

# Welcome!

COMP 557 Fundamentals of Computer Graphics  
Prof: Michael Langer

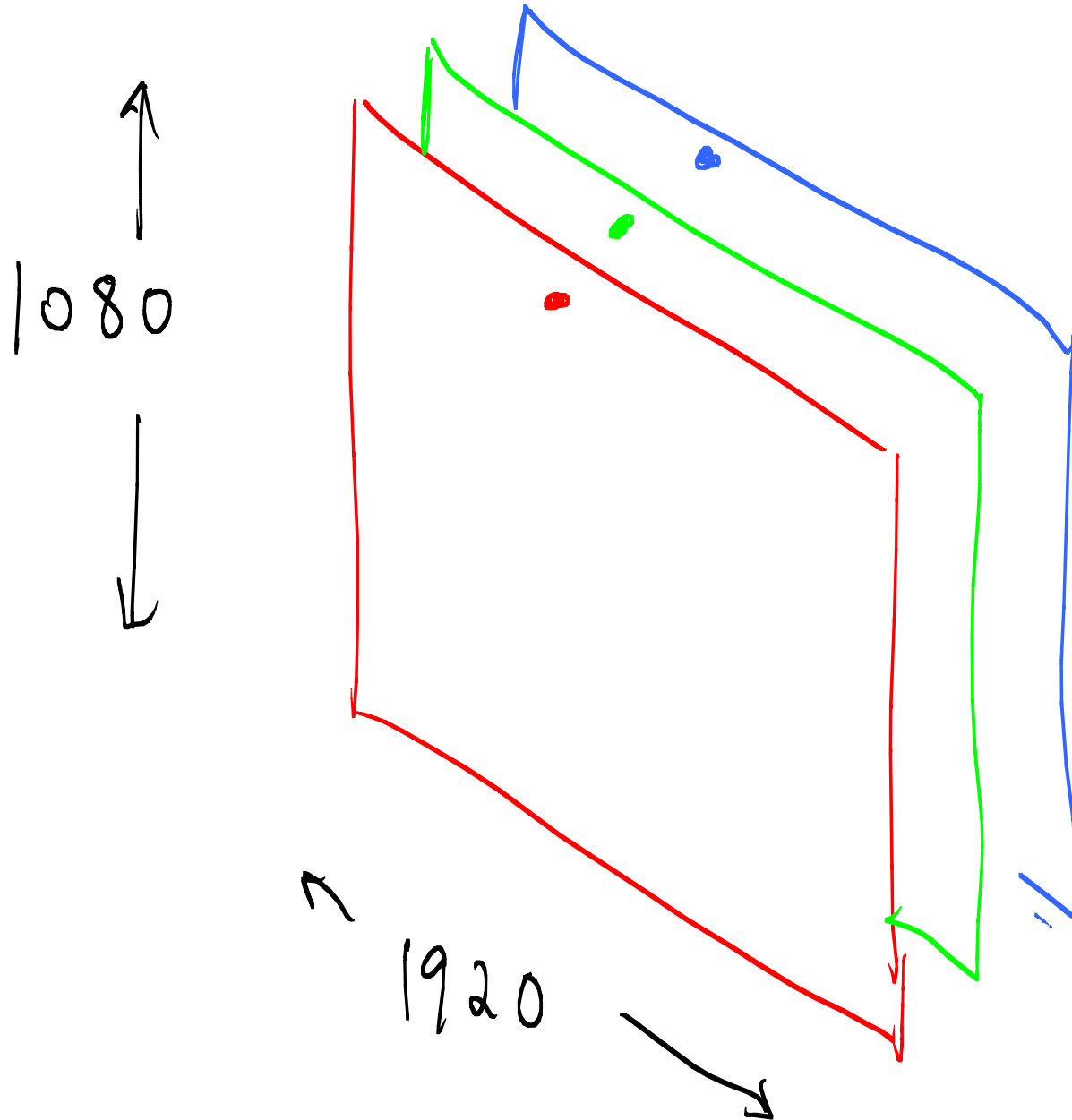
See public web page for this course:

<http://www.cim.mcgill.ca/~langer/557.html>

Q: What is computer graphics (CG) ?

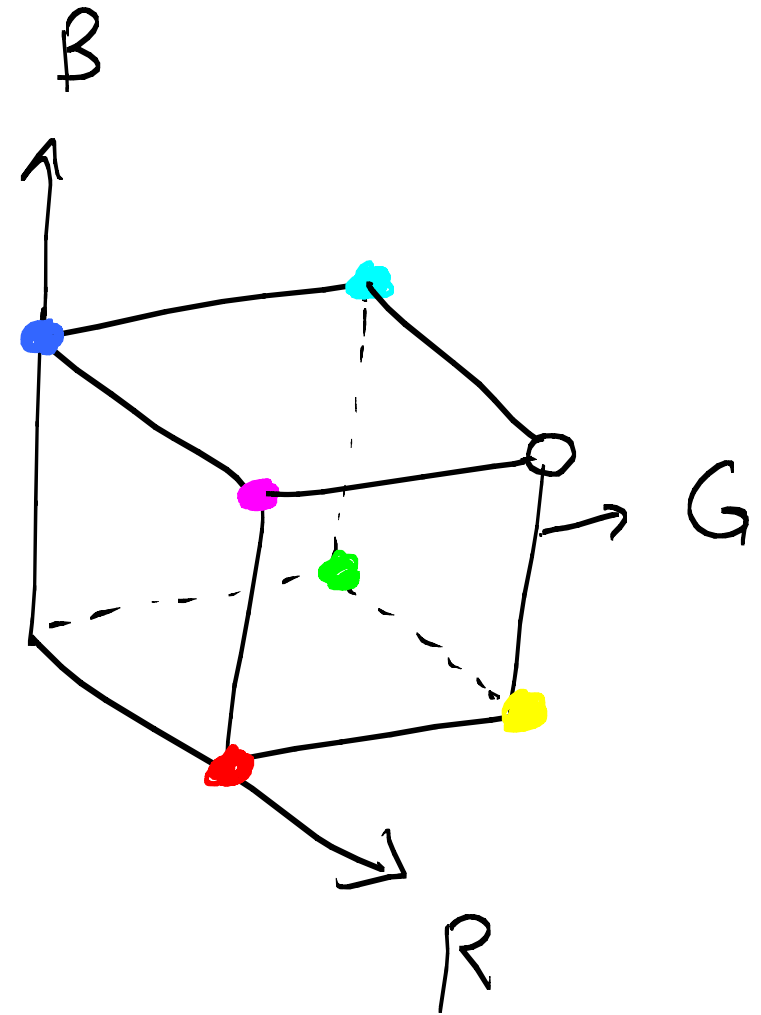
A: Using computers to create and manipulate images.

An image is an array of RGB pixels  
("pixel" = picture element)



Pixels typically use 8 bits per R, G, B value, i.e. [0, 255].

( 0, 0, 0)	black
(255, 255, 255)	white
(255, 0, 0)	red
( 0, 255, 0)	green
( 0, 0, 255)	blue
(255, 255, 0)	yellow
(255, 0, 255)	magenta
( 0, 255, 255)	cyan



# Applications of Computer Graphics

## Entertainment

- Games, Movies, Advertising, VR (Oculus Rift)

## Photo and video editing

## Design

- cars, furniture, machine parts, architecture, 3D printing

## Simulation

- training in virtual environments (VR)

## Visualization

- <http://ieeevis.org/>

<http://www.siggraph.org/>

(Annual conference attracts 20,000 people. Last year was 40th.)

<https://www.youtube.com/user/ACMSIGGRAPH>



## Recent events

### Montréal REWIND 2014

This event will be a review of last year's accomplishments of the Montreal Computer Graphics community. Several short presentations will...

 Date: **11/24/2014 - 19:00 to 21:00**

 Place: **NAD. 405 Ogilvy, 3th floor, Montreal.**

[Read more](#)

-  English
-  Français

### Resources

[Computer Graphics in Montreal](#)

[Jobs in Computer Graphics](#)

[Recent Content](#)

internships

<https://www.facebook.com/pages/Montreal-ACM-SIGGRAPH/139255796191166>

Browser address bar: <https://www.facebook.com/pages/Montreal-ACM-SIGGRAPH/139255796191166>

Navigation bar: Bookmarks, Bookmarks, Dropbox, Suggested Sites, Imported From IE

Facebook search bar: Montreal ACM SIGGRAPH

User profile: Michael Home 20+ Find Friends



**ACM SIGGRAPH**  
**MONTREAL**

# ACM SIGGRAPH

# MONTREAL

**Montreal ACM SIGGRAPH**  
Non-Profit Organization

Like Follow Message

Navigation tabs: Timeline About Events Likes More



## What COMP 557 is:

- fundamental theory (models, algorithms)
- programming assignments (OpenGL 1.0, ...)

## What COMP 557 is NOT:

- modern computer graphics programming ('shading languages', WebGL)

*You can learn that on your own. It will be easier after you have taken this course.*

# Bit of History

- 1970's & 80's - dawn of computer graphics. Many fundamental algorithms/concpts invented, but programming was difficult. (Standards were very basic, and hardware was slow.)
- 1992 - OpenGL 1.0 API released (GL = graphics library).  
'Fixed function pipeline'.
- 1990's - NVidia, ATI, Intel emerge as GPU leaders.  
GPU programming remained difficult (assembly language).
- 2004 - OpenGL 2.0 released. Included GLSL - high level C-like "shading language" for GPU programming.
- 2011 - WebGL released (graphics for web browsers)

# Course Outline (Please read)

<http://www.cim.mcgill.ca/~langer/557/CourseOutline.pdf>

- Reference material (my lecture notes and slides)
- Prerequisites (linear algebra, Cal III (?), COMP 206, 251)
- Evaluation (4 assignments 40%, midterm 20% or 0%,  
final exam 40 or 60% )
- Policies ( grading, discussion board, .. )

# Overview of Course

# 1. Viewing Transformations (lectures 2-6)

What is the geometric relationship between 2D pixel coordinates of an image and 3D coordinates of the points in the scene that we want to draw ?

Geometry can be defined in several coordinate systems (object, world, eye).

**Linear Algebra is used heavily here.** The main new concept is "homogeneous coordinates".

## 2. Visibility, Object and Scene Modelling (lectures 6-12)

Which objects can be seen in an image ? (how to discard "hidden surfaces" ?)

How to model object shapes, e.g. hierarchical (plants), smooth (cubics and bicubics) ?

This is more **data structures and algorithms** (sorting, trees, stacks). Some linear algebra and Cal III.

---> **MIDTERM EXAM (THURS. FEB 19)**

### 3. Rendering: shading, texture, transparency (lectures 13-20)

How do add light source?

How do we define surface materials (glossy, matte)?

How do we add texture patterns to surfaces (wallpaper) ?

How do we combine image layers ? (compositing)

**Cal III (partial derivatives) & linear algebra used here.**

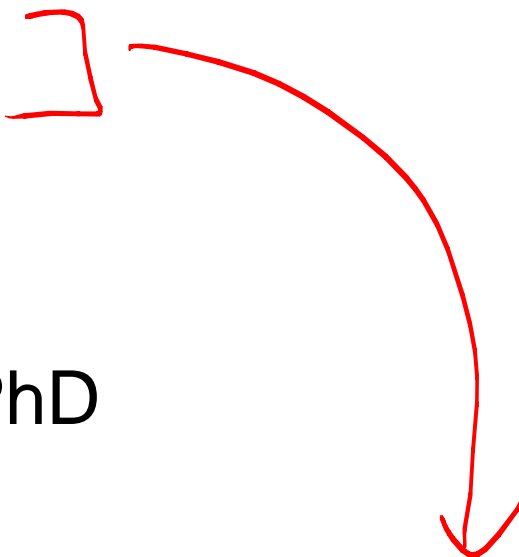
## 4. Image Capture and Display (lectures 21-25)

- cameras and monitors/projectors
- perceptual issues (including some of my own research)



# Who are you ?

120 students currently registered:

~25 U1 or U2 

~75 U3

~20 MSc or PhD

You have the option to wait for Fall 2015. Paul Kry will be back from sabbatical and will offer it then.

His version of the course will be more challenging than mine. (He covers more material. He plans to introduce some basic GLSL.)

He will also offer COMP 599 (Computer Animation) in Winter 2016.

# Want to get involved in research ?

See [www.cim.mcgill.ca/~langer/resources-gradschool.html](http://www.cim.mcgill.ca/~langer/resources-gradschool.html)

Undergraduates:

- COMP 400 Honours Project in CS
- COMP 396 Undergraduate Research Project

*Either can be done over the summer.*

Graduate students (M.Sc.):

- Project or Thesis

# Who am I ?

- grew up in Toronto
- BSc at McGill in early 1980s (Math Major, CompSci Minor)
- MSc in CompSci at U of T (Toronto) in late 1980s (thesis on image coding)
- PhD at McGill in early 1990s (thesis in computer vision, topic "shape from shading")
- postdoc in late 1990s
  - computer vision, at NEC Research in NJ, USA (3 years)
  - human vision, at MPI in Germany (2 years)
- professor here since 2000 (I have taught COMP 557 four times)  
<http://www.cim.mcgill.ca/~langer/>

# My Research Interests

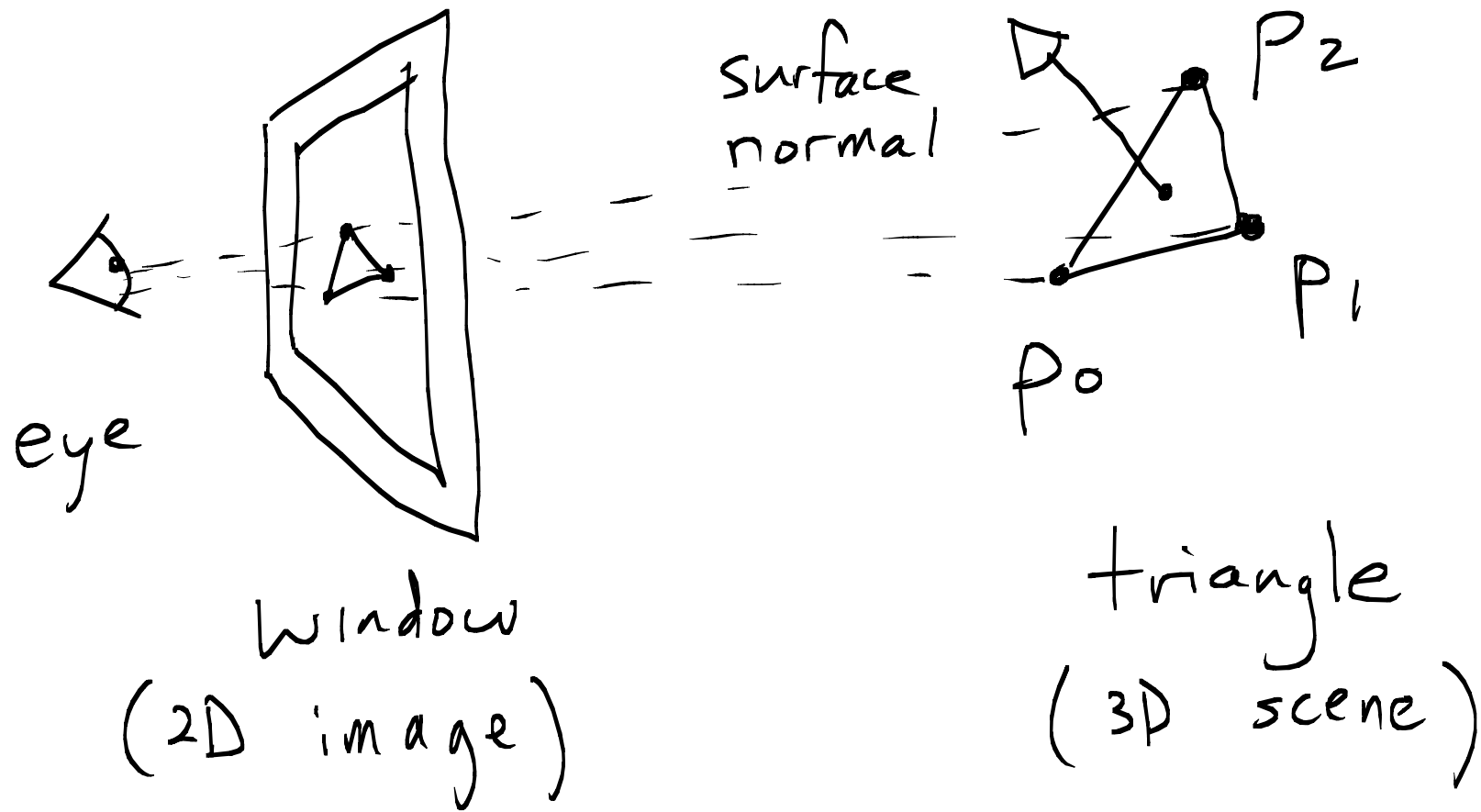
- Computational Models of Human Vision
  - in particular, depth perception
  - (COMP 546 Computational Perception. Offered in 2015-16)
- Computer Vision
  - may also teach COMP 558 next year
- Applications of Perception in Computer Graphics

If there are no questions, then  
we can get started...

Today:

- review of basic linear algebra:  
dot product, cross product

# Why do we need linear algebra?



# 2D Dot Product (Inner Product)

$$\text{Given } \begin{cases} \vec{u} = (u_x, u_y) \\ \vec{v} = (v_x, v_y) \end{cases}$$

$$\vec{u} \cdot \vec{v} \equiv \begin{bmatrix} u_x & u_y \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

$$= u_x v_x + u_y v_y$$

For fixed  $\vec{u}$ ,  $\vec{u} \cdot \vec{v}$  is  
a linear transform on  $V$ .

$$\text{i.e. } \vec{u} \cdot (a\vec{v}_1 + \vec{v}_2) = a\vec{u} \cdot \vec{v}_1 + \vec{u} \cdot \vec{v}_2$$

For fixed  $\vec{v}$ ,  $\vec{u} \cdot \vec{v}$  is  
a linear transform on  $U$

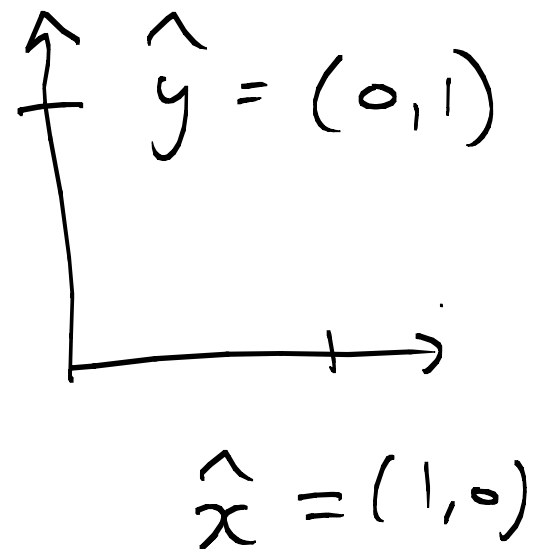


$$\begin{aligned}
 & (u_x \hat{x} + u_y \hat{y}) \cdot (v_x \hat{x} + v_y \hat{y}) \\
 &= u_x v_x + u_y v_y
 \end{aligned}$$

$$\hat{x} \cdot \hat{x} = 1$$

$$\hat{y} \cdot \hat{y} = 1$$

$$\hat{x} \cdot \hat{y} = 0$$

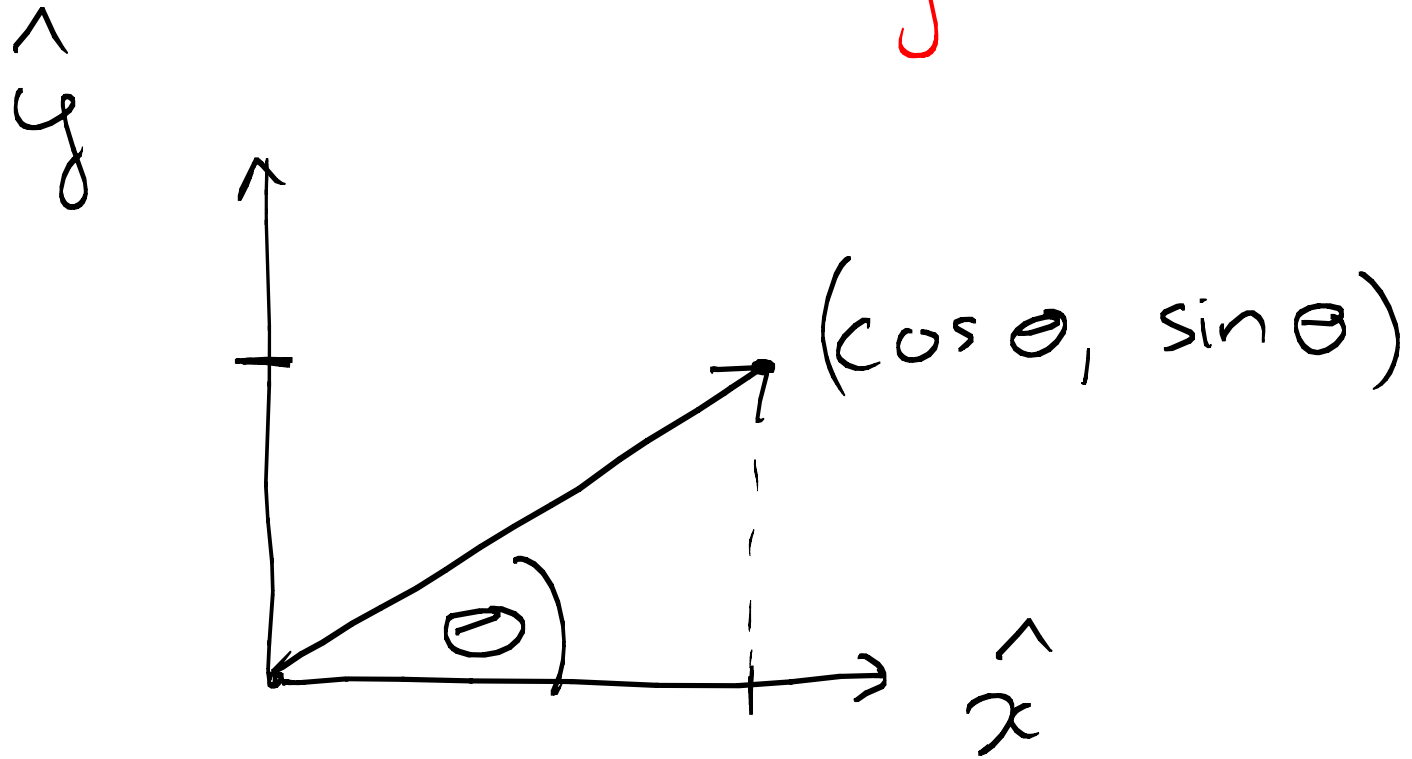


$$\begin{aligned}\vec{u} \cdot \vec{v} &= u_x v_x + u_y v_y \\ &= |\vec{u}| |\vec{v}| \cos \theta\end{aligned}$$

EXERCISE  
why?

where  $\theta$  is angle between  
 $u$  and  $v$ .

Hint (but how to generalize?)



$$(\cos \theta, \sin \theta) \cdot (1, 0)$$

$$= \cos \theta$$

# 3D Dot product (inner product)

Given

$$\vec{u} = (u_x \ u_y \ u_z)$$

$$\vec{v} = (v_x \ v_y \ v_z)$$

Define

$$\vec{u} \cdot \vec{v} \equiv u_x v_x + u_y v_y + u_z v_z$$

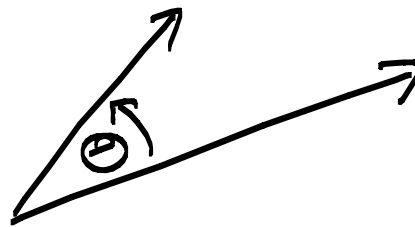
$$\equiv |\vec{u}| |\vec{v}| \cos \theta$$

EXERCISE  
why?

## "2D Cross Product"

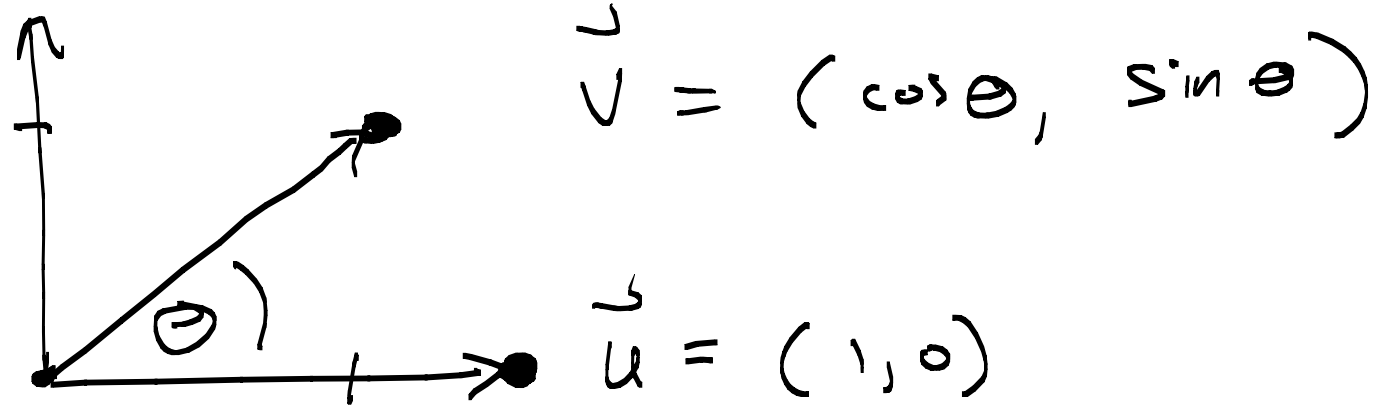
As was pointed out by a student during class, the term "cross product" is an operation on two 3D vectors. When I write "2D cross product" here, I mean  $(u_x, u_y, 0)$  and  $(v_x, v_y, 0)$ , i.e. two vectors in the XY plane. Their cross product is a vector in the Z direction.

$$\vec{v} = (v_x, v_y)$$


$$\vec{u} = (u_x, u_y)$$

$$\begin{aligned} |\vec{u} \times \vec{v}| &\equiv u_x v_y - v_x u_y \\ &\equiv |\vec{u}| |\vec{v}| \sin \theta \end{aligned} \quad \left. \vphantom{\begin{aligned} |\vec{u} \times \vec{v}| &\equiv u_x v_y - v_x u_y \\ &\equiv |\vec{u}| |\vec{v}| \sin \theta \end{aligned}} \right\} \begin{array}{l} \text{not} \\ \text{obvious} \end{array}$$

## "2D Cross Product": Example



$$|\vec{u} \times \vec{v}| = \sin \theta$$

# 3D Cross Product

$\vec{u} \times \vec{v}$  is a 3D vector that:

- is perpendicular to  $u$  and  $v$   
(and plane spanned by  $\vec{u}, \vec{v}$ )
- has length  $|u||v| \sin \theta$  where  
 $\theta$  is angle between  $\vec{u}, \vec{v}$

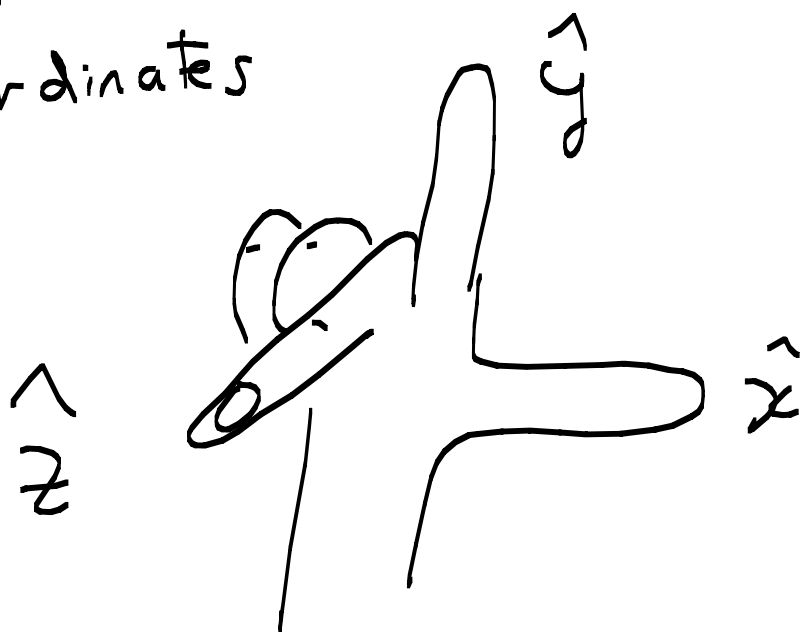
# 3D Cross Product

$$\hat{x} \times \hat{y} = \hat{z}$$

$$\hat{y} \times \hat{z} = \hat{x}$$

$$\hat{z} \times \hat{x} = \hat{y}$$

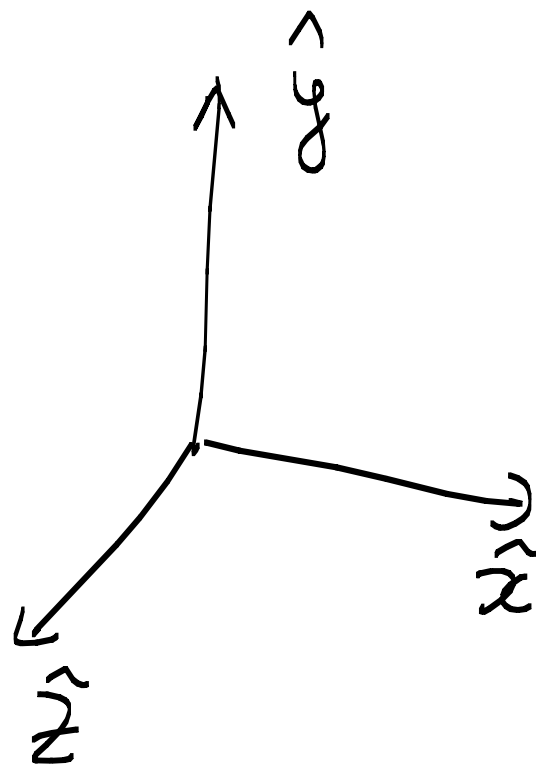
right handed  
coordinates



for all  $\vec{u}, \vec{v}$ ,

$$\vec{u} \times \vec{v} = -\vec{v} \times \vec{u}$$

$$\vec{u} \times \vec{u} = \vec{0}$$





$$\vec{u} \times \vec{v}$$

$$= (u_x \hat{x} + u_y \hat{y} + u_z \hat{z}) \times (v_x \hat{x} + v_y \hat{y} + v_z \hat{z})$$

= multiplying out gives 9 terms,  
3 of which are 0.

Grouping the remaining 6 terms gives:

$$\vec{u} \times \vec{v} = \begin{bmatrix} u_y v_z - u_z v_y \\ -(u_x v_z - u_z v_x) \\ u_z v_y - u_y v_x \end{bmatrix}$$

← recall  
earlier  
formula  
for 2D

Or if you prefer:

$$\vec{u} \times \vec{v} \equiv \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix}$$

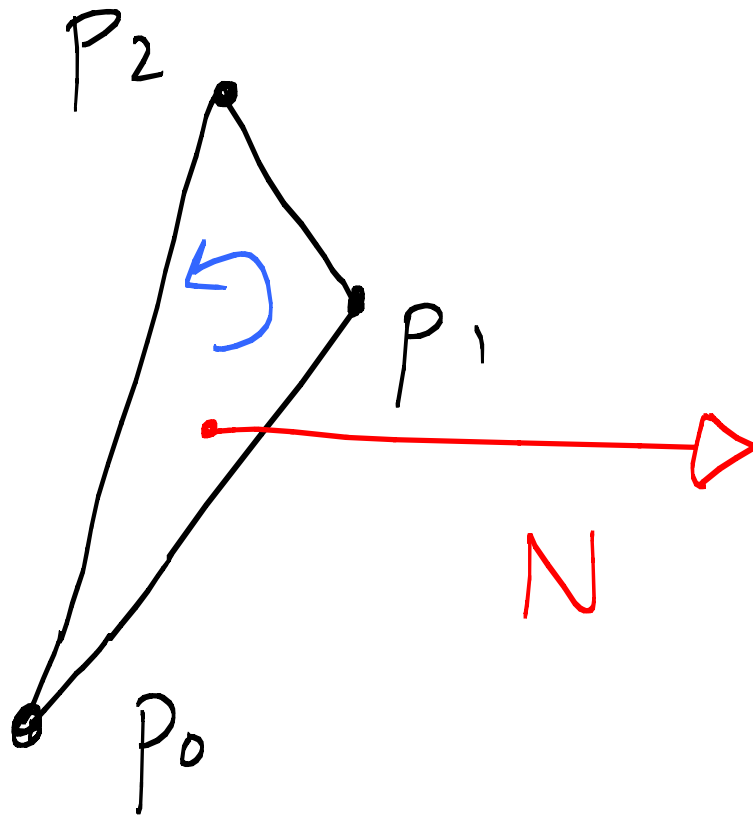
↑  
determinant

Or if you prefer:

$$\vec{u} \times \vec{v} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

i.e. For given  $\vec{u}$ ,  $\vec{u} \times \vec{v}$  is a  
linear transform on  $\vec{v}$ .

How to use cross product to  
define **surface normal** of triangle?

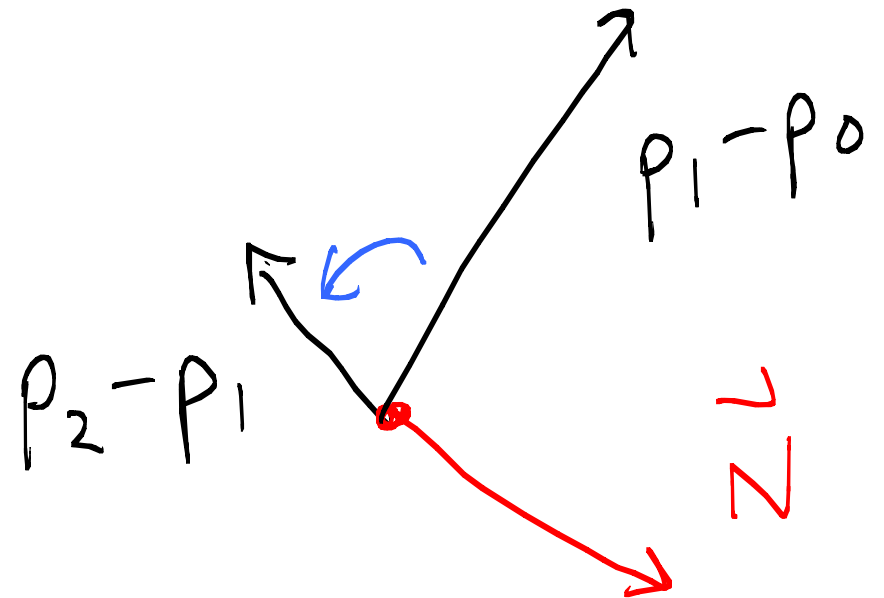
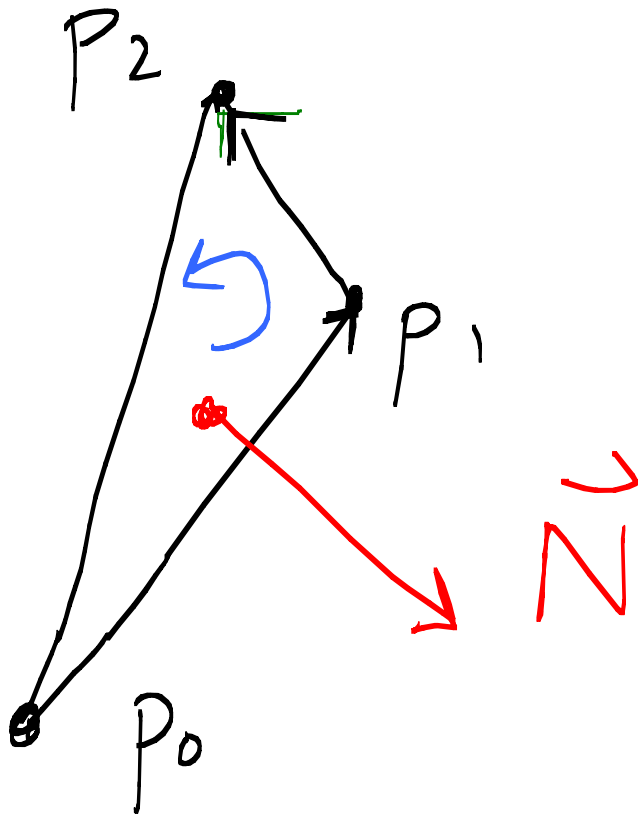


eye

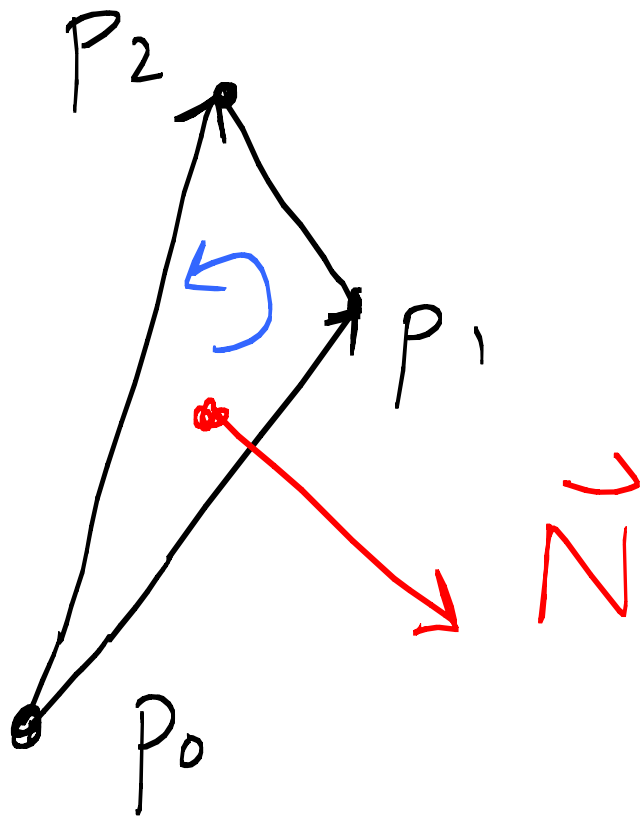


Assume vertices  
are labelled  
counter clockwise.

$$\vec{N} = (p_1 - p_0) \times (p_2 - p_1)$$



What is the equation of the  
plane containing the polygon?



Take any  $\vec{p} = (x, y, z)$  in plane.

$$\vec{N} \cdot (\vec{p} - \vec{p}_0) = 0$$

$$N_x (x - p_{0x}) + N_y (y - p_{0y}) + N_z (z - p_{0z}) = 0$$