

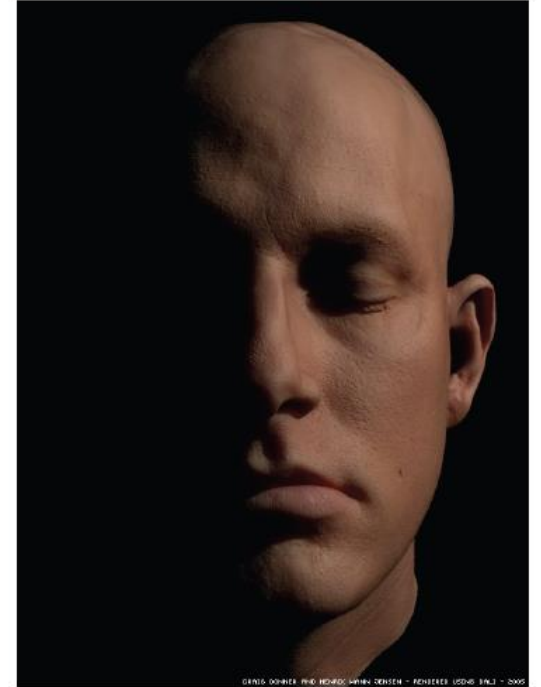
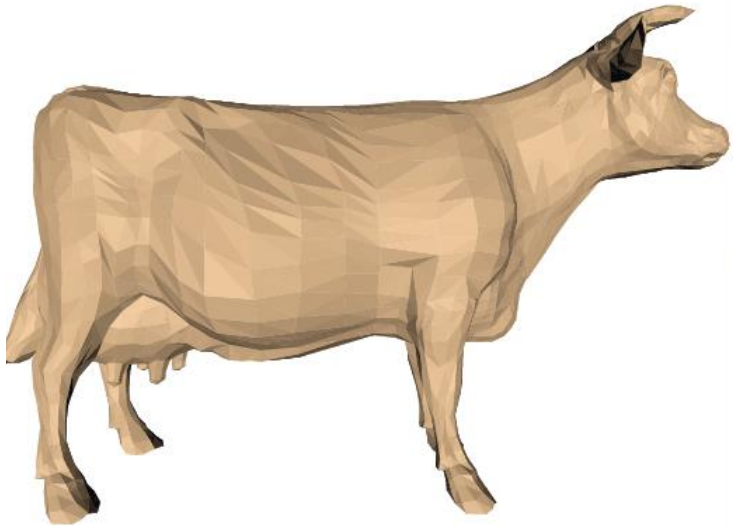
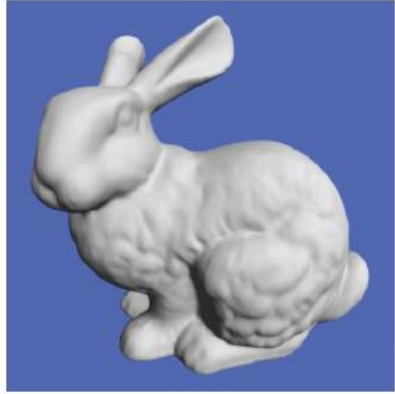
# CSE160 – Oct 8

- Everything is triangles
- OpenGL Primitives
- How is this stored in buffers
- Rasterization
- Normals
- Interpolation
- Non triangle modeling
- Assignment 1
- Administrative
- Q&A

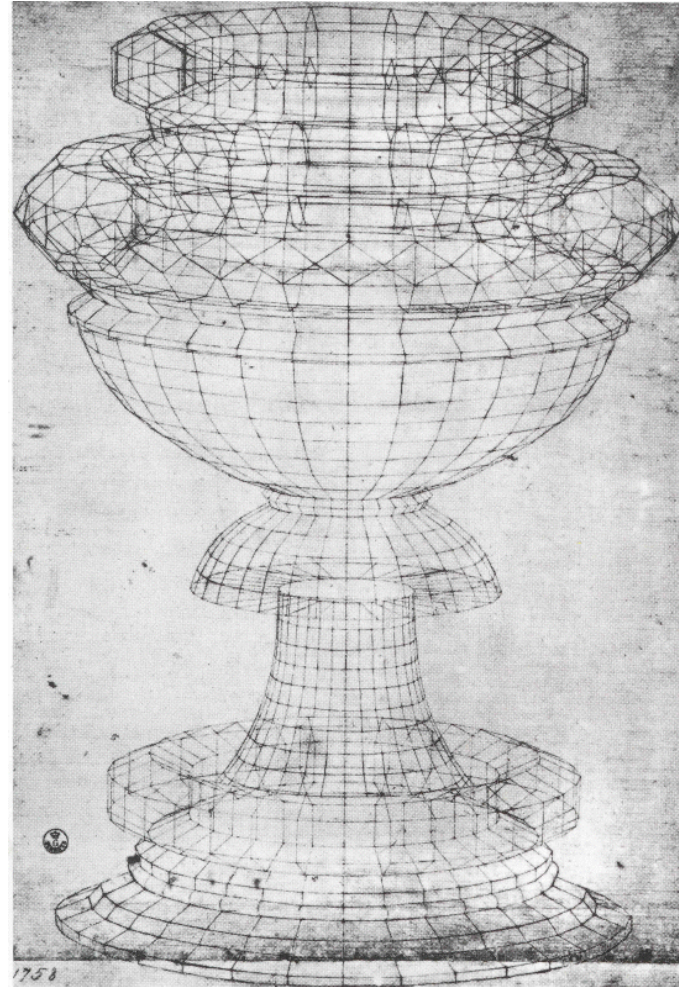
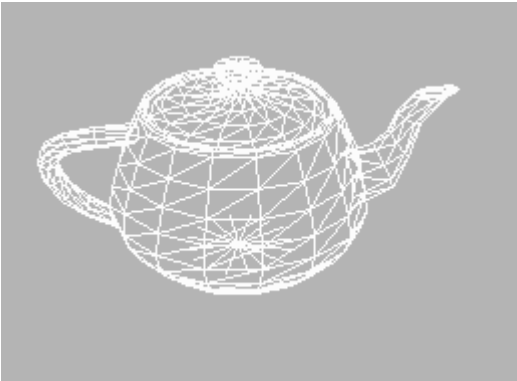
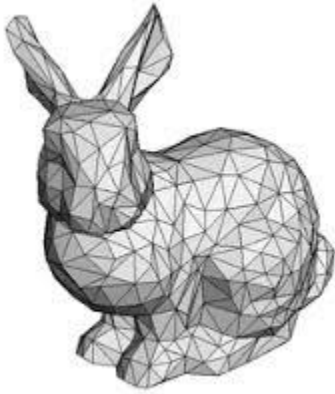
Everything is made of ~~atoms~~  
triangles

# Triangle Meshes

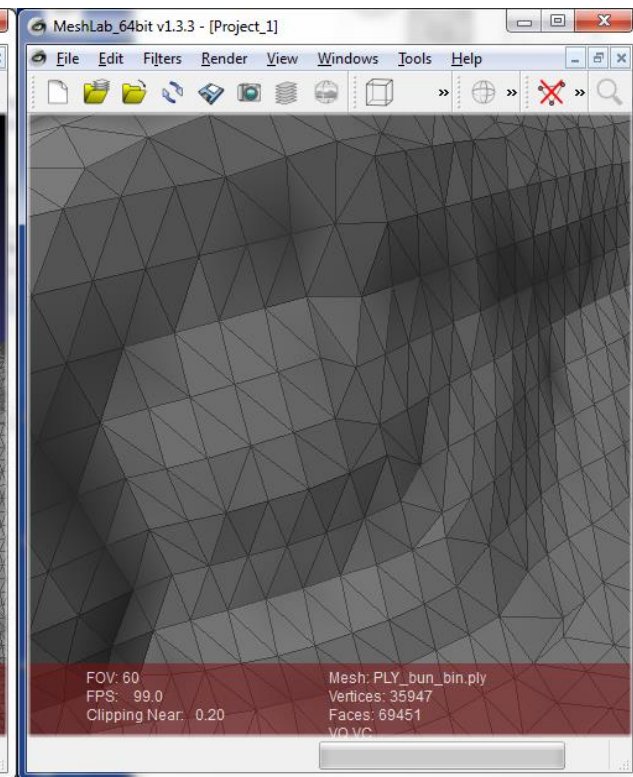
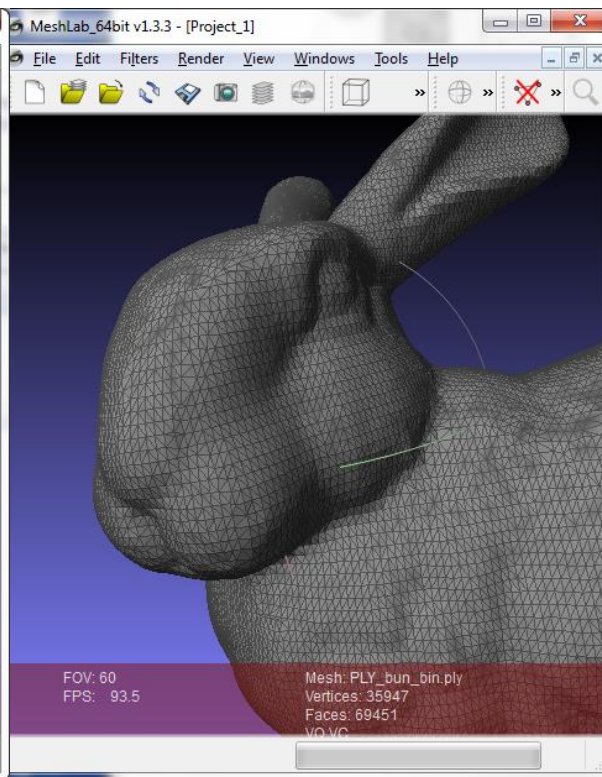
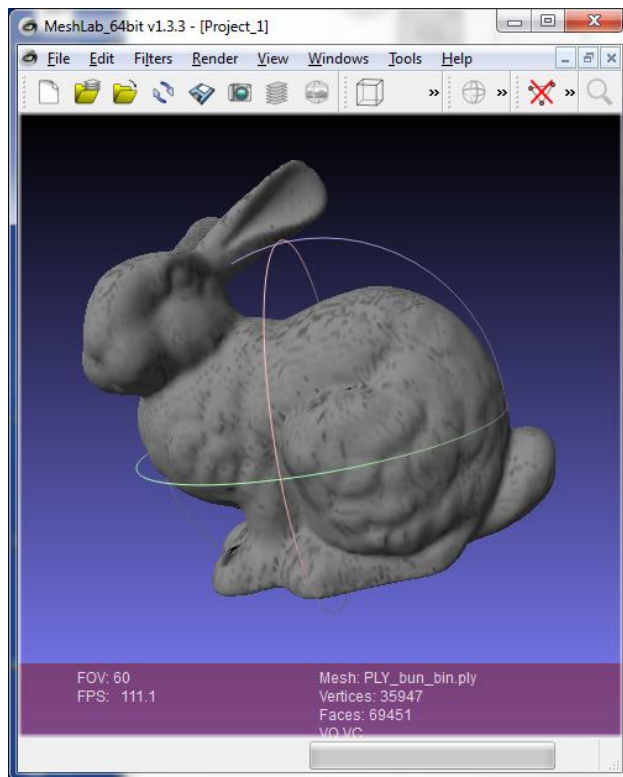
---



# Everything made of triangles



Pen and ink drawing of a wireframe chalice ("Perspective Study of a Chalice"), done by Paolo Uccello in 1430-1440, Florence, Italy.

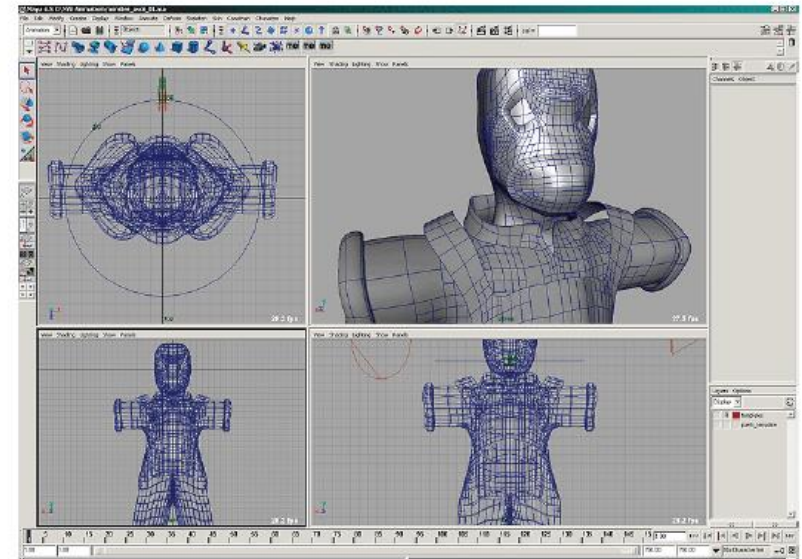




# Mesh Generation

## Modeling

- Software packages like Maya, Blender, etc. are powerful but hard to use
- Tremendous time investment needed to create complex models



## Laser Scanning

- Good for capturing real objects
- Scanners are expensive
- Registering multiple scans is difficult
- Turning point data into triangles is also non-trivial

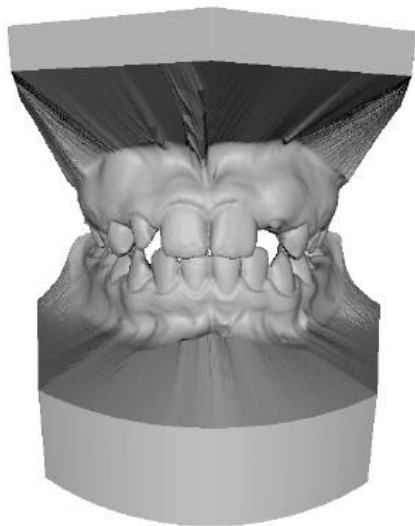


# Level of Detail

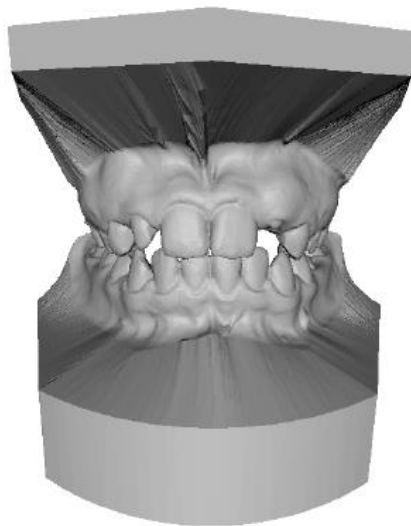
---

## Far Away Objects Need Less Detail

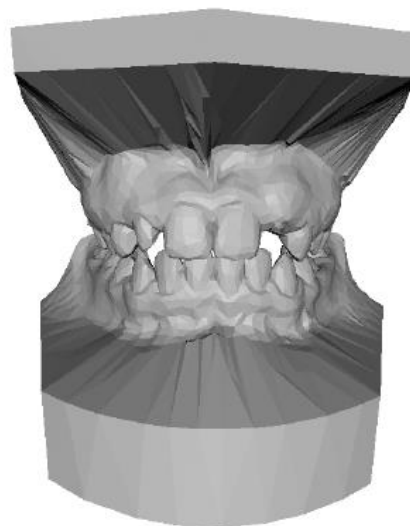
- Acquisition systems often produce huge models
- Create multiple versions of models
- Pick the correct version for each view
- Can result in substantial performance gains
- Simplification is nontrivial



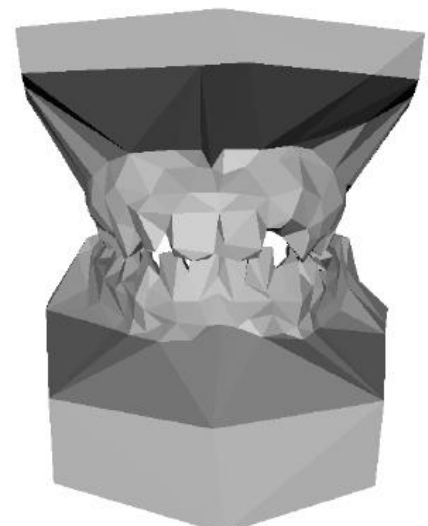
424,376



60,000



8,000



1,000

# OpenGL Primitives

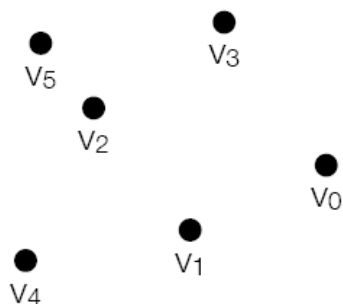


# Points in OpenGL

---

## GL\_POINTS

- Draws square pixel region on screen
- One pixel wide by default
- With antialiasing, circular region drawn with smooth edges
- Size controllable with `glPointSize()`
- Easy, efficient way to activate pixels



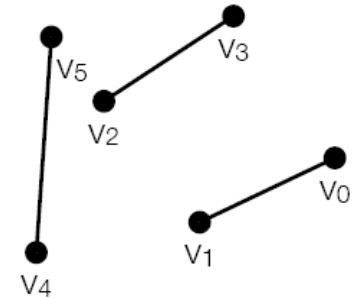
```
glPointSize(10.0f);  
glBegin(GL_POINTS);  
    for(int i=0; i<N; i++)  
        glVertex3fv(v[i]);  
glEnd();
```

# Lines in OpenGL

---

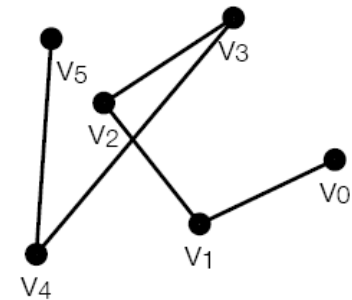
## GL\_LINES

- Draws lines one pixel wide
- Width can be controlled by `glLineWidth()`
- Successive *pairs* of vertices specify segments



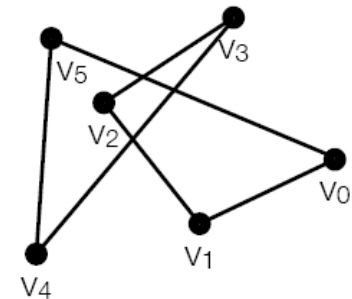
## GL\_LINE\_STRIP

- Like GL\_LINES, but successive vertices specify next connected segment in strip



## GL\_LINE\_LOOP

- Like GL\_LINE\_STRIP, but also connects last and first vertex

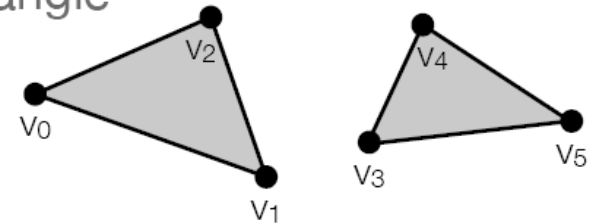


# Triangles in OpenGL

---

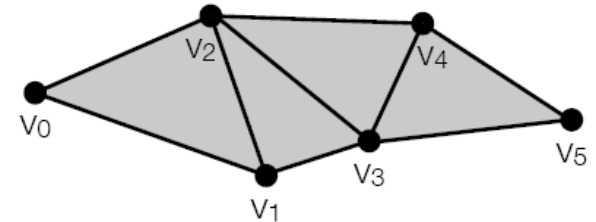
## GL\_TRIANGLES

- Successive vertex triples specify individual triangles
- Requires three vertices to be emitted for every triangle



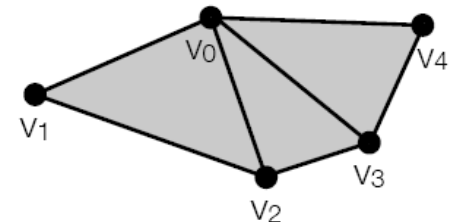
## GL\_TRIANGLE\_STRIP

- First triple specifies first triangle
- Subsequent vertices *each* specify new triangle, along with previous two vertices
- One vertex emitted per triangle in long strips
- But stripifying meshes is nontrivial



## GL\_TRIANGLE\_FAN

- First vertex is center of fan
- Subsequent vertices form ordered boundary
- One vertex emitted per triangle for dense fans
- But few such fans arise in practice

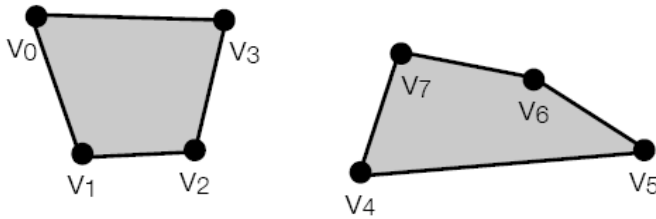


# Other Primitives

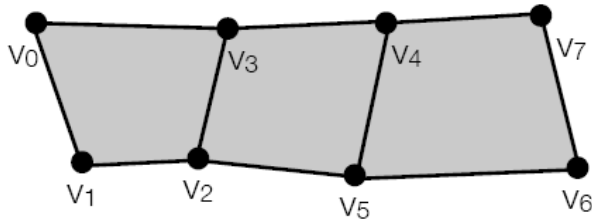
*Not in WebGL*

## GL\_QUADS

- OpenGL only handles **planar** quadrilaterals properly



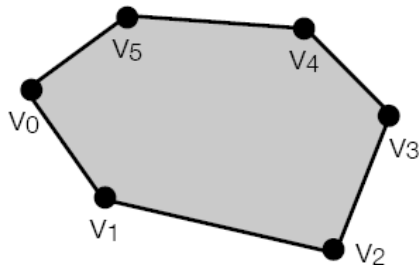
## GL\_QUAD\_STRIP



These primitives are trouble.  
It's safer to stick to triangles.

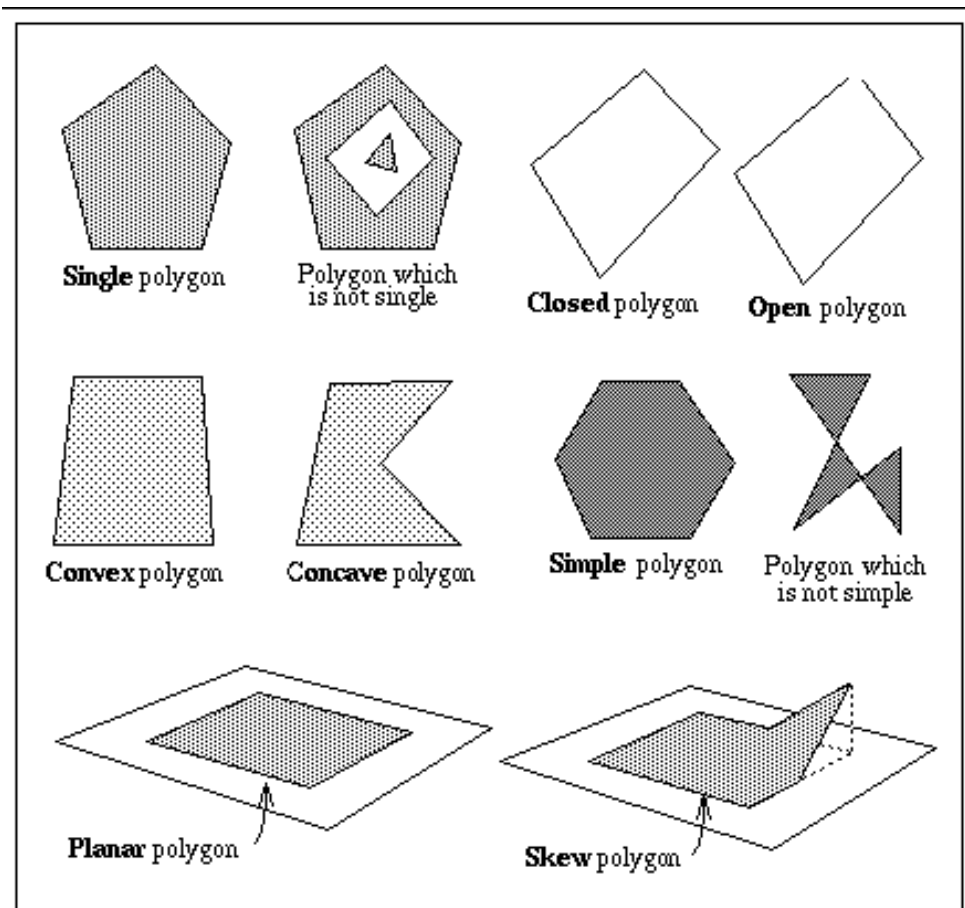
## GL\_POLYGON

- OpenGL only handles **convex** polygons properly

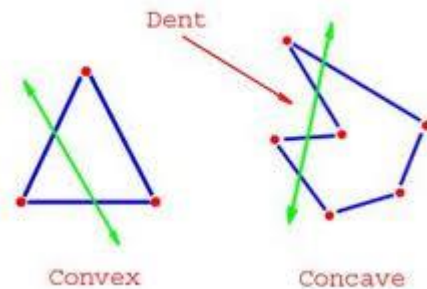


## Reasons triangles are better

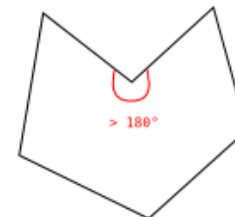
- Definitely planar
- Definitely convex
- Definitely not self intersecting
- Exactly 3 vertices always



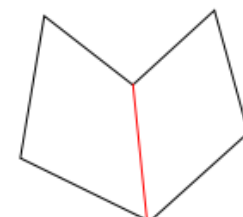
## Testing convexity



Concave polygon has interior angle(s)  $> 180^\circ$



Must be split up into multiple convex polygons. For example:





# Polygonal Meshes

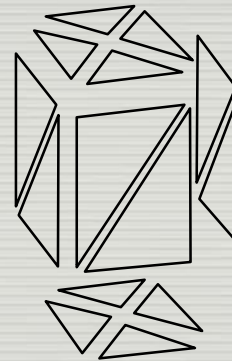
*You only specify vertices*



vertices



edge



faces

(image courtesy of Wikipedia)

## Q: What will render a square?

(A)

```
glBegin(GL_TRIANGLES)
  glVertex3f(0,0,0);
  glVertex3f(1,1,0);
  glVertex3f(1,0,0);
  glVertex3f(0,0,0);
  glVertex3f(0,1,0);
  glVertex3f(1,1,0);
glEnd();
```

(C)

```
glBegin(GL_TRIANGLES)
  glVertex3f(0,0,0);
  glVertex3f(1,1,0);
  glVertex3f(1,0,0);
  glVertex3f(0,0,0);
  glVertex3f(1,1,0);
  glVertex3f(0,1,0);
glEnd();
```

(B)

```
glBegin(GL_QUADS)
  glVertex3f(0,0,0);
  glVertex3f(0,1,0);
  glVertex3f(1,0,0);
  glVertex3f(1,1,0);
glEnd();
```

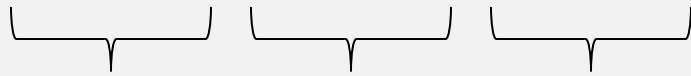
(D)

```
glBegin(GL_QUADS)
  glVertex3f(0,0,0);
  glVertex3f(0,1,0);
  glVertex3f(1,1,0);
  glVertex3f(1,0,0);
glEnd();
```

(E) I just really don't know

How is this stored in buffers

[V0.x, V0.y, V1.x, V1.y, V2.x, V2.y, V3.x, V3.y, V4.x, V4.y, V5.x, V5.y, V6.x, V6.y, ...]



Point 0

Point 1

Point 2

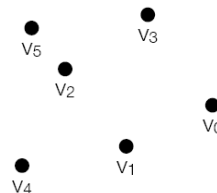
*Attribute Vec2 a\_Position;*

`drawArrays(gl.POINTS, 0, n);`

## Points in OpenGL

### GL\_POINTS

- Draws square pixel region on screen
- One pixel wide by default
- With antialiasing, circular region drawn with smooth edges
- Size controllable with `glPointSize()`
- Easy, efficient way to activate pixels



```
glPointSize(10.0f);  
glBegin(GL_POINTS);  
    for(int i=0; i<N; i++)  
        glVertex3fv(v[i]);  
glEnd();
```

[V0.x, V0.y, V1.x, V1.y, V2.x, V2.y, V3.x, V3.y, V4.x, V4.y, V5.x, V5.y, V6.x, V6.y, ...]

Line 0

Line 1

Line 2

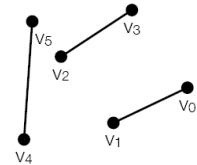
*Attribute Vec2 a\_Position;*

`drawArrays(gl.LINES, 0, n/2);`

## Lines in OpenGL

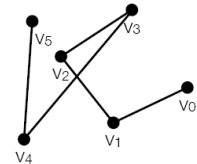
### GL\_LINES

- Draws lines one pixel wide
- Width can be controlled by `glLineWidth()`
- Successive *pairs* of vertices specify segments



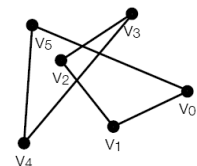
### GL\_LINE\_STRIP

- Like GL\_LINES, but successive vertices specify next connected segment in strip



### GL\_LINE\_LOOP

- Like GL\_LINE\_STRIP, but also connects last and first vertex





[V0.x, V0.y, V1.x, V1.y, V2.x, V2.y, V3.x, V3.y, V4.x, V4.y, V5.x, V5.y, V6.x, V6.y, ...]

Triangle 0

Triangle 1

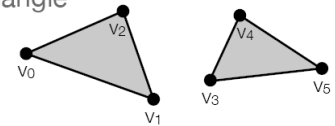
*Attribute Vec2 a\_Position;*

```
drawArrays(gl.TRIANGLES, 0, n/3);
```

## Triangles in OpenGL

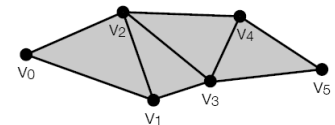
### GL\_TRIANGLES

- Successive vertex triples specify individual triangles
- Requires three vertices to be emitted for every triangle



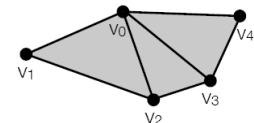
### GL\_TRIANGLE\_STRIP

- First triple specifies first triangle
- Subsequent vertices *each* specify new triangle, along with previous two vertices
- One vertex emitted per triangle in long strips
- But stripifying meshes is nontrivial

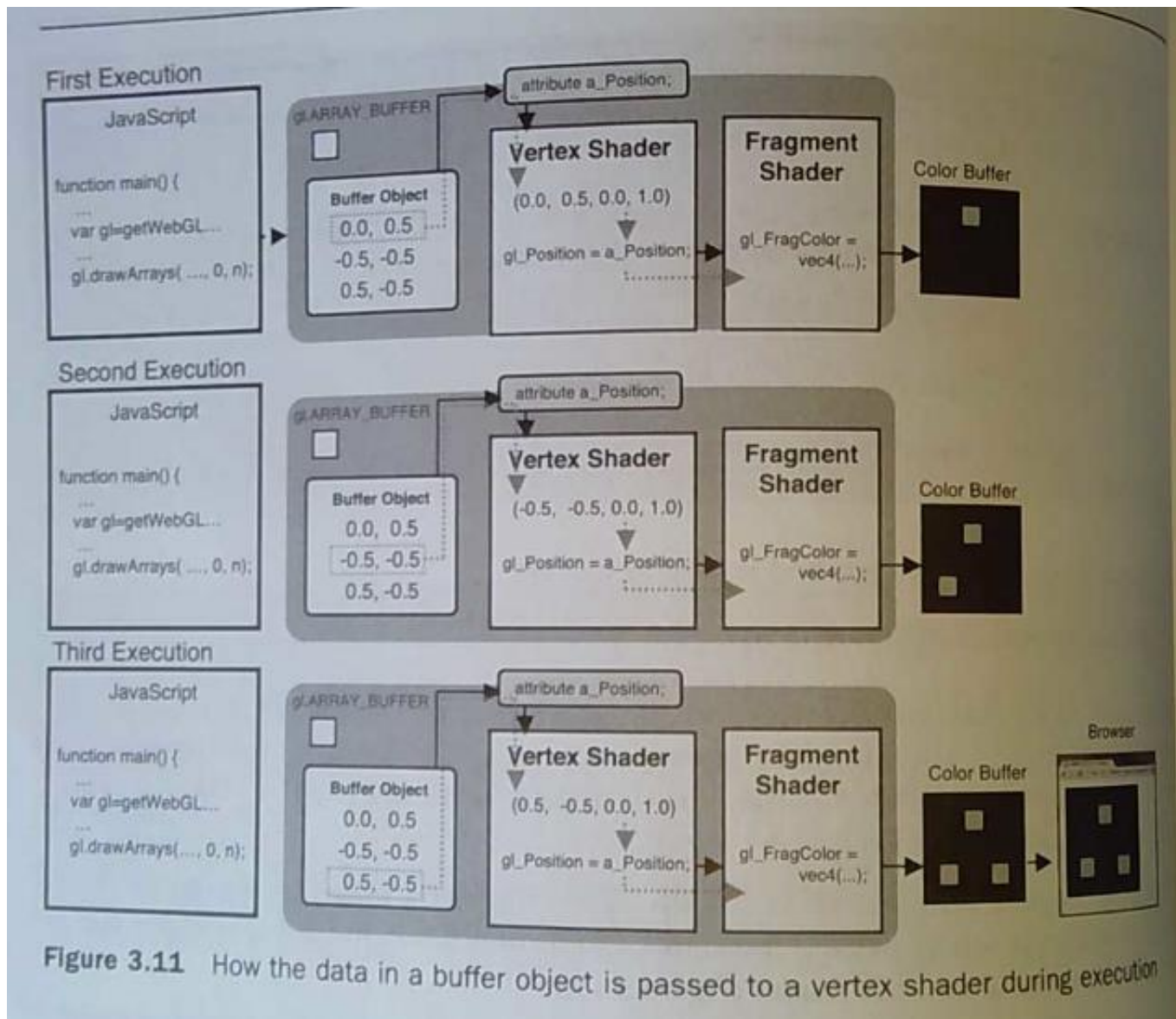


### GL\_TRIANGLE\_FAN

- First vertex is center of fan
- Subsequent vertices form ordered boundary
- One vertex emitted per triangle for dense fans
- But few such fans arise in practice



# Vertex shader runs for *each* item in the buffer



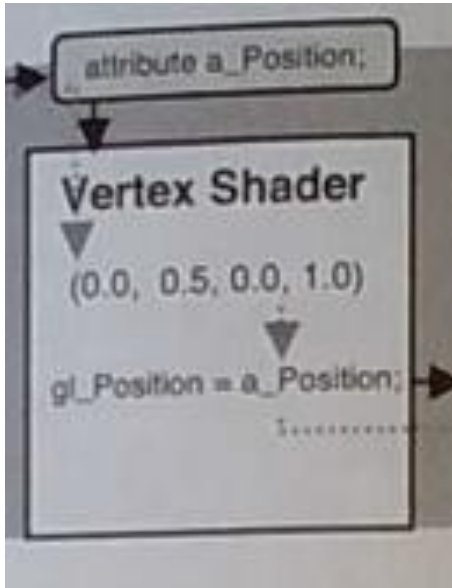
[V0.x, V0.y, V1.x, V1.y, V2.x, V2.y, V3.x, V3.y, V4.x, V4.y, V5.x, V5.y, V6.x, V6.y, ...]

Triangle 0

Triangle 1

*Attribute Vec2 a\_Position;*

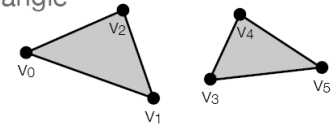
`drawArrays(gl.TRIANGLES, 0, n/3);`



## Triangles in OpenGL

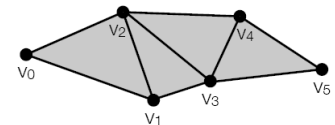
### GL\_TRIANGLES

- Successive vertex triples specify individual triangles
- Requires three vertices to be emitted for every triangle



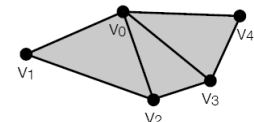
### GL\_TRIANGLE\_STRIP

- First triple specifies first triangle
- Subsequent vertices *each* specify new triangle, along with previous two vertices
- One vertex emitted per triangle in long strips
- But stripifying meshes is nontrivial



### GL\_TRIANGLE\_FAN

- First vertex is center of fan
- Subsequent vertices form ordered boundary
- One vertex emitted per triangle for dense fans
- But few such fans arise in practice



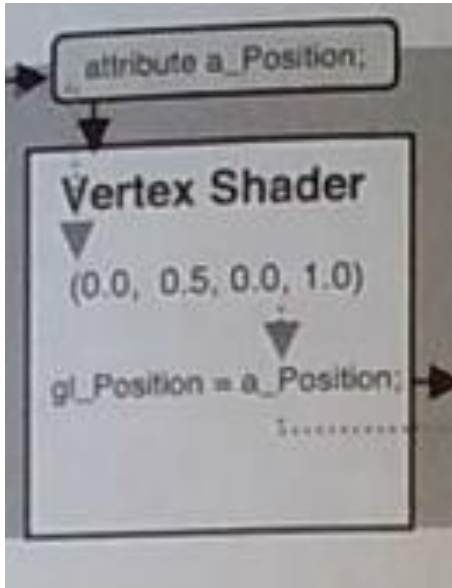
[V0.x, V0.y, V1.x, V1.y, V2.x, V2.y, V3.x, V3.y, V4.x, V4.y, V5.x, V5.y, V6.x, V6.y, ...]

Triangle 0

Triangle 1

*Attribute Vec2 a\_Position;*

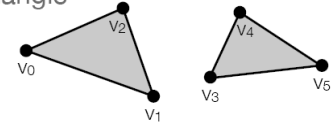
drawArrays(gl.TRIANGLES, 0, n/3);



## Triangles in OpenGL

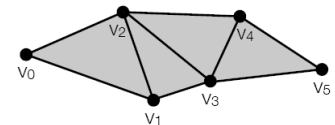
### GL\_TRIANGLES

- Successive vertex triples specify individual triangles
- Requires three vertices to be emitted for every triangle



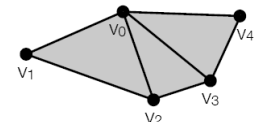
### GL\_TRIANGLE\_STRIP

- First triple specifies first triangle
- Subsequent vertices *each* specify new triangle, along with previous two vertices
- One vertex emitted per triangle in long strips
- But stripifying meshes is nontrivial



### GL\_TRIANGLE\_FAN

- First vertex is center of fan
- Subsequent vertices form ordered boundary
- One vertex emitted per triangle for dense fans
- But few such fans arise in practice



[V0.x, V0.y, V1.x, V1.y, V2.x, V2.y, V3.x, V3.y, V4.x, V4.y, V5.x, V5.y, V6.x, V6.y, ...]

Triangle 0

Triangle 1

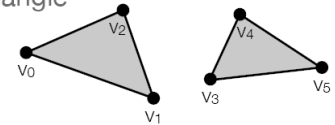
*Attribute Vec2 a\_Position;*

`drawArrays(gl.TRIANGLES, 0, n/3);`

## Triangles in OpenGL

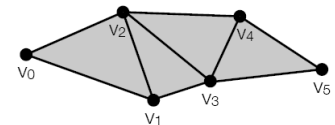
### GL\_TRIANGLES

- Successive vertex triples specify individual triangles
- Requires three vertices to be emitted for every triangle



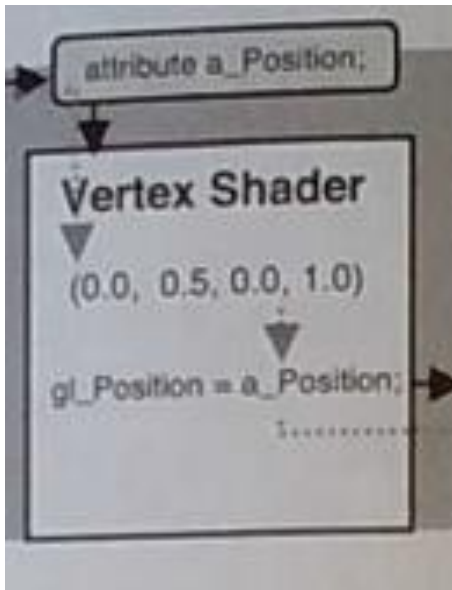
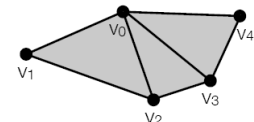
### GL\_TRIANGLE\_STRIP

- First triple specifies first triangle
- Subsequent vertices *each* specify new triangle, along with previous two vertices
- One vertex emitted per triangle in long strips
- But stripifying meshes is nontrivial



### GL\_TRIANGLE\_FAN

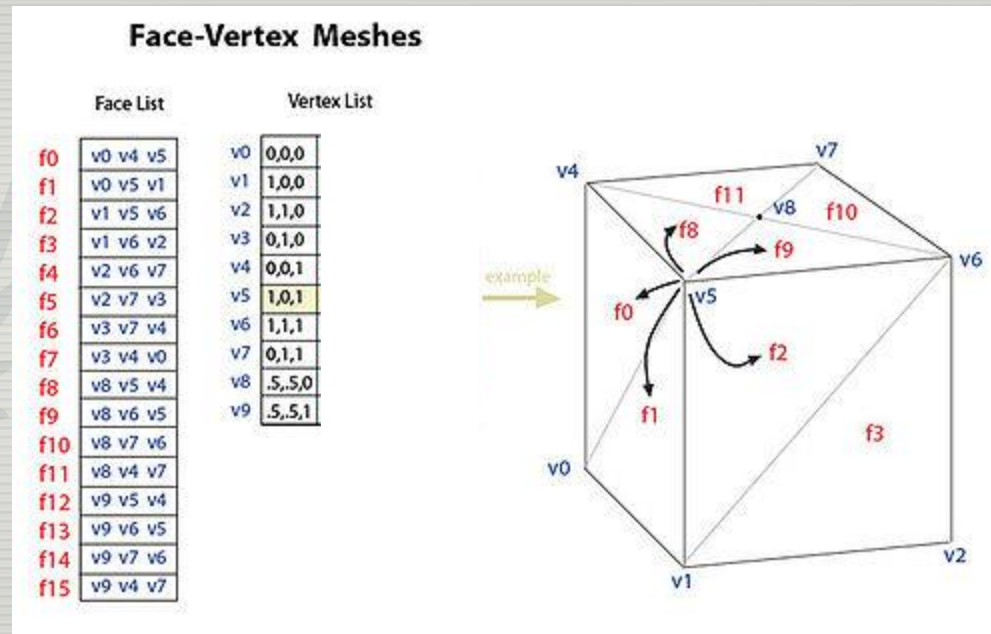
- First vertex is center of fan
- Subsequent vertices form ordered boundary
- One vertex emitted per triangle for dense fans
- But few such fans arise in practice





# Face-Vertex Meshes (FV)

- List of faces defined by vertex indices



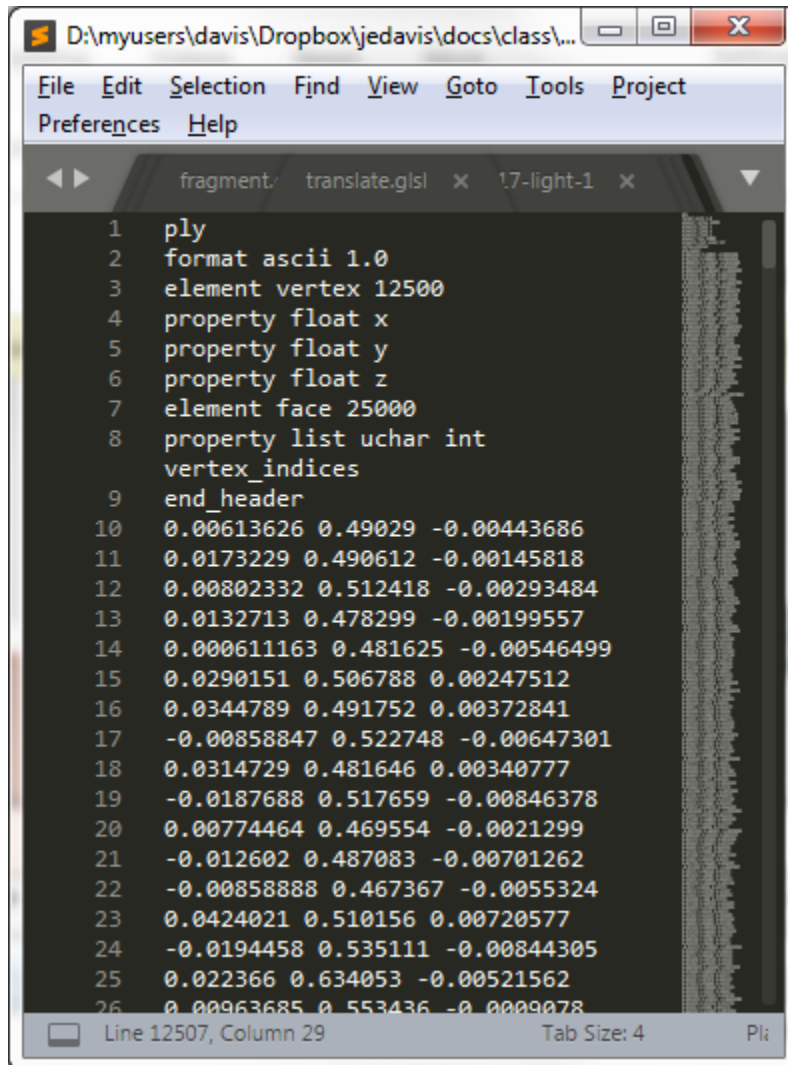
(image courtesy of Wikipedia)

```
drawArrays(gl.TRIANGLES, 0, n/3);  
drawElements(...);
```

# 3D Scene/Model File Formats

- Wavefront OBJ (.obj)
- 3DS Max (.3ds)
- Geomview OFF (Object File Format) (.off)
- PLY (.ply) for scanned data
- ... and more

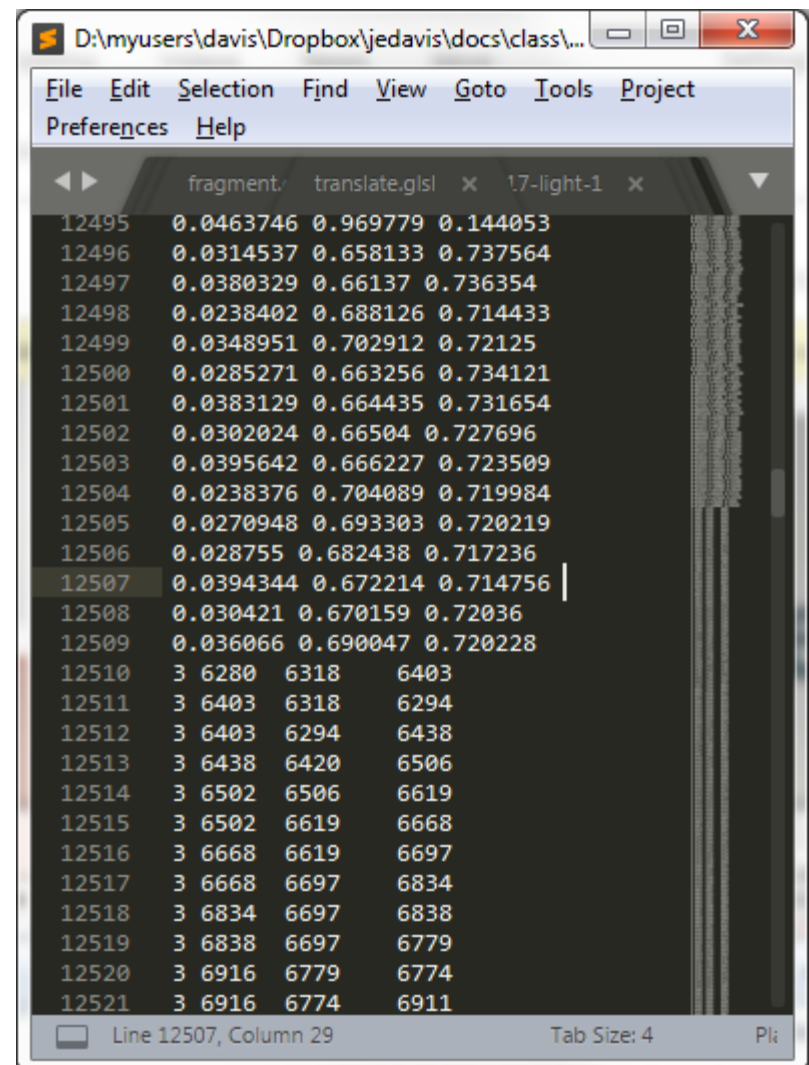
# Example data in a 3D file (.ply)



A screenshot of a text editor window showing the header section of a .ply file. The window title is "D:\myusers\davis\Dropbox\jedavis\docs\class\...". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The toolbar shows navigation and tab management icons. The text content is as follows:

```
1 ply
2 format ascii 1.0
3 element vertex 12500
4 property float x
5 property float y
6 property float z
7 element face 25000
8 property list uchar int
  vertex_indices
9 end_header
10 0.00613626 0.49029 -0.00443686
11 0.0173229 0.490612 -0.00145818
12 0.00802332 0.512418 -0.00293484
13 0.0132713 0.478299 -0.00199557
14 0.000611163 0.481625 -0.00546499
15 0.0290151 0.506788 0.00247512
16 0.0344789 0.491752 0.00372841
17 -0.00858847 0.522748 -0.00647301
18 0.0314729 0.481646 0.00340777
19 -0.0187688 0.517659 -0.00846378
20 0.00774464 0.469554 -0.0021299
21 -0.012602 0.487083 -0.00701262
22 -0.00858888 0.467367 -0.0055324
23 0.0424021 0.510156 0.00720577
24 -0.0194458 0.535111 -0.00844305
25 0.022366 0.634053 -0.00521562
26 0.00963685 0.553436 -0.0009078
```

The status bar at the bottom indicates "Line 12507, Column 29", "Tab Size: 4", and "Pli".



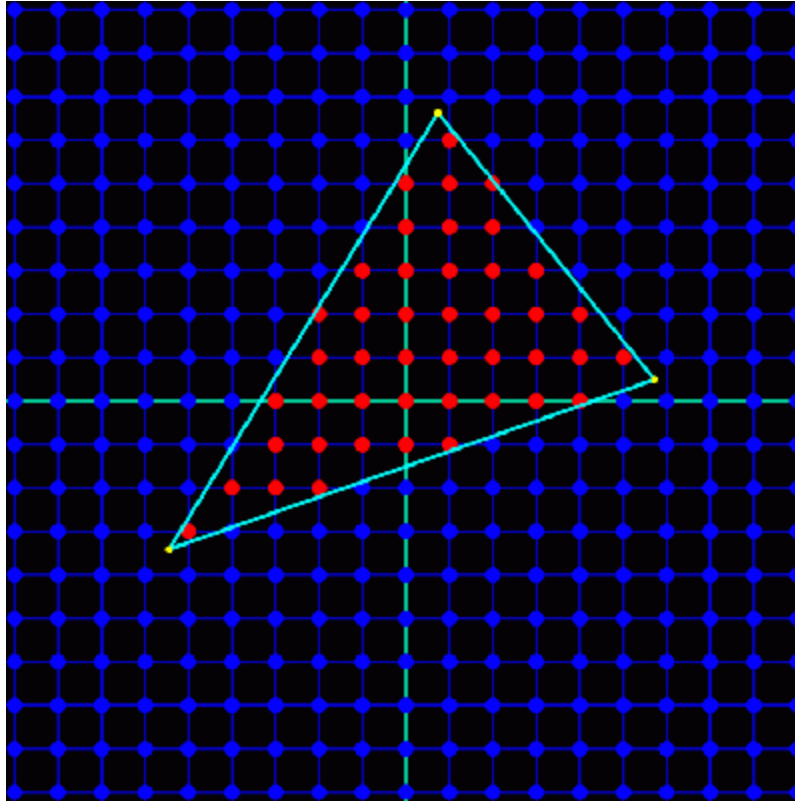
A screenshot of a text editor window showing the data section of a .ply file. The window title is "D:\myusers\davis\Dropbox\jedavis\docs\class\...". The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The toolbar shows navigation and tab management icons. The text content is as follows:

```
12495 0.0463746 0.969779 0.144053
12496 0.0314537 0.658133 0.737564
12497 0.0380329 0.66137 0.736354
12498 0.0238402 0.688126 0.714433
12499 0.0348951 0.702912 0.72125
12500 0.0285271 0.663256 0.734121
12501 0.0383129 0.664435 0.731654
12502 0.0302024 0.66504 0.727696
12503 0.0395642 0.666227 0.723509
12504 0.0238376 0.704089 0.719984
12505 0.0270948 0.693303 0.720219
12506 0.028755 0.682438 0.717236
12507 0.0394344 0.672214 0.714756
12508 0.030421 0.670159 0.72036
12509 0.036066 0.690047 0.720228
12510 3 6280 6318 6403
12511 3 6403 6318 6294
12512 3 6403 6294 6438
12513 3 6438 6420 6506
12514 3 6502 6506 6619
12515 3 6502 6619 6668
12516 3 6668 6619 6697
12517 3 6668 6697 6834
12518 3 6834 6697 6838
12519 3 6838 6697 6779
12520 3 6916 6779 6774
12521 3 6916 6774 6911
```

The status bar at the bottom indicates "Line 12507, Column 29", "Tab Size: 4", and "Pli".

# Rasterization

Rasterization = Turn on all pixels inside the triangle



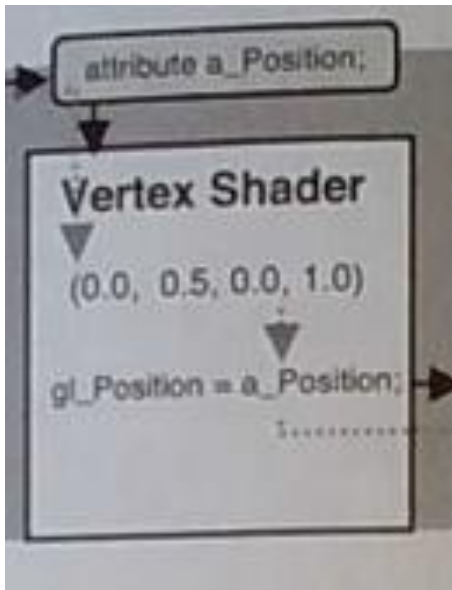
[V0.x, V0.y, V1.x, V1.y, V2.x, V2.y, V3.x, V3.y, V4.x, V4.y, V5.x, V5.y, V6.x, V6.y, ...]

Triangle 0

Triangle 1

*Attribute Vec2 a\_Position;*

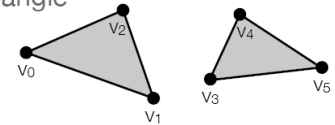
`drawArrays(gl.TRIANGLES, 0, n/3);`



## Triangles in OpenGL

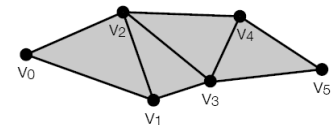
### GL\_TRIANGLES

- Successive vertex triples specify individual triangles
- Requires three vertices to be emitted for every triangle



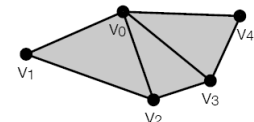
### GL\_TRIANGLE\_STRIP

- First triple specifies first triangle
- Subsequent vertices *each* specify new triangle, along with previous two vertices
- One vertex emitted per triangle in long strips
- But stripifying meshes is nontrivial



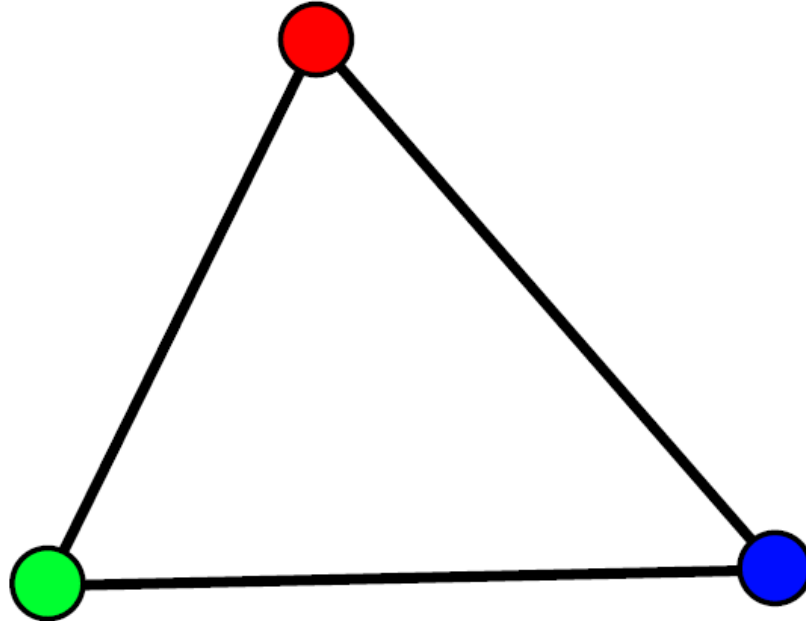
### GL\_TRIANGLE\_FAN

- First vertex is center of fan
- Subsequent vertices form ordered boundary
- One vertex emitted per triangle for dense fans
- But few such fans arise in practice



## ***How is shading done in OpenGL?***

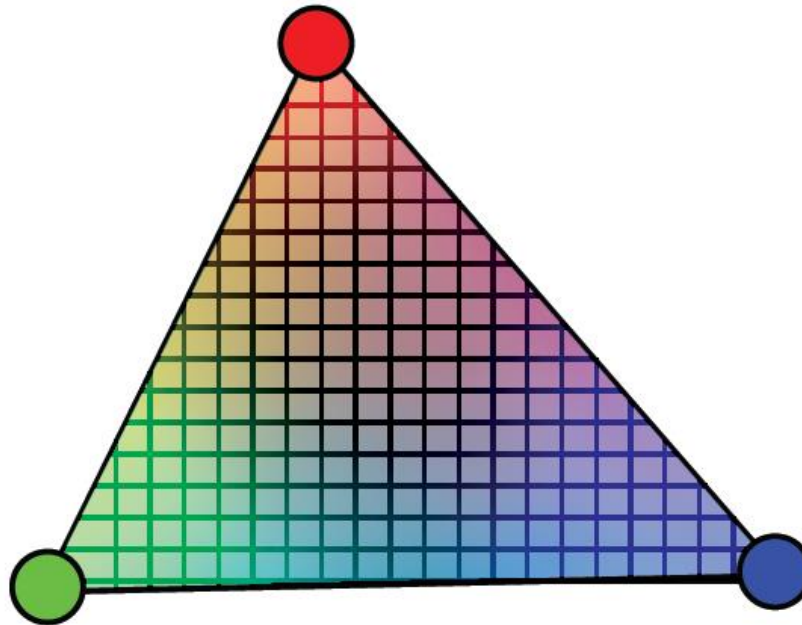
1. Attributes are specified on vertices.



## ***How is shading done in OpenGL?***

2. Attributes are interpolated across triangles by the rasterizer

(see appendix for details)

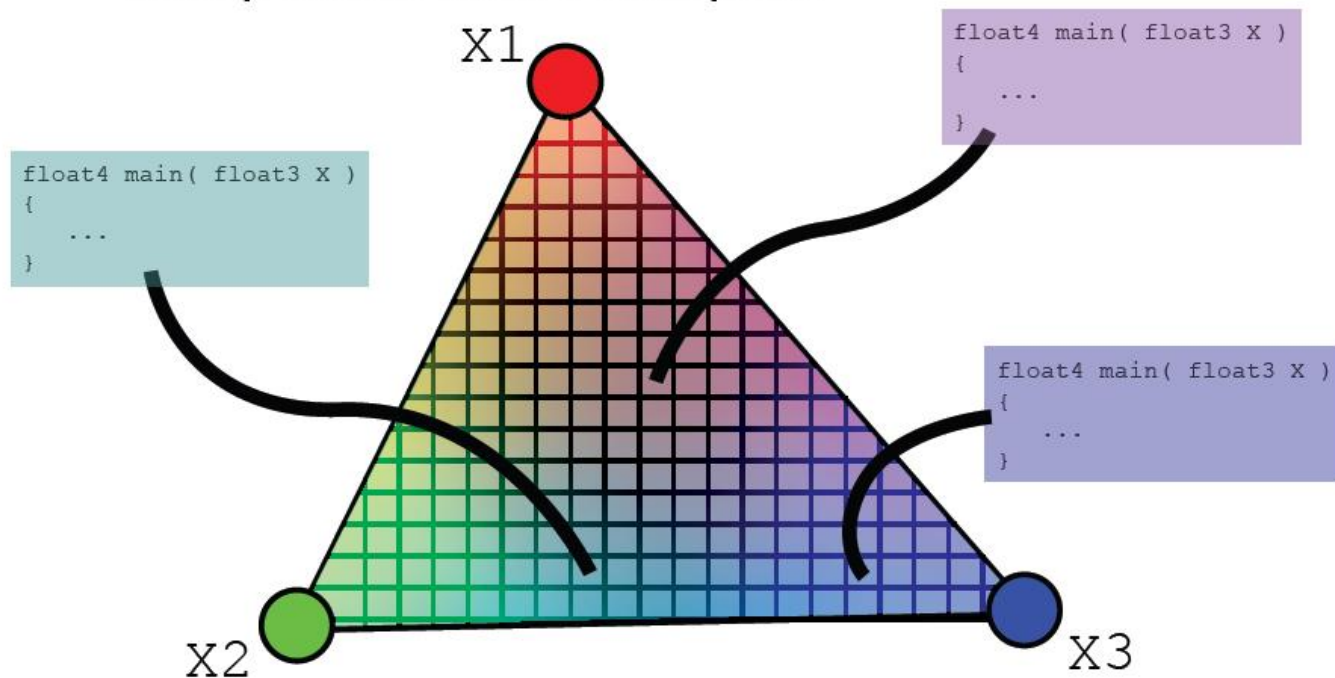


Rasterizer also breaks the triangle into “fragments.”



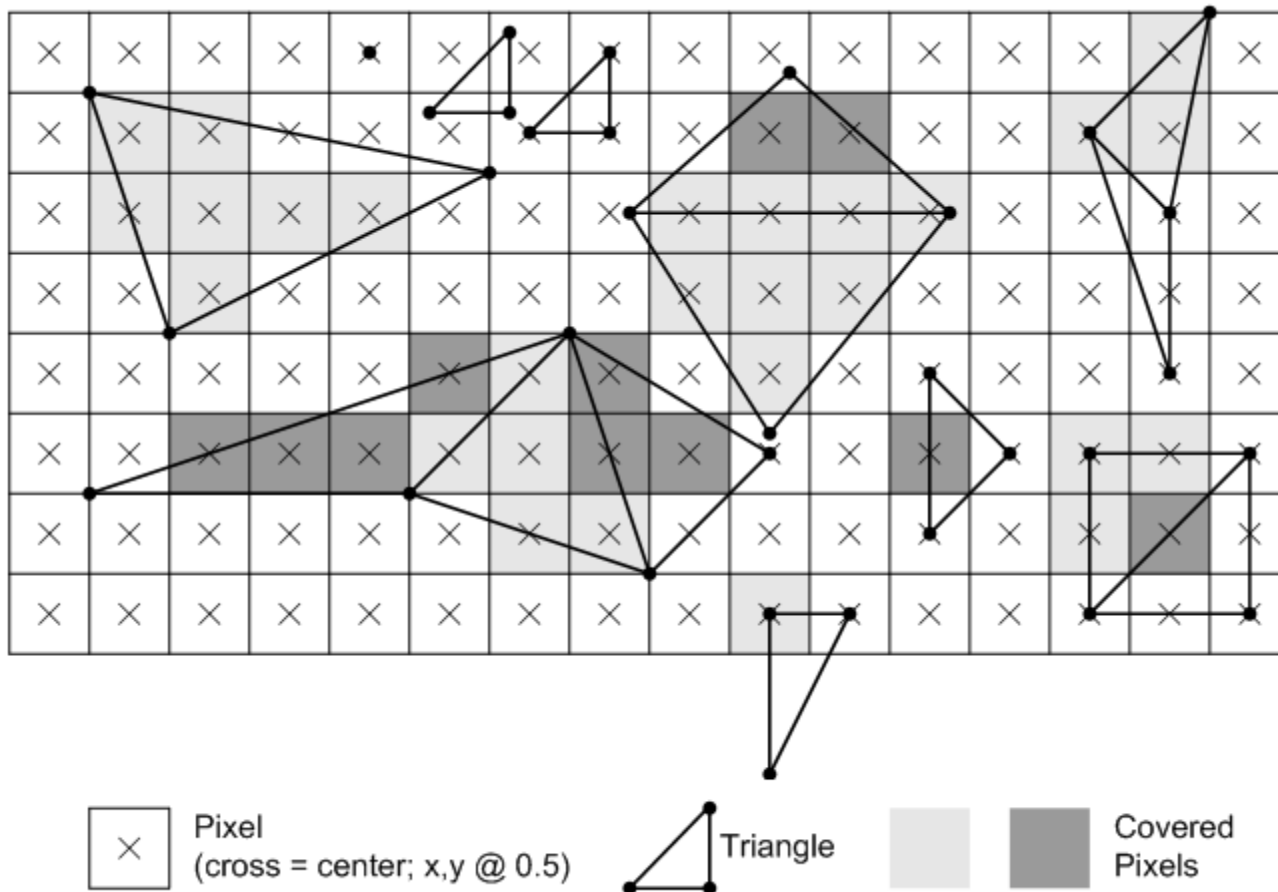
## ***How is shading done in OpenGL?***

3. Each fragment runs the shader using interpolated values as inputs.



*Exact same routine, different inputs.*

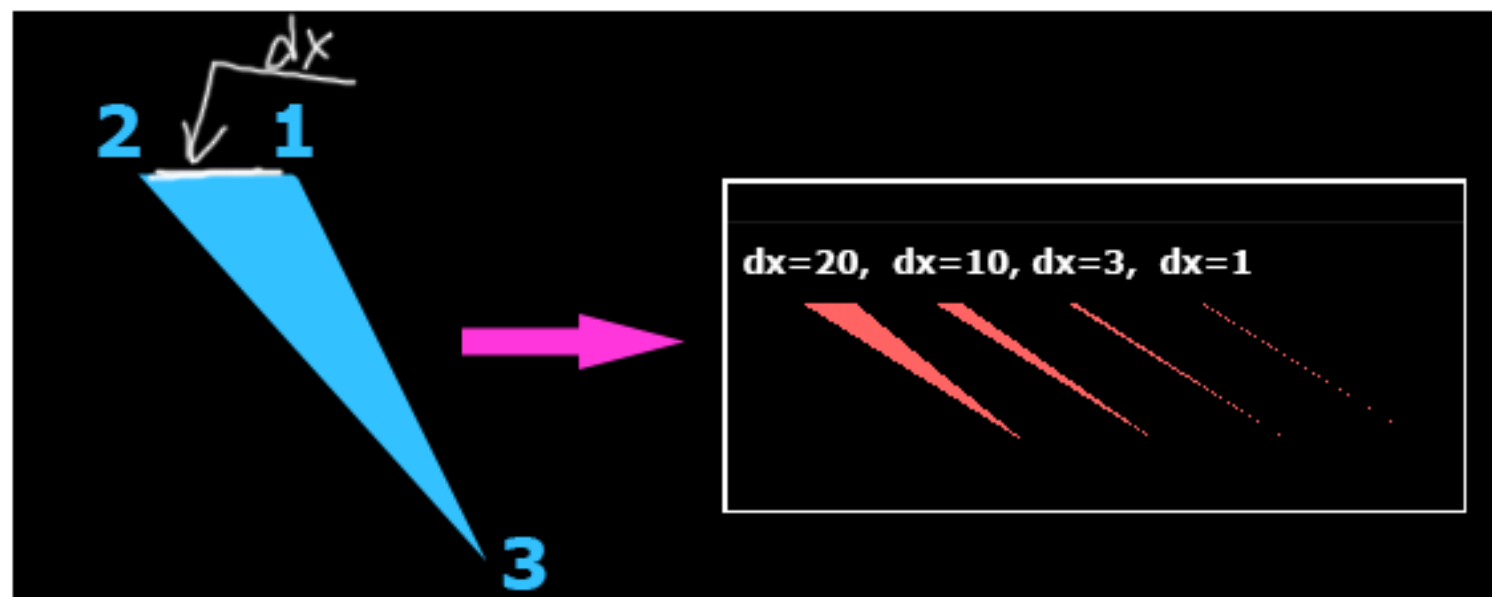
## Top-Left Rasterization Rule



2

I'm using TriangleList to output my primitives. Most all of the time I need to draw rectangles, triangles, circles. From time to time I need to draw very thin triangles (width=2px for example). I thought it should look like a line (almost a line) but it looks like separate points :)

Following picture shows what I'm talking about:



First picture at the left side shows how do I draw a rectangle (counter clockwise, from top right corner). And then you can see the "width" of the rectangle which I call "dx".

How to avoid this behavior? I would it looks like a straight (almost straight) line, not as points :)

opengl

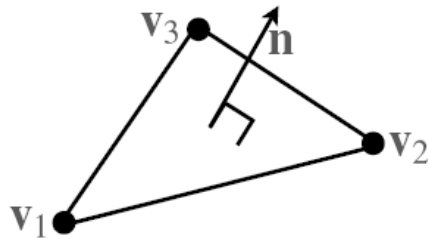
gl-triangle-strip

**Normals**

# Triangle Normals

## Per-Triangle

- Triangle defines unique plane:
- Can easily compute **unit** normal vector from vertices:

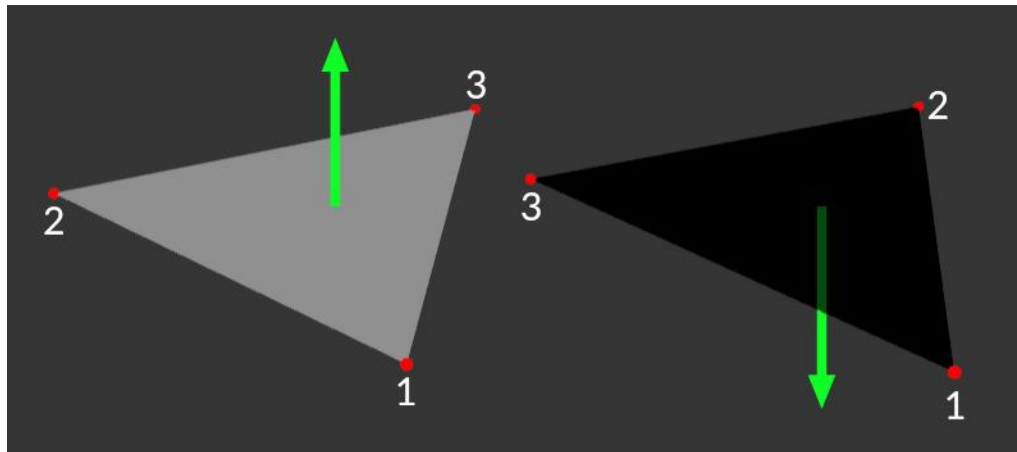


$$\mathbf{a} = \mathbf{v}_2 - \mathbf{v}_1$$

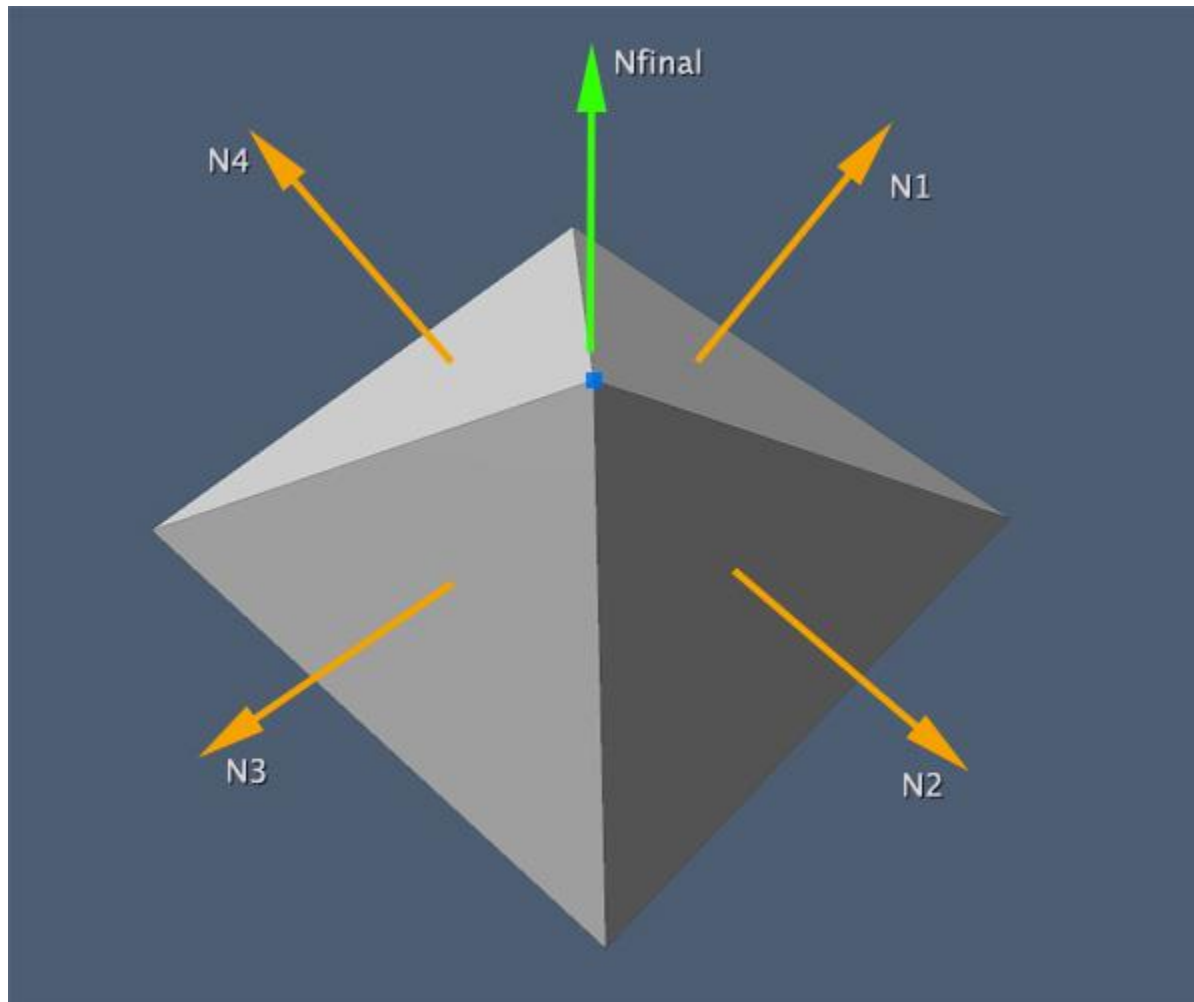
$$\mathbf{b} = \mathbf{v}_3 - \mathbf{v}_1$$

$$\mathbf{n} = \frac{\mathbf{a} \times \mathbf{b}}{\|\mathbf{a} \times \mathbf{b}\|}$$

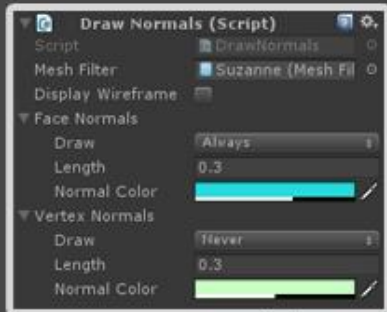
- Orientation depends on vertex order (clockwise yields  $-\mathbf{n}$ )



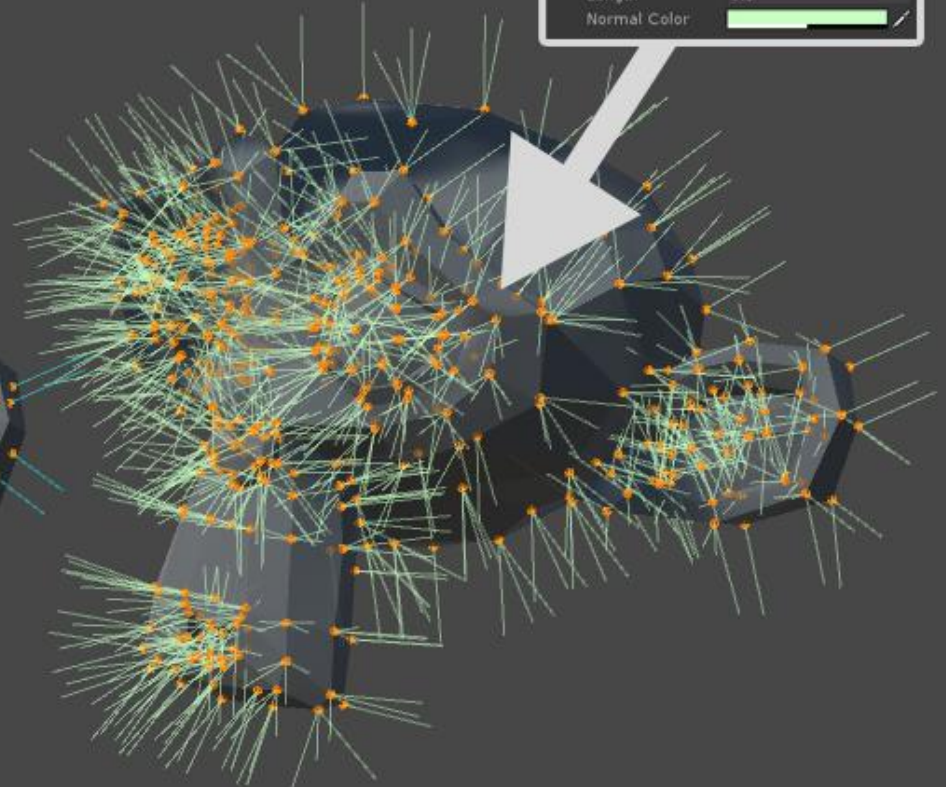
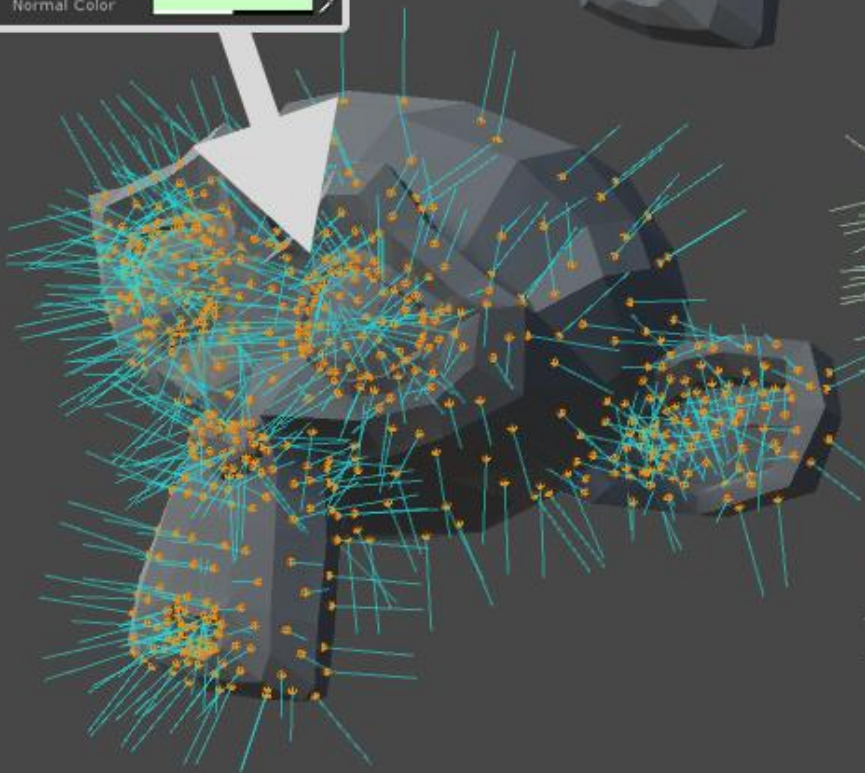
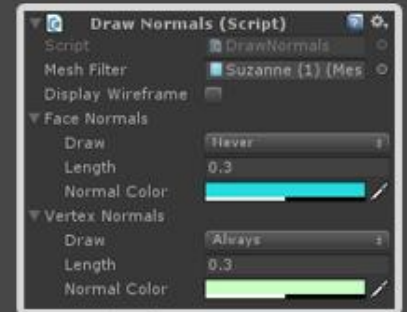
# How can a vertex have a normal?

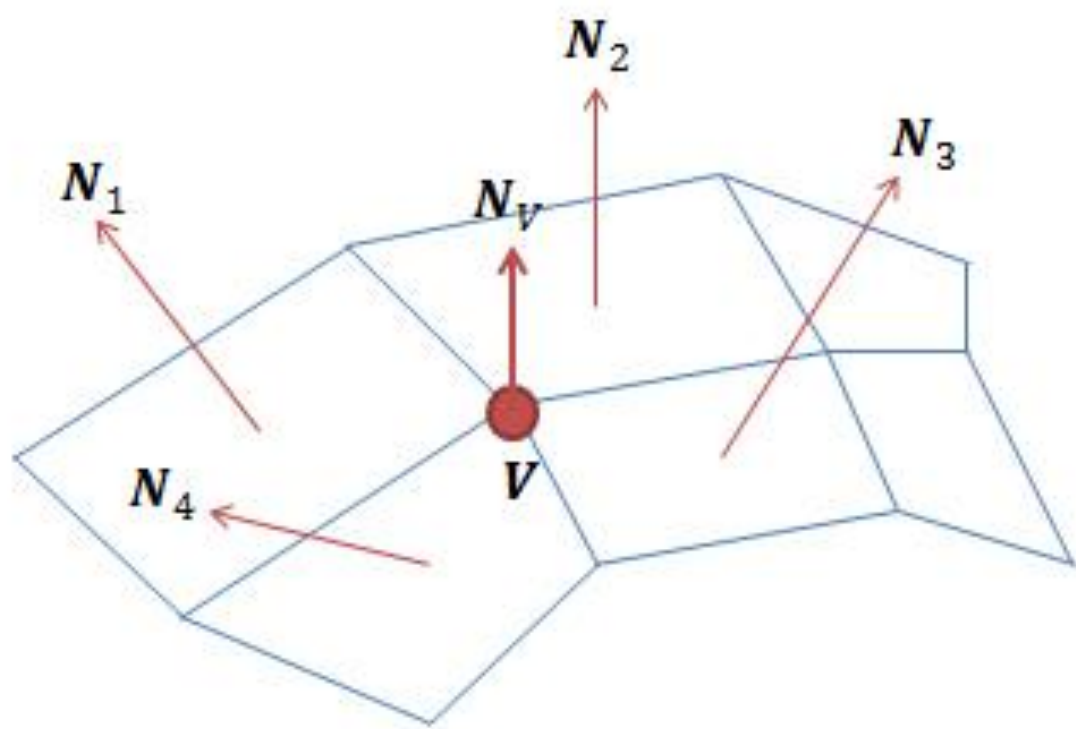


## (FACE NORMALS)



## (VERT NORMALS)

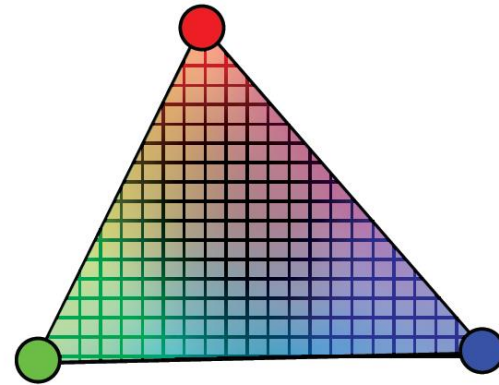




$$N_V = \frac{\sum_{k=1}^n N_k}{\left| \sum_{k=1}^n N_k \right|}$$

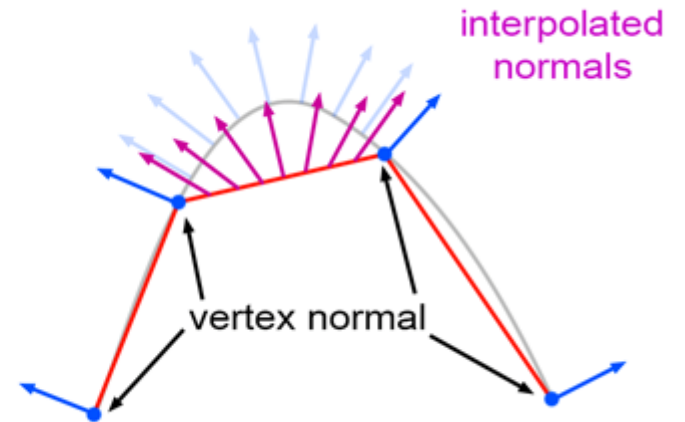
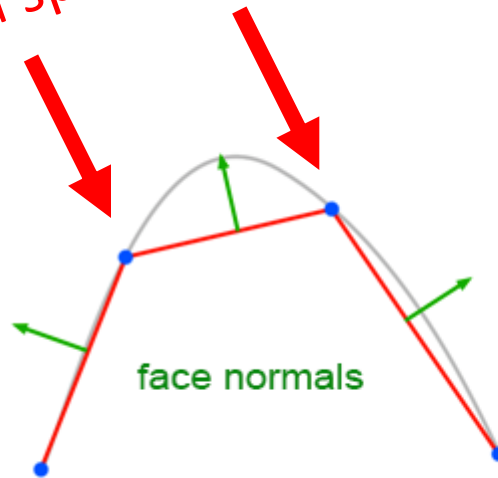


Interpolating  
color

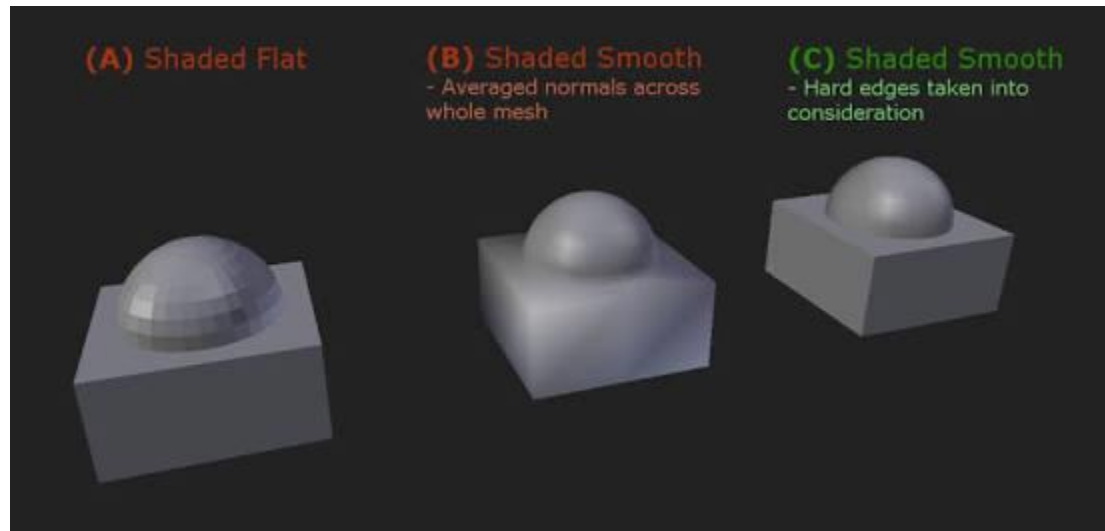
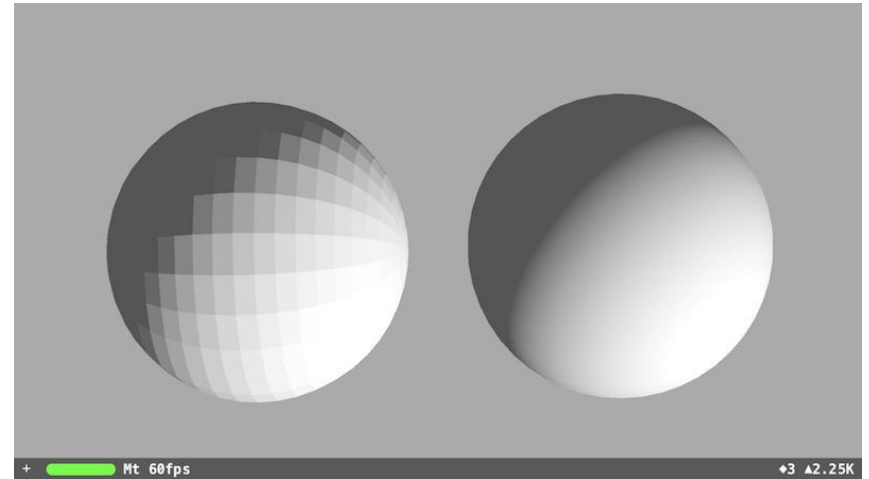
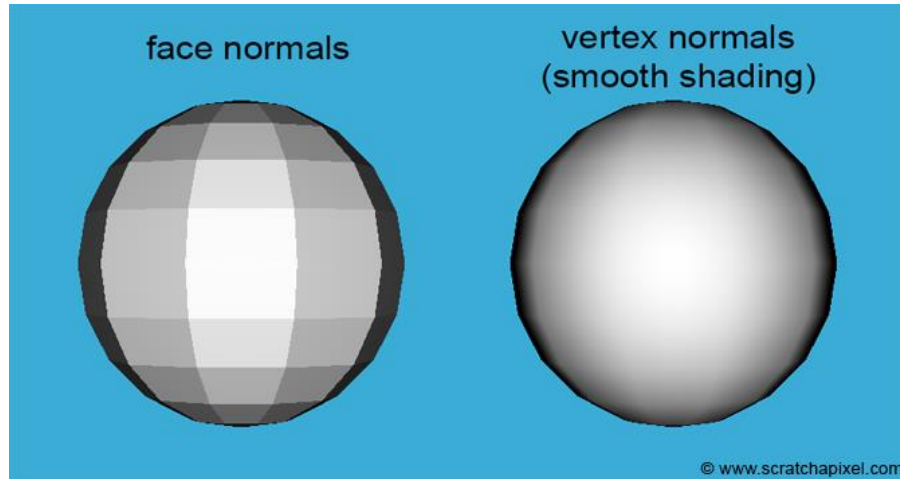


You specify these

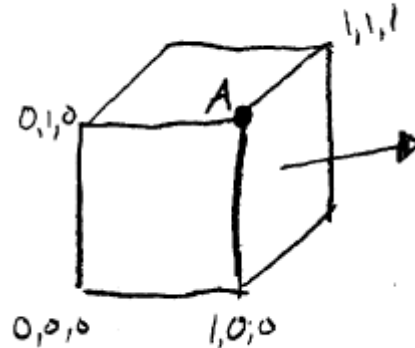
Interpolating  
normals



# Per face vs per vertex normals



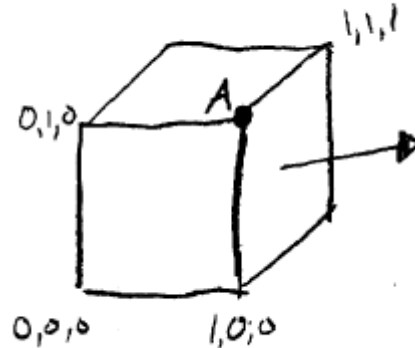
# Q about Normals



What is the per-polygon normal shown?

- (A) 1,1,1
- (B) 0,0,1
- (C) 1,0,0
- (D) 0,1,0
- (E) Don't know

# Q about Normals



What is the per-vertex normal at point A?

- (A) 1,1,1
- (B) 1,1,-1
- (C)  $1/\sqrt{3}$ ,  $1/\sqrt{3}$ ,  $1/\sqrt{3}$
- (D)  $-1/\sqrt{3}$ ,  $1/\sqrt{3}$ ,  $1/\sqrt{3}$
- (E) Don't know

# http://tiny.cc/160108

Google Calendar x Launch Meeting x Launch Meeting x CSE160 - Spring x Assignment x Syllabus for x Attendance x Participation x Call for Summer x Apr2 x CSE- 160-01 x Channel videos x Tiny URL | Free x + -

docs.google.com/forms/d/1uoCwxd5f9a83TTu-XDdDoSydUMn5rzafhWDeuTv9ZJM/edit

Automated Mesh G... Bookmarks My Profile - Zoom Channel videos - Y... 01:09:14 - slides for... 95033talk@groups... LaunchPad Central... Inuitive All Discussions - Bel... LaunchPad Central... Intel PerC H/W Sup... Bellus3D Internal >> Other bookmarks

Participation Apr 7 All changes saved in Drive

Questions Responses

### Participation Apr 7

Form description

This form is automatically collecting email addresses for UC Santa Cruz users. [Change settings](#)

I was in class Apr 7

☐ Yes

☐ No

I have finished Lab [A0](#)

☐ Yes

☐ No, but I will pretty soon

☐ No, and I am really stuck

I have finished [HW1](#)

☐ Yes

☐ No, but I will pretty soon

☐ No, and I am really stuck

I have started [A1](#) (Paint Program)

☐ Yes

☐ No, but I will pretty soon

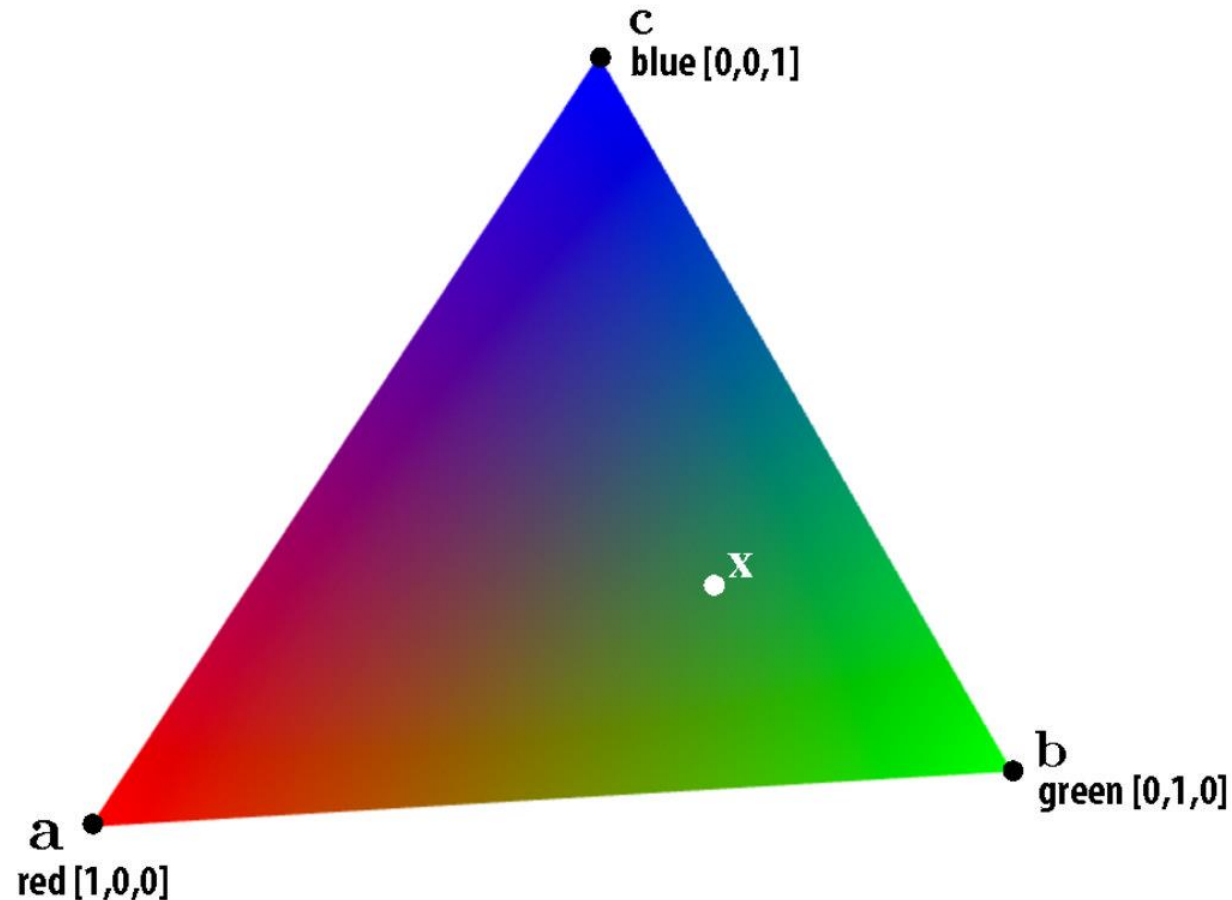
☐ No, I guess the weekend before its due is about right for me

☐ Other...

Send

# Interpolation

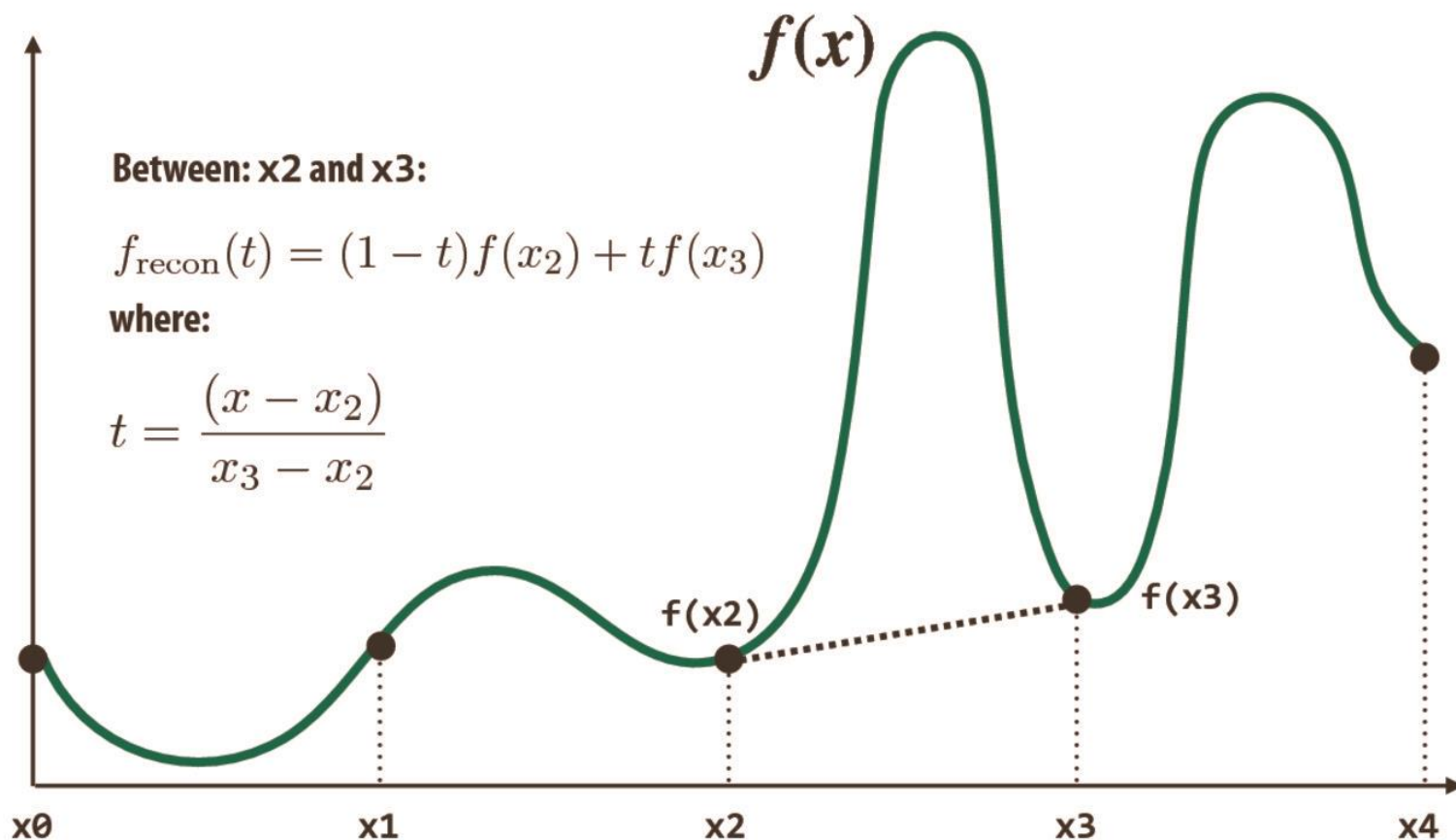
# Consider sampling color( $x, y$ )



What is the triangle's color at the point  $x$  ?

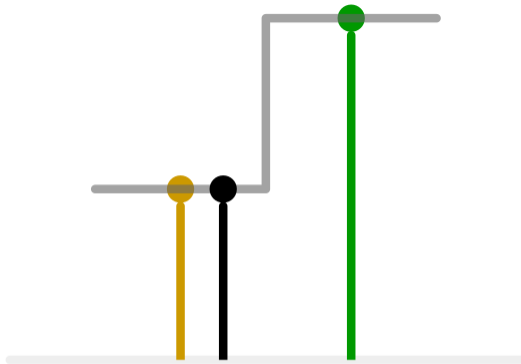
# Review: interpolation in 1D

$f_{\text{recon}}(x)$  = linear interpolation between values of two closest samples to  $x$

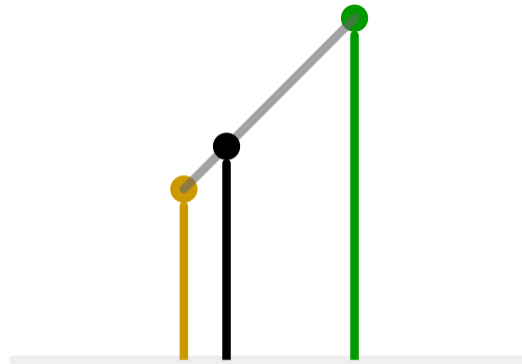




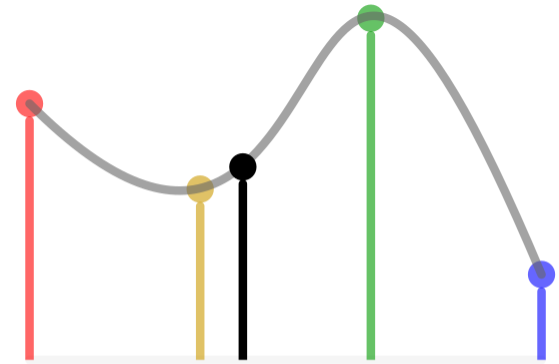
# Linear interpolation



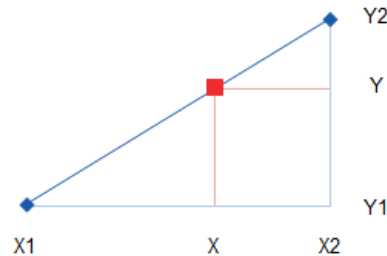
1D nearest-neighbour



Linear



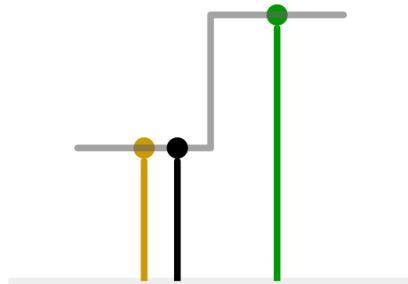
Cubic



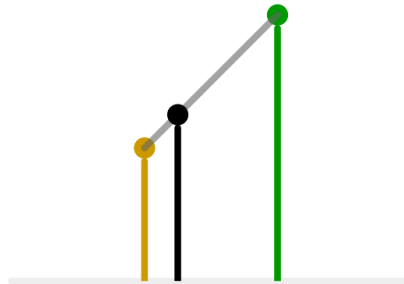
$$\frac{(X - X1)}{(X2 - X1)} = \frac{(Y - Y1)}{(Y2 - Y1)}$$

$$Y = Y1 + (X - X1) \frac{(Y2 - Y1)}{(X2 - X1)}$$

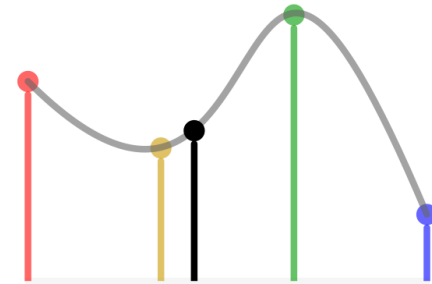
# Bi-linear interpolation



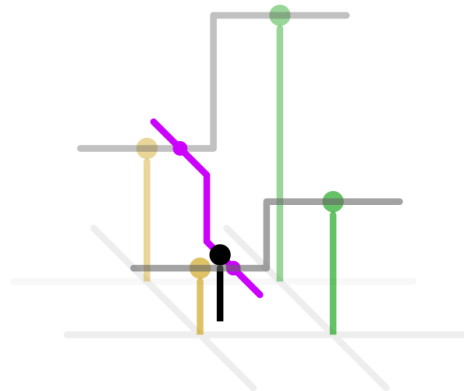
1D nearest-neighbour



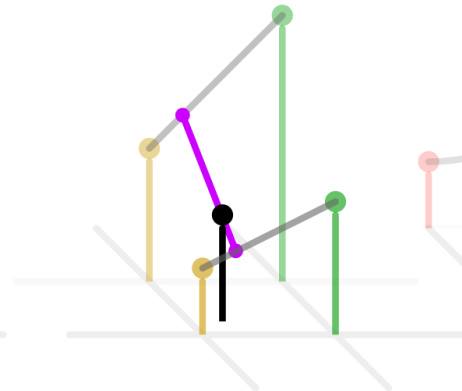
Linear



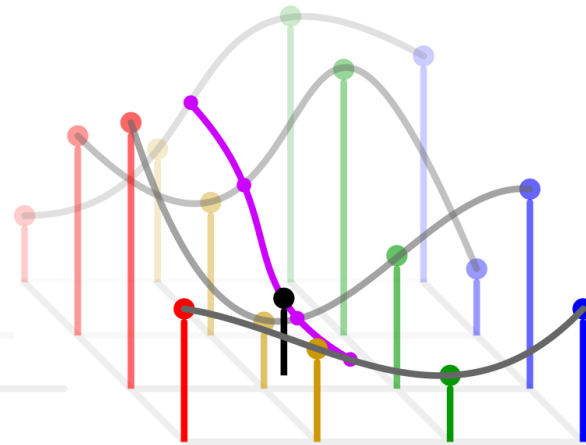
Cubic



2D nearest-neighbour

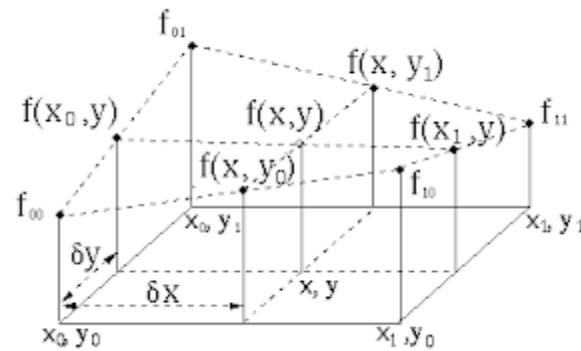
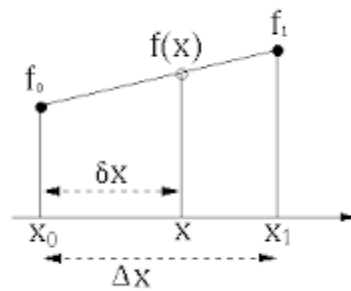


Bilinear



Bicubic

# Bi-linear interpolation



Interpolation is not just about surfaces, used heavily in images

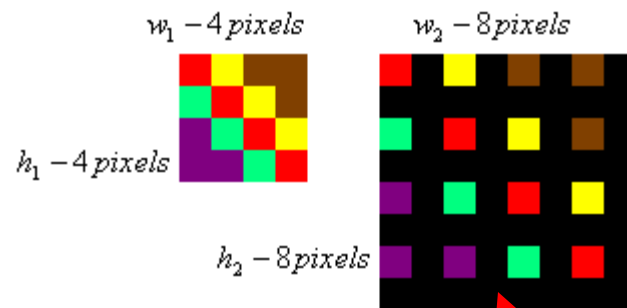
---

Image sizes



© 2006 blog.forret.com

Suppose you start with the smallest image and need a big one?



What value goes in  
between

Simple algorithm (just resize)



original

simple-2

simple-4



simple-8

simple-16

simple-32

Nearest, bilinear, b-spline interpolation

orthm (Triangle filter)



a\_original

triangle-2

triangle-4



triangle-8

triangle-16

triangle-32

a\_original

bspline-2

bspline-4

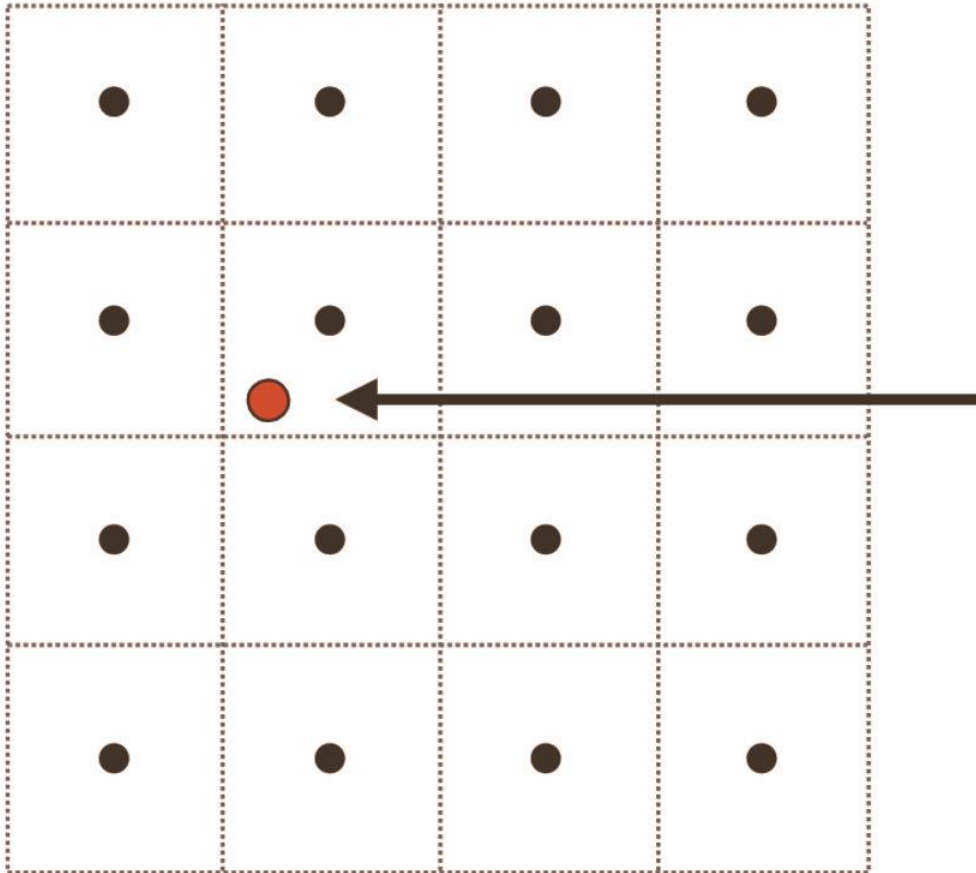


bspline-8

bspline-16

bspline-32

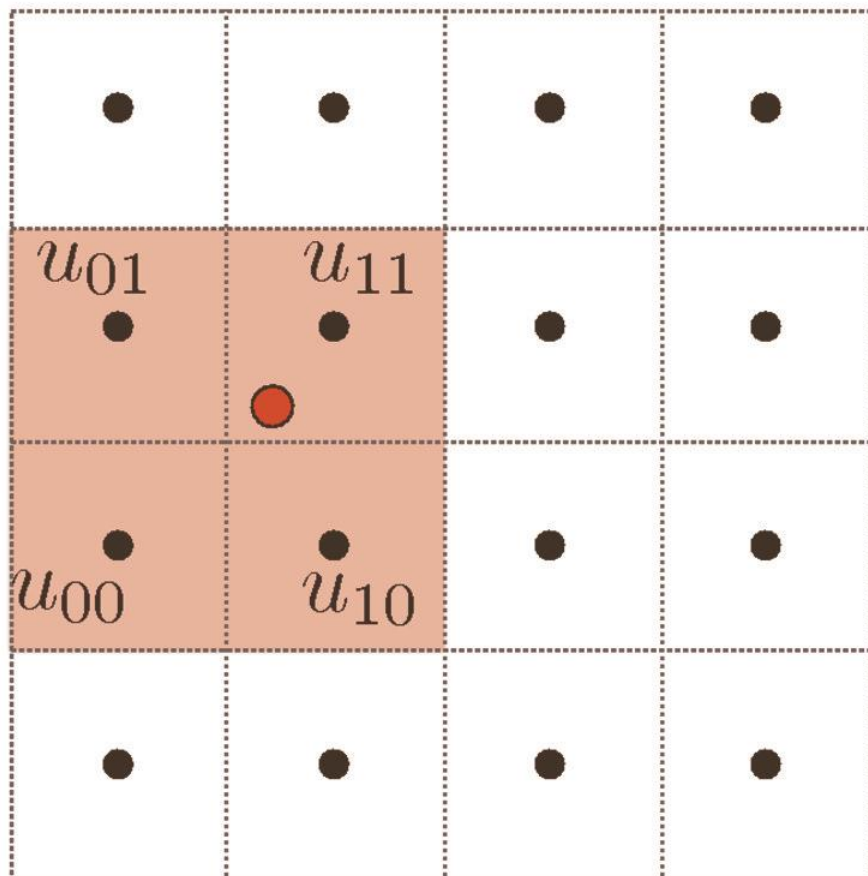
# Bilinear Filtering



Want to sample  
texture value  $f(x,y)$  at  
red point

Black points indicate  
texture sample  
locations

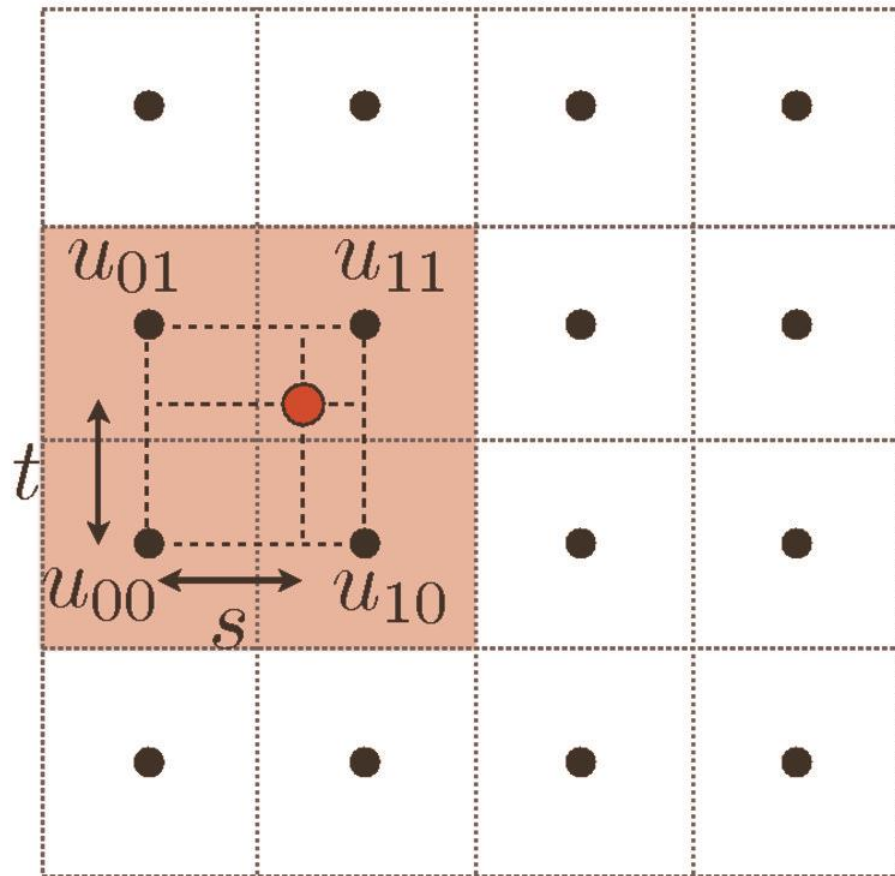
# Bilinear filtering



Take 4 nearest sample locations, with texture values as labeled.

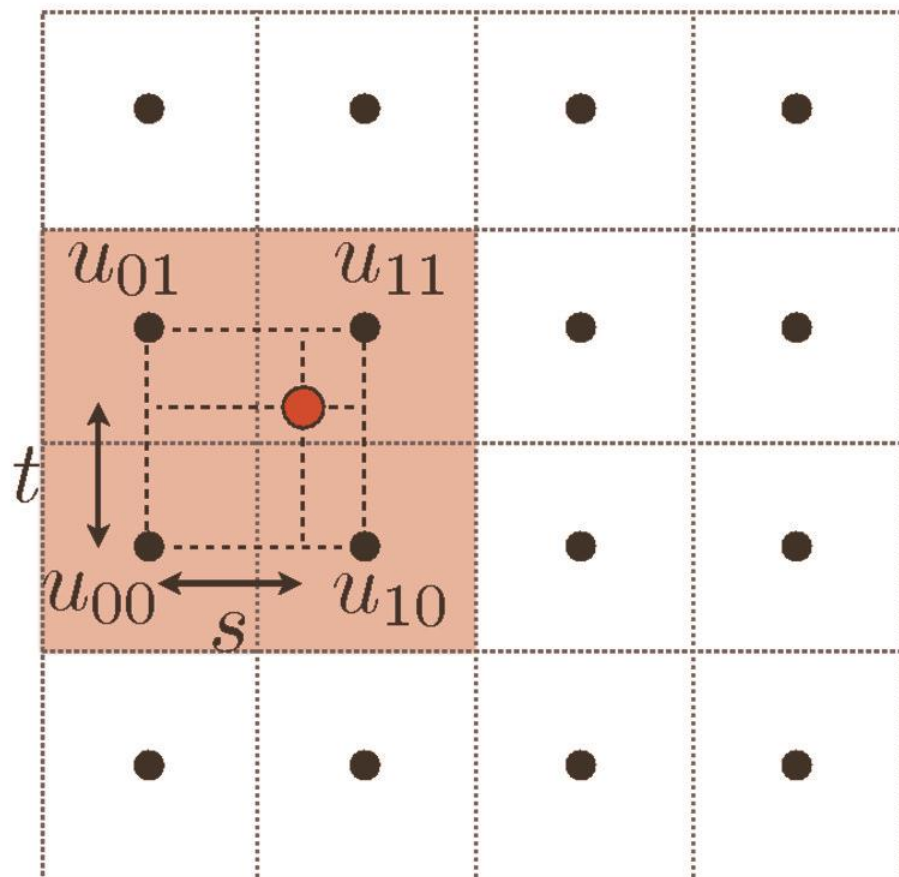


# Bilinear filtering



And fractional offsets,  $(s,t)$  as shown

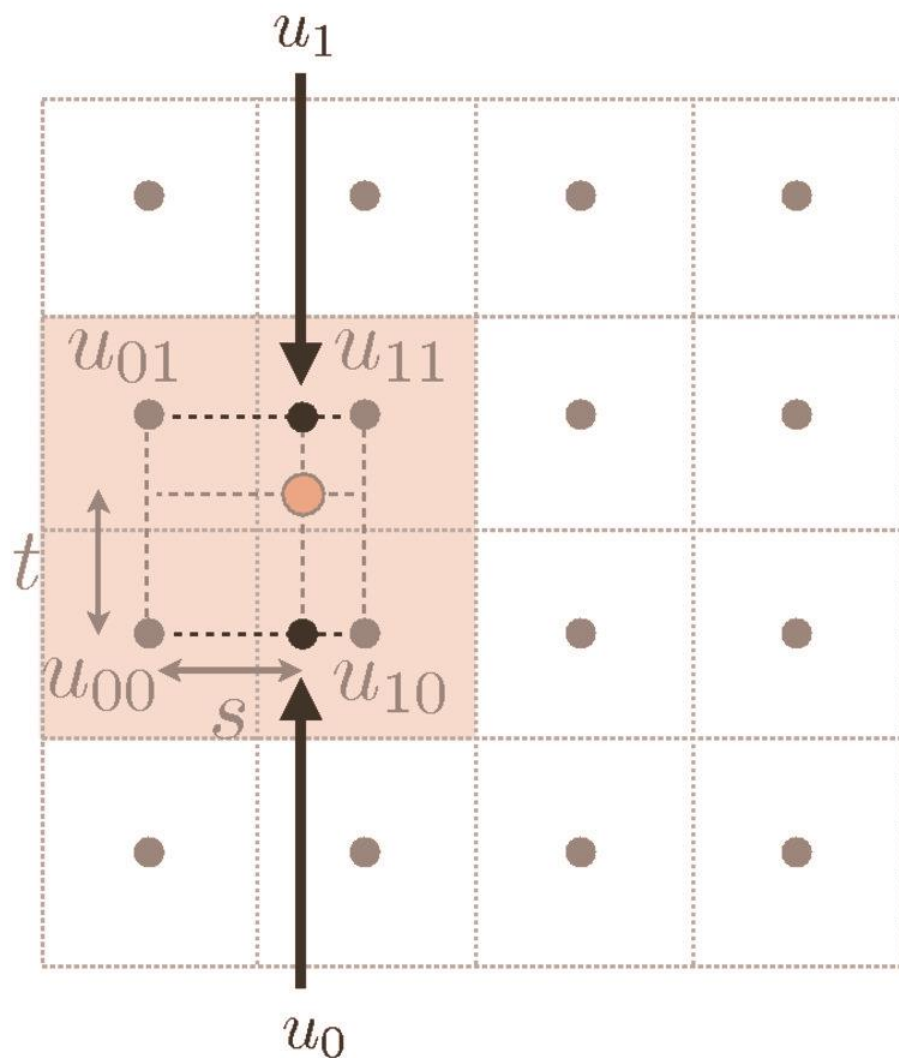
# Bilinear filtering



**Linear interpolation (1D)**

$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

# Bilinear filtering



## Linear interpolation (1D)

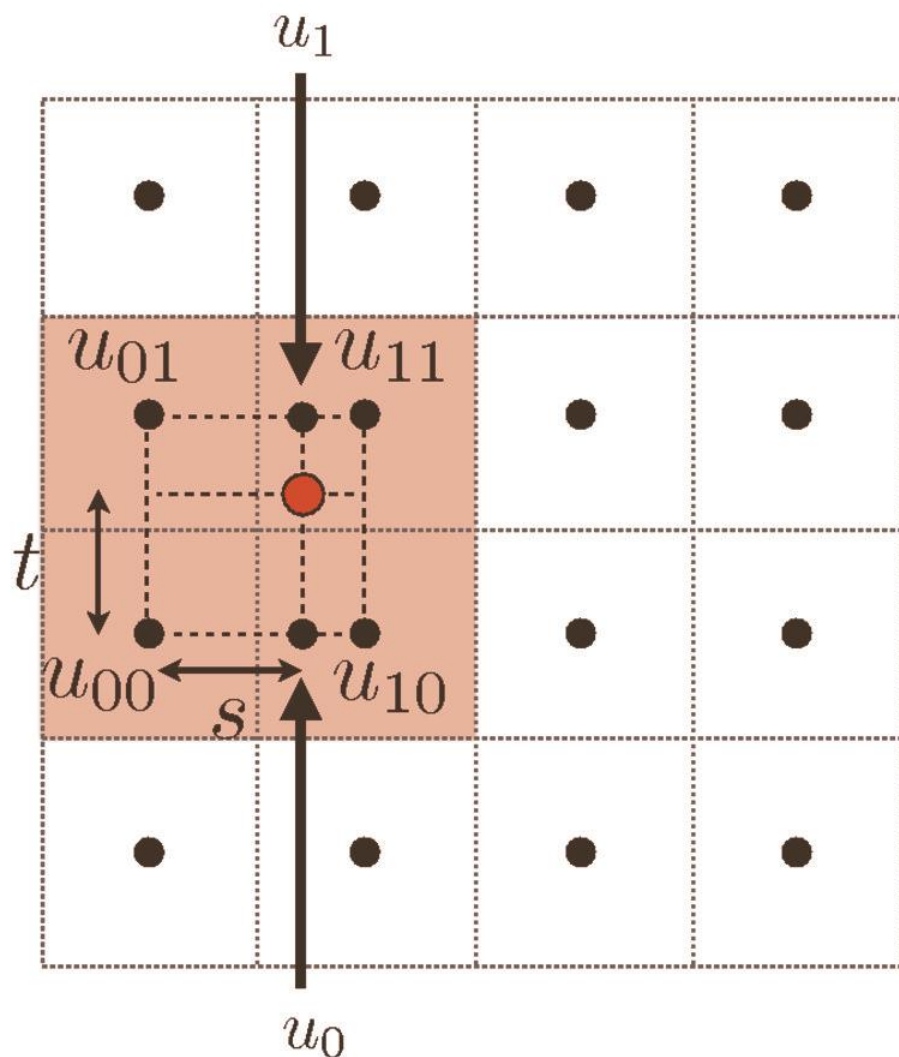
$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

## Two helper lerps (horizontal)

$$u_0 = \text{lerp}(s, u_{00}, u_{10})$$

$$u_1 = \text{lerp}(s, u_{01}, u_{11})$$

# Bilinear filtering



## Linear interpolation (1D)

$$\text{lerp}(x, v_0, v_1) = v_0 + x(v_1 - v_0)$$

## Two helper lerps

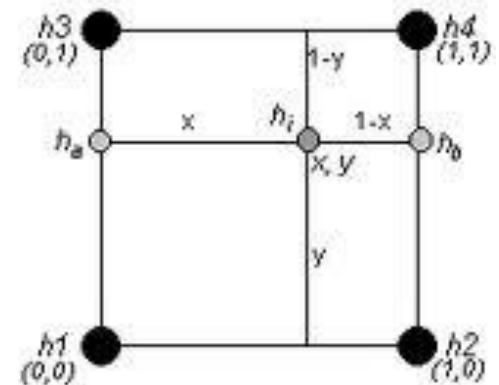
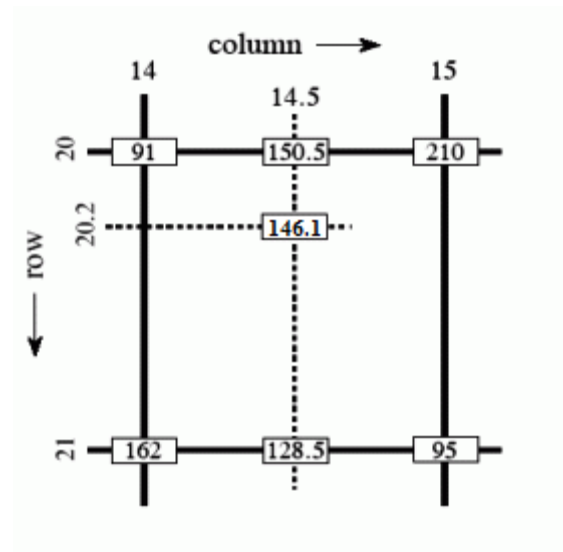
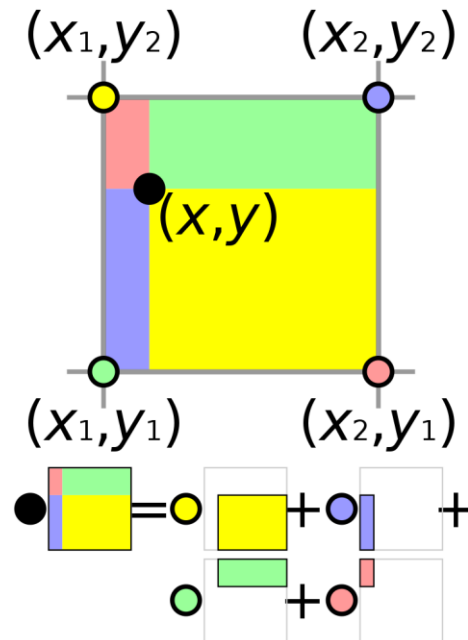
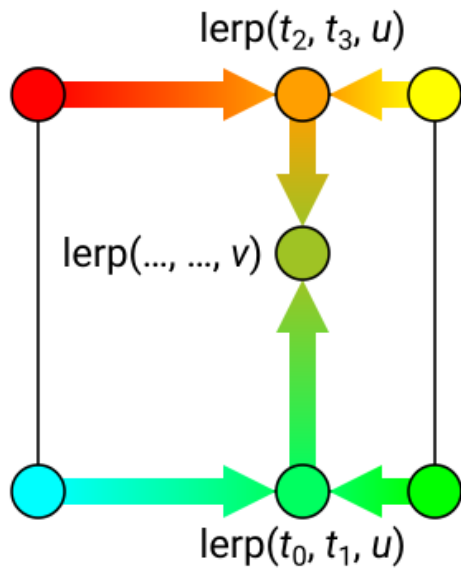
$$u_0 = \text{lerp}(s, u_{00}, u_{10})$$

$$u_1 = \text{lerp}(s, u_{01}, u_{11})$$

## Final vertical lerp, to get result:

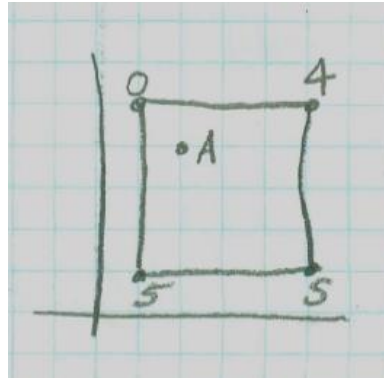
$$f(x, y) = \text{lerp}(t, u_0, u_1)$$

# Bilinear interpolation



# Q about bi-linear interpolation

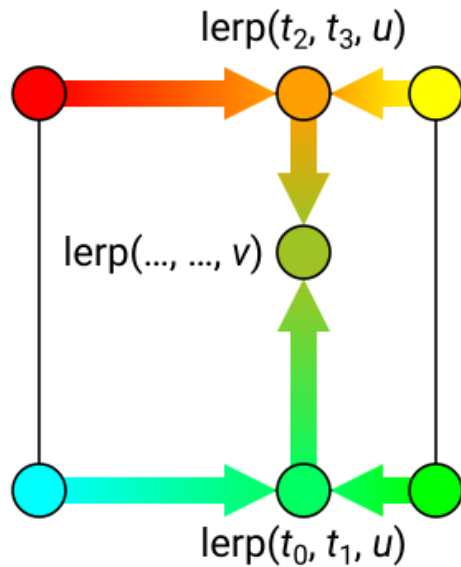
---



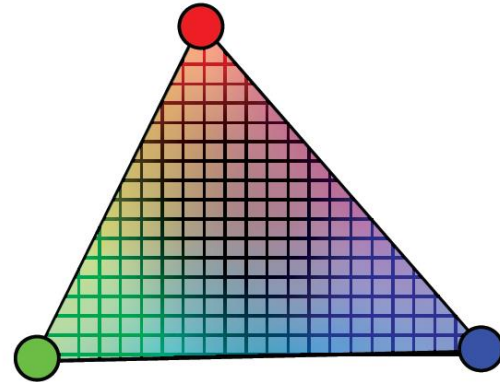
What is the value at point A?

- (A) 1
- (B) 2
- (C) 3
- (D) 4
- (E) Don't know

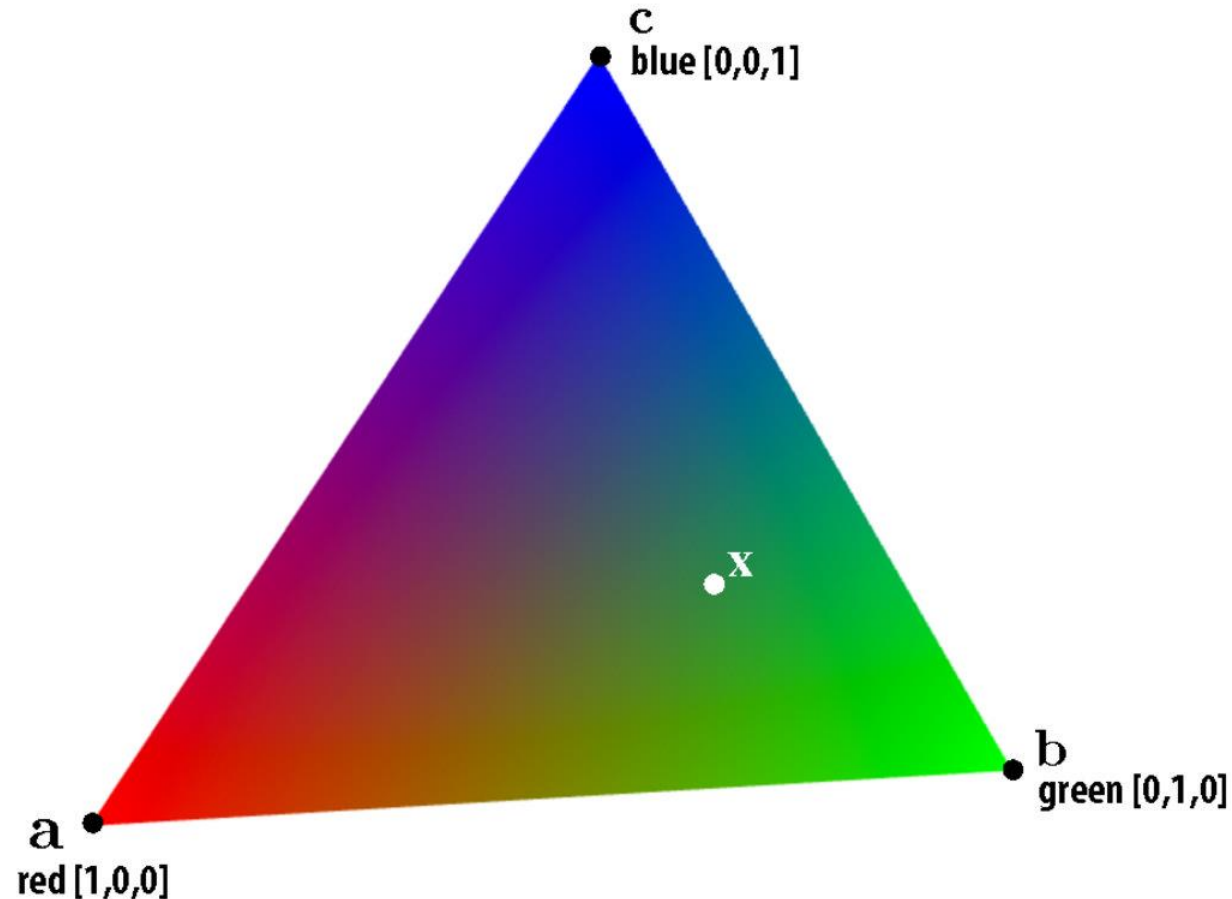
Umm.. So how do I use this on triangles?



Bilinear interpolation



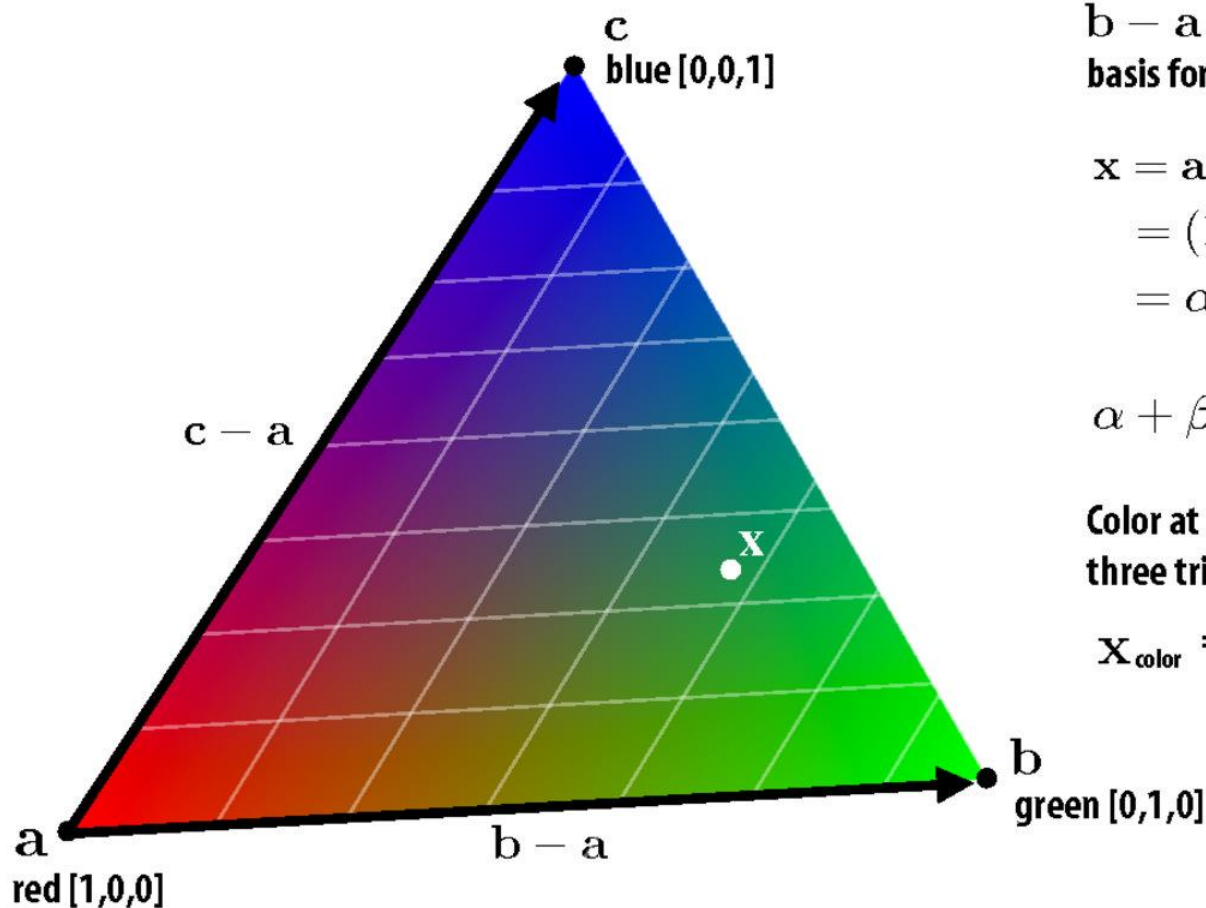
# Consider sampling color( $x, y$ )



What is the triangle's color at the point  $x$  ?



# Interpolation via barycentric coordinates



$b - a$  and  $c - a$  form a non-orthogonal basis for points in triangle (origin at  $a$ )

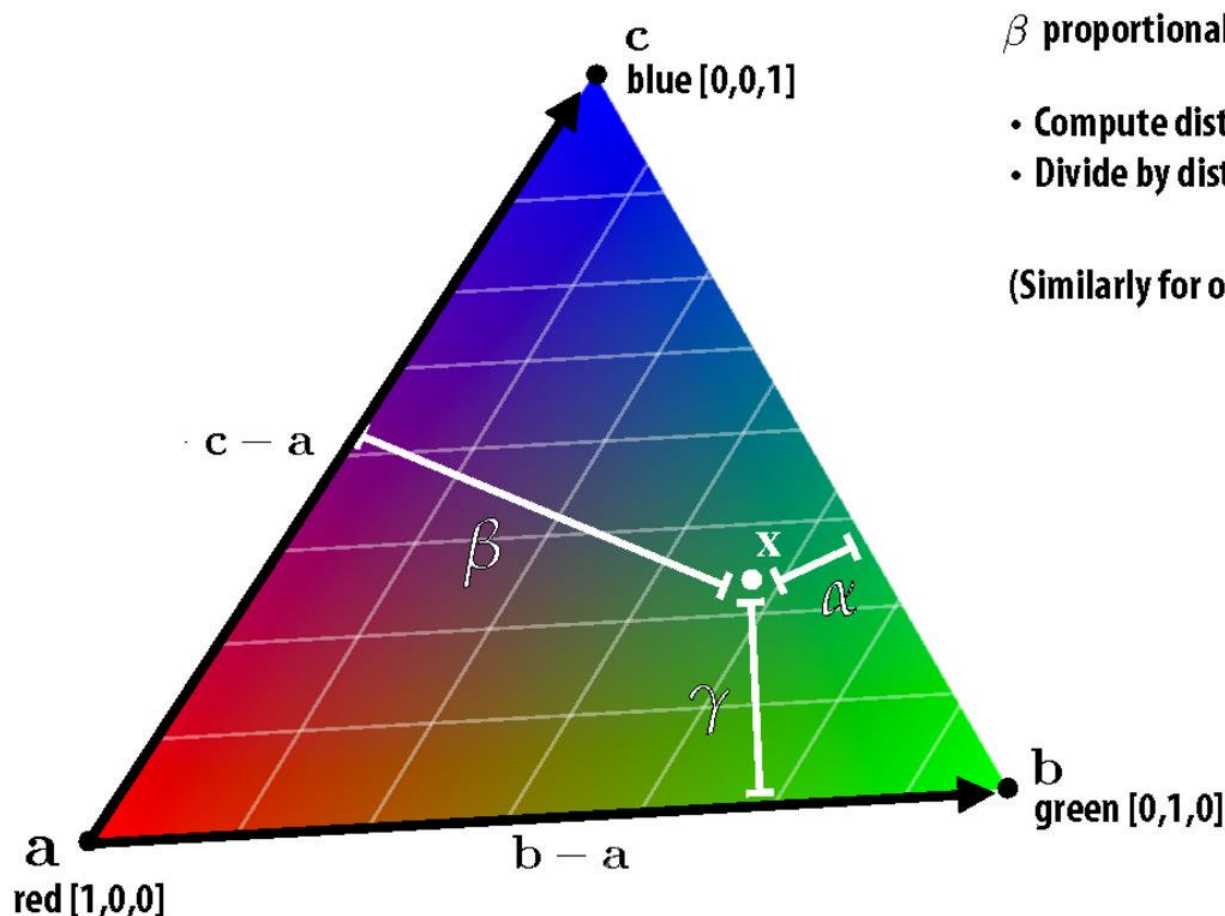
$$\begin{aligned} \mathbf{x} &= \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}) \\ &= (1 - \beta - \gamma)\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \\ &= \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \end{aligned}$$

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

Color at  $x$  is linear combination of color at three triangle vertices.

$$\mathbf{x}_{\text{color}} = \alpha\mathbf{a}_{\text{color}} + \beta\mathbf{b}_{\text{color}} + \gamma\mathbf{c}_{\text{color}}$$

# Barycentric coordinates as scaled distances

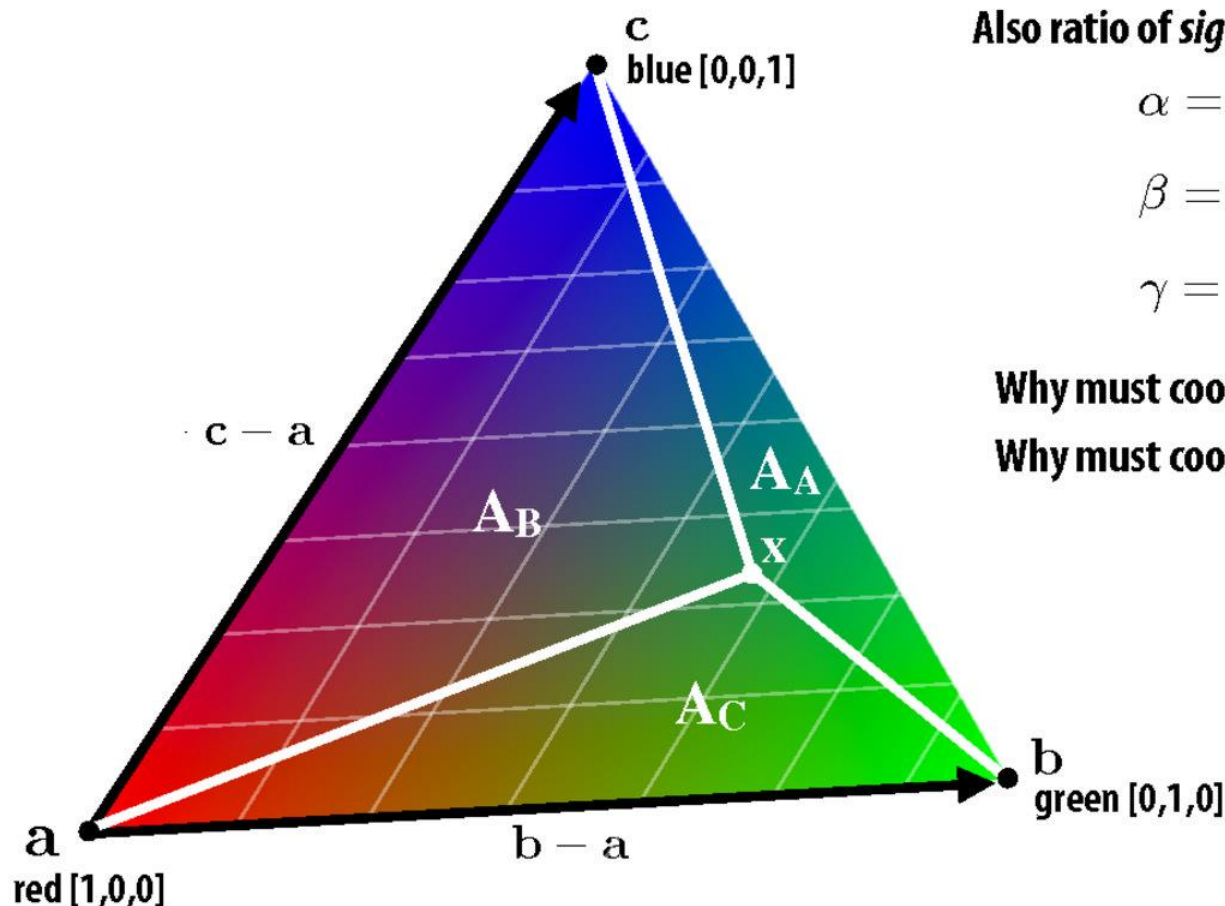


$\beta$  proportional to distance from  $x$  to edge  $c - a$

- Compute distance of  $x$  from line  $ca$
- Divide by distance of  $b$  from line  $ca$  ("height")

(Similarly for other two barycentric coordinates)

# Barycentric coordinates as ratio of areas



Also ratio of *signed* areas:

$$\alpha = A_A/A$$

$$\beta = A_B/A$$

$$\gamma = A_C/A$$

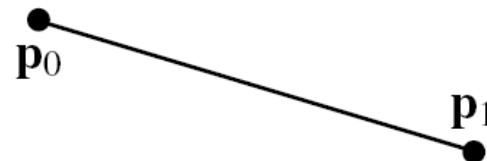
Why must coordinates sum to one?

Why must coordinates be between 0 and 1?

# Barycentric Coordinates

## Linear Interpolation

- Pick points along line:  $\mathbf{p}(u) = (1 - u)\mathbf{p}_0 + u\mathbf{p}_1$



## Barycentric Coordinates

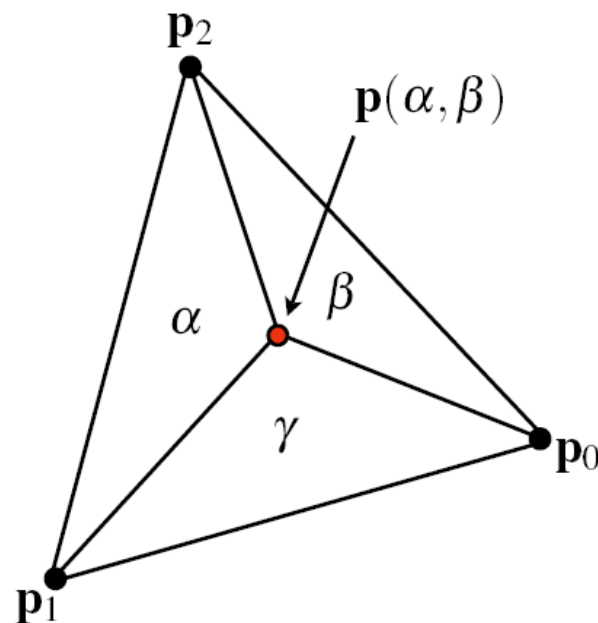
- Points in a triangle satisfy the following equation:

$$\mathbf{p} = \alpha\mathbf{p}_0 + \beta\mathbf{p}_1 + \gamma\mathbf{p}_2 \quad \text{where} \quad \alpha + \beta + \gamma = 1$$

- Coefficients are area ratios:

$$\alpha = \frac{\text{Area}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p})}{\text{Area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)} \quad \beta = \frac{\text{Area}(\mathbf{p}_0, \mathbf{p}_2, \mathbf{p})}{\text{Area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)}$$

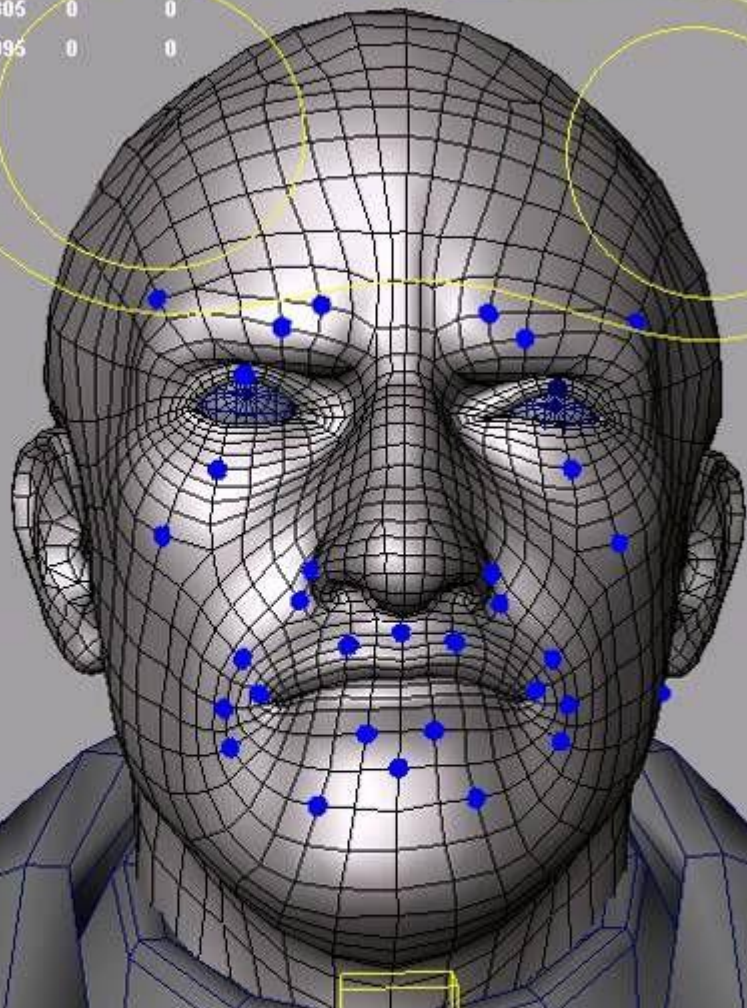
$$\gamma = \frac{\text{Area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p})}{\text{Area}(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)} = 1 - \alpha - \beta$$



# Non triangle modeling



Verts: 7042 0 0  
Edges: 14428 0 0  
Faces: 7400 0 0  
Tris: 13806 0 0  
UVs: 10096 0 0



nose emo



stretcher



funneler



pucker



mouth emo



chill



cheeks



up lip



low lip



muzzle



lips pushed

nostril



lips part

mouth close

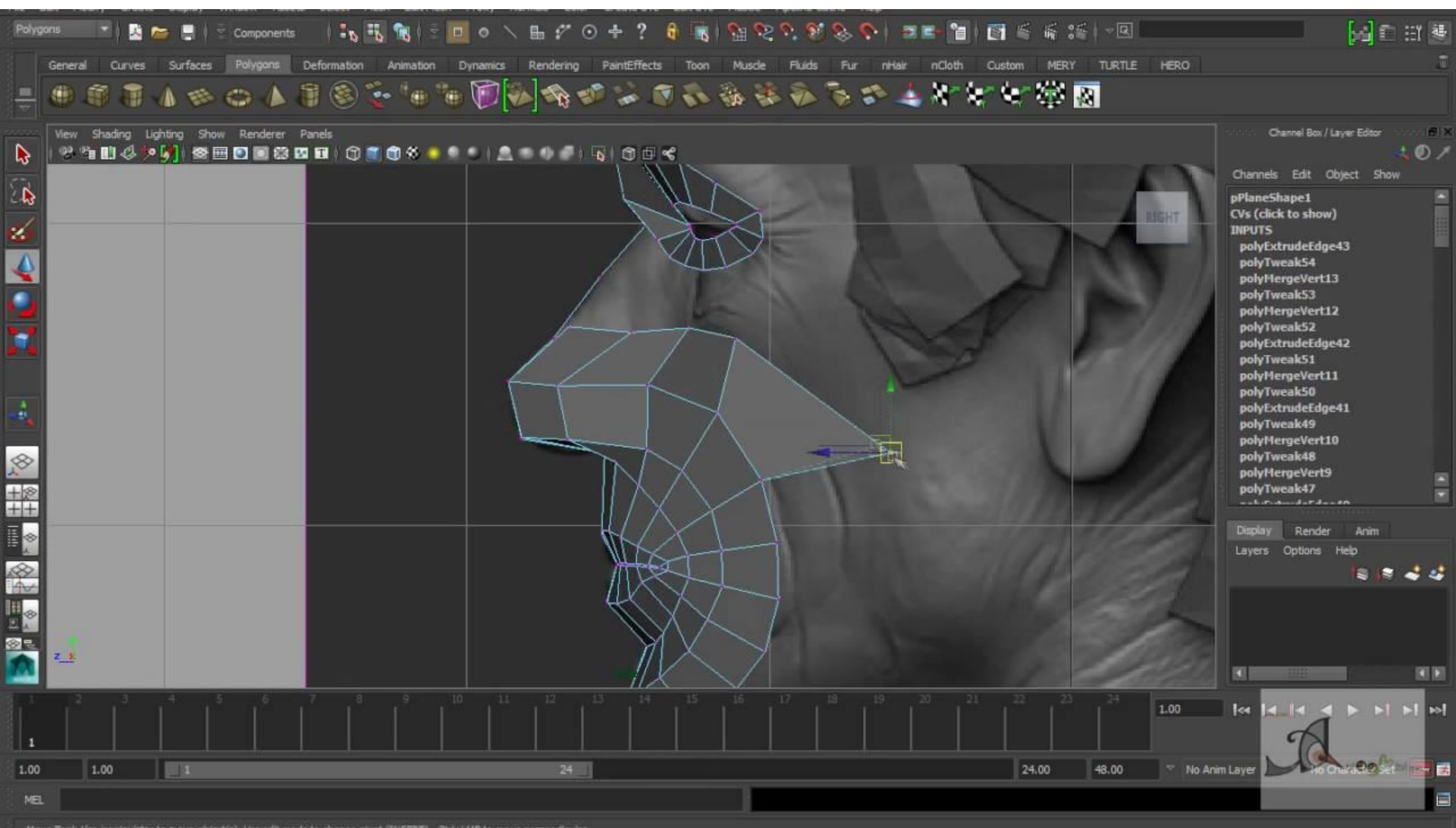
pressor

compressor

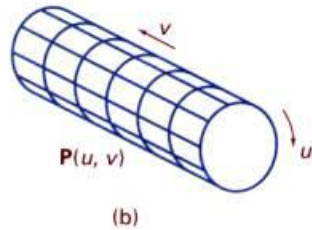
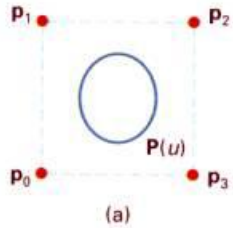
suck

persp

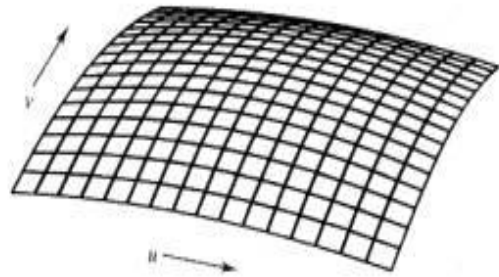
FRONT



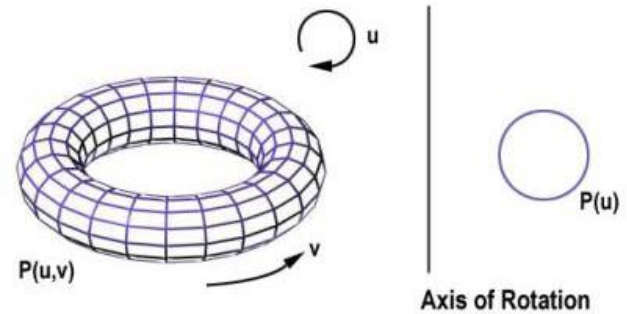
# Some Non-Polygonal Modeling Tools



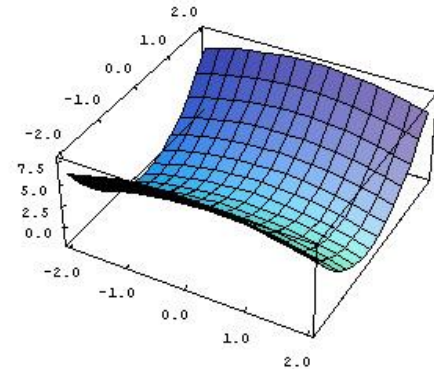
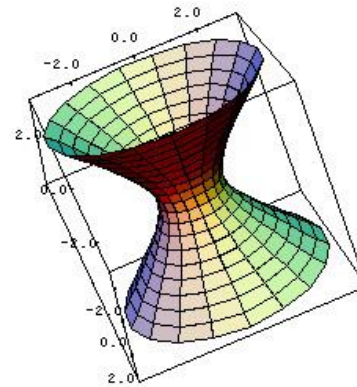
Extrusion



Spline Surfaces/Patches



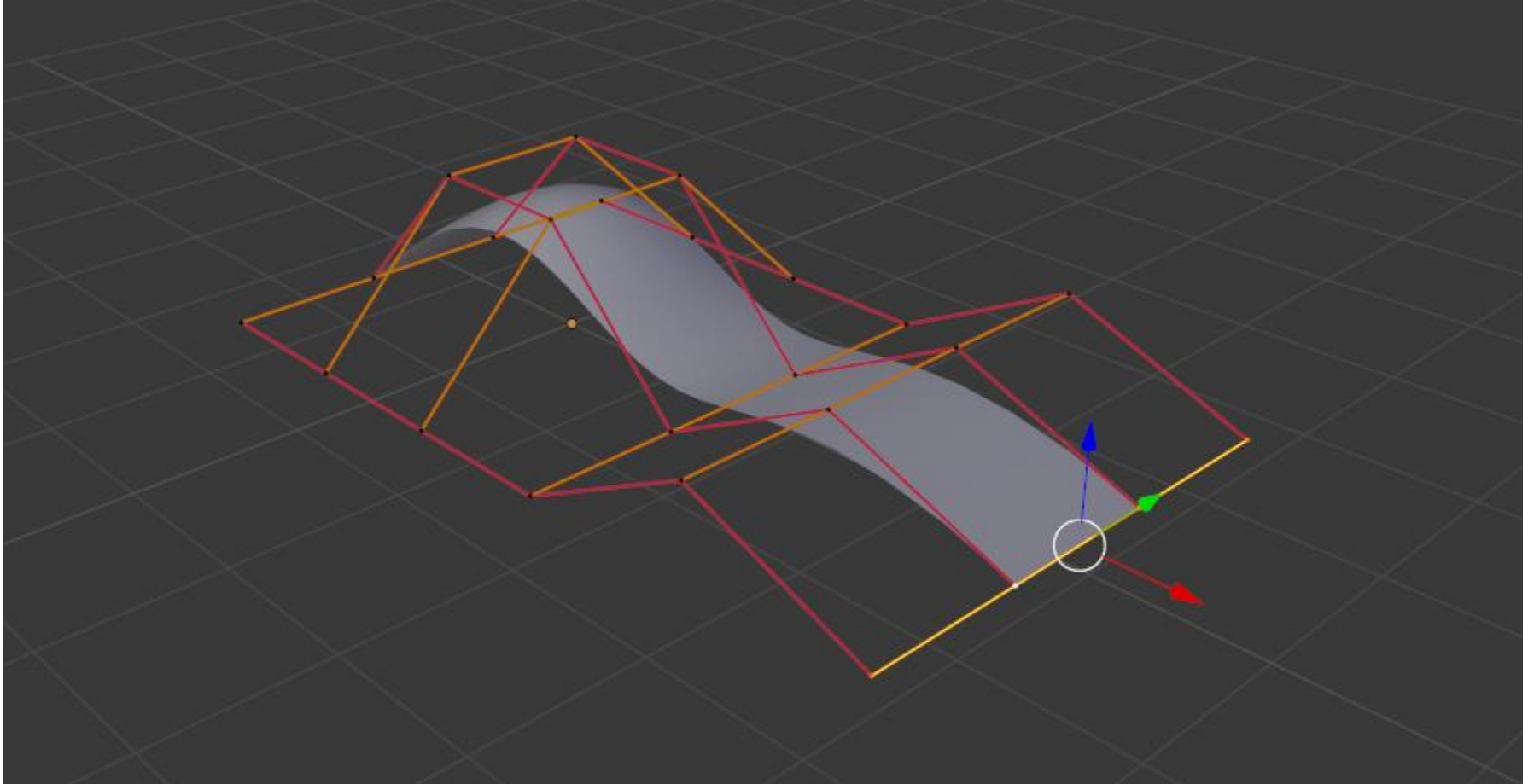
Surface of Revolution



Quadrics and other  
implicit polynomials



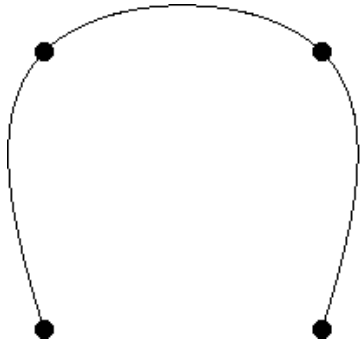
# Splines and patches



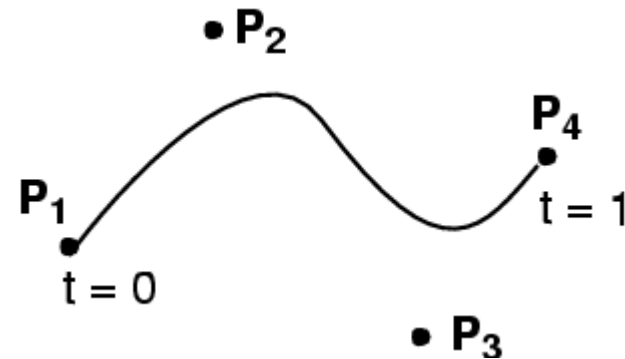
# Definition: What's a Spline?

---

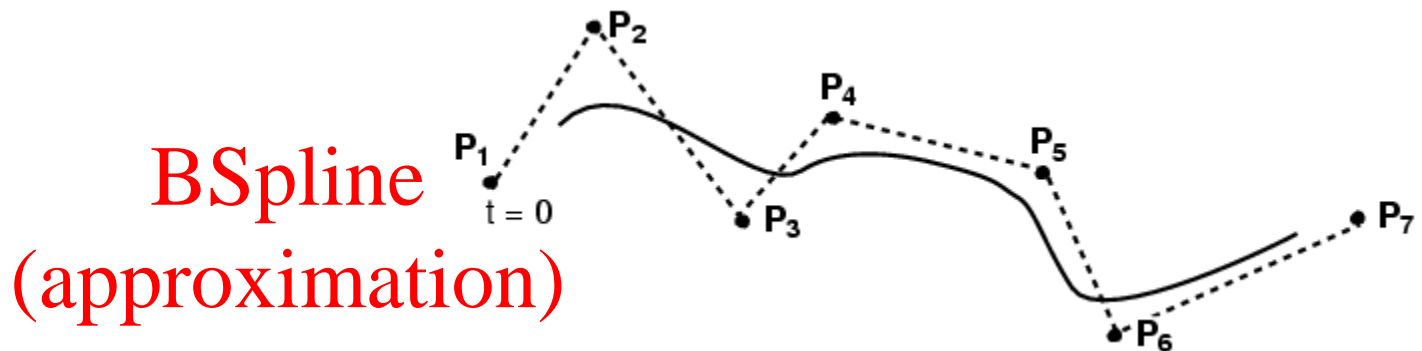
- Smooth curve defined by some control points
- Moving the control points changes the curve



Interpolation



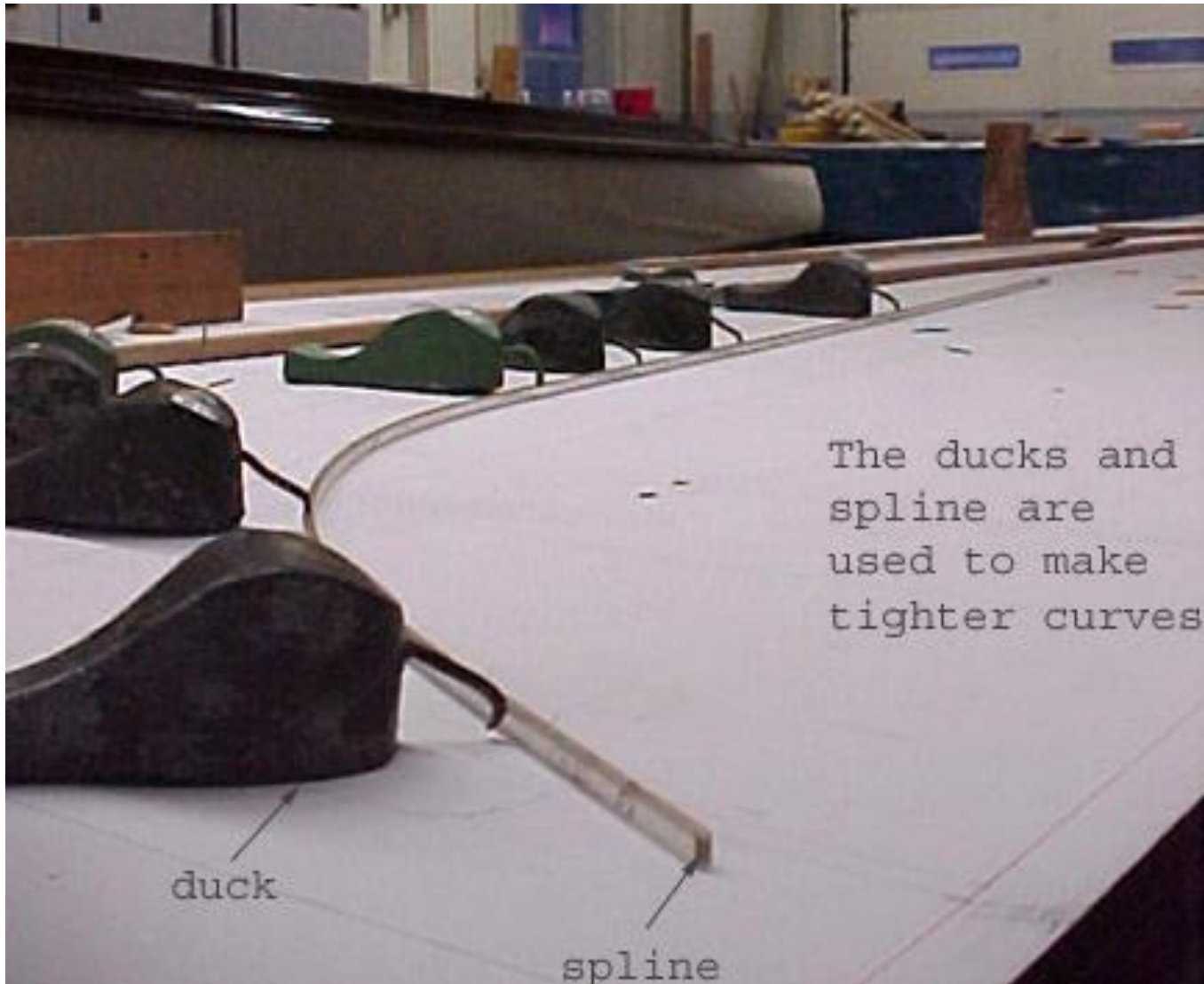
Bézier (approximation)



BSpline  
(approximation)

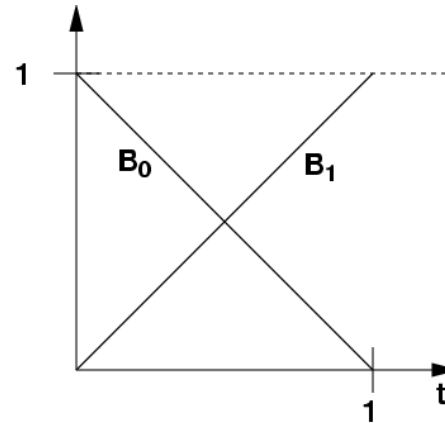
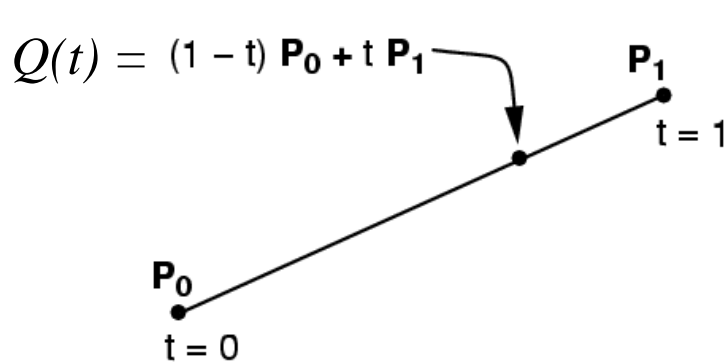
# Interpolation Curves / Splines

---



# Linear Interpolation

- Simplest "curve" between two points



Spline Basis  
Functions

a.k.a. Blending  
Functions

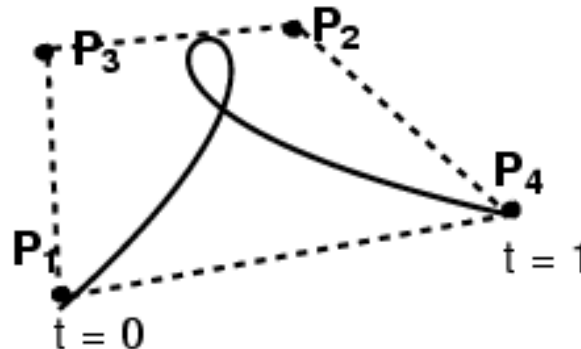
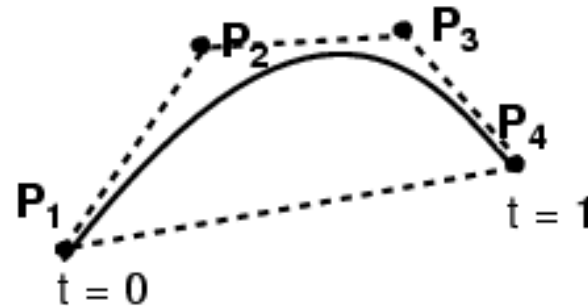
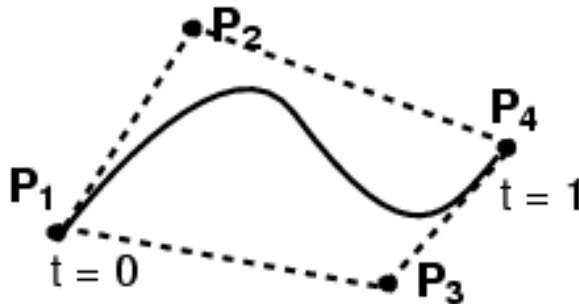
$$Q(t) = \begin{pmatrix} Q_x(t) \\ Q_y(t) \\ Q_z(t) \end{pmatrix} = \begin{pmatrix} (P_0) & (P_1) \end{pmatrix} \begin{pmatrix} -1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} t \\ 1 \end{pmatrix}$$

$$Q(t) = \mathbf{G}\mathbf{B}\mathbf{T}(t) = \text{Geometry } \mathbf{G} \cdot \text{Spline Basis } \mathbf{B} \cdot \text{Power Basis } \mathbf{T}(t)$$

# Cubic Bézier Curve

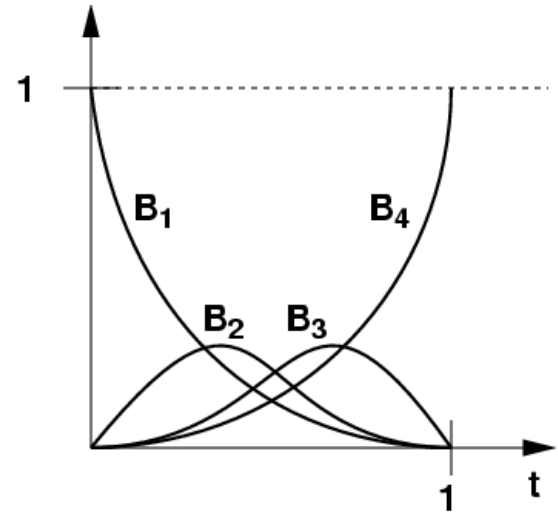
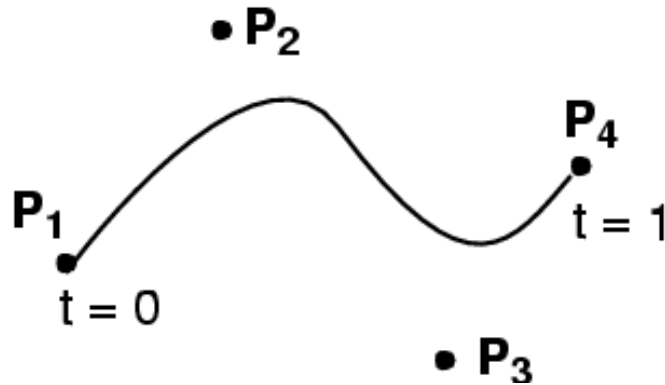
---

- 4 control points
- Curve passes through first & last control point
- Curve is tangent at  $\mathbf{P}_0$  to  $(\mathbf{P}_0 - \mathbf{P}_1)$  and at  $\mathbf{P}_4$  to  $(\mathbf{P}_4 - \mathbf{P}_3)$



A Bézier curve is bounded by the convex hull of its control points.

# Cubic Bézier Curve



$$Q(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$$

$$Q(t) = \mathbf{GBT}(t) \quad B_{Bezier} = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

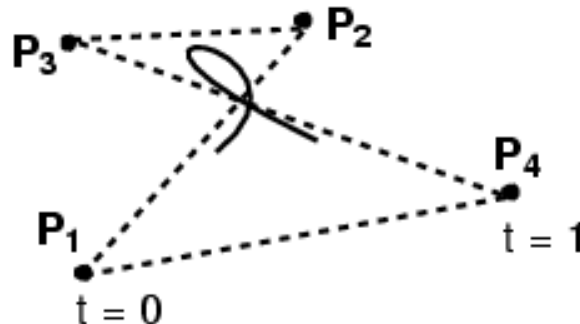
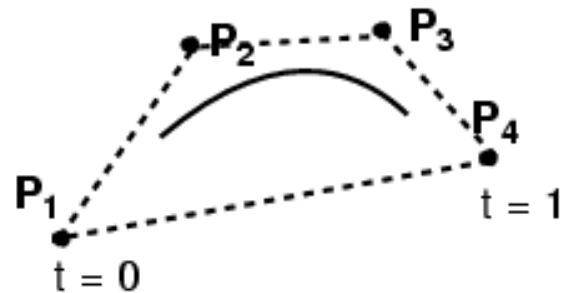
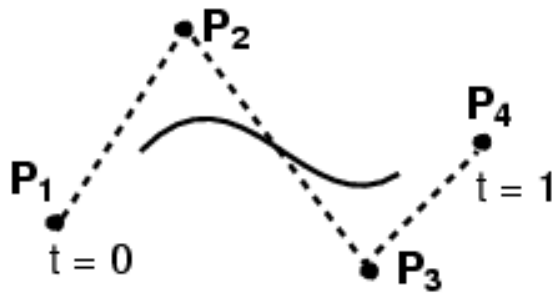
Bernstein  
Polynomials

→  $B_1(t) = (1-t)^3; B_2(t) = 3t(1-t)^2; B_3(t) = 3t^2(1-t); B_4(t) = t^3$

# Cubic BSplines

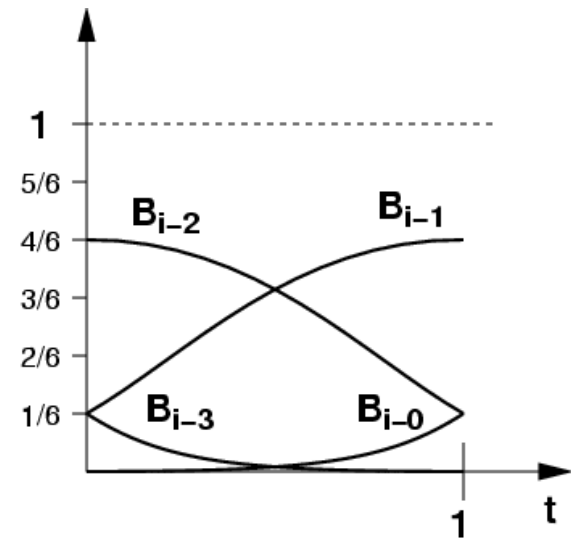
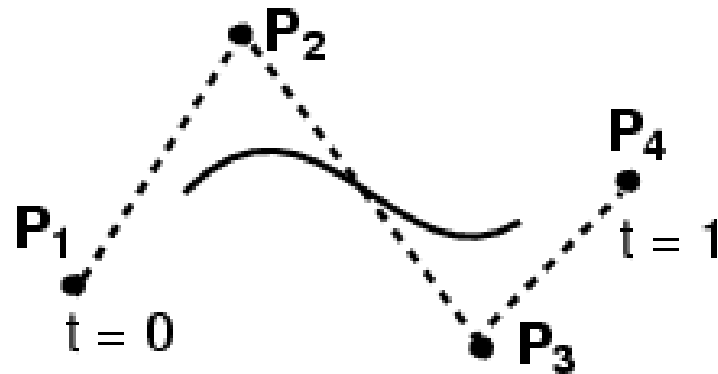
---

- $\geq 4$  control points
- Locally cubic
- Curve is not constrained to pass through any control points



A BSpline curve is also bounded by the convex hull of its control points.

# Cubic BSplines



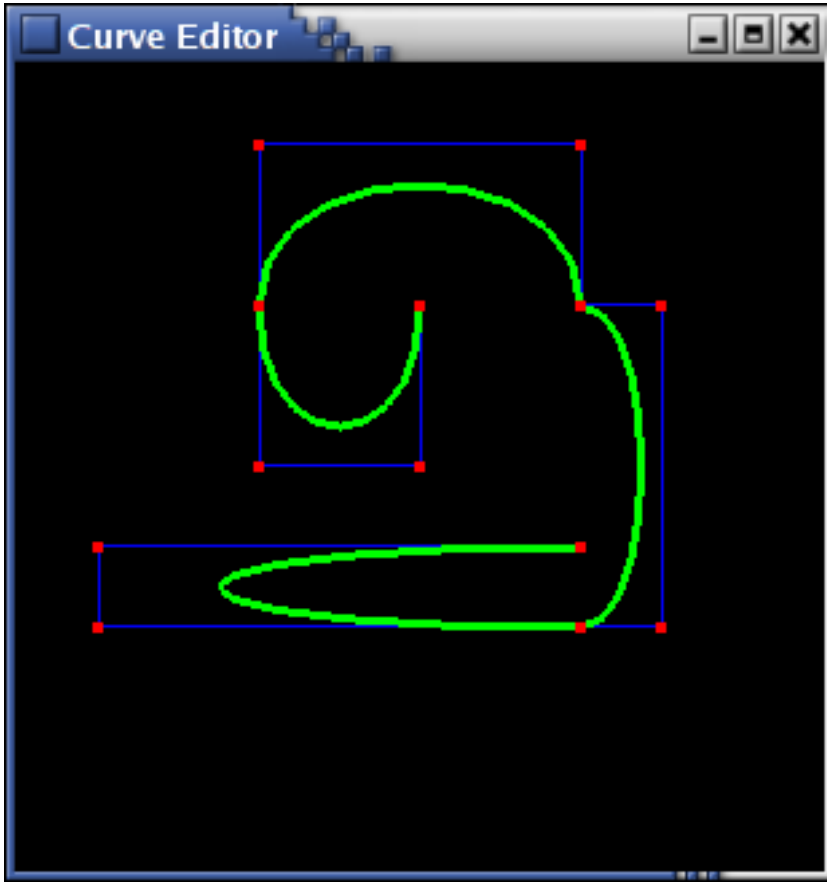
$$Q(t) = \frac{(1-t)^3}{6}P_{i-3} + \frac{3t^3 - 6t^2 + 4}{6}P_{i-2} + \frac{-3t^3 + 3t^2 + 3t + 1}{6}P_{i-1} + \frac{t^3}{6}P_i$$

$$Q(t) = \mathbf{GBT}(t) \quad B_{B-Spline} = \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 0 & 4 \\ -3 & 3 & 3 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

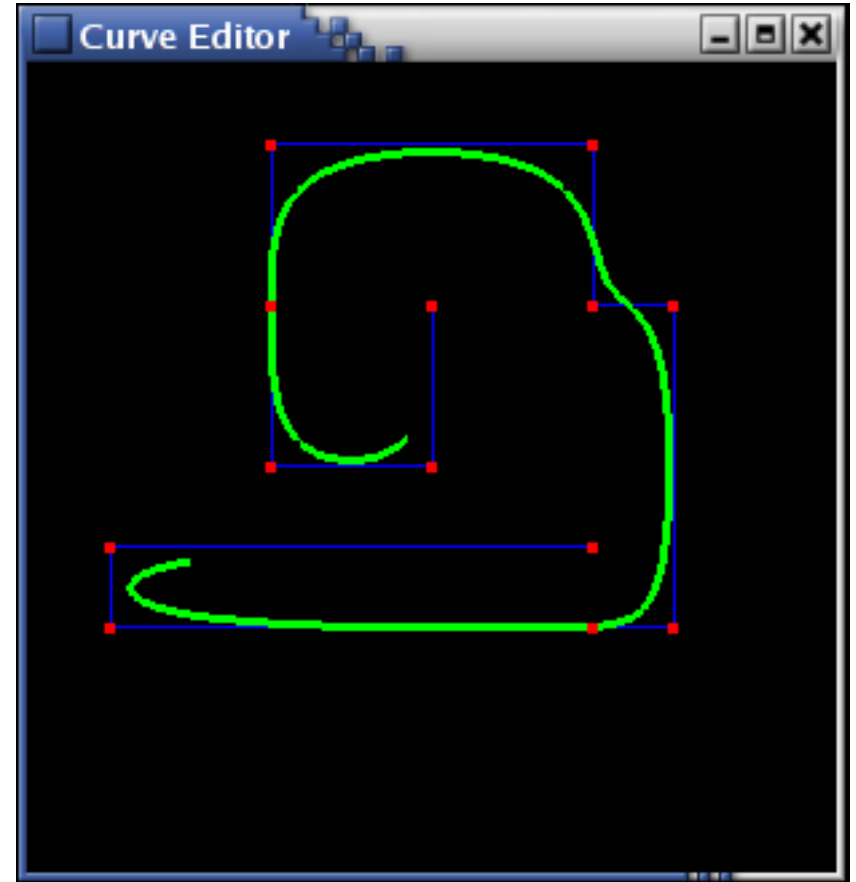


# Bézier is not the same as BSpline

---



Bézier

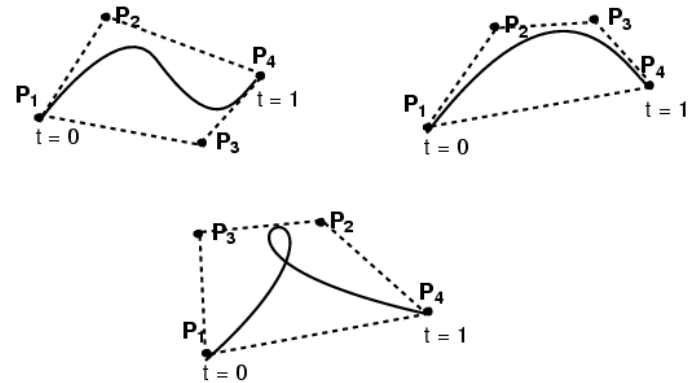
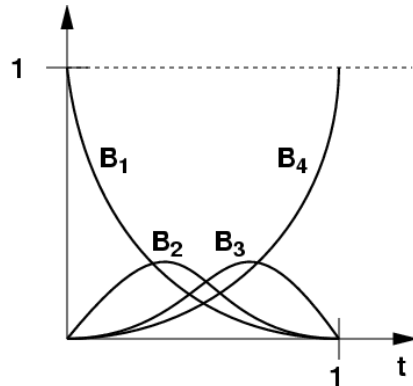


BSpline

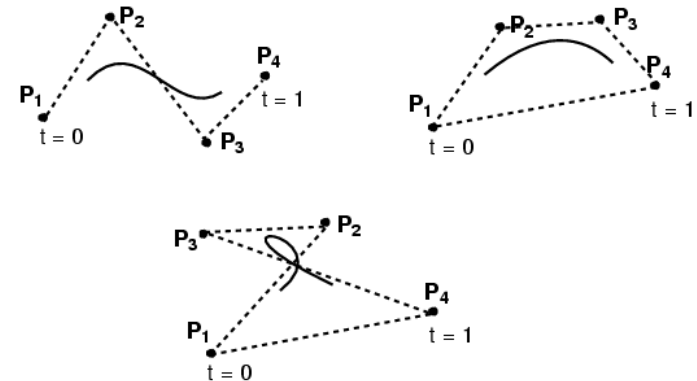
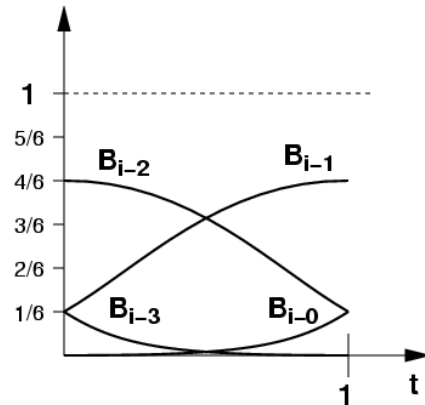
# Bézier is not the same as BSpline

- Relationship to the control points is different

Bézier



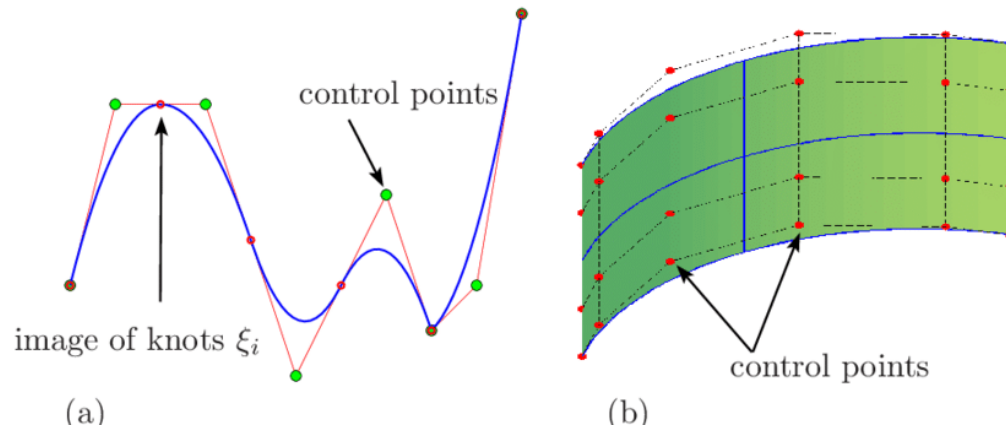
BSpline



# NURBS (generalized BSplines)

---

- BSpline: uniform cubic BSpline
- NURBS: Non-Uniform Rational BSpline
  - non-uniform = different spacing between the blending functions, a.k.a. knots
  - rational = ratio of polynomials (instead of cubic)

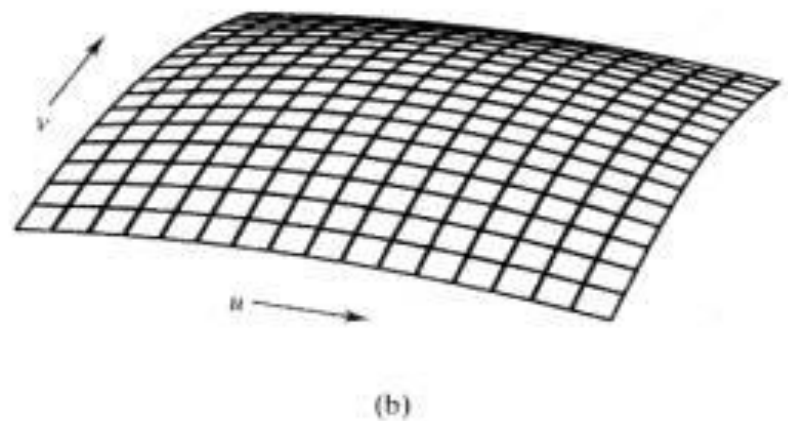
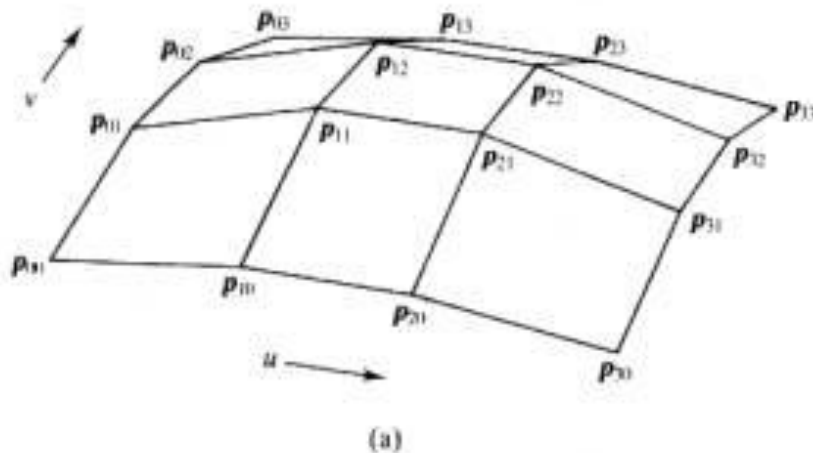


# Bicubic Bezier Patch

Notation:  $\mathbf{CB}(P_1, P_2, P_3, P_4, \alpha)$  is Bézier curve with control points  $P_i$  evaluated at  $\alpha$

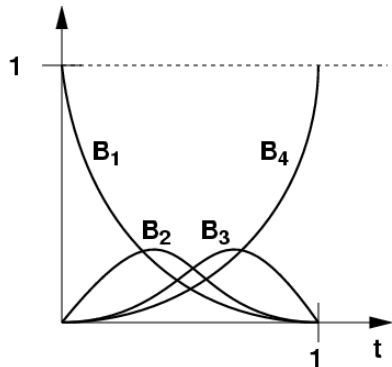
Define “Tensor-product” Bézier surface

$$Q(s, t) = \mathbf{CB}(\begin{array}{l} \mathbf{CB}(P_{00}, P_{01}, P_{02}, P_{03}, t), \\ \mathbf{CB}(P_{10}, P_{11}, P_{12}, P_{13}, t), \\ \mathbf{CB}(P_{20}, P_{21}, P_{22}, P_{23}, t), \\ \mathbf{CB}(P_{30}, P_{31}, P_{32}, P_{33}, t), \\ s) \end{array}$$

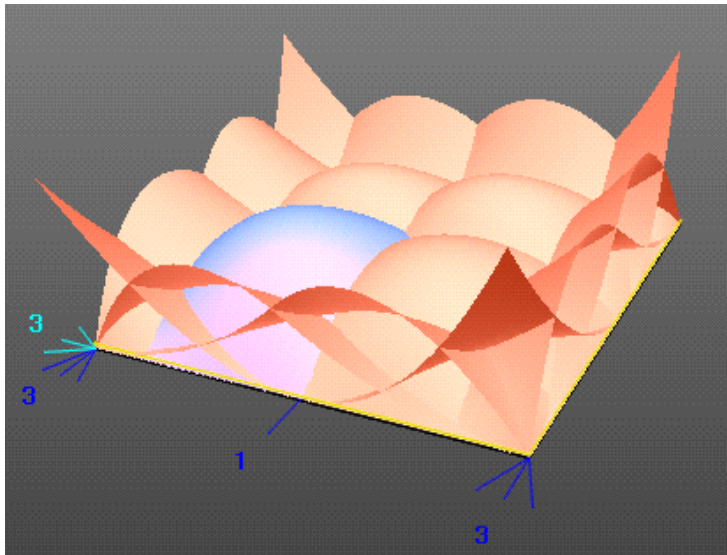


# Editing Bicubic Bezier Patches

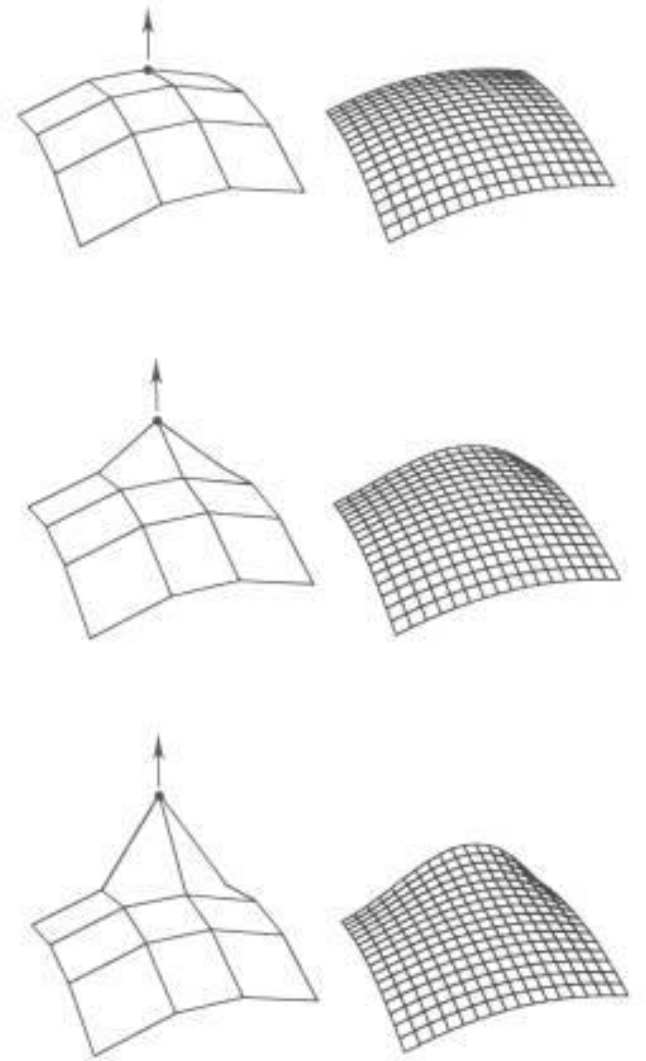
---



Curve Basis Functions



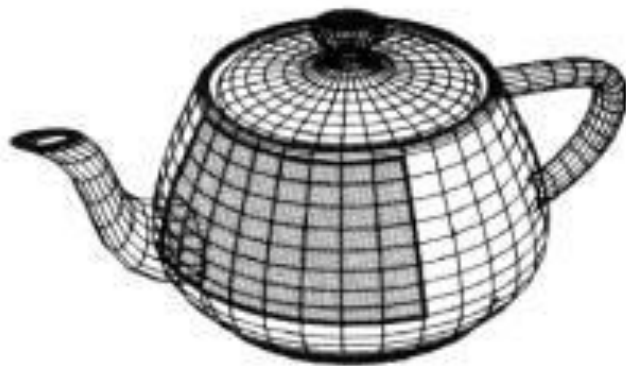
Surface Basis Functions



# Modeling with Bicubic Bezier Patches

---

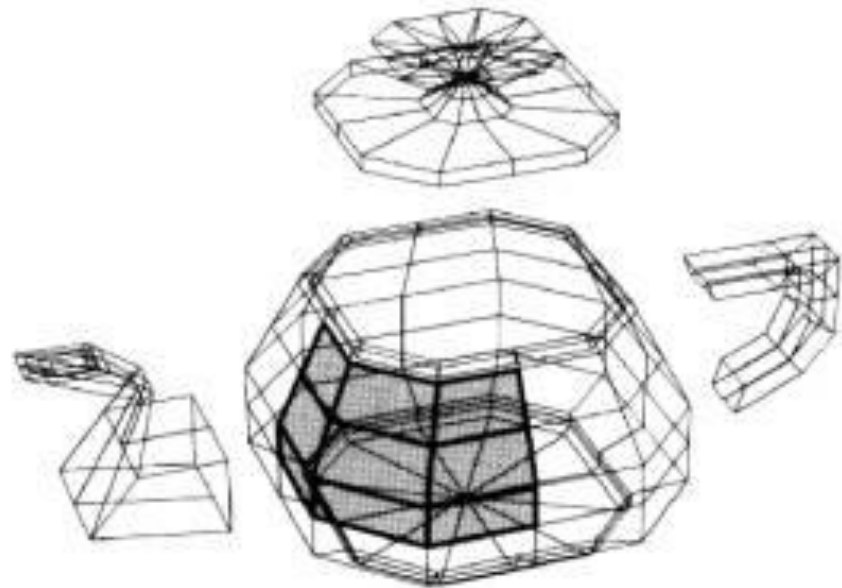
- Original Teapot specified with Bezier Patches



(a)



(b)



(c)

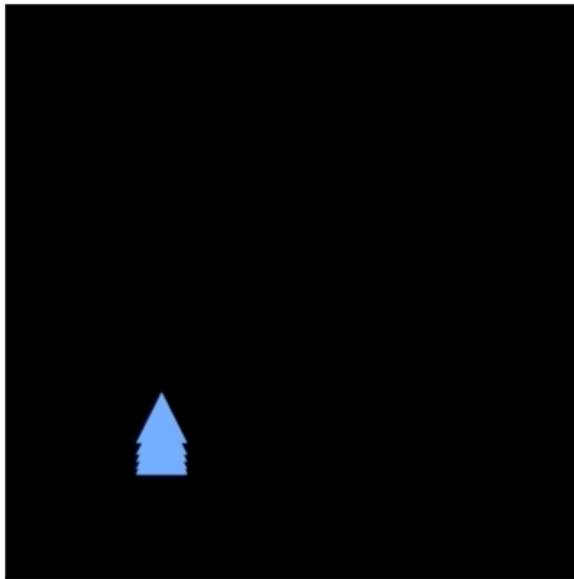
**Administrative**

# Due Dates

- Due Monday
  - HW 1
  - Lab Assignment 0
- Tuesday: NO ZOOM (watch videos)
- Due Wed
  - Quiz 1 (open until Wed)
- Due the following Monday
  - Assignment 1 (Paint Program)



# Assignment 1 (watch videos)



Clear Canvas

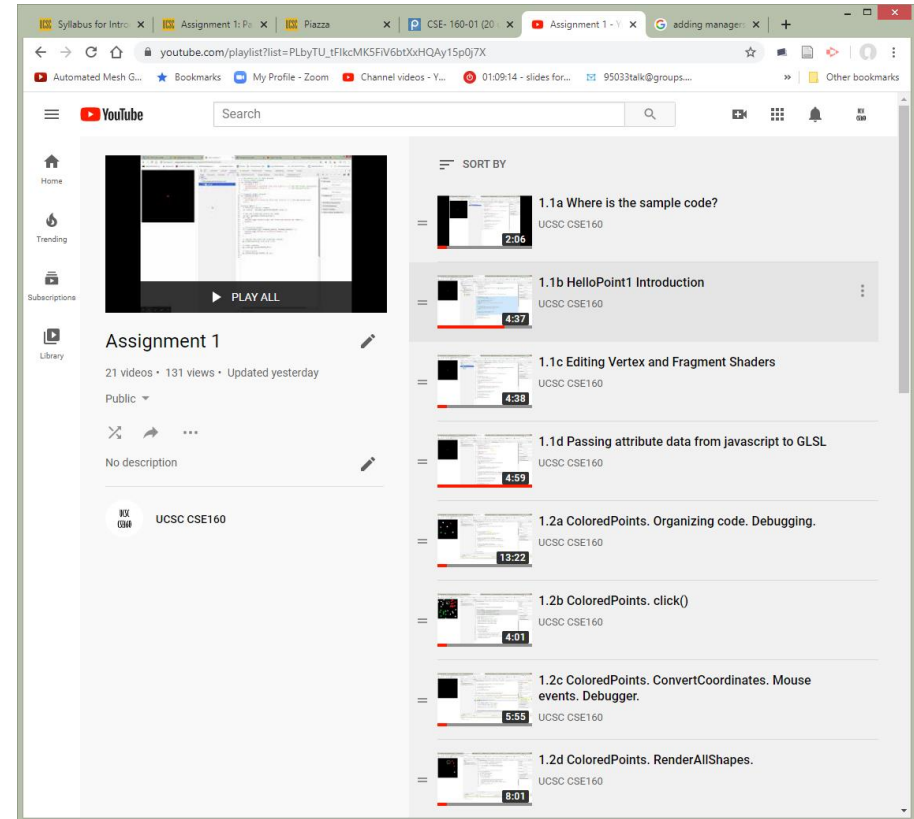
Drawing Mode:

Squares Triangles Circles

Shape Color:

Red Green Blue

Shape Size: (Circles) Segment Count:



# Check Piazza for Announcements

**PIAZZA** CSE- 160-01 Q & A Resources Statistics Manage Class James Davis

Drafts 1 polls hw1 22 hw2 7 hw3 5 hw4 2 project 13 exam 1 logistics 3 other 5

Ban User Console • Note History: disable history

New Post Search or add a post...

**Instr What will be on the final?** 3/4/20  
What will be on the final? I've been asked several times, and frankly from my perspective it indicates you are missi

**TODAY**

**Instr Gigi's Office Hours Over Zo...** 3:41PM  
Since in-person classes have been canceled for the school, I'll be hosting tomorrow's office hours over Zoom. I&

**Spotted at the 185 poster session** 3:23PM

**drawElements vs drawArrays can I use...** 3:14PM  
Prof. Davis advised us on using drawArrays for our cubes/shapes because he said it would be easier. However, the book us

**Private Show this Photo to Future C...** 11:48AM  
If you ever want to show future classes how important it is to check piazza daily you can show them this photo...I didn't  
• 1 Unresolved Followup

**TA Office Hours due to school update** 11:13AM  
Given the school has moved classes online but is allowing sections/office hours in person if necessary, will Wednesday,

**Instr Complete silliness.. Minecra...** 10:28AM  
My son loves Minecraft. I mostly do what your parents most likely did - "Get off the game!" However I recentl  
• 3 Unresolved Followups

**YESTERDAY**


**Midterm\_key question #4** 11:51PM  
for #4, how did you get the values 0.5/3 and 2.5/3 for P bilinear interpolation with the formula given in number 3 solut

**Lighting help?** 10:15PM  
Hello, I seem to be having a bit of a hard time

private note @208 2 views

## Show this Photo to Future Classes

If you ever want to show future classes how important it is to check piazza daily you can show them this photo...I didn't check piazza until 10 minutes ago when the class was empty.



other

[WebGL programming guide: interactive 3D graphics programming with WebGL](#)  
[Kouichi Matsuda and Rodger Lea](#)

The assignments follow the flow of this book closely, and I assume you are reading the chapters. You'll want a copy.

## Youtube Channel

This was new in Spring20 and students reported that it was useful, so we are keeping it. There will be playlists for different topics. e.g. Assignment1 playlist has videos to walk you through the "lab" assignment.

[https://www.youtube.com/channel/UCSynd9Z5RdlpKfvTCITV\\_8A/playlists](https://www.youtube.com/channel/UCSynd9Z5RdlpKfvTCITV_8A/playlists)

## Schedule

[Raw lecture recordings \(unedited, needs a @ucsc.edu login\) Spr20](#)

[Raw Lectures Fall20](#)

<ul style="list-style-type: none"><li>Lecture 1 (Oct 1): Introduction to Class<ul style="list-style-type: none"><li><a href="#">Slides(Fall20)</a></li></ul></li></ul>	
<ul style="list-style-type: none"><li>Lecture 2 (Oct 6): Linear Algebra Review, Intro to Shaders<ul style="list-style-type: none"><li><a href="#">Slides(Fall20)</a></li></ul></li><li>Lecture 3 (Oct 8): Object Modeling, triangles and meshes.<ul style="list-style-type: none"><li><a href="#">Videos</a></li><li><a href="#">Slides</a></li><li><a href="#">Intro to Javascript Slides</a></li></ul></li></ul>	<p>Reading:</p> <ul style="list-style-type: none"><li><a href="#">Shirley - Fundamentals of Computer Graphics - Ch 2.4-2.4.4, Ch 5.2-5.2.2</a></li><li>Matsuda Ch 1, Ch2 (9-16 pages)</li></ul> <p>Due: Due: Lab 0, HW1, Quiz 1 (Linear Algebra)</p>
<ul style="list-style-type: none"><li>Lecture 4 (Oct 13): NO ZOOM Watch the YouTube for Asg1</li><li>Lecture 5 (Oct 15): Transformations.<ul style="list-style-type: none"><li><a href="#">Videos</a></li><li><a href="#">Slides</a></li><li><a href="#">transformation_game.jar</a></li></ul></li></ul>	<p>Reading:</p> <ul style="list-style-type: none"><li>Matsuda Ch 2, Ch03 (67-91).</li></ul> <p>Due: Lab 1 - Paint Program</p>
<ul style="list-style-type: none"><li>Lecture 6 (Oct 20): NO ZOOM Watch the Youtube for Asg2.</li><li>Lecture 7 (Oct 22): Colors.<ul style="list-style-type: none"><li><a href="#">Videos</a></li><li><a href="#">Slides</a></li></ul></li></ul>	<p>Reading:</p> <ul style="list-style-type: none"><li>Shirley Ch 6 p135-158 - Transformation Matrices.</li><li>Matsuda Chap 3(91-113)</li></ul> <p>Due: HW 2, Quiz 2 (ObjectModeling, Transforms)</p>
<ul style="list-style-type: none"><li>Lecture 8 (Oct 27): Discuss quiz2, Non-triangle Modeling</li></ul>	<p>Reading:</p>

2003.13

JLuo96

3D Scar

CSE- 16

Live Q&

Inbox (4

CruzID

Google

My Me

My Driv

CSE160

CSE160

Attende

Particip

A1 Sem

Introdu

(4) | x

youtube.com/playlist?list=PLbyTU\_tFikcMOSf73r6rJpCwuf8lygfKS

Automated Mesh G... CSE101 - Intro to D... CSE101: Intro to Da... Bookmarks My Profile - Zoom Zoom JED CSE160 - Fall 2020 |... Channel videos - Y... 01:09:14 - slides for... Other bookmarks

YouTube

Search

Home

Trending

Subscriptions

Library

History

Your videos

Watch later

Lab Section: Assign...

Show more

SUBSCRIPTIONS

BassBuzz

Browse channels

MORE FROM YOUTUBE

YouTube Premium

Movies & Shows

Gaming

Live

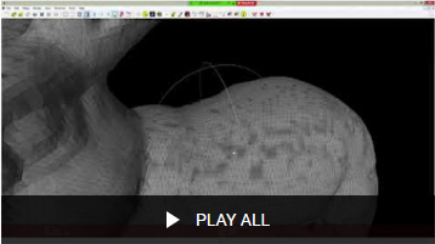
Fashion & Beauty

Learning

Settings

Report history

Help



PLAY ALL

### L3: Object Modeling

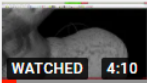
8 videos • 18 views • Last updated on Apr 25, 2020

Public

No description

UCSC CSE160


SORT



WATCHED 4:10

Overview of Meshes and Triangles


UCSC CSE160



6:48

OpenGL Primitives


UCSC CSE160



4:05

What Will Render A Square Sample Problem with ExplanationOpenGL Primitives

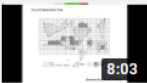
UCSC CSE160



10:12

Storing Vertex Data into Buffers


UCSC CSE160



8:03

Rasterization


UCSC CSE160



6:08

Normals

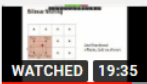
UCSC CSE160



11:11

Normals Sample Problem With Explanation

UCSC CSE160



WATCHED 19:35

Interpolation, Color, and Barycentric Coordinates

UCSC CSE160

Stuff missing

Q&A

End