CS519 Cheat Sheet

1) CIE XYZ and xyY color spaces

CIE XYZ color space (Week 1: Perceptual Color Spaces - 15:30)

- no negative values
- separate luminance from chromaticity

RGB to XYZ:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 2.768892 & 1.751748 & 1.130160 \\ 1 & 4.590700 & 0.060100 \\ 0 & 0.056508 & 5.594292 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

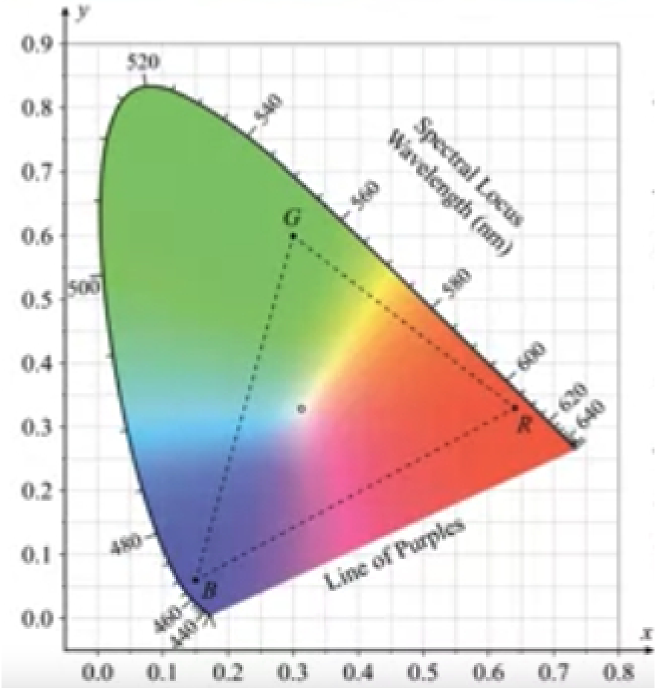
Y corresonds to **brightness**.

CIE xyY color space (Week 1: Perceptual Color Spaces-18:10)

- normalized chromaticity values in [0,1]
- no change luminance
  Convertion:

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad \text{and} \quad z = \frac{Z}{X+Y+Z}$$

CIE xy chromaticity diagram:



2) Gamma correction (know that gamma means raising a value to some exponent, and correction is raising a value to the reciprocal)

Week 1-Gamma Correction

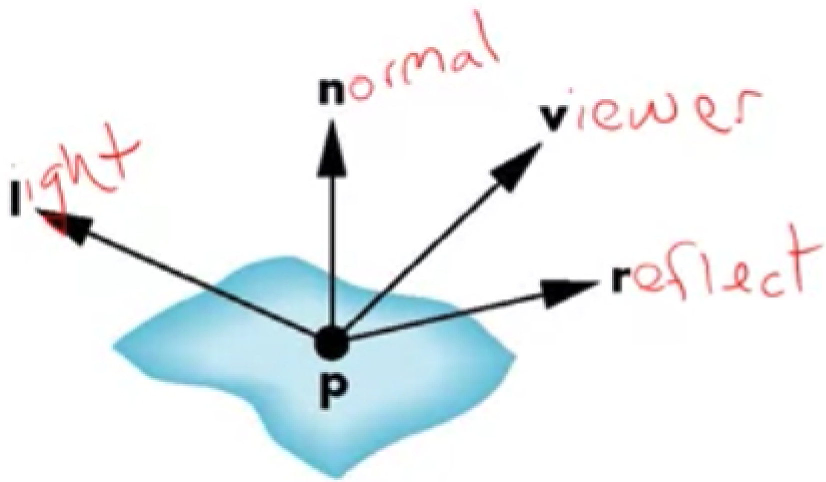
In CRT disaplays, $V_{display} = V_{signal}^{\gamma}$, where γ varied by display, but **2.2** was a typical value ==> displayed colors were darker than input color.

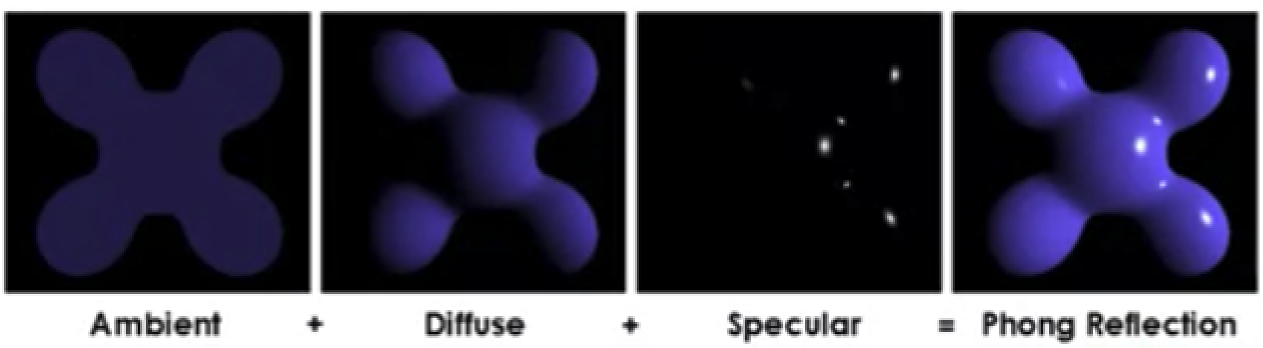
So **Gamma Correction** is: $V_{signal}^{1/\gamma}$

- LCD-LEDs don't use gamma.
- sRGB standard uses gamma.

3) Phong and Blinn-Phong reflection models

Phong reflection model: (Week 2-Shading-8:20)



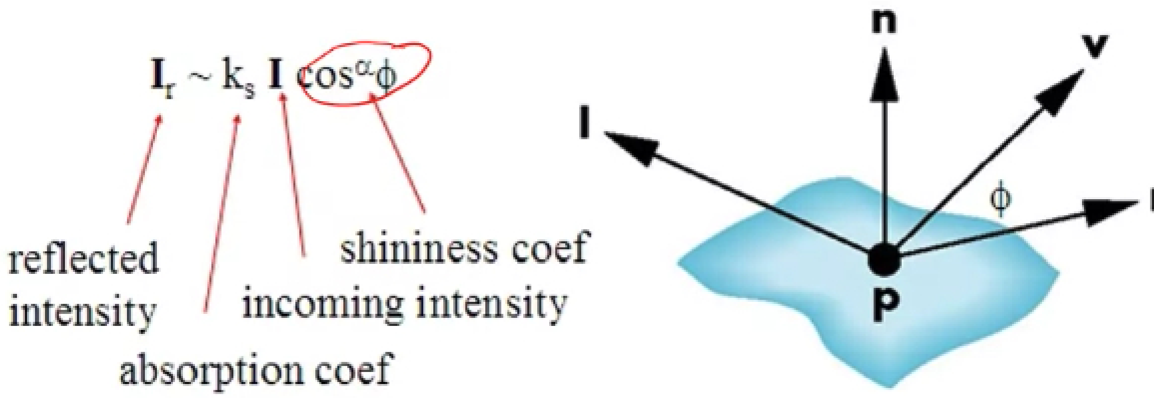


$$I_{\mathrm{p}} = k_{\mathrm{a}} i_{\mathrm{a}} + \sum_{m \in \mathrm{lights}} \left(k_{\mathrm{d}} (\hat{L}_m \cdot \hat{N}) i_{m,\mathrm{d}} + k_{\mathrm{s}} (\hat{R}_m \cdot \hat{V})^{\alpha} i_{m,\mathrm{s}} \right)$$

ka is the reflectance of material; i is the intensity;

Specular reflection:

- High coefficient means smoother look
  - Maybe 100 for metal
  - Maybe 10 for plastic



Diffuse reflection:

- Light scattered equally in all directions
- Amount of light reflected is affected by the angle of incidence
  - reflected light proportional to ***cosine of angle between l and n***
  - if vectors normalized

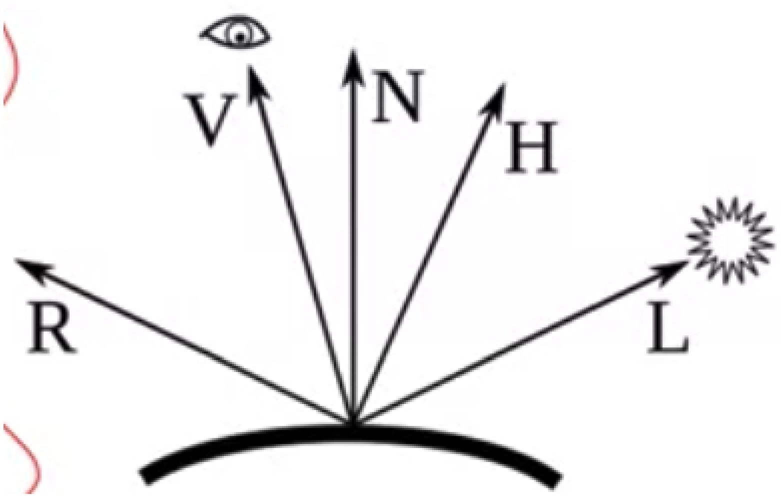
$$\cos(\theta) = n \cdot l$$

Blinn-Phong Refrelection model: (Week 2-Shading-20:30)

Replace the $(V.R)^a$ term with $(N.H)^b$ term where H is the halfway vector.

- More efficient.
- Use **higher** $b > a$ make output similar to Phong with a

$$H = \frac{L+V}{||L+V||}$$

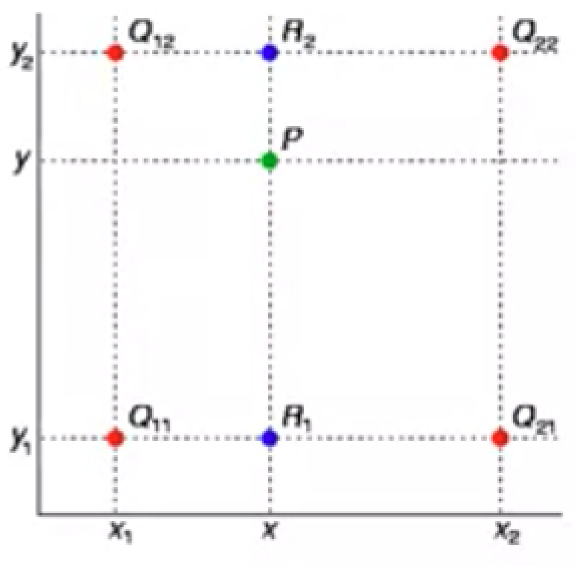


4) Linear and Bilinear interpolation (memorize how to do it)

Linear Interpolation (Week 3-Linear interpolation-10:00)

2 points: p_0 and p_1 where $f(p_0) = v_0$ and $f(p_1) = v_1$, then $f(t) = (1-t)v_0 + tv_1$ and the t is:
$t = \dfrac{dist(p_i, p_0)}{dist(p_1, p_0)}$

Bilinear Interpolation (Week 3-Linear interpolation-13:00)



$$f(x, y_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}),$$
$$f(x, y_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}).$$

$$f(x, y) \approx \frac{y_2 - y}{y_2 - y_1} f(x, y_1) + \frac{y - y_1}{y_2 - y_1} f(x, y_2)$$

where $Q_{11} = (x_1, y_1), Q_{12} = (x_1, y_2), Q_{21} = (x_2, y_1), Q_{22} = (x_2, y_2)$

5) Barycentric coordinates

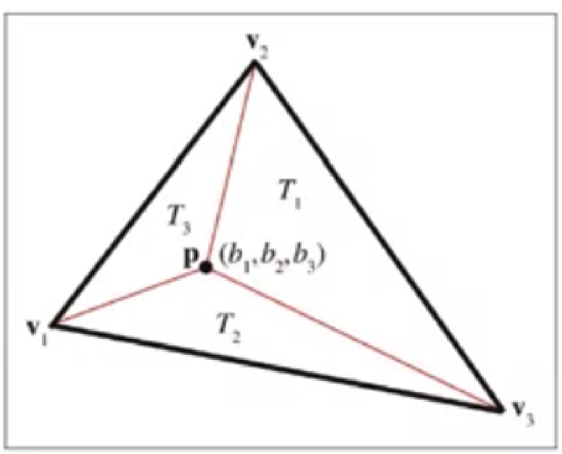
Week 3-Barycentric Coordinates and interpolation-4:00

describe location of a point p in relation to the vertices of a given triangle.
$p = (\lambda_1, \lambda_2, \lambda_3)$ where
$p = \lambda_1 a + \lambda_2 b + \lambda_3 c$ and
$\lambda_1 + \lambda_2 + \lambda_3 = 1$

Interpolation:

$f(p) = \lambda_1 f(a) + \lambda_2 f(b) + \lambda_3 f(c)$

Coordinates are the signed area of the opposite subtriangle divided by area of the triangle



$$b_1 x_1 + b_2 x_2 + b_3 x_3 = p_x,$$
$$b_1 y_1 + b_2 y_2 + b_3 y_3 = p_y,$$
$$b_1 + b_2 + b_3 = 1.$$

$$b_1 = \frac{(p_y - y_3)(x_2 - x_3) + (y_2 - y_3)(x_3 - p_x)}{(y_1 - y_3)(x_2 - x_3) + (y_2 - y_3)(x_3 - x_1)},$$
$$b_2 = \frac{(p_y - y_1)(x_3 - x_1) + (y_3 - y_1)(x_1 - p_x)}{(y_1 - y_3)(x_2 - x_3) + (y_2 - y_3)(x_3 - x_1)},$$
$$b_3 = \frac{(p_y - y_2)(x_1 - x_2) + (y_1 - y_2)(x_2 - p_x)}{(y_1 - y_3)(x_2 - x_3) + (y_2 - y_3)(x_3 - x_1)}.$$

$$b_1 = A(T_1)/A(T), \qquad b_2 = A(T_2)/A(T), \qquad b_3 = A(T_3)/A(T)$$

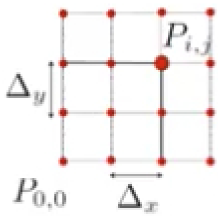
where A means area.

6) Structured grid representations of domains

Week 4-Meshes and Elements-10:00

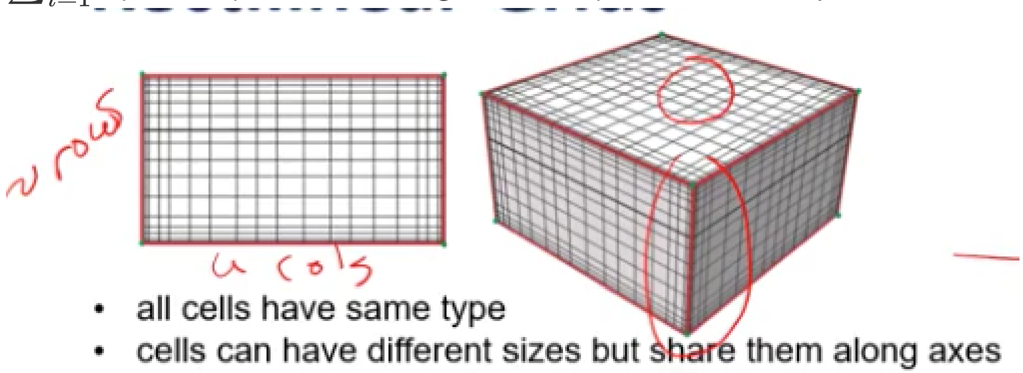
Structured grid:

- uniform grid: need m integers for #vertices along each of the m dimensions and 2 corrner points.

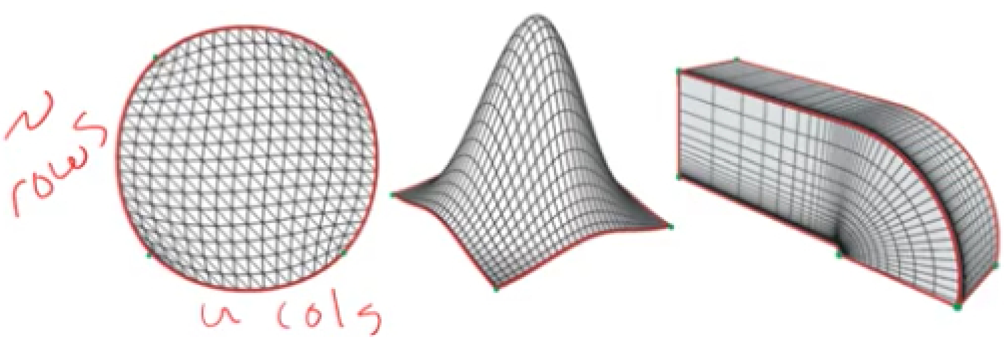


$$P_{i,j.} = P_{0,0} + i\Delta_x \vec{e}_x + j\Delta_y \vec{e}_y \cdot$$

- rectilinear grids: all cells have same type but can have different sizes sharing along axes; need $\sum_{i=1}^{m} d_i$ floats (#vertices along each axis) and 1 corrner points



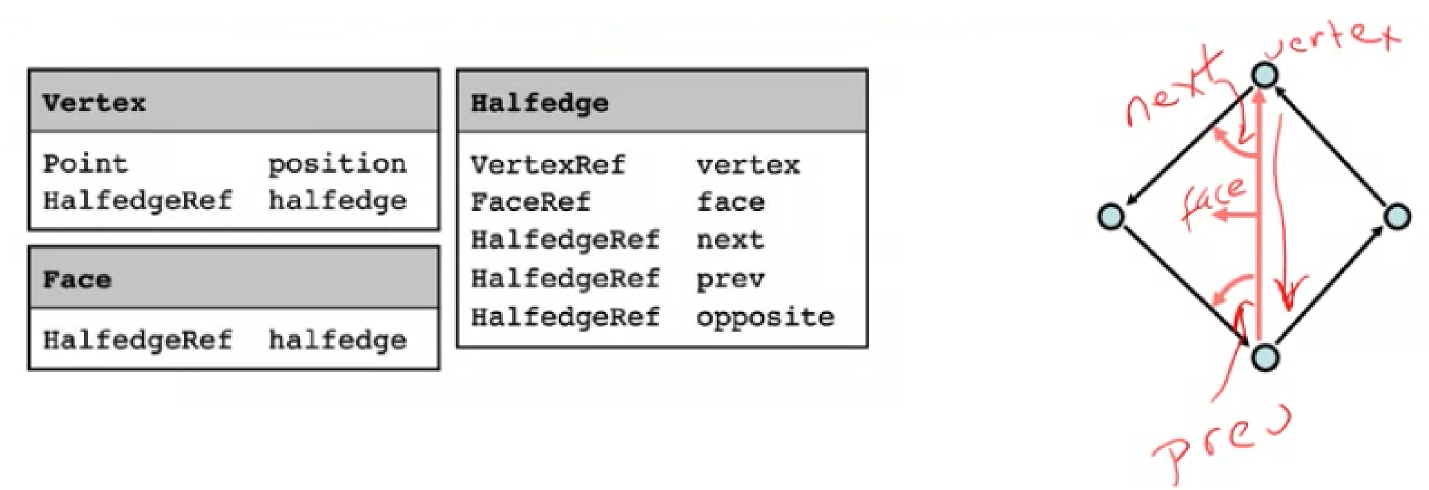
- Curvilinear grids: all cells have same shape and cell vertex coordinates are **freely** specified; need $\prod_{i=1}^{m} d_i$ floats (coordinates of all vertices) and 1 for each axis (#vertices along each axis)



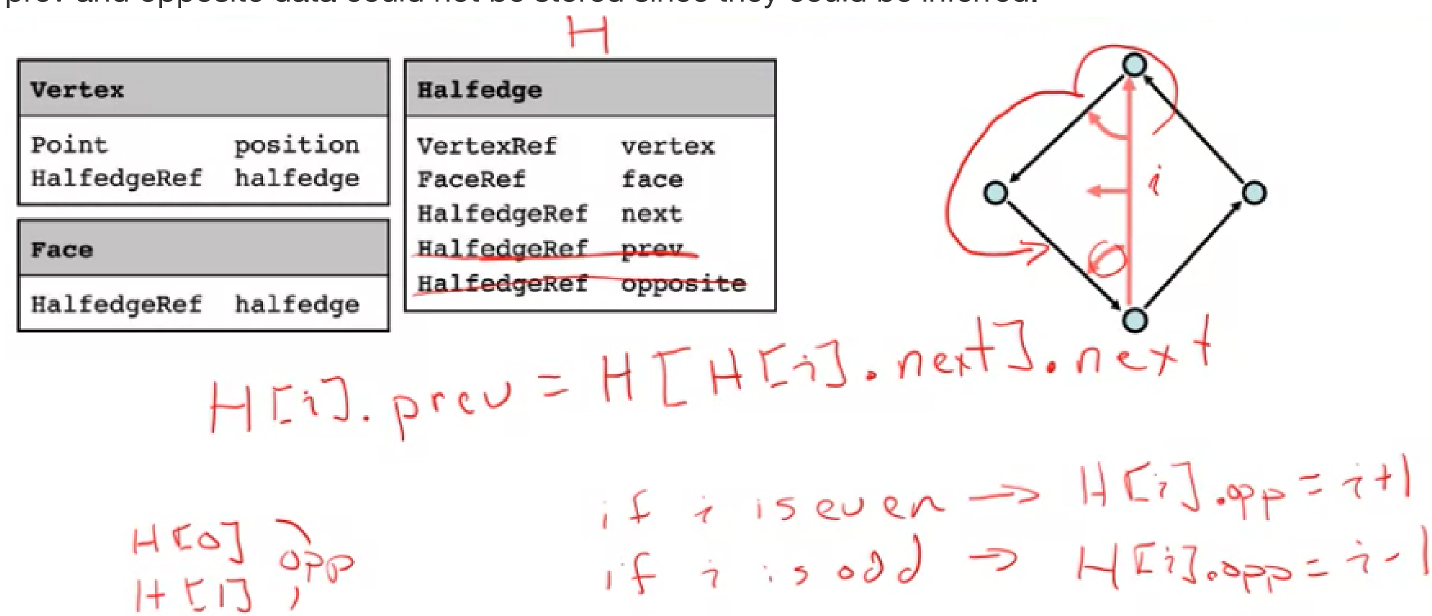
7) Half-edge data structure

Week 4-Data structures for polygonal meshes-13:00

Data structure to save ploygon mesh coordinates: simple and efficient traversal of vertex **neighbourhoods**.



prev and opposite data could not be stored since they could be inferred.



Data storage: $vertex + face + half edge$

- vertex: x,y,z, halfhedge ref ==> 4 * 4 bytes/vertex
- face: 4 bytes/face ==> 2*4 bytes/vertex
- halfedge: 3*4 bytes/halfedge ==> since $E \approx 3V$, $HalfE \approx 6V$, then 6*3*4 bytes/vertex
  Euler Characteristic

V-E+F= 2(1-G)

V = number of vertices
E = number of edges
F = number of faces
G = genus (number of holes in the surface)

Traverse the mesh in **counterclock** around the face using halfedge **next** reference and **vertex** reference for each halfedge.

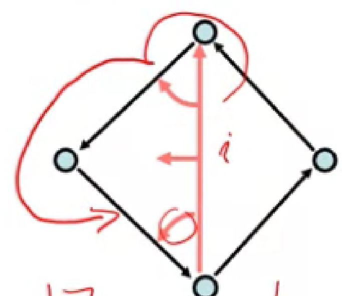
Traverse the vertices on the face by one-ring traversal:

Vertex	
Point	position
HalfedgeRef	halfedge

Face	
HalfedgeRef	halfedge

H

Halfedge	
VertexRef	vertex
FaceRef	face
HalfedgeRef	next
~~HalfedgeRef~~	~~prev~~
~~HalfedgeRef~~	~~opposite~~



H[i].prev = H[H[i].next].next

H[0] ⌉ opp
H[1] ⌋

if i is even → H[i].opp = i+1
if i is odd → H[i].opp = i-1

8) Colormap construction

Week 2-Colormap

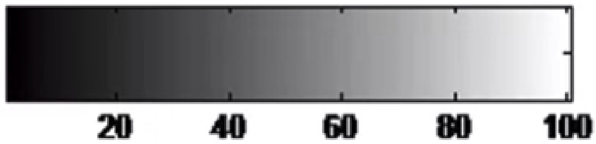
Color table: pre-compute colors and store colors

color mapping function: N colors and index them into [0, N-1]

$i = min(\lfloor \frac{x - x_{min}}{\frac{x_{max} - x_{min}}{N}} \rfloor, N - 1)$, get the floor value.

Transfer functions: define colors at certain scalar values (knots), then use interpolation to define colors between knots.

- Consider a function with a range of [0,100]
- c(0) = (0,0,0) and c(100)=(1,1,1) and use linear interpolation in between



This is a simple but super effective colormap!

rainbow color map is not linear.

diverging colormap: have breakpoint in the middle.

Colormap design advice:

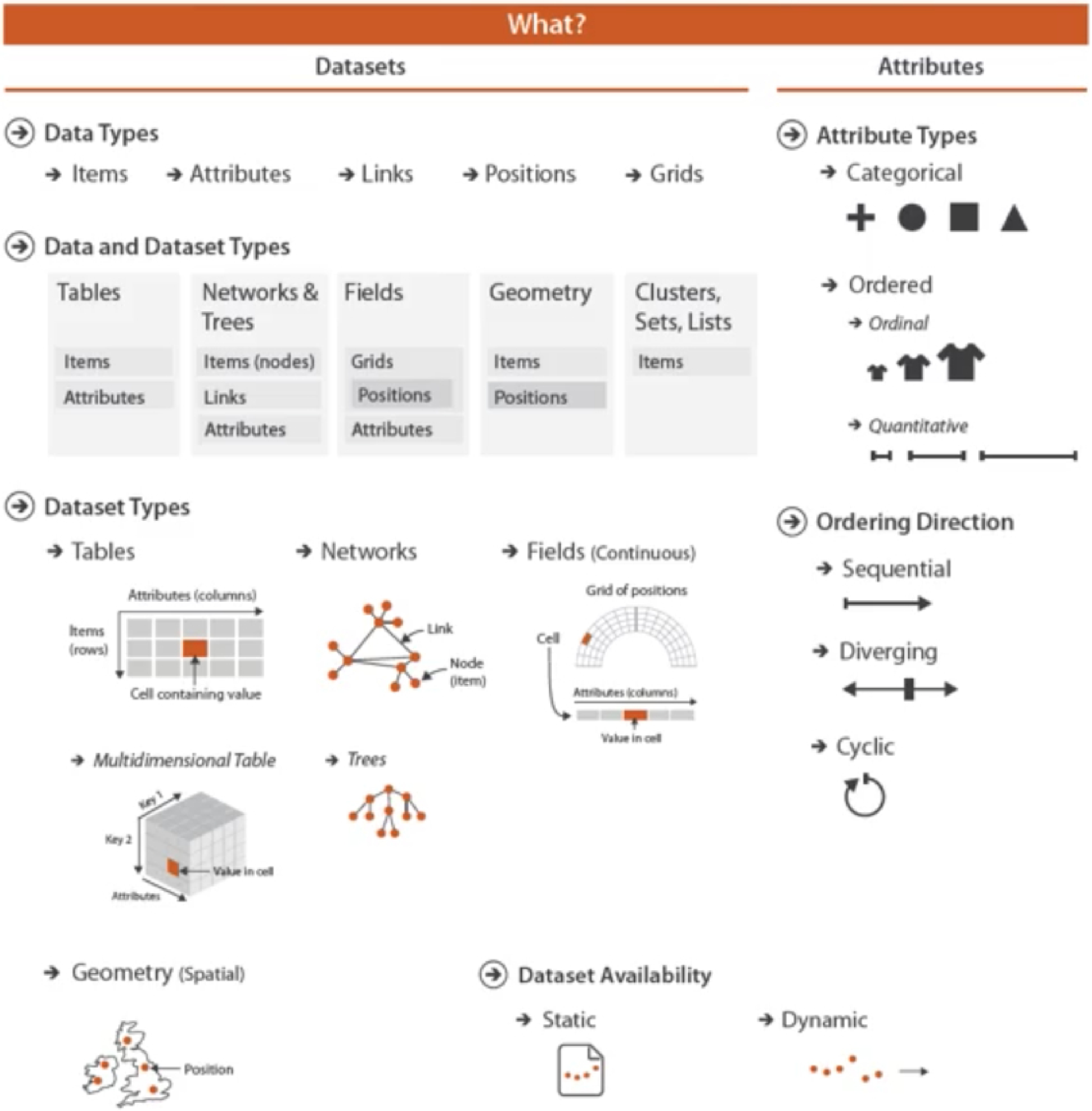
- Design for accessibility
  - minimally, don't depend on red-green differentiation
- Use your knowledge of the data set (*e.g.*, is there a critical value?)
- If there is a standard in the field the audience may be expecting?
- Often a perceptually uniform colormap is the best choice
  - equal steps in data are perceived as equal steps in the color space
- We perceive change in lightness as changes in the data pretty well
  - better than changes in hue.
- Use colormaps with monotonically increasing lightness

9) Munzner's data taxonomy

Week 3-A Data Taxonomy

Data types: structure or mathematical interpretation of data

- item: individual entity.
- attribute: property of item
- links: relationship between items
- positions: spatial data / pixel data
- grids: sampling strategy for continuous data



10) DEM to TIN conversion process

Week 4-Terrain Generation for Geographic information systems-

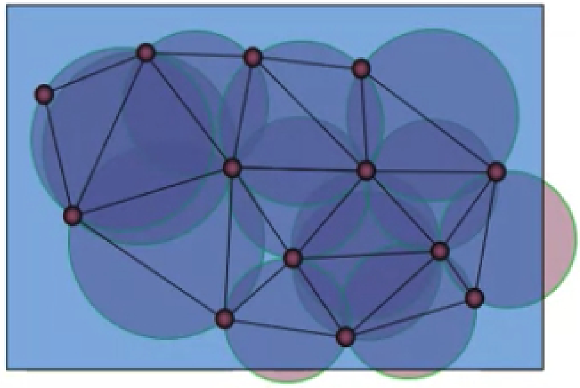
DEM: digital elevation model

TIN: triangular irregular network

LIDAR points --> triangulating --> TIN --> interpolation --> raster DEM.

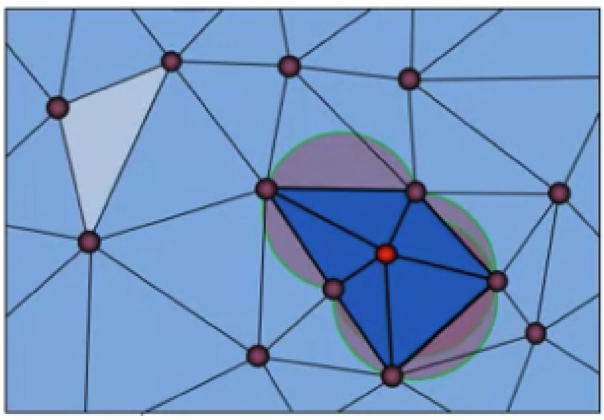
- compute TIN in places where all points have already arrived
- raster, output & deallocate
- keep parts that miss neighbourhoods

Delaunay Triangulation



- a triangulation in which every triangle has an empty circumscribing circle

Incremental point insertion in delaunay triangulation



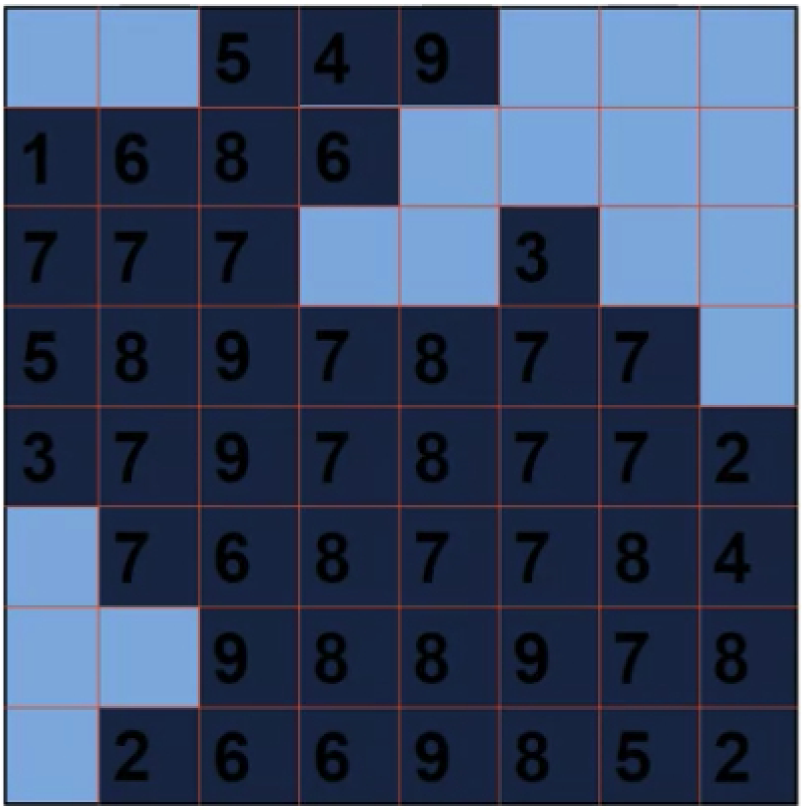
[Lawson '77]
[Bowyer '81]
[Watson '81]

- locate triangle enclosing the point

- find and remove all triangles with non-empty circumcircles
- triangulate by connecting new point

Spatial finalization of points

Spatial Finalization of Points



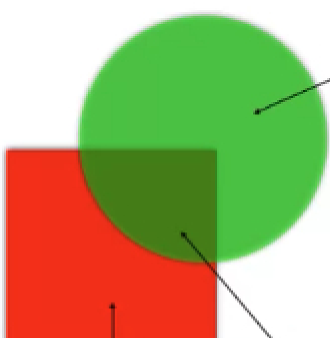
1. compute bounding box
2. create finalization grid
   - count number of points per cell
3. output finalized points
   - buffer per grid cell
   - if full, output points in one randomized chunk followed by finalization tag

11) Compositing with the over operator

Week 6-Compositing-5:00

$\mathbf{c}_f = (0, 1, 0)$
$\alpha_f = 0.4$

front



$$\boxed{\begin{aligned} \mathbf{c} &= \alpha_f \mathbf{c}_f + (1-\alpha_f)\alpha_b \mathbf{c}_b \\ \underline{\alpha} &= \alpha_f + (1-\alpha_f)\alpha_b \end{aligned}}$$

$$\mathbf{c} = 0.4\begin{pmatrix}0\\1\\0\end{pmatrix} + \underline{(1-0.4)\times 0.9}\begin{pmatrix}1\\0\\0\end{pmatrix} = \begin{pmatrix}0.54\\0.4\\0\end{pmatrix}$$

$$\alpha = 0.94$$

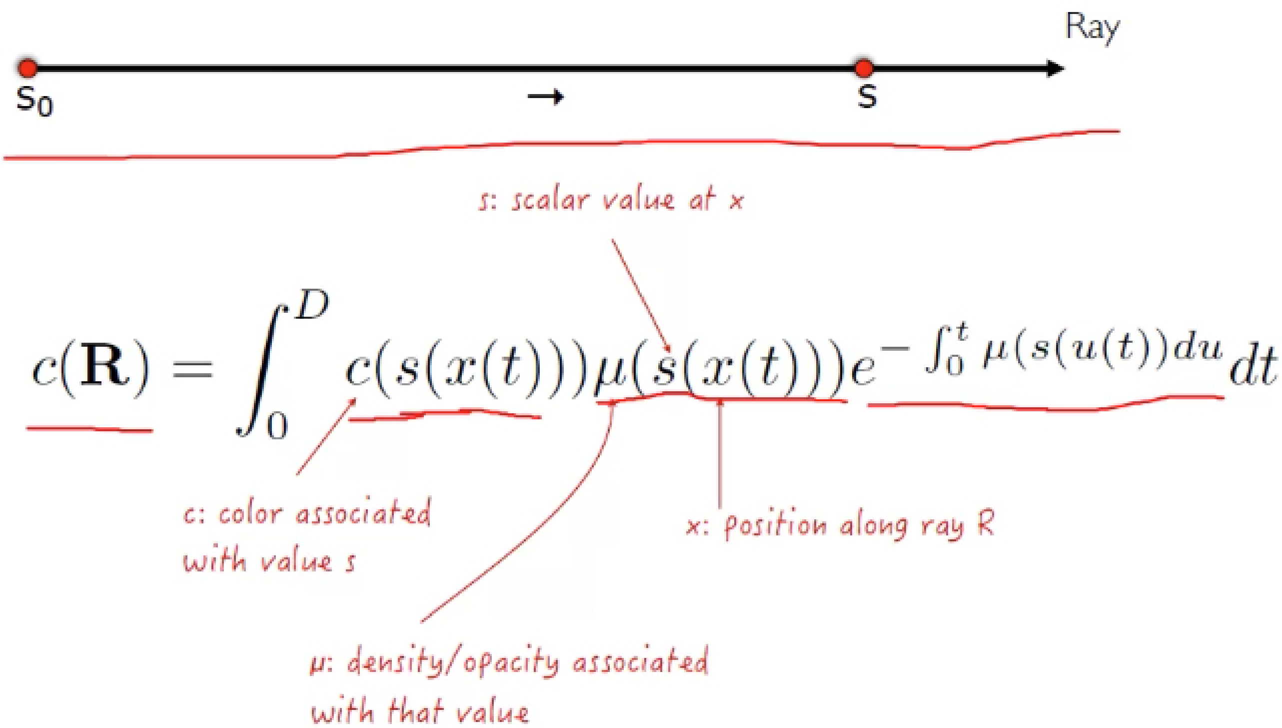
$\mathbf{c}_b = (1, 0, 0)$
$\alpha_b = 0.9$

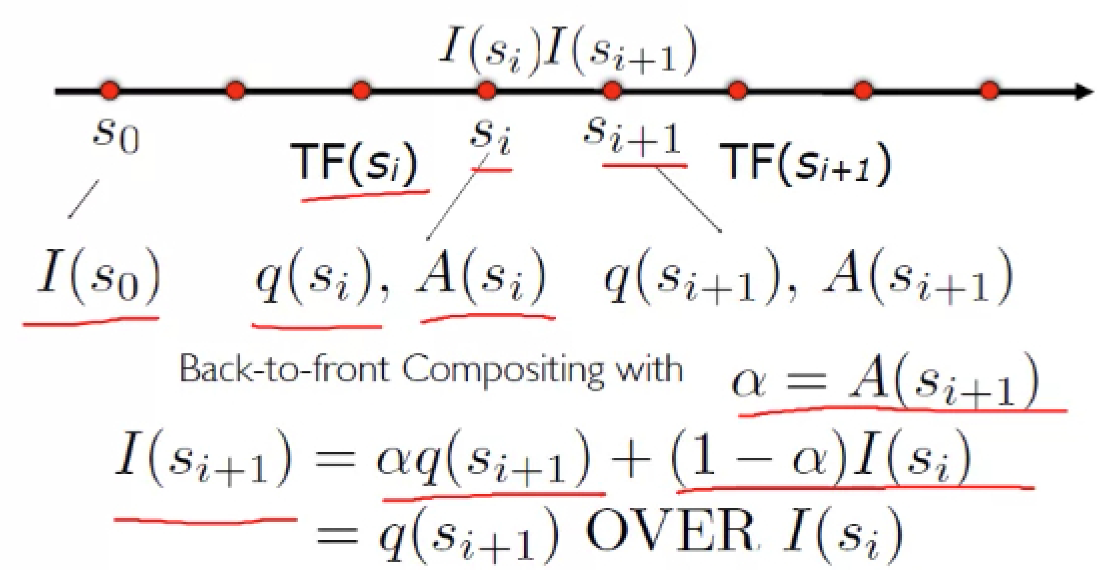
back

Order matters.

12) Volume rendering using ray-casting

Week 6-Ray casting-2:00





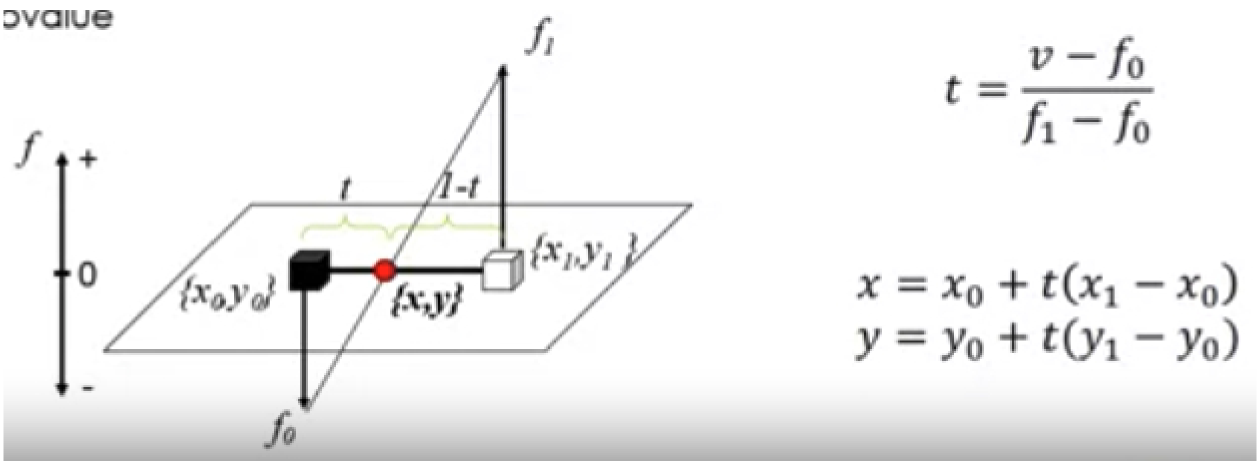
I is total illumination, q is the active emission of light at a point. TF is the transfer function, A is the opacity value generated by TF.

13) Marching squares and cubes

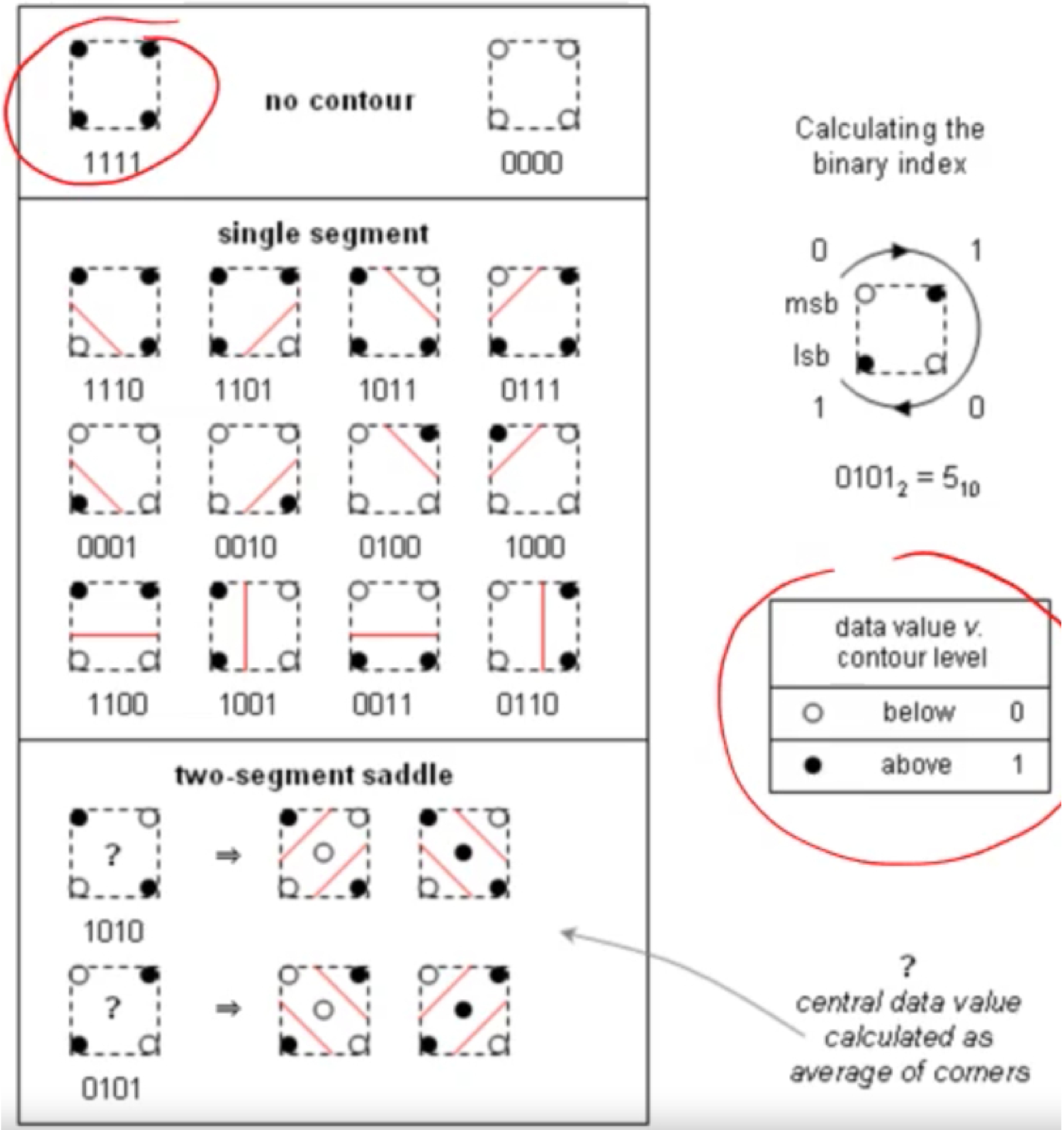
Marching squares: (Week 3-Marching Squares-9:00)

For creating contour line vertices (x,y)

- assume the underlying, continuous functions is **linear** on the grid edge
- linear interpolation
- v is the isovalue



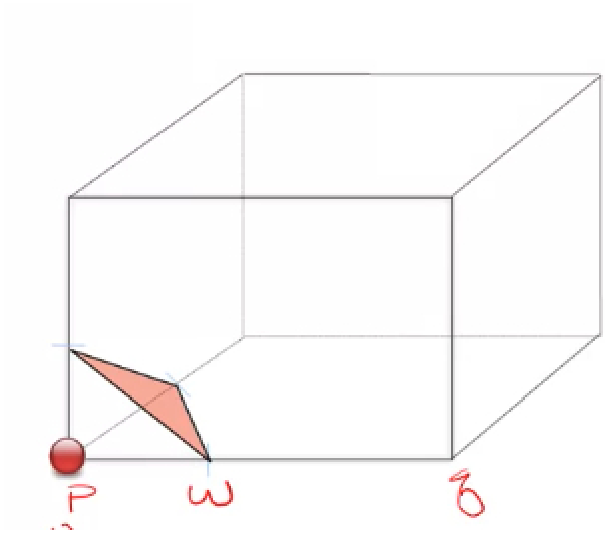
Encode the state of cell vertices in a **4-bit** id.



Marching cubes: (Week 5-marching cubes: algorithm)

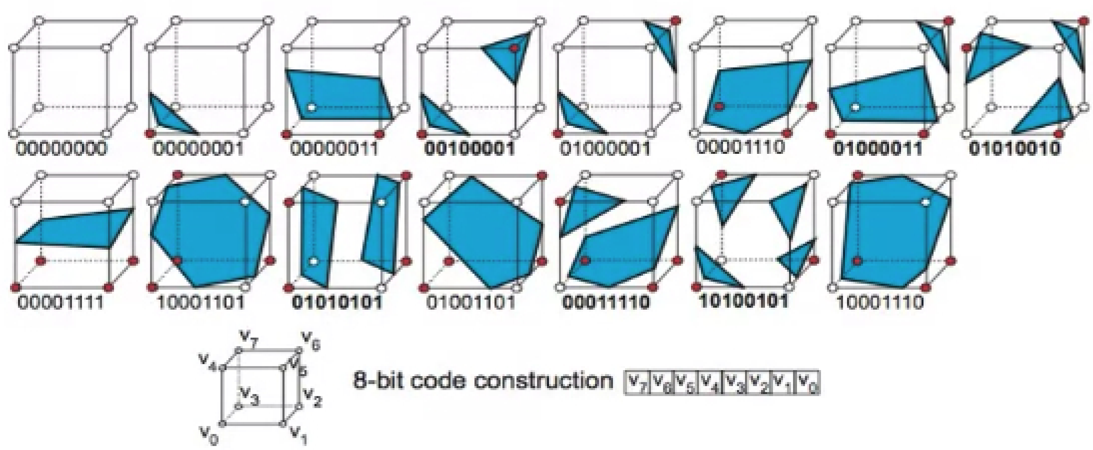
bipolar edges: edge with two differently classified endpoints

place polygon vertices on the edges, $w = (1-t)p + tq$, where $t = \dfrac{f_0 - s_p}{s_q - s_p}$



8 bits to represent cube configuration:

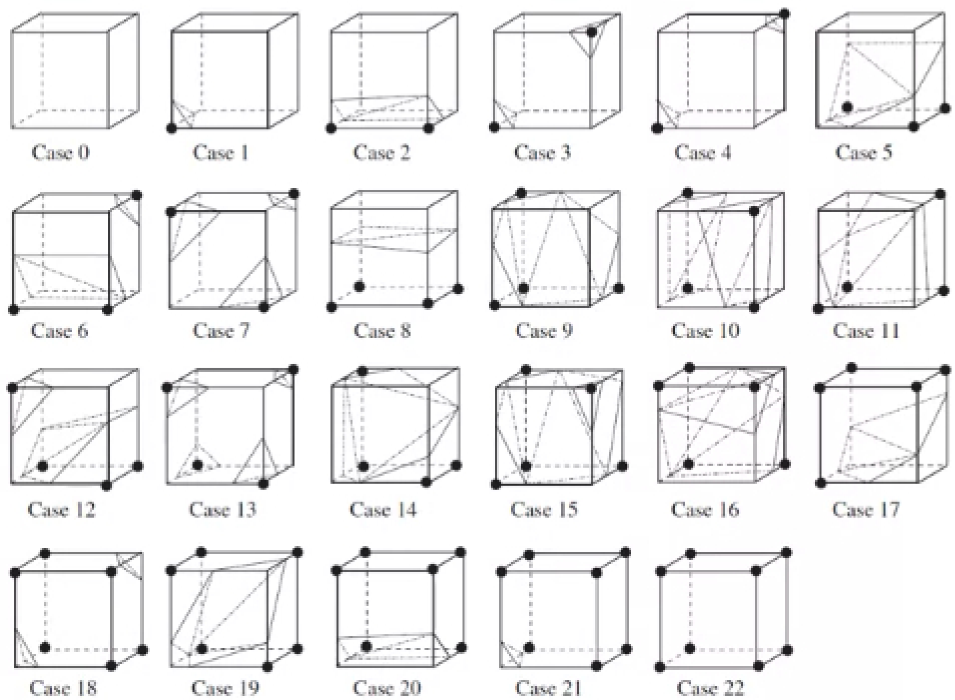
if using **complementary** (swap positive and negative) and **rotational symmetry**, there are 15 cases.



Steps:

- classify vertices of a cube and generate bitcode
- read isosurface lookup table using bit code
- retrieve triangles
- compute vertex coordinates using linear interpolation
- store triangles

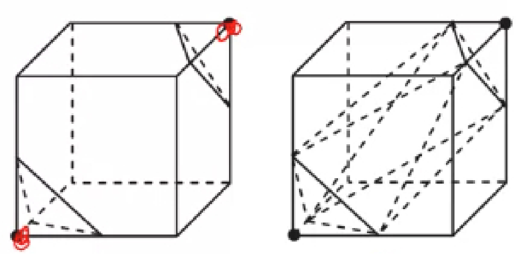
To keep consistency (face ambiguity) and correctness, use rotational symmetry only (23 cases).



internal ambiguity:

Internal ambiguity does not cause any topological inconsistency but it can yield an incorrect isosurface

Internal ambiguity can arise in cases 4, 6, 7, 10, 12, and 13.



There are many, many ways people have addressed both kinds of ambiguity…we'll just discuss one

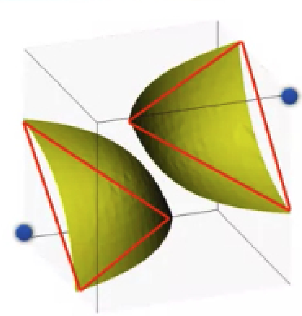
To determine correctness, use trilinear interpolation.

$$
\begin{aligned}
T(x,y,z) &= (1-x)(1-y)(1-z)T(0,0,0) \\
&\quad + (1-x)(1-y)zT(0,0,1) \\
&\quad + (1-x)y(1-z)T(0,1,0) + (1-x)yzT(0,1,1) \\
&\quad + x(1-y)(1-z)T(1,0,0) + x(1-y)zT(1,0,1) \\
&\quad + xy(1-z)T(1,1,0) + xyz(1,1,1) \\
&= \left(\frac{x_1-x}{x_1-x_0}\right)\left(\frac{y_1-y}{y_1-y_0}\right)\left(\frac{z_1-z}{z_1-z_0}\right)T(x_0,y_0,z_0) \\
&\quad + \left(\frac{x_1-x}{x_1-x_0}\right)\left(\frac{y_1-y}{y_1-y_0}\right)\left(\frac{z-z_0}{z_1-z_0}\right)T(x_0,y_0,z_1) \\
&\quad + \left(\frac{x_1-x}{x_1-x_0}\right)\left(\frac{y-y_0}{y_1-y_0}\right)\left(\frac{z_1-z}{z_1-z_0}\right)T(x_0,y_1,z_0) \\
&\quad\quad \vdots \\
&\quad + \left(\frac{x-x_0}{x_1-x_0}\right)\left(\frac{y-y_0}{y_1-y_0}\right)\left(\frac{z-z_0}{z_1-z_0}\right)T(x_1,y_1,z_1).
\end{aligned}
$$

True isosurface of a trilinear function is a cubic curved surface

Contours are hyperbola

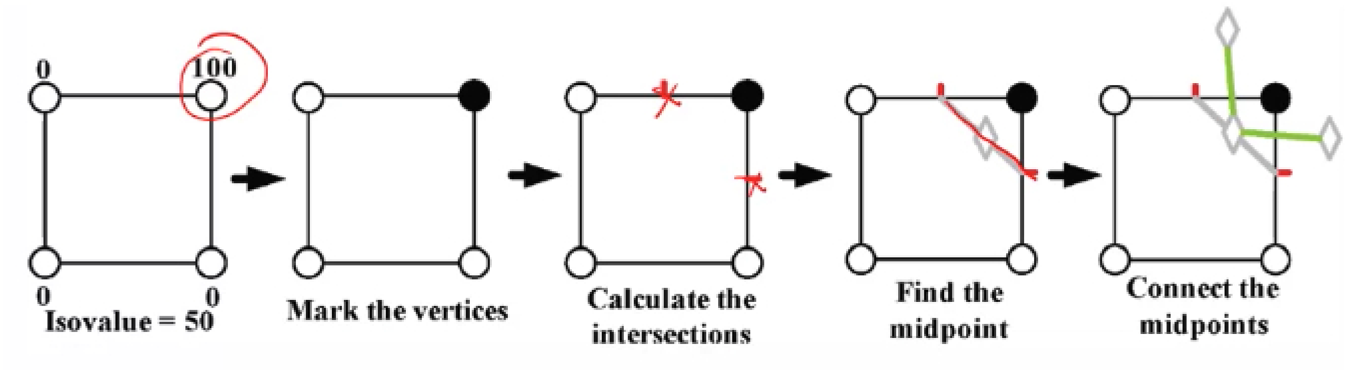
MC approximates with triangles



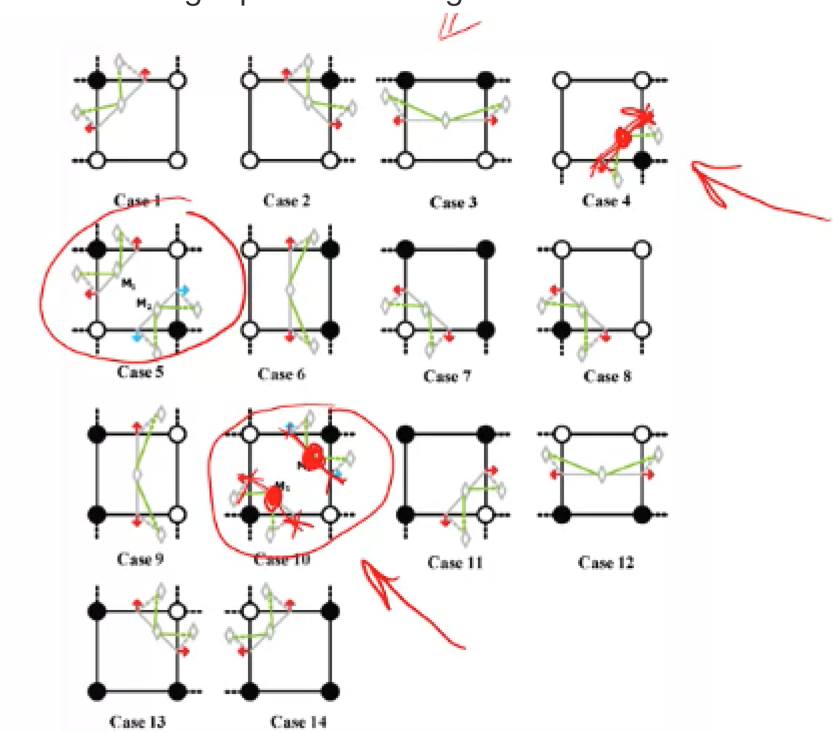
14) Dual marching squares

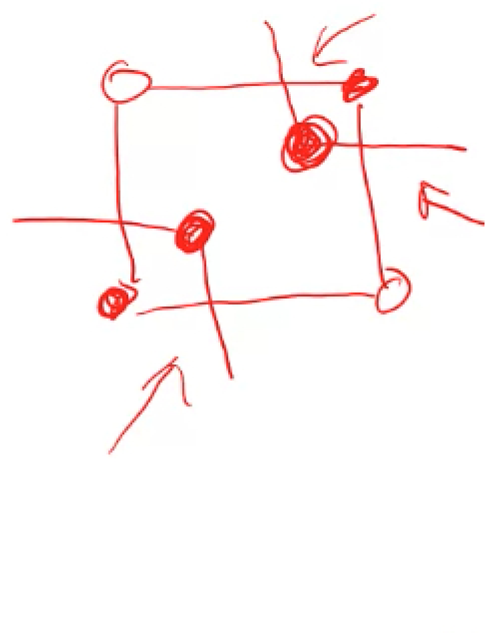
Week 5-Dual marching squares

dual contouring places isosurface vertices inside mesh elements. Isosurfaces vertices in **adjecent elements** are with edges.



dual marching square cell configurations





No ambiguity

15) Scattered data interpolation using RBFs

Week 3-Scattered Data Interpolation-9:00

Radial Basis Function: function dependent on distance from a center is radial.
radial function: $\phi(x,p) = \phi(||x-p||)$
intepolation function: $f(x) \approx \sum_{i=1} N w_i \phi(x, pi)$

Some popular kernel functions

$r = \|x - p\|$

$$\phi(r) = e^{-\lambda r^2}$$ Gaussian

$$\phi(r) = \frac{1}{1+r^2}$$ Inverse distance

compute weights:

$$f(p_j) = \sum_{i=1}^{N} w_i \phi(p_j, p_i)$$

$$Aw = p$$

$$A = \begin{bmatrix} \phi(p_1, p_1) & \cdots & \phi(p_1, p_N) \\ \cdots & \cdots & \cdots \\ \phi(p_N, p_1) & \cdots & \phi(p_N, p_N) \end{bmatrix}$$

$$w = \begin{bmatrix} w_1 \\ \cdots \\ w_N \end{bmatrix}$$

$$p = \begin{bmatrix} f(p_1) \\ \cdots \\ f(p_N) \end{bmatrix}$$

$$w = A^{-1}p$$