

Solution for distance:

$$\mathbf{BC} = \mathbf{vC} - \mathbf{vB};$$

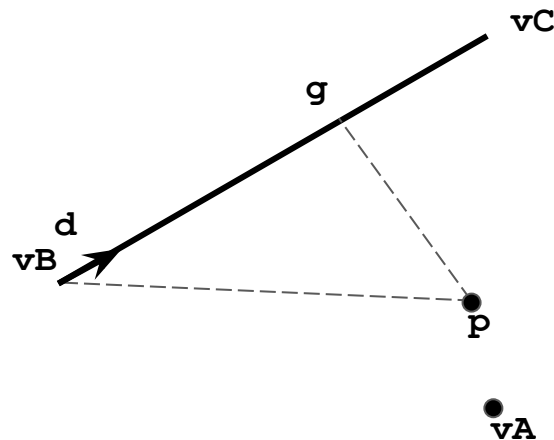
$$\mathbf{d} = \text{normalize}(\mathbf{BC}); \text{ //the direction of BC}$$

$$\mathbf{Bp} = \mathbf{p} - \mathbf{vB};$$

$$|\mathbf{Bg}| = \text{dot}(\mathbf{Bp}, \mathbf{d});$$

$$\mathbf{g} = \mathbf{vB} + |\mathbf{Bg}| \mathbf{d};$$

$$\text{distance\_alpha} = |\mathbf{pg}|;$$



# Vector Arithmetics Solution

Is  $\mathbf{vA}$  and  $\mathbf{p}$  on the same side of  $\mathbf{vB}-\mathbf{vC}$ ?

$$\mathbf{BC} = \mathbf{vC} - \mathbf{vB};$$

$$\mathbf{Bp} = \mathbf{p} - \mathbf{vB};$$

$$\mathbf{BA} = \mathbf{vA} - \mathbf{vB};$$

$$\text{outer1} = \text{cross}(\mathbf{Bp}, \mathbf{BC});$$

$$\text{outer2} = \text{cross}(\mathbf{BA}, \mathbf{BC});$$

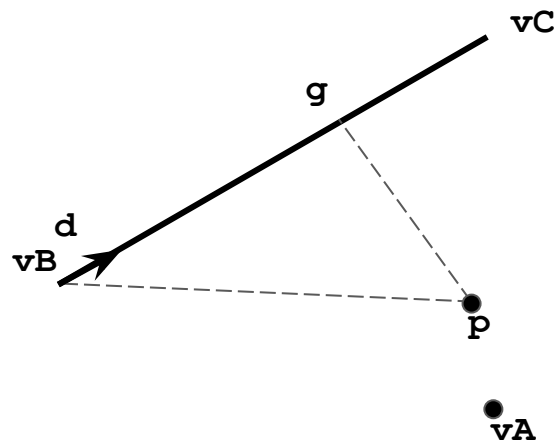
**if** ( $\text{sign}(\text{outer1}) == \text{sign}(\text{outer2})$ )

=> same side

**else**

=> different side,  $\text{distance\_alpha} *= -1;$

Also calculate  $\text{distance\_beta}$  and  $\text{distance\_gamma}$  in a similar way.





# Reading Assignment

Textbook Section 4.4

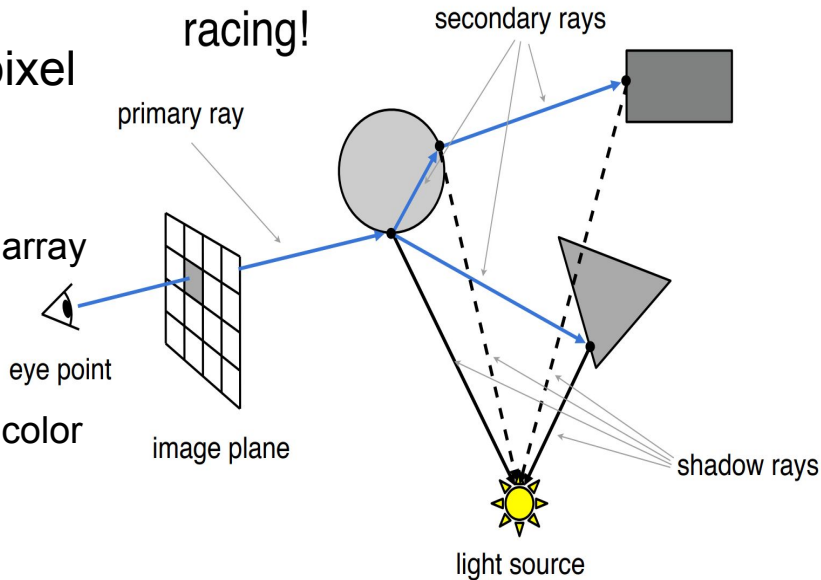
Compare the vector arithmetics solution with the algebraic solution in the textbook  
(e.g. establish a few unknowns, then solve an equation)

Also uploaded Peter Shirley's chapter in Resources (contents are similar)

# Ray tracing in a nutshell

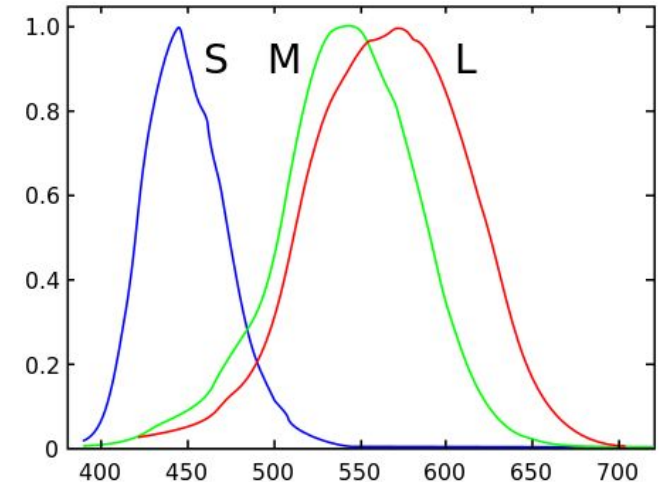
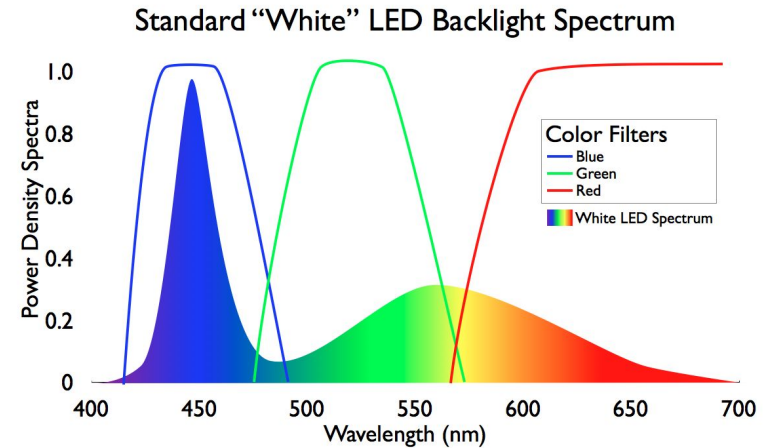
**foreach** pixel in the Screen Space:

1. Cast a ray from PoV that go through that pixel
2. **foreach** object in the World Space:
  - a. Test if the ray hit/intersect with the object
  - b. If it hits, store the intersection point (t only) in an array
3. Pick the **nearest** intersection point and calculate a pixel color.
  - a. If it doesn't hit (e.g. array is empty), set the pixel color to the **Background Color**



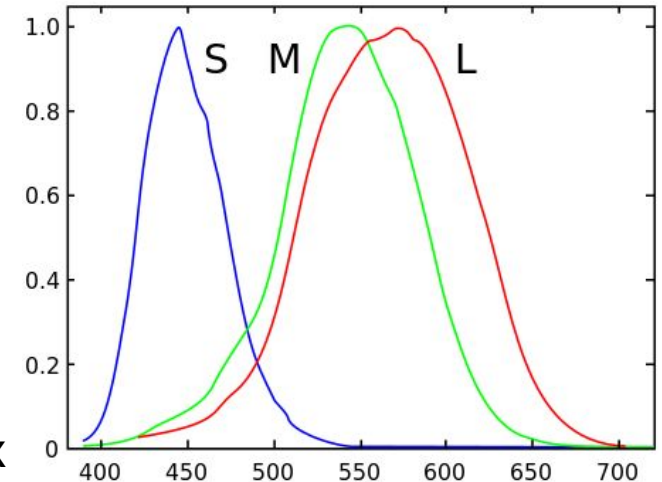
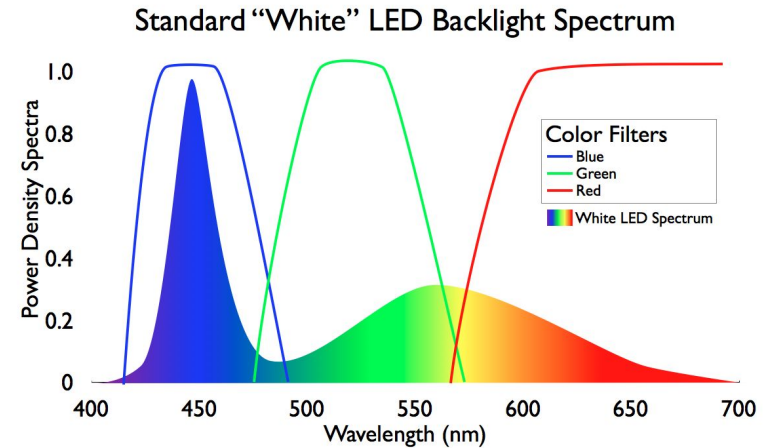
# Summary: RGB Color Space

1. Color is a psychological, inner feeling; and it can be correlated to the excitation levels of the 3 visual cones.
2. To reproduce all color impression we got from the natural world, we need to elicit all possible combinations of excitation levels of the 3 cones with minimum number of artificial light sources.
3. The simplest way is to use also 3 artificial light sources, each of them only excites a single type of cones. (the Dirac function thingy...)



# Summary cont'd

4. When the primary light only excite one type of cone, its strength of light can be approximately considered as the excitation levels in that type of cone, and therefore approximately equals to the feeling of color. (tons of simplifications)
5. The RGB values in computer graphics are strengths of lights, (LCD sub-pixel) emission powers.
6. Grassmann's law states that as long as primary lights appears with the correct color, we can mix them for any colors and the eye cannot tell a spectral difference.



# How to calculate perception based color parameters?

From previous section we have

1. Feeling of brightness is from the **overall/average** excitation of all three kinds of cones.
2. Feeling of color hue is from the **relative difference** of cone excitation levels.
  - a. The larger the difference, the stronger the feeling of color  $\Rightarrow$  higher saturation
3. RGB per-channel light strengths approximately correspond to cone excitation levels.

# How to calculate perception based color parameters?

Therefore:

Luminance =  $(R+G+B) / 3$ , OR  $\max(R, G, B)$ , OR  $\frac{1}{2} (\max(RGB) + \min(RGB))$

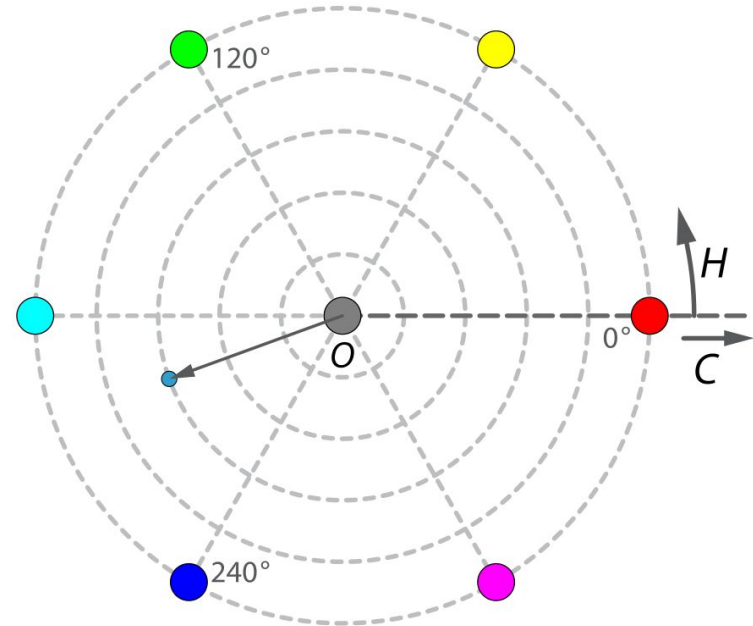
Saturation =  $\max(R, G, B) - \min(R, G, B) / \text{Luminance}$  (for normalization)

Hue???

# How to calculate color hue?

wiki

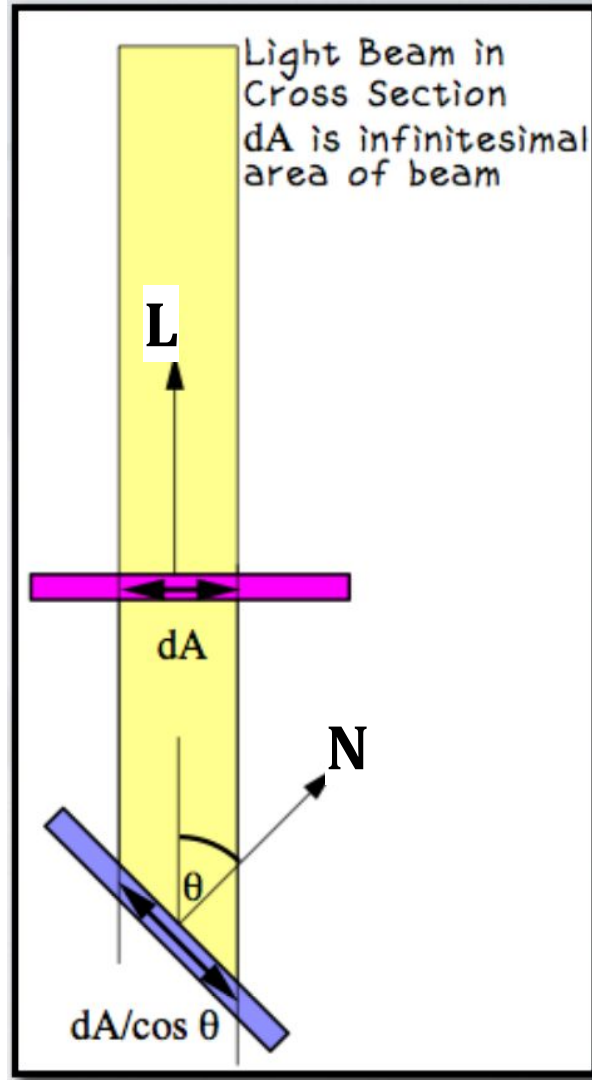
1. **Define** a color circle according to human perception.
  - a. Red =  $0^\circ$
  - b. Green =  $120^\circ$
  - c. Blue =  $240^\circ$
2. Find the maximum RGB component as the **starting point**:  
 $M = \max(R, G, B)$
3. **Move an offset** on the circle according to the remaining two components:
  - a. Assuming  $M = R$ , we start at  $0^\circ$ .
  - b. If  $G > B$ , move towards G according to their difference,  $(G-B)$
  - c. If  $G < B$ , move towards B according to  $(B-G)$



# Lambertian Reflection Model

- Due to *Johann Heinrich Lambert*, A Swiss scientist/polymath in 18th century.
- $K$  is a 3-vector **albedo**, depends on the surface's light reflectance. (Recall: the albedo part in the color lecture).
- Calculated for each R-G-B channels

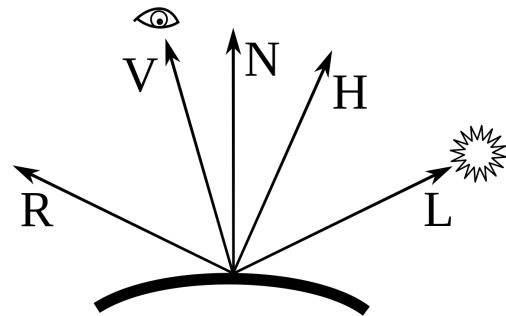
$$R = I_0 K \cos \theta$$
$$= I_0 K (\vec{L} \cdot \vec{N})$$





# Phong-Blinn Specular Reflection Model

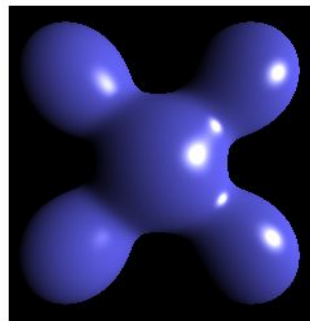
- ***Just add more light*** around the mirror reflection direction... and fall out the added light strength according to angular difference.
- The diffuse light is not touched ...



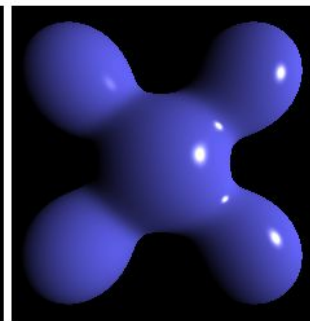
$$R_{+} = I_0 K_s (\vec{R} \cdot \vec{V})^{\alpha} \quad (\text{Phong Original})$$

$$\text{or } + = I_0 K_s (\vec{N} \cdot \vec{H})^{\alpha}$$

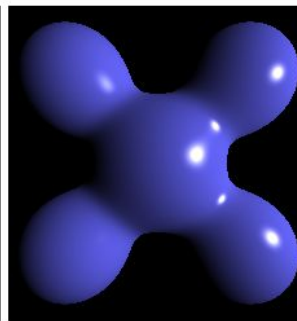
(Phong-Blinn)



Blinn-Phong



Phong



Blinn-Phong  
(higher exponent)

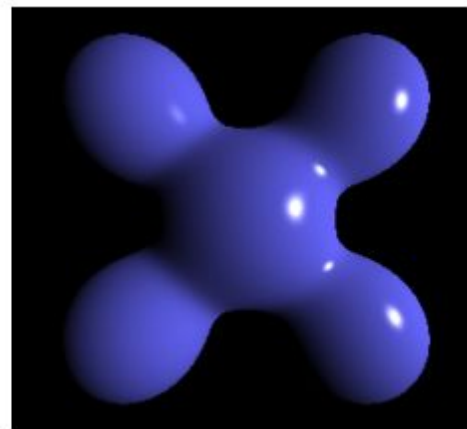
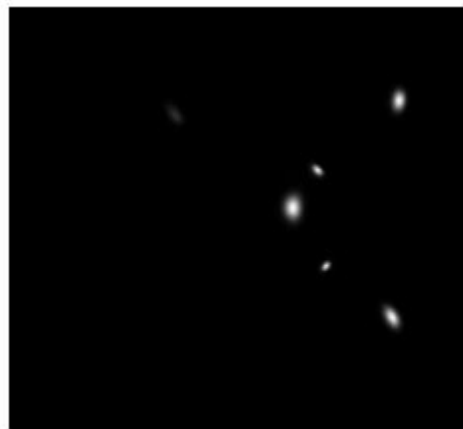
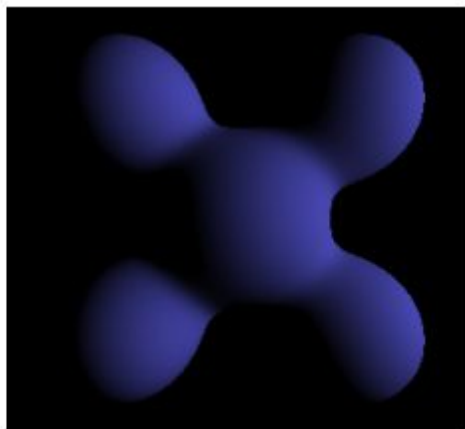
# Full Phong Reflection Model (single light source)

$$R = I_0 (K_a + K_d(\vec{L} \cdot \vec{N}) + K_s(\vec{H} \cdot \vec{N})^\alpha)$$

ambient

diffuse

specular(Blinn formation)



Ambient

+

Diffuse

+

Specular

=

Phong Reflection

# Assignment 1(c)

Given:

1. Texture Coordinate  $(u_a, v_a)$  for vertex A,  $(u_b, v_b)$  &  $(u_c, v_c)$ .
2. The barycentric coordinate  $(\alpha, \beta, \gamma)$  of the intersection point p.

Solve for:

Texture Coordinate  $(u_p, v_p)$  of the intersection point p.

Solution:

$$(u_p, v_p) = \alpha * (u_a, v_a) + \beta * (u_b, v_b) + \gamma * (u_c, v_c)$$

# Assignment 1(c) continued

Given:

1. A raster image that is  $N$  pixels by  $M$  pixels to use as a texture.
2.  $p$ 's UV (texture) coordinate,  $(u_p, v_p)$ .

Solve for:

$p$ 's texture color from the  $N \times M$  image.

Solution:

$p$ 's texture color is the pixel  $(u_p * N, v_p * M)$  on the image.

# What did we learn...

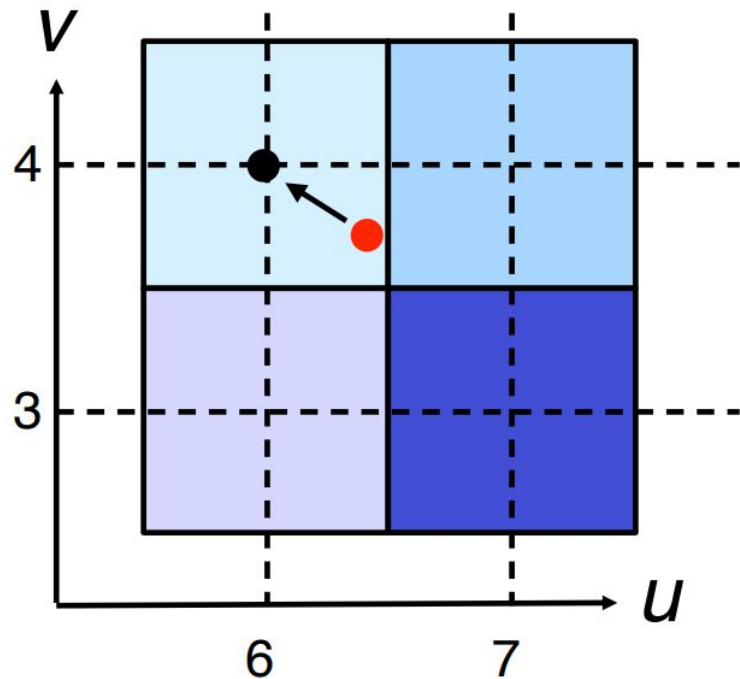
- In practise, texture coordinate is always in the range of  $(0, 1)$ .
  - $u \in [0, 1], v \in [0, 1]$
  - Resolution independent - in this way we don't care the pixel size of the texture image
  - If the texture is not a square, aspect ratio will change, e.g. (squeezed).
- What if  $\mathbf{u}_p * \mathbf{N}$  and  $\mathbf{v}_p * \mathbf{M}$  are not integers?
  - **Texture filtering** - blending between neighbouring pixels
- What if  $\mathbf{u}_p$  and  $\mathbf{v}_p$  go outside of  $[0, 1]$ ?
  - **Texture Tiling & Wrapping**

# Texture Filtering

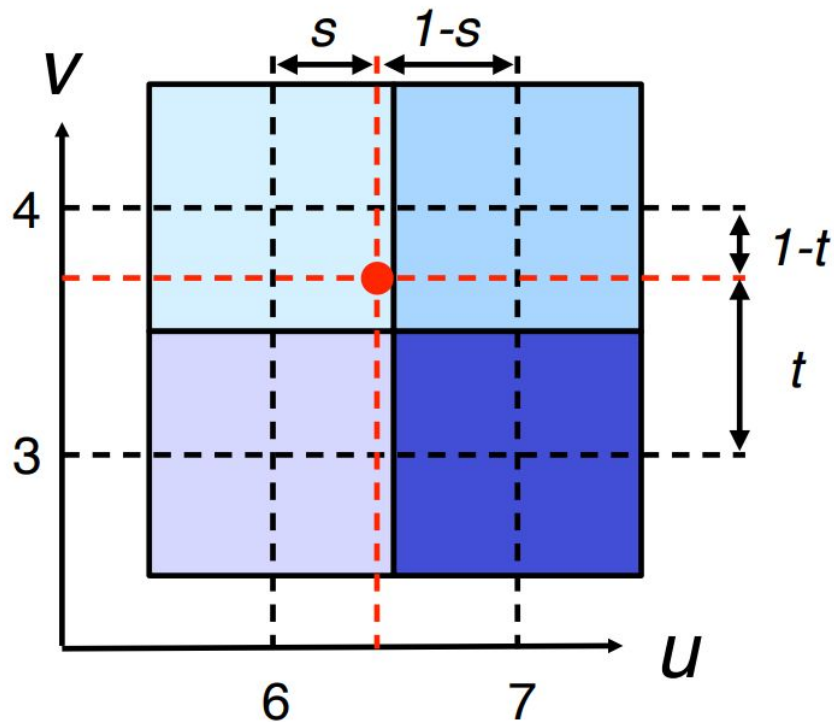
The intersection point **p** corresponds to pixel  $(u_p * N, v_p * M) = (6.4, 3.2)$  on the image...

We can :

- ★ Round to **nearest** pixel (integer coordinate)
- ★ ***Bilinear Interpolation*** with 4 neighboring pixels (Textbook Section 11.3.2  
Blackboard demo with next slide)
- ★ fancier methods (cubic interpolation etc...)




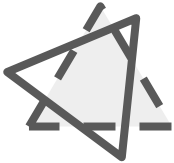
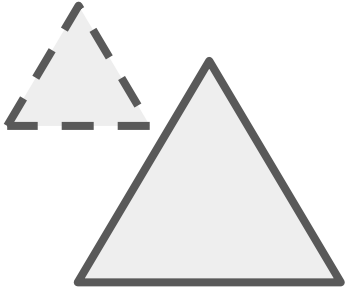
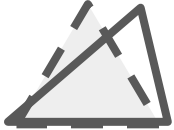
color = tex[6,4]



$$\text{color} = (1-s) \cdot (1-t) \cdot \text{tex}[6,3] + (1-s) \cdot t \cdot \text{tex}[6,4] \\ + s \cdot (1-t) \cdot \text{tex}[7,3] + s \cdot t \cdot \text{tex}[7,4]$$

# Transformations - done by matrices

- Instead of manipulate coordinate by hand, we can **multiply a 3x3 matrix onto the coordinate vector**. (Post multiplication with column vectors)
- $\vec{p}' = \mathbf{M}\vec{p}$
- Capable of Rotate, Scale and Shear (but NOT translate)

Translate	Rotate	Scale	Shear
			



# Homogeneous Coordinates

That 4x3 matrix is kinda of ugly... we want to make it into a nice 4x4 square matrix.


The following are called the Homogeneous Coordinates:

(Transform is now a linear operation...)


$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} m_{00} & m_{01} & m_{02} & t_x \\ m_{10} & m_{11} & m_{12} & t_y \\ m_{20} & m_{21} & m_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# Homogeneous Coordinates

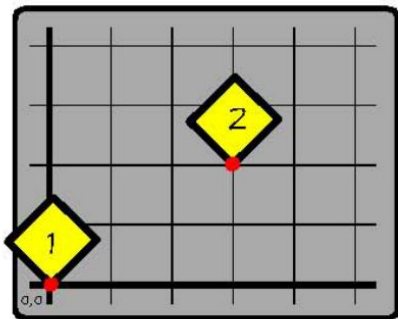
“Homogeneous” means that multiplying by a scalar does not affect the meaning of the coordinate.

Multiply by a scalar 

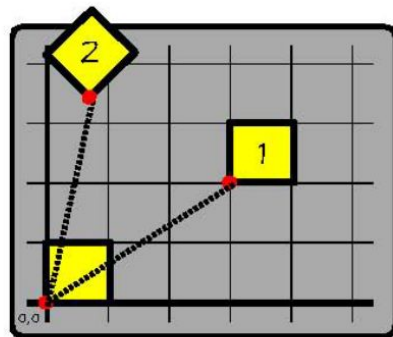
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = w \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} wx' \\ wy' \\ wz' \\ w \end{bmatrix}$$

 Get back to Cartesian coordinates

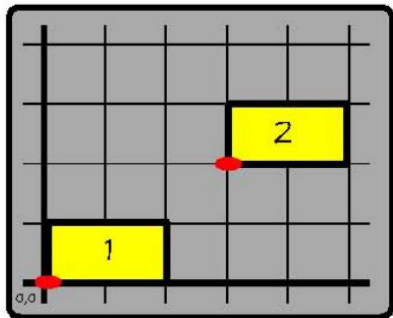
# Order of Concatanation



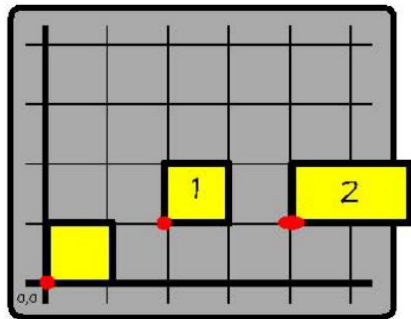
Start with a unit square at the origin.  
Rotatate about the origin  
translate to 3,2



Translate to 3,2 rotate  
Rotate about the origin



Scale about the origin by 2,1  
translate to 3,2



Translate to 2,1  
Scale about the origin by 2,1

# What to do for reading break?

Unity tutorials - esp. the currently first 2.

We won't teach much unity details in the class, but these tutorial help to write excellent assignments+personal projects in Unity!



## Interactive Tutorials (4)

Get Started with Unity.



## Roll-a-ball tutorial (9)

Build your first simple game and Learn to code in C#