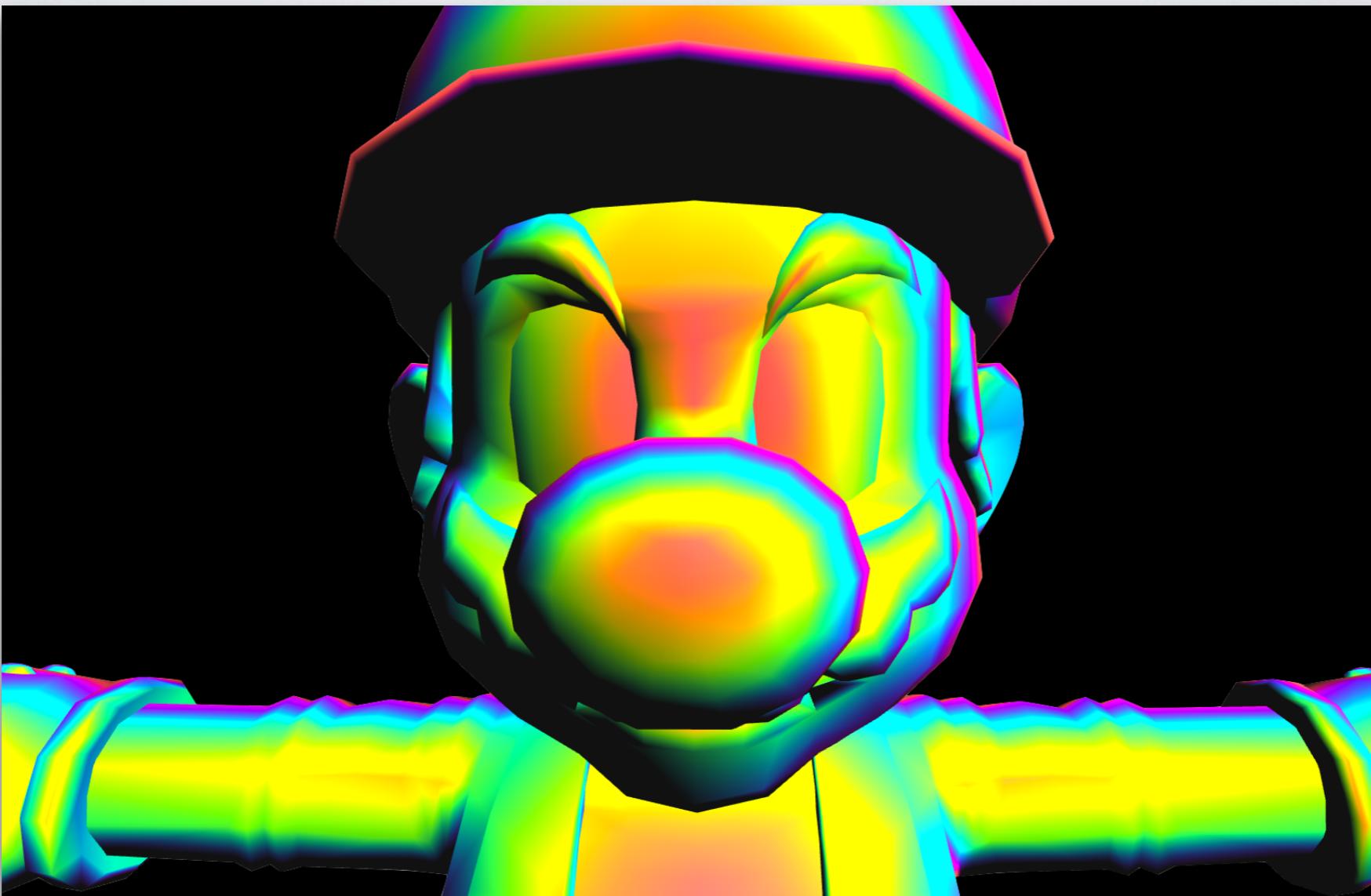


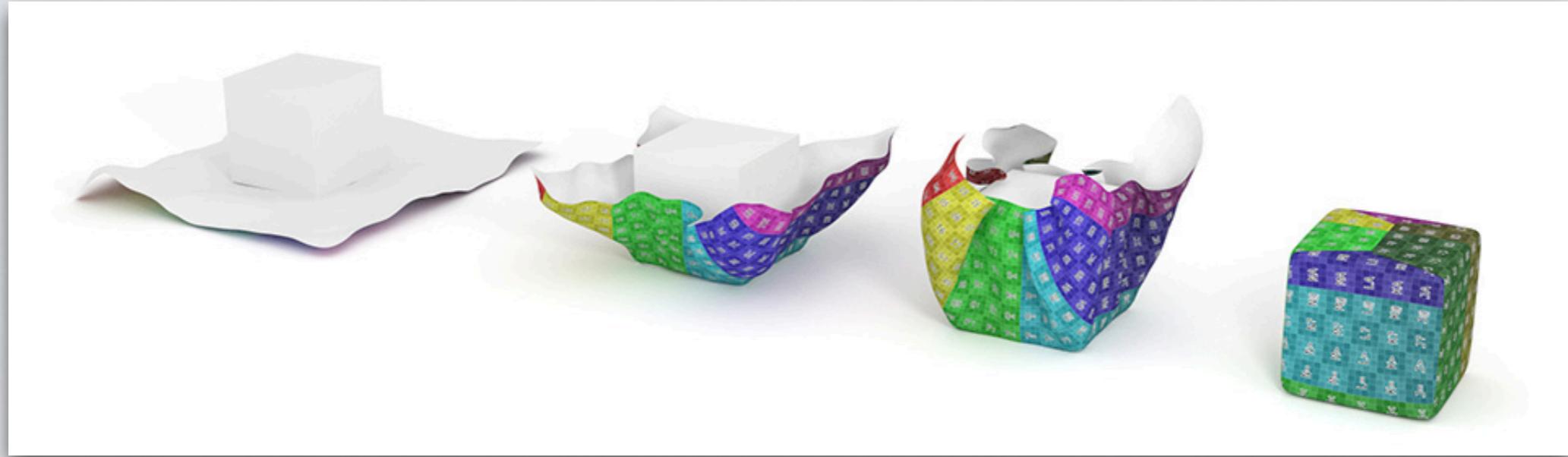
COLOR

Tips and tricks for the artistic mapping of color values



University of Pennsylvania - CIS 700 Procedural Graphics
Rachel Hwang

MAPPING PROBLEMS



Chocofur ([source](#))

- Graphics is full of mapping problems. For example:
 - Geometry vertices to transformed positions
 - Then transformations through time as animation data
 - UV coordinates to texture coordinates
- Often we have some dataset, manually or procedurally specified
 - Then we use some mapping function to determine extra attributes, like normals or color
 - “Procedural graphics” is misleading — much of graphics is “procedural”! Proceduralism is fuzzy.

COLOR IS POWERFUL

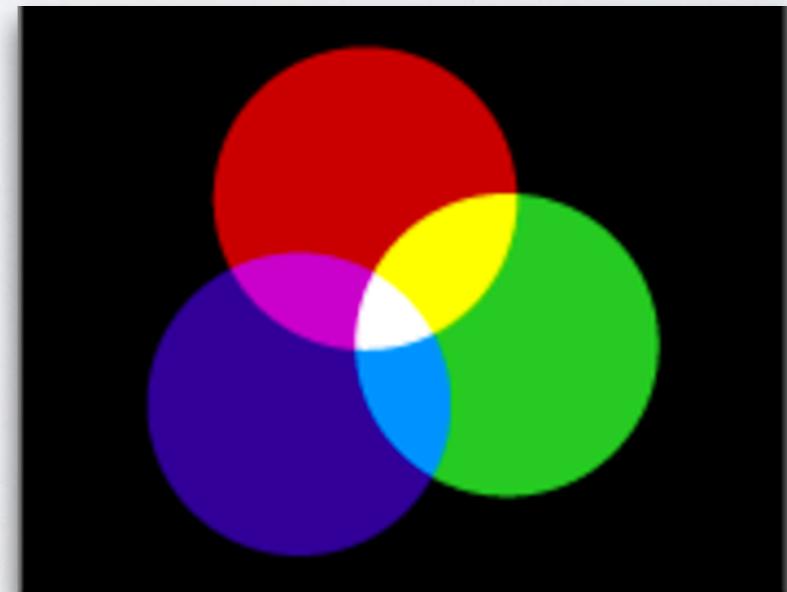
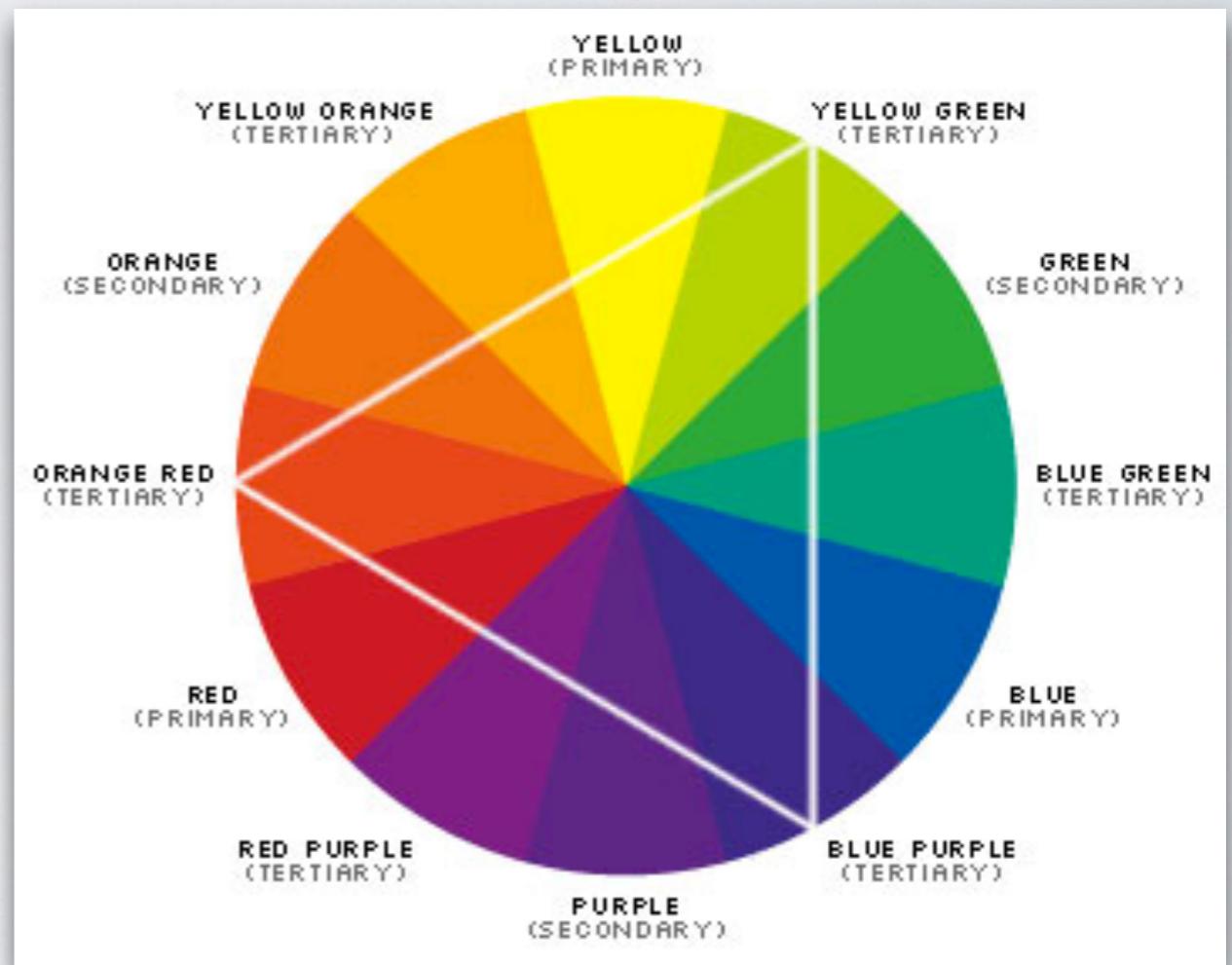


Leonid Afremov ([source](#))

- Influences human perception tremendously
- In CG, we're just creating a sequence of pixel colors to put on-screen based on some data.

SUPER BRIEF COLOR THEORY

- Colors: primary, secondary, tertiary composition
- Pretty color palettes usually consist of colors with some logical relationship. Eg.
 - Complementary - opposing colors
 - Analogous - adjacent colors
 - Color Triad - three equidistant colors
 - Generally, limit range of hue, luminance, etc.
 - Adobe color-palette picker
- To us procedural artists, we can treat color as just another dataset
 - 3D values (r, g, b)
 - We can map colors to or from any space we like procedurally!



nyu (source)

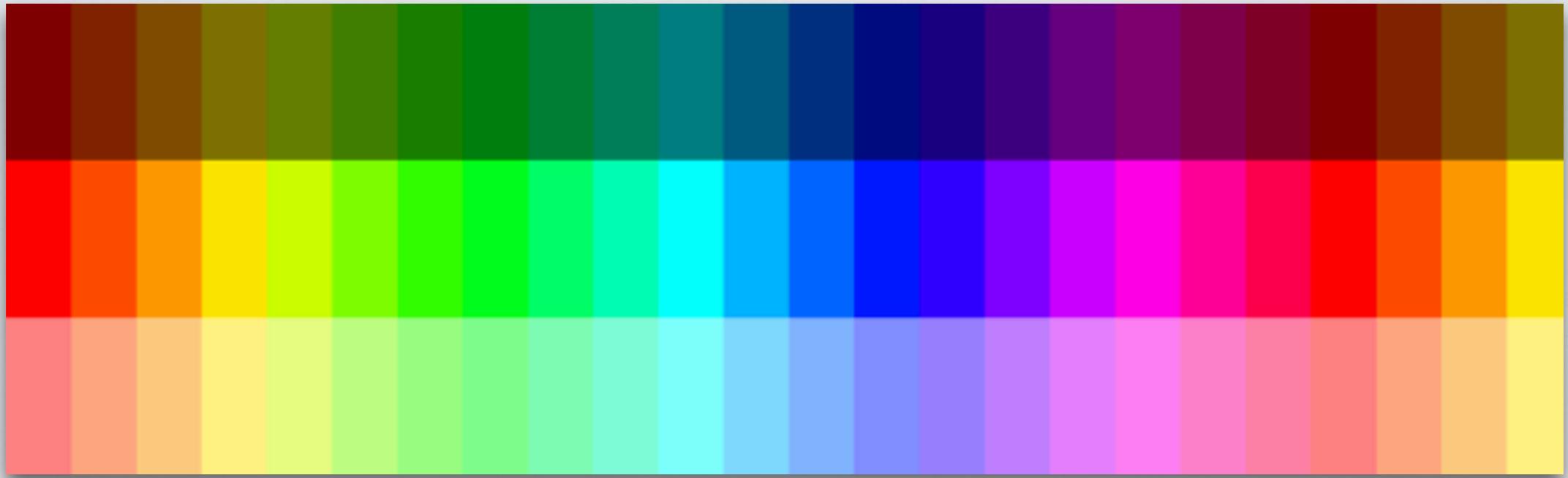
~DISCLAIMER~

The following is a motley assortment of shading and image-processing effects,
not a canonical selection, and not the most efficient algorithms.

Take these as inspiration in developing an intuition for programming visual
effects.

COLOR PALETTE TECHNIQUES

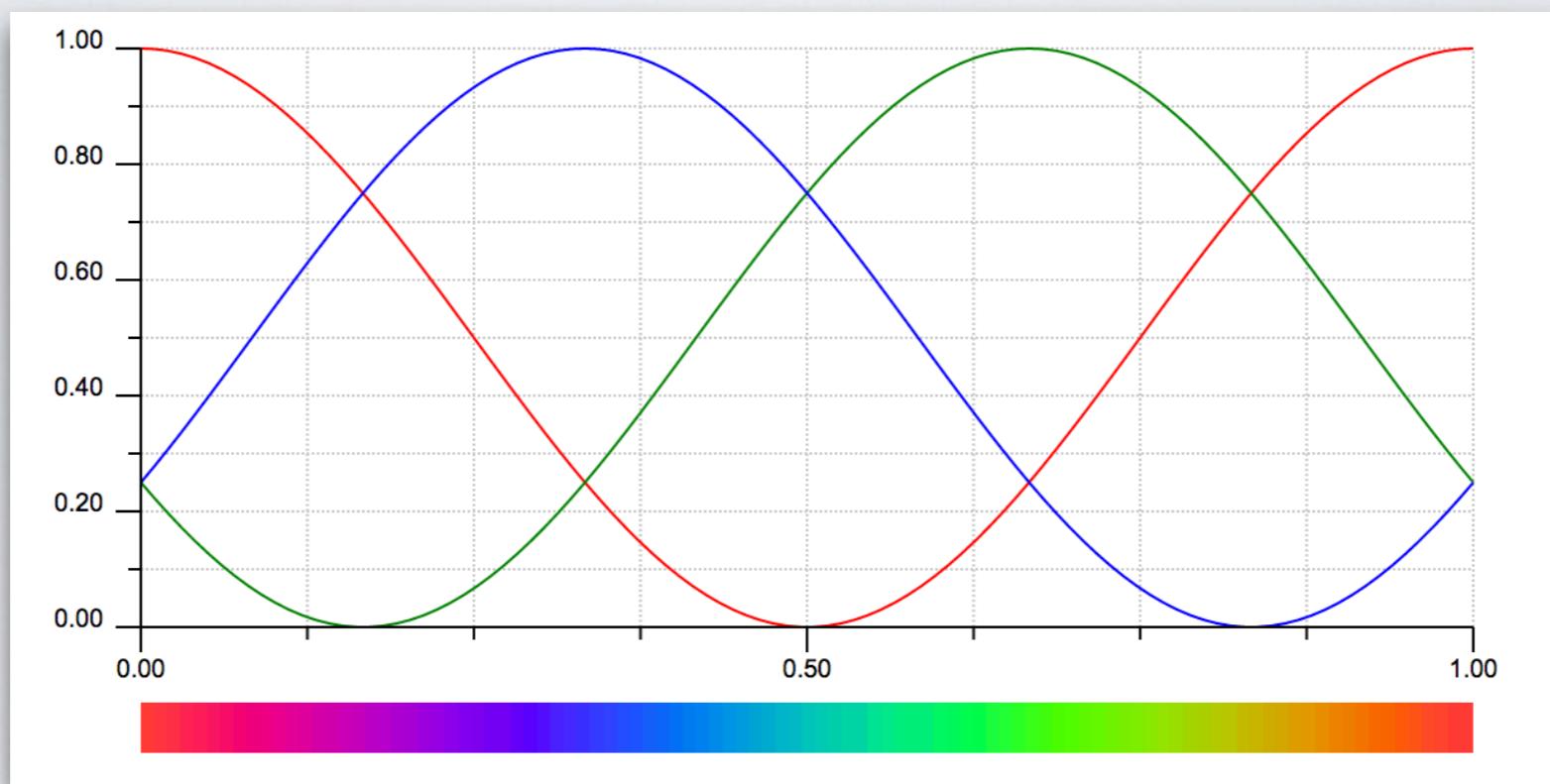
CONSIDERATIONS



Devmag (source)

- Vector distance in color space (rgb, hsv, etc) usually doesn't match difference in perception.
- Digital color representation is imperfect and cannot capture all natural colors.
- Convenient to generate 1D or 2D color palettes to create color variation as we vary some parameter(s). (Like with noise values!)
 - Vary color across an object that would have a constant color
 - Apply color to things that have no color defined for them, (eg. originally greyscale)

COSINE COLOR PALETTES



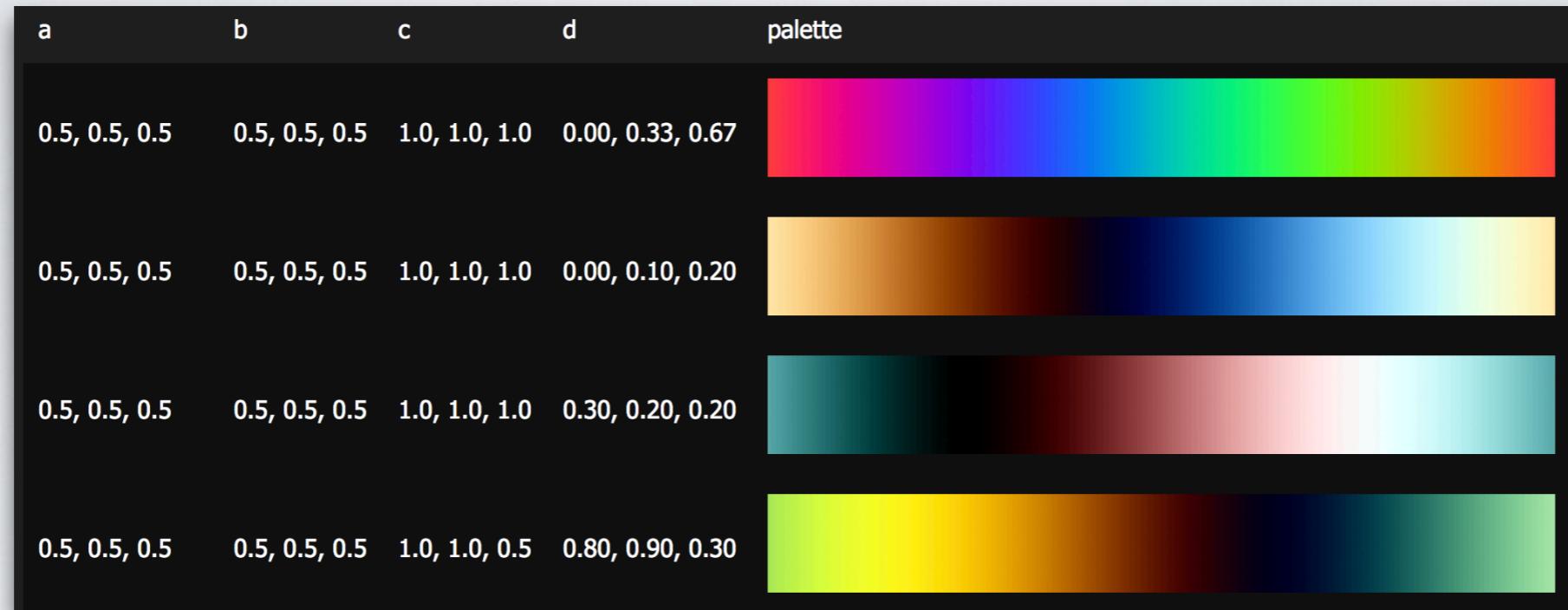
- Observation: Varying rob component(s) smoothly will result in a smooth change in color.
- IQs idea: modulate color value using cosine functions!

QUICK COSINE REVIEW

$$\text{color}(t) = a + b * \cos(2 \pi(c * t + d))$$

- These are the four knobs with which we can control our cosine function
 - **a** : shifts the entire curve up or down along the y-axis
 - **b** : scales the amplitude. Increase makes the wave taller; decrease for shorter
 - **c** : scales the frequency. Increase makes the cycle faster; decrease for slower
 - **d** : shifts the entire curve left or right along the x-axis
- Try for yourself here.

COSINE COLOR PALETTES



- For vec3 output, we use vec3 input parameters. [Try it here.](#)
- Using parameterized cosine waves for each channel, we can smoothly move between various colors.

- **A** = [0.50, 0.50, 0.50]
- **B** = [0.50, 0.50, 0.50]
- **C** = [1.00, 1.00, 1.00]
- **D** = [0.00, 0.33, 0.67]

Parameters for red curve

Parameters for blue curve

Parameters for green curve

INSTAGRAM-Y FILTERS



Photodoto ([source](#))

Original

“Brannan” filter

- Effects like Instagram filters are just more color-remapping effects!
- Brannan filter: increase contrast + brightness, blend with yellow color, remap full value range to exclude darkest values.

INSTAGRAM-Y FILTERS

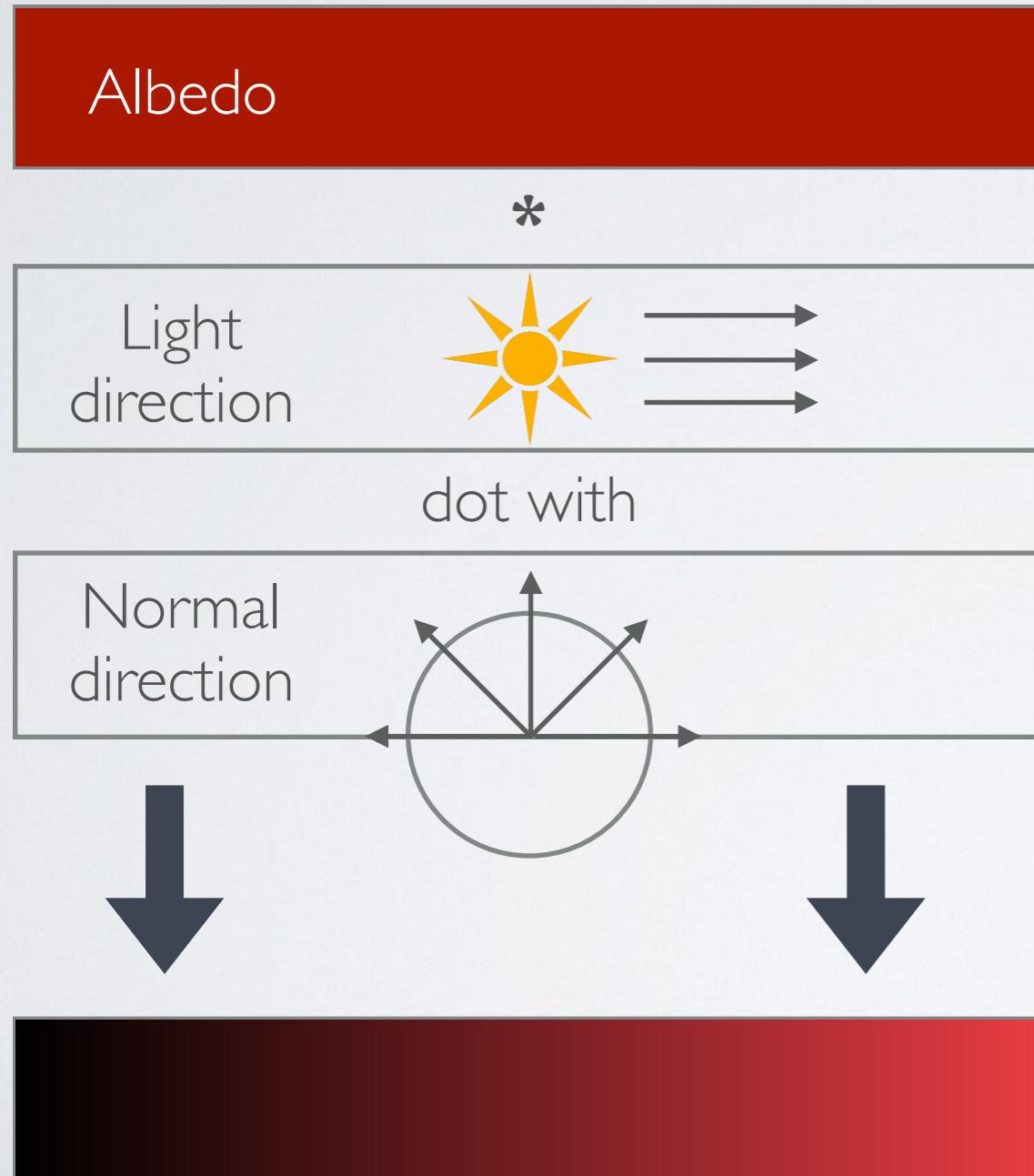


- Software like Photoshop provide handy visualization tools for color remapping
- Above: increasing contrast. Pinch the curve so the middle tones are less represented

SHADING IDEAS

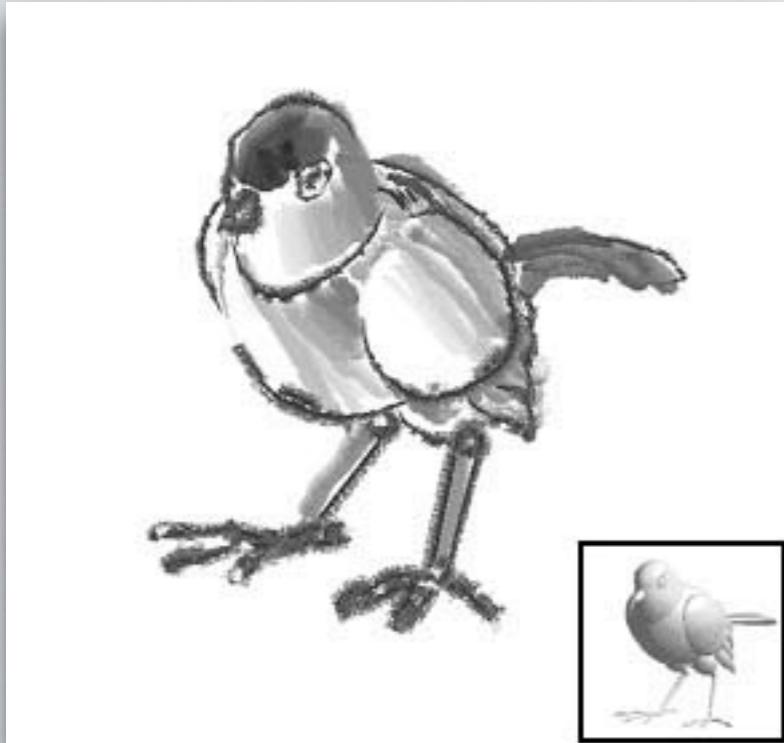
THE RENDERING STATUS QUO

Lambert shading

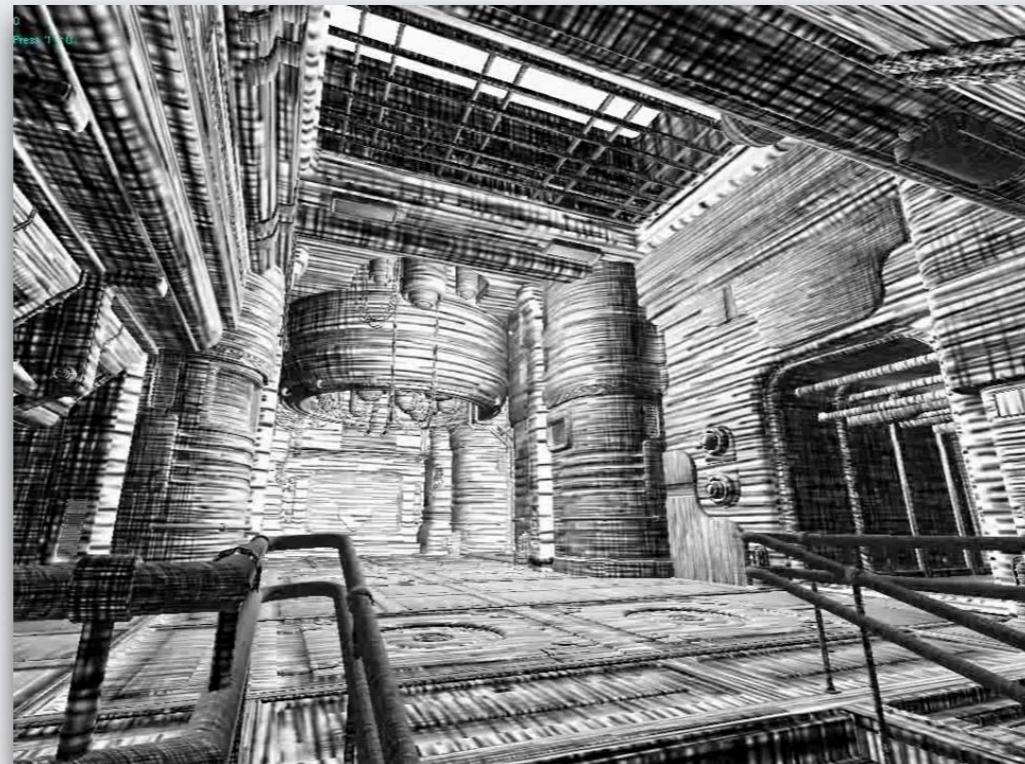


- Rendering is already full of procedural shading!
- We just usually try to mimic the physical properties of light for realism eg. light diffusion
- Lambert shading is just a procedural formula: scale base color by $\text{dot}(\text{normal}, \text{vecToLight})$
- But, why stick to physically-based shading models?

NON-PHOTOREALISTIC RENDERING (NPR)



Ming OuhYoung ([source](#))



Franz Peschel ([source](#))



Cmu ([source](#))

- With some creative programming, we can mimic many artistic styles
- Much stylization is defined by shading and coloring...how does a material respond to light?
- [Hatch-shader demo](#), [various NPR effects demo](#)

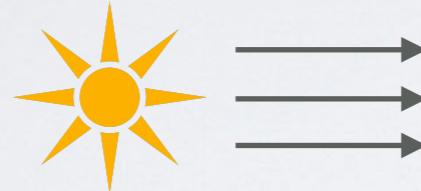
TOON SHADING

cell shading

Albedo (optional)

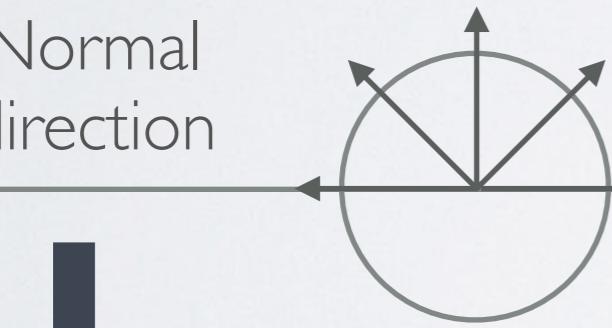
*

Light
direction

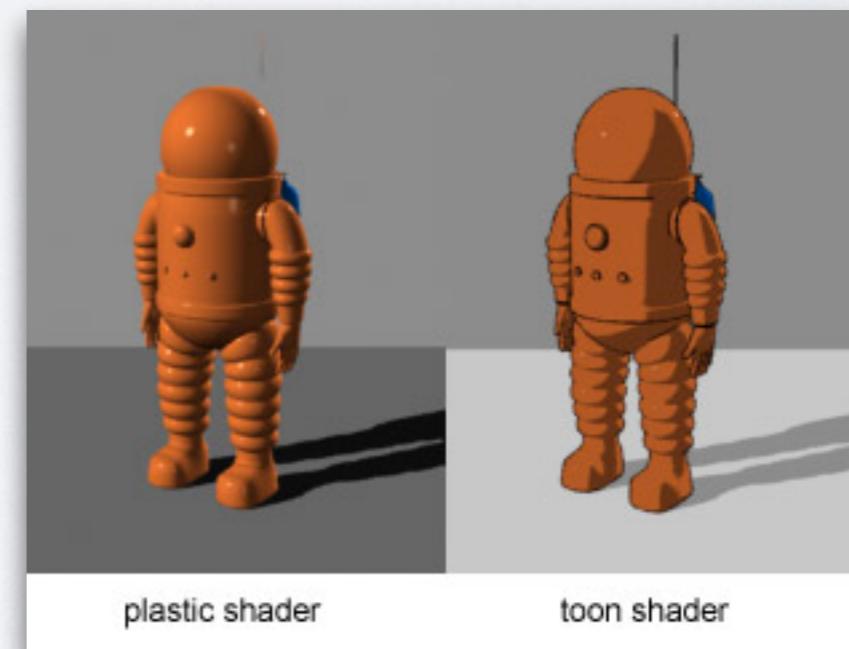


dot with

Normal
direction



- Mimics cell-shading, an artistic simplification
- Similar to lambert shading, but discretizes the output color space
- Output color: procedurally or manually specified
- Can add an outline by coloring black when surface normal is perpendicular to view vector
- [Windwaker demo](#)

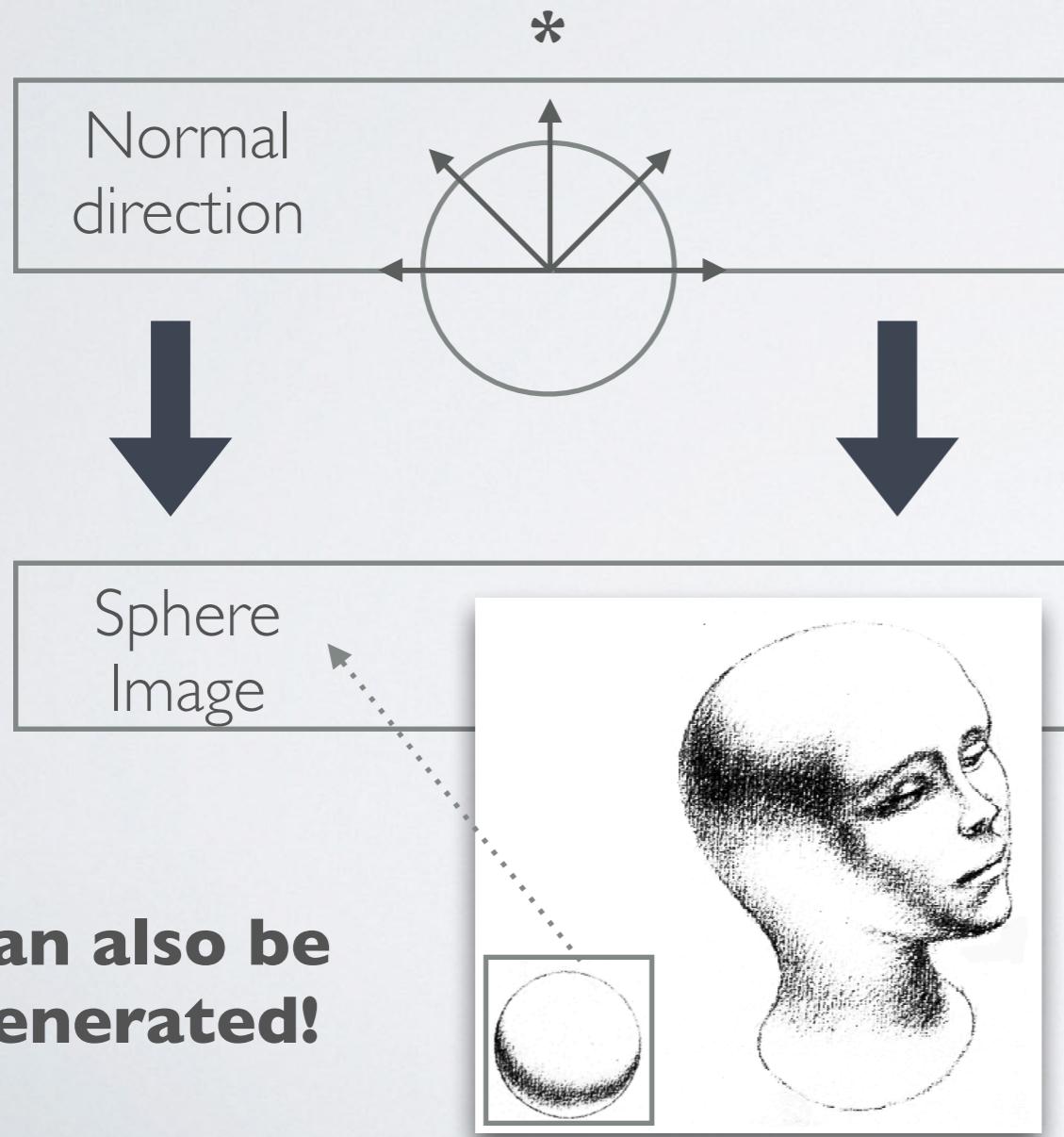


T-Tus (source)

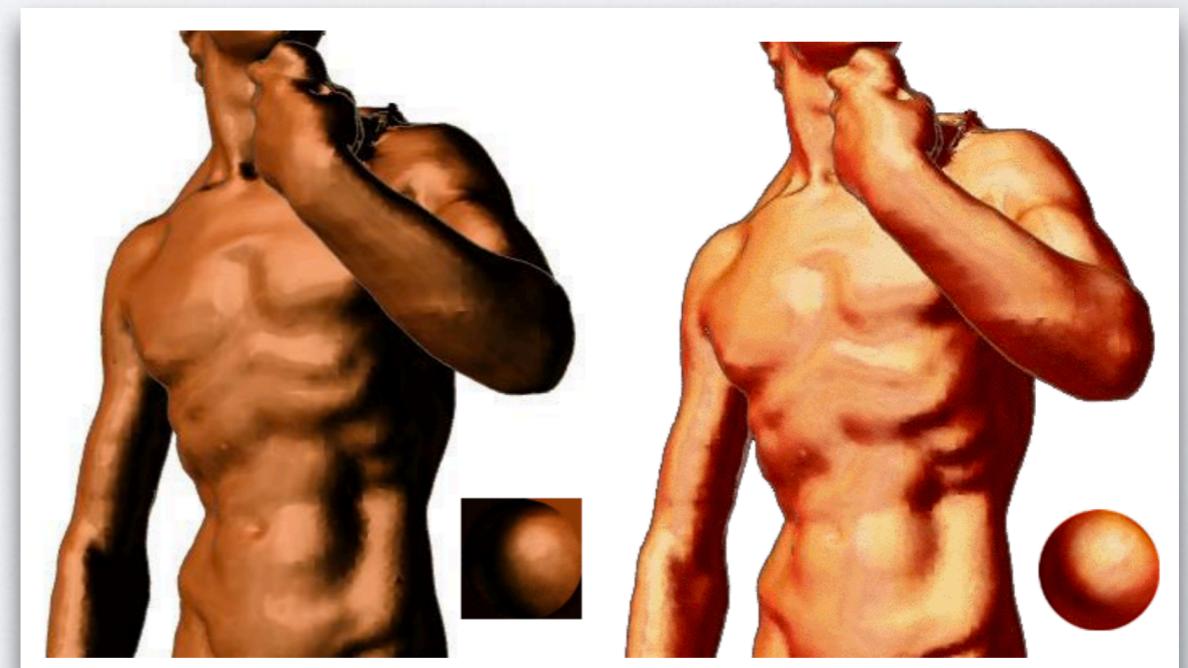
LIT SPHERE SHADING

lit sphere shading

Albedo (optional)



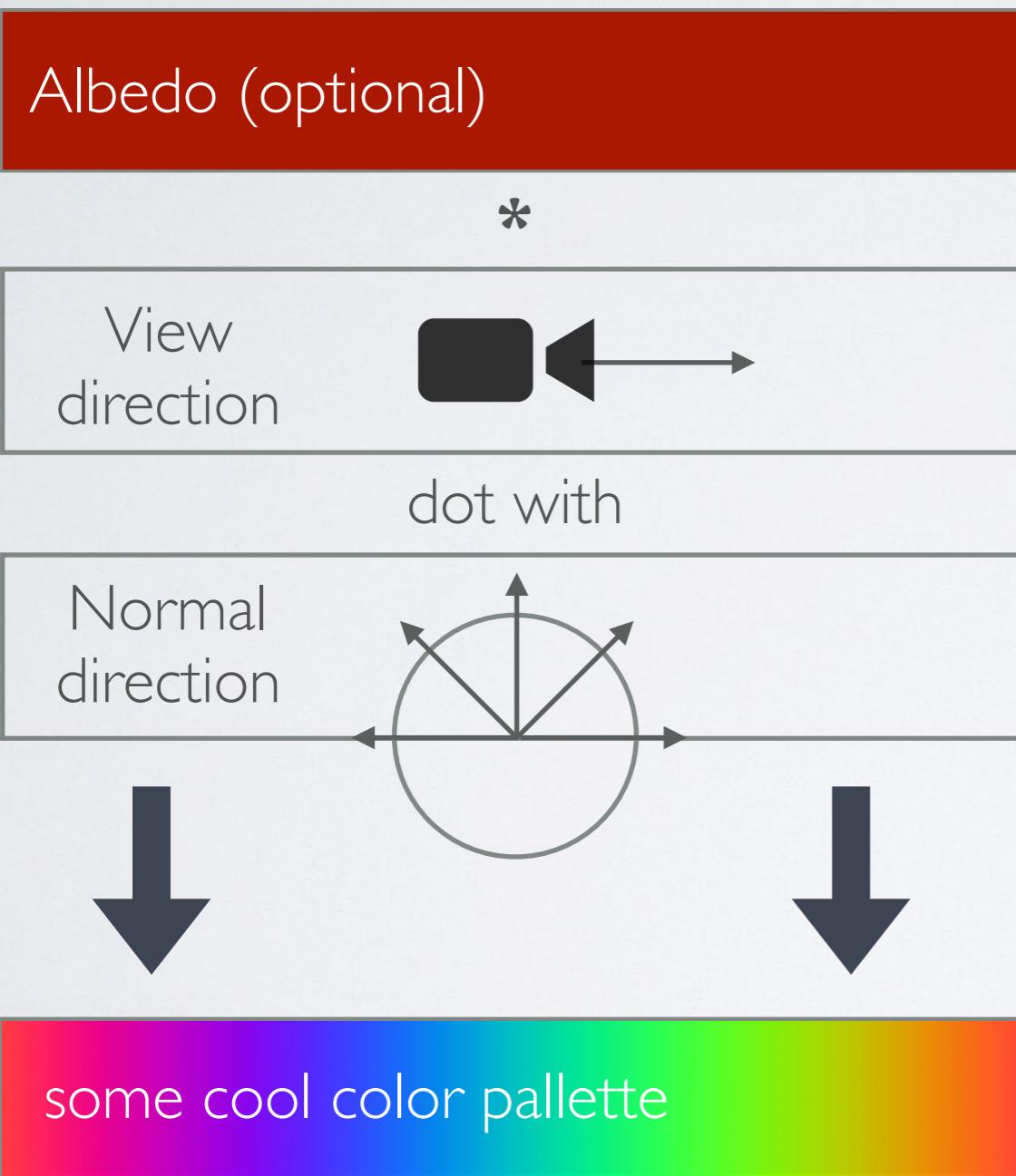
- A sphere covers the full set of unit normals.
- “Fake” reflectance models by reading color from a sphere-image texture rather than computing.
- Map normal directions to image texture of a sphere. (Like normal-mapping backwards!)
- Can encode a huge variety of styles! Allows you to draw your own reflectance model!
- [Lit-sphere demo](#)



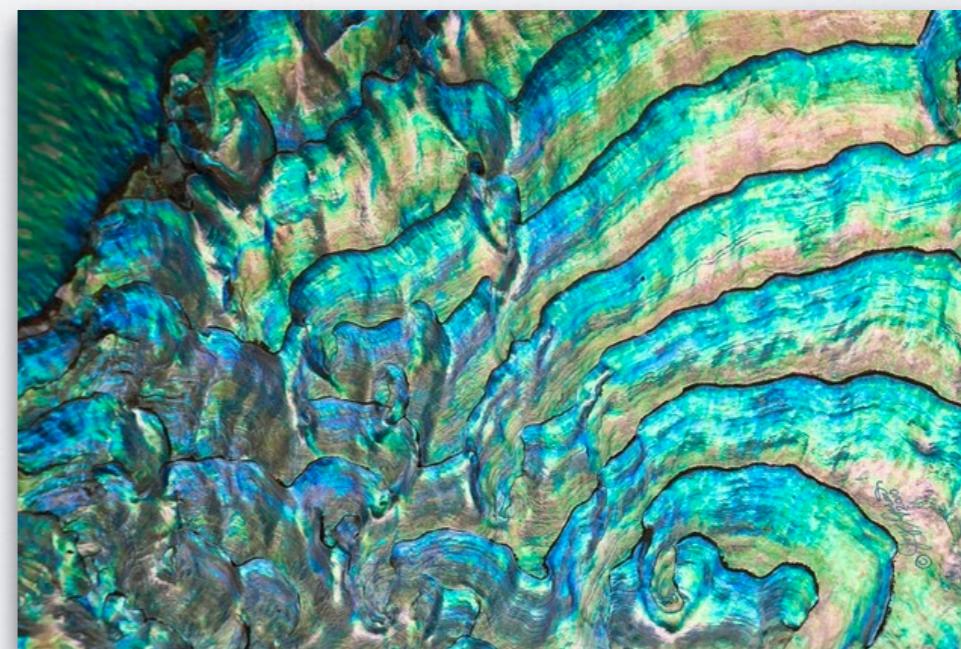
Lit Sphere ([source](#))

IRIDESCENT SHADING

iridescent shading



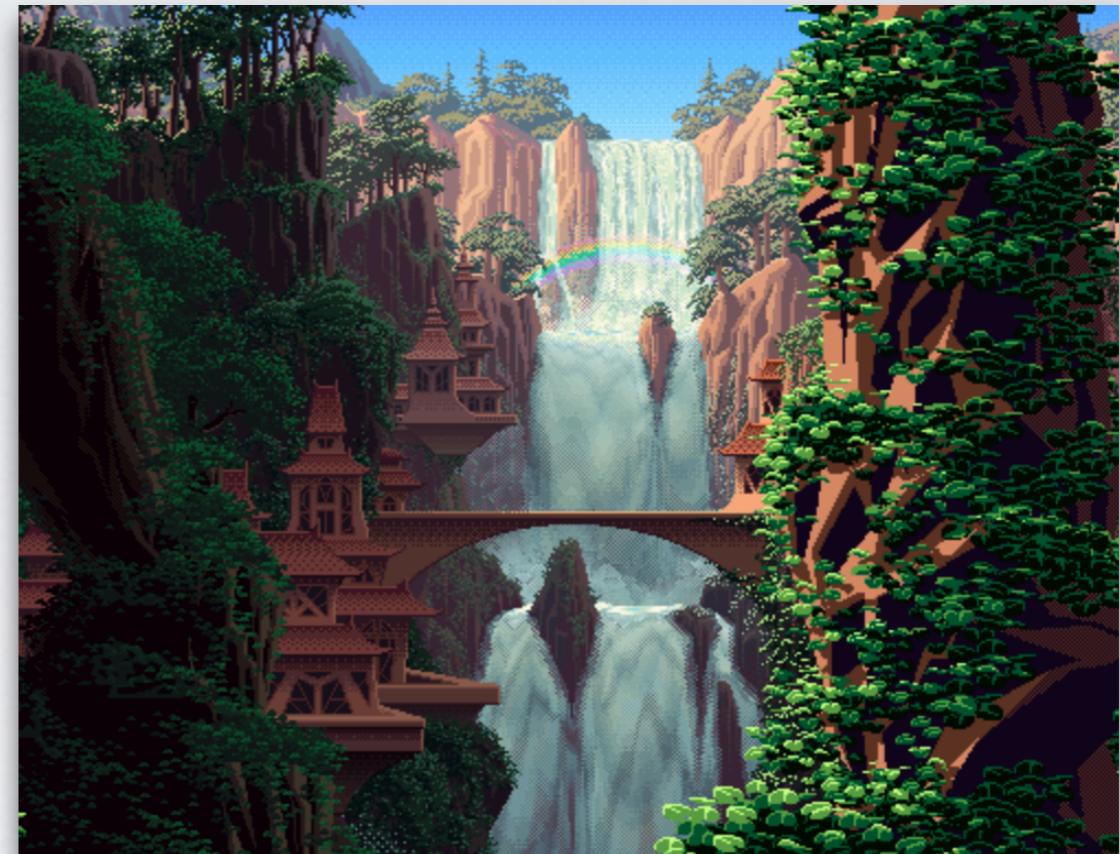
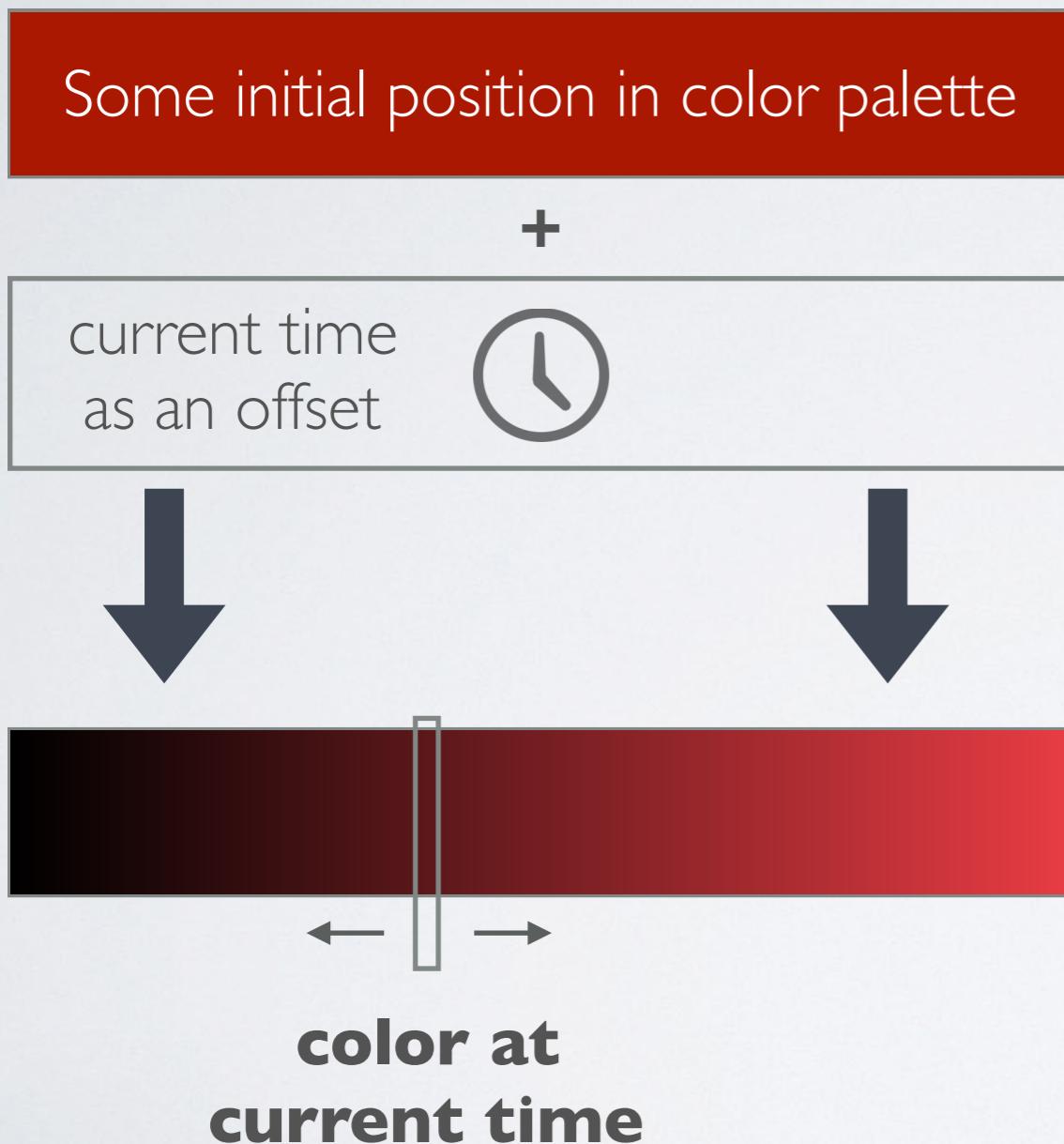
- Color is dependent on the viewing angle, appears to shimmer and change with rotation
- Implementation is similar to lambert, but with the view vector instead of the light vector.
- Looks like this. Ok, physically-based.



ASSORTED POST-PROCESSING EFFECTS

COLOR CYCLING

color cycling

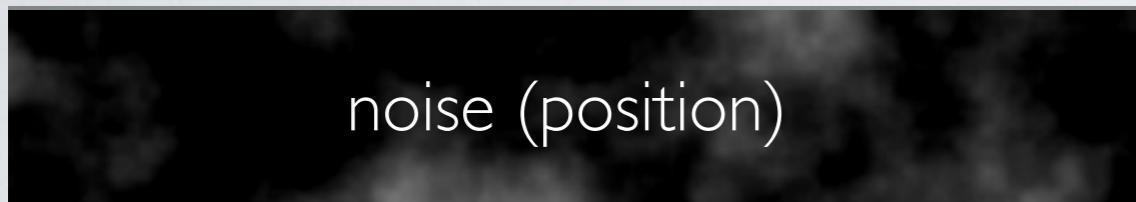


Effectgames ([source](#))

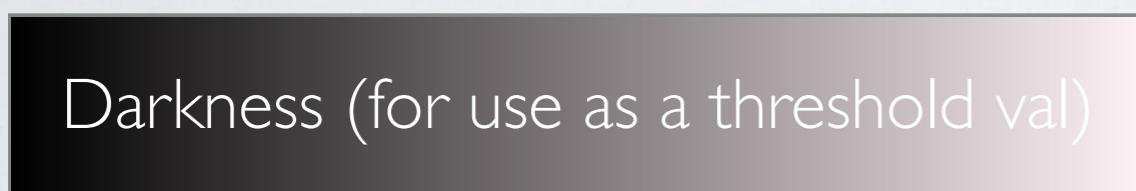
- Aka. “palette shifting”
- [See effect here.](#) (look at options)
- Create cheap animation by offsetting over time into a texture or color map.

POINTILLISM

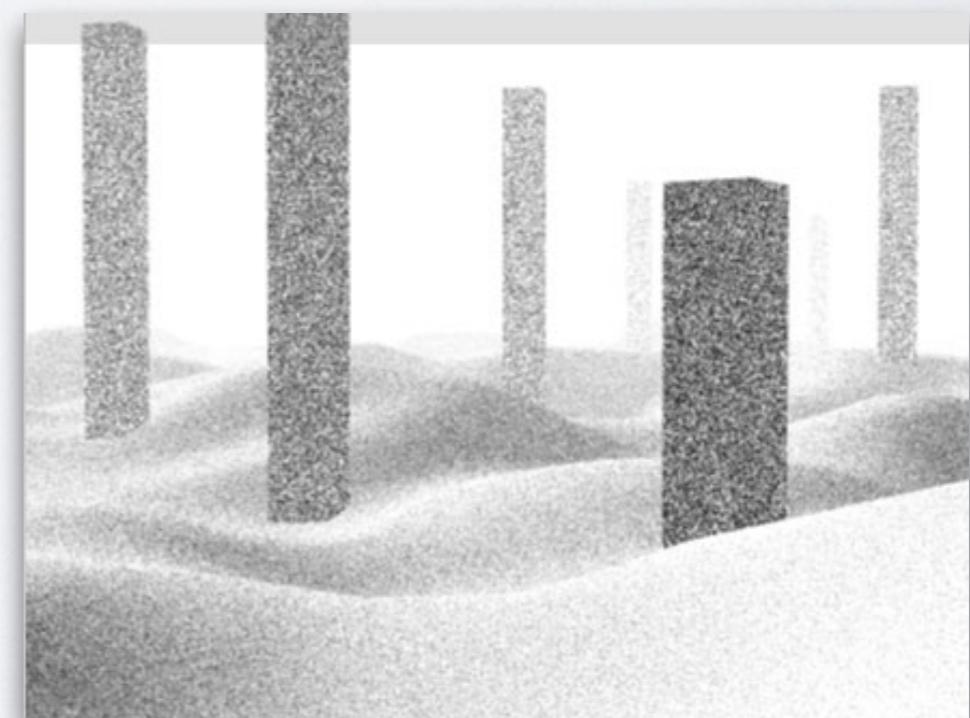
Pointillism shading



Compare to



- The speckled effect — basically represents darkness using density of speckles
- We can approximate this density by applying color with some probability.
- Probability scales with original darkness value. Eg. if the original color was pure black, we color the pixel with 100% probability



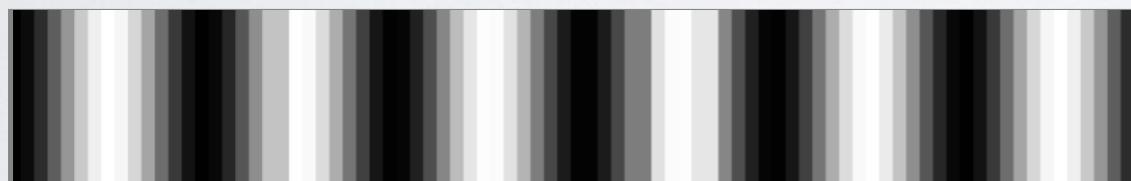
Effect:

HATCHING

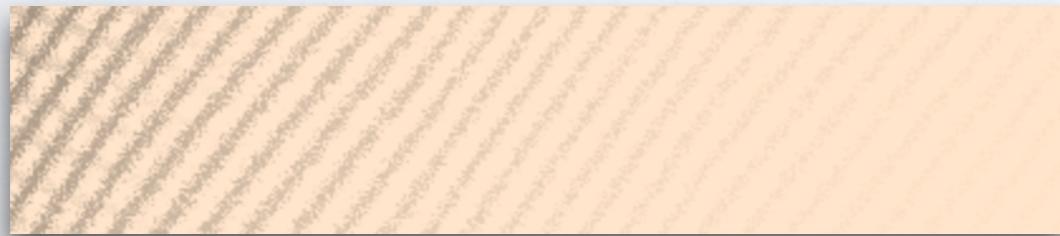
hatching shading



*



Effect:



- Mimics the illustration technique of representing darkness with diagonal lines
- We can use a sine function to make periodic dark lines which look like hatches
- Use both $x + y$ as an input to get a diagonal hatch
- Not perfect — would be better to also vary distance between hatches, but more complex
- [See effect here](#)

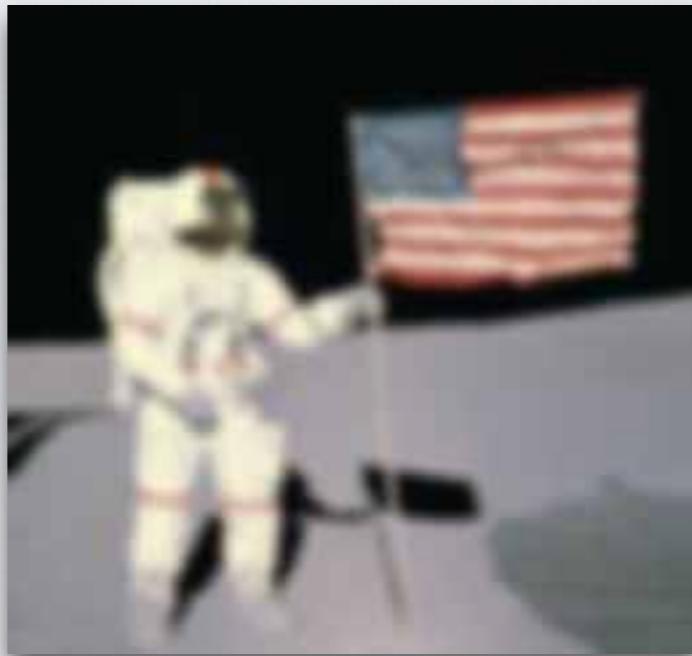


mstworkbooks ([source](#))

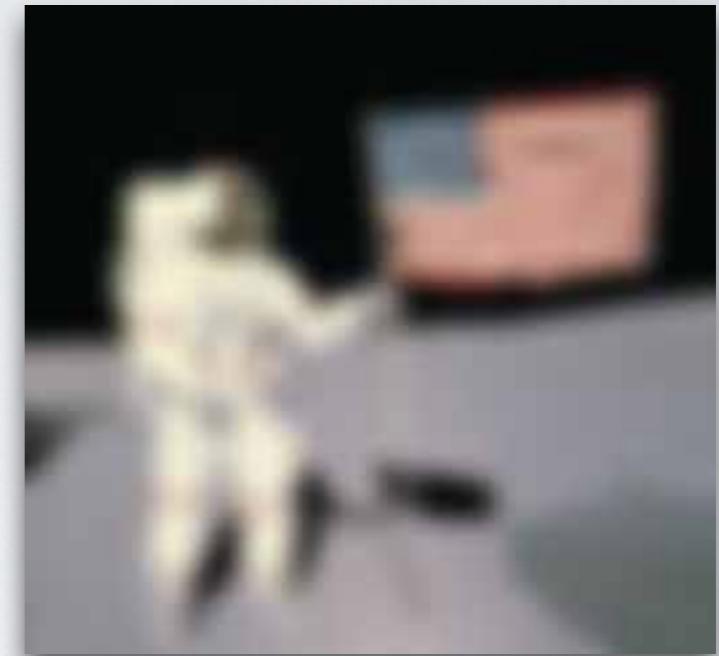
GAUSSIAN BLUR



original



radius 2.5 px

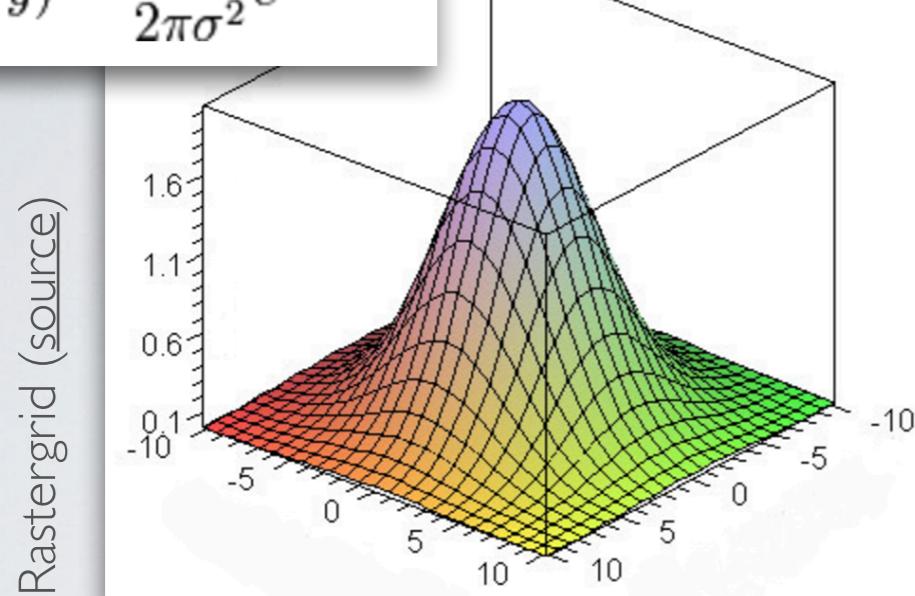


radius 5 px

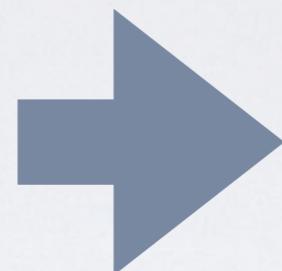
- We can achieve blur/smooth by averaging neighboring pixel values.
 - The more neighbors (larger radius within to consider), the more dramatic the blur effect
 - Closer pixels should contribute more, and further pixels less. We want to scale contribution by distance (like interpolation!)

GAUSSIAN BLUR

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



the (2D) kernel shape

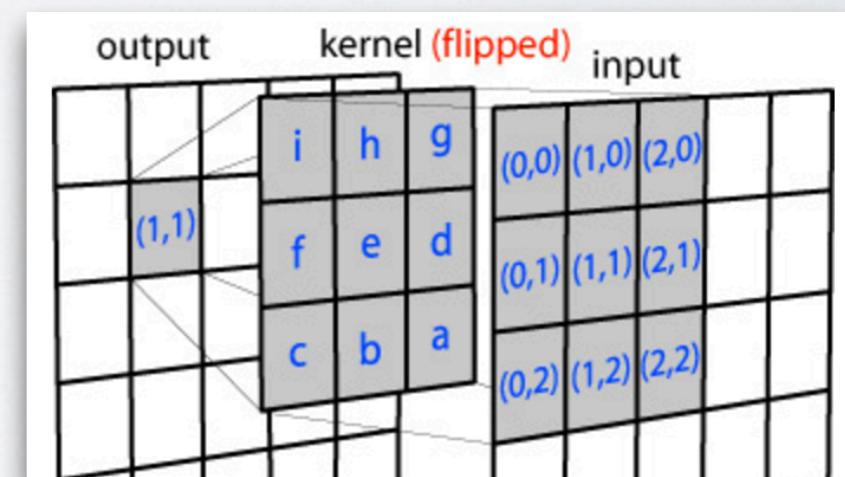


$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

possible pixel weights

- Like much in signal processing, notation makes this look scary, but the concept is simple.
- This gaussian equation describes the weight that cells contribute based on their spatial relationship to the center.
- The Gaussian is a normal distribution — using it to weight neighboring pixel values in averaging does a smooth blend.

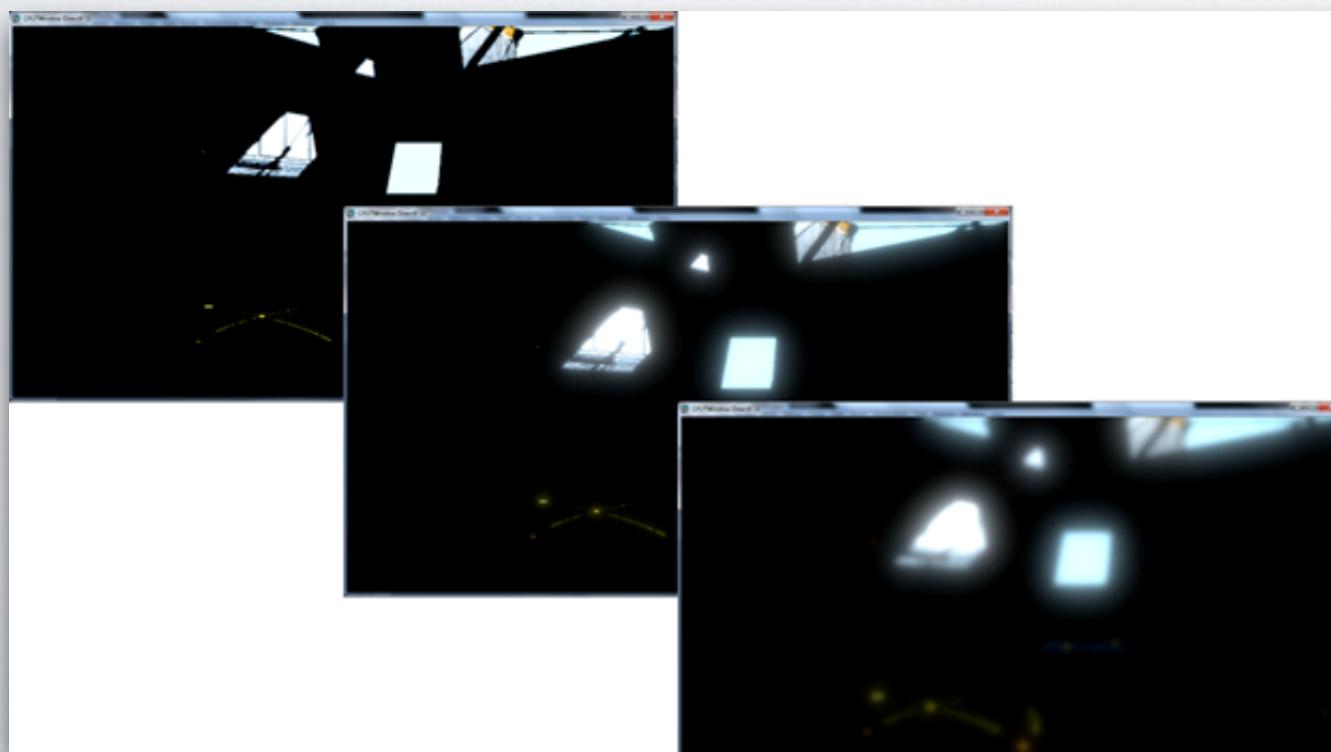


kernel application

BLOOM



gamespot ([source](#))



Intel ([source](#))

- Creates a glow around bright parts of an image.
- First do a “high pass filter”, extract pixels of high luminosity from your image. Store these bright pixels in a texture.
- Apply gaussian blur to the bright pixel image to create fake glow effect
- Blend the blurred bright image with the original image.

WARP EFFECTS



image for $f(p) = f\text{bm}(p)$

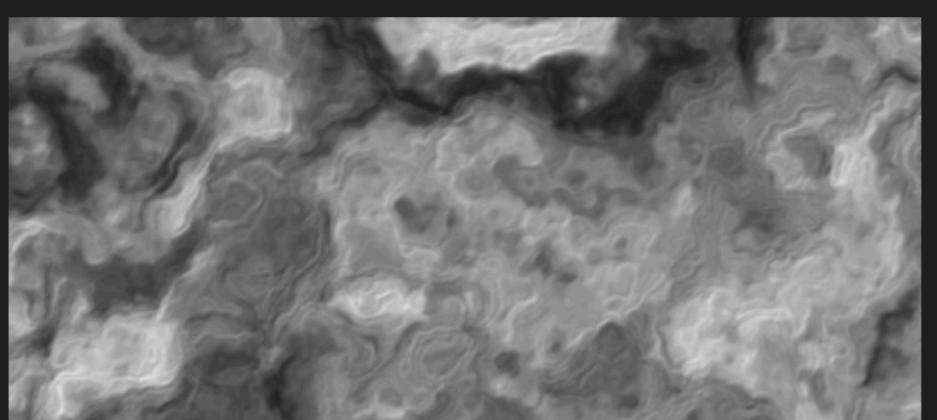


image for $f(p) = f\text{bm}(p + f\text{bm}(p))$

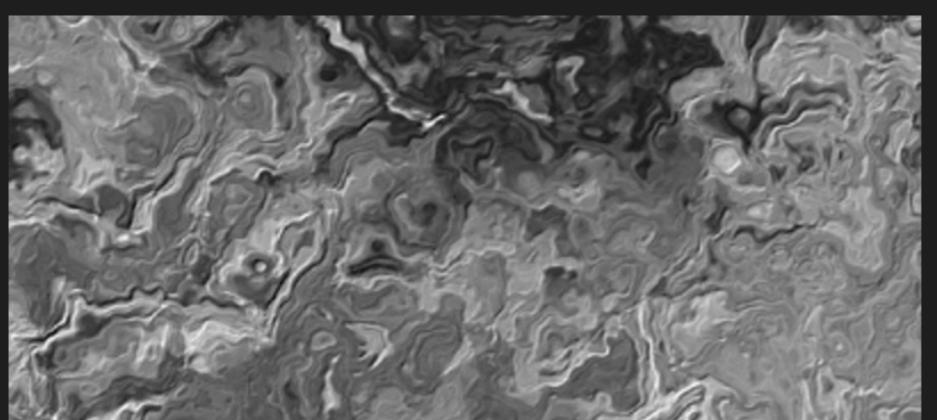


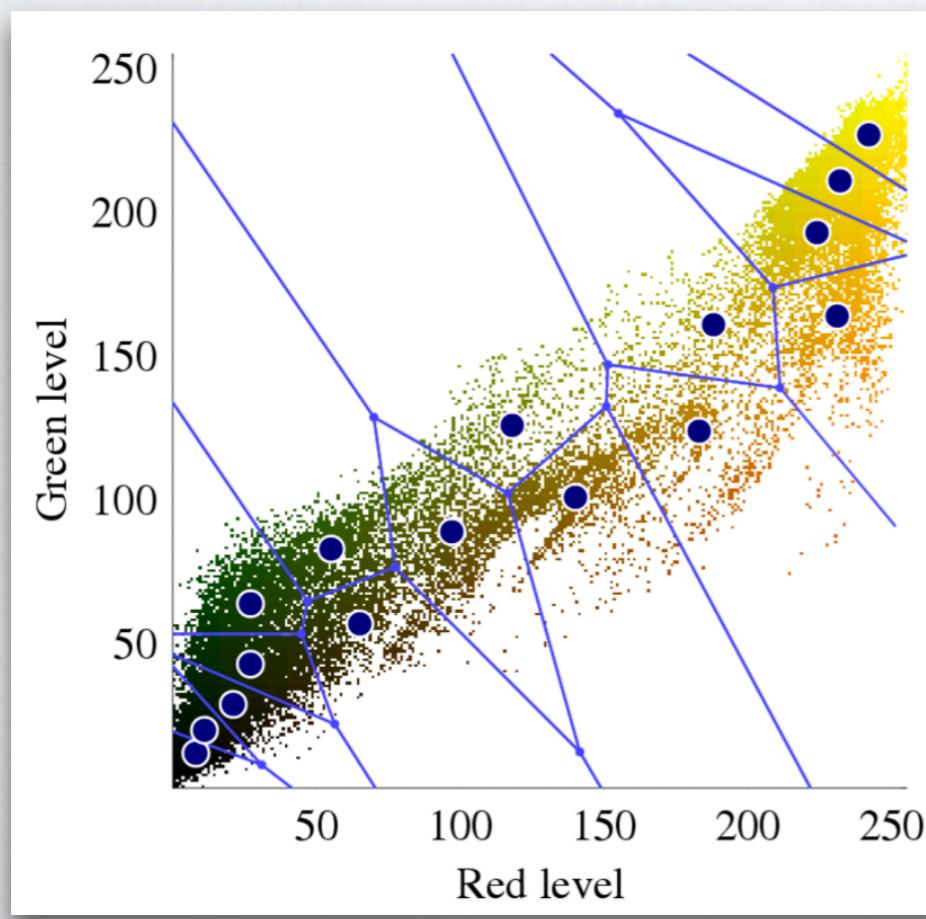
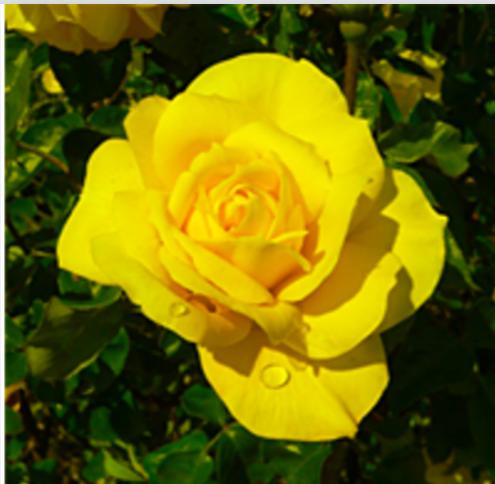
image for $f(p) = f\text{bm}(p + f\text{bm}(p + f\text{bm}(p)))$

IQ (source)

- A simple example
- Like color mapping, you can remap position/spatial input
- In general, ordinarily, we have some **output = $f(\mathbf{position})$**
- However, we can arbitrarily modify the input position:
- $\mathbf{output} = \mathbf{f}(\mathbf{other_f}(\mathbf{position}))$
- More on spatial domain transforms from IQ

K-MEANS COLOR COMPRESSION

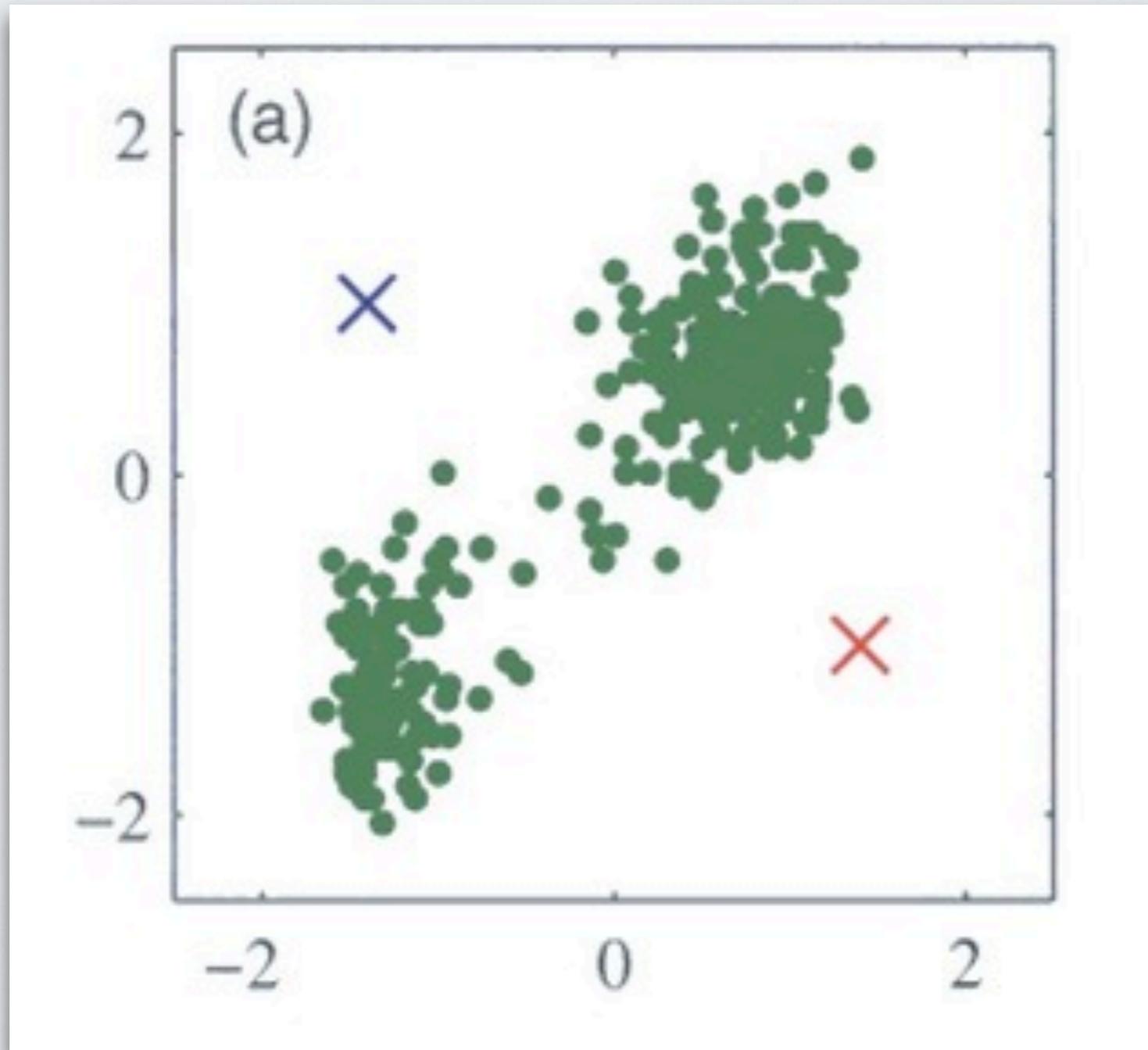
Dzcoetzee ([source](#))



- Want to reduce the number of colors in an image
- But want to minimize the distance from original colorization
- Well, if we were reducing to a single color, we'd use the average of all pixel colors
- Need a way to find k averages, for compression to k colors.
- We can use the k-mean algorithm: clustering technique applicable in many domains. Works in virtually any feature space

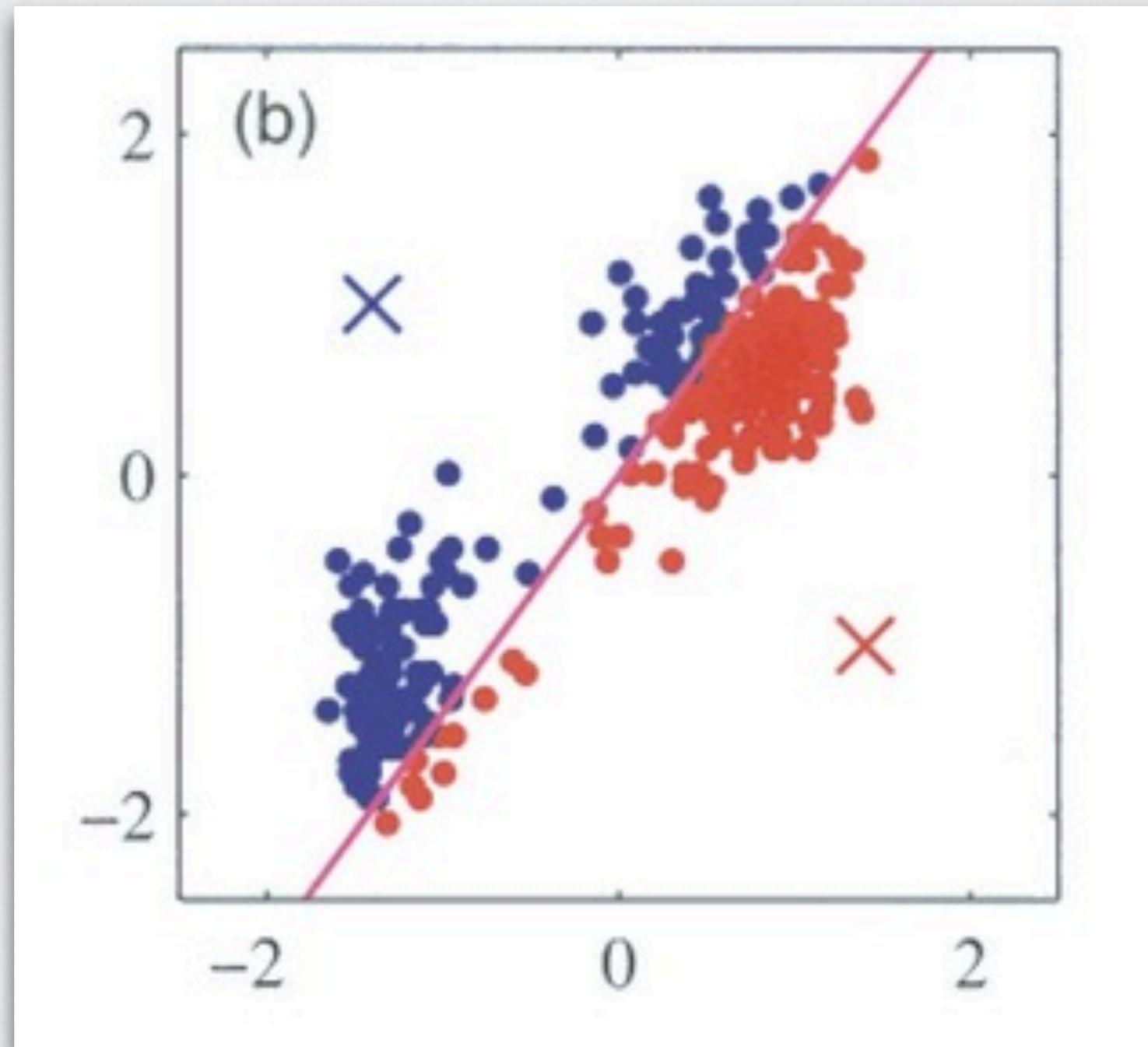
K-MEANS COLOR COMPRESSION

Initialize with k randomly-selected centroids



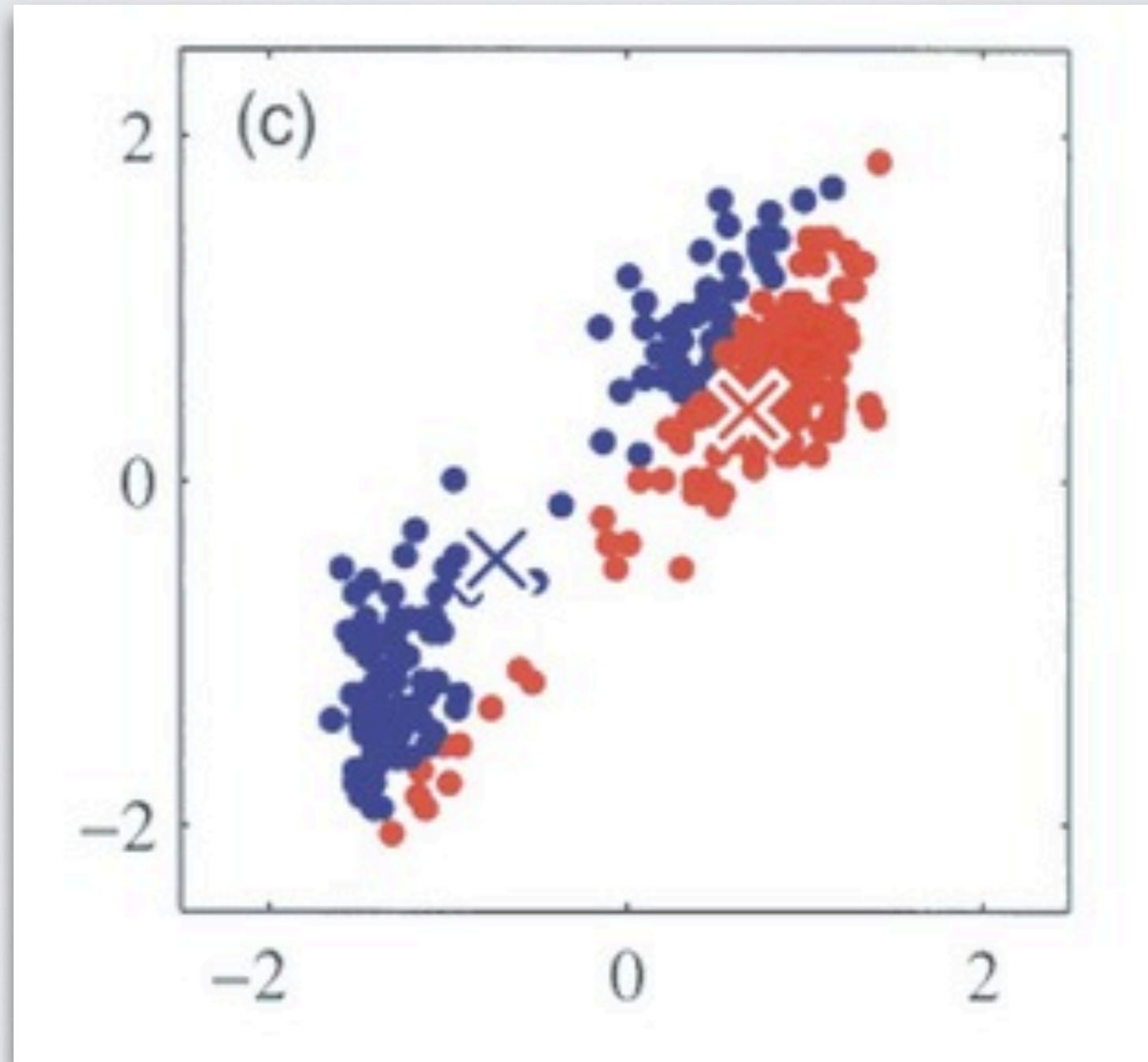
K-MEANS COLOR COMPRESSION

Assign every point to a cluster based on distance to centroid



K-MEANS COLOR COMPRESSION

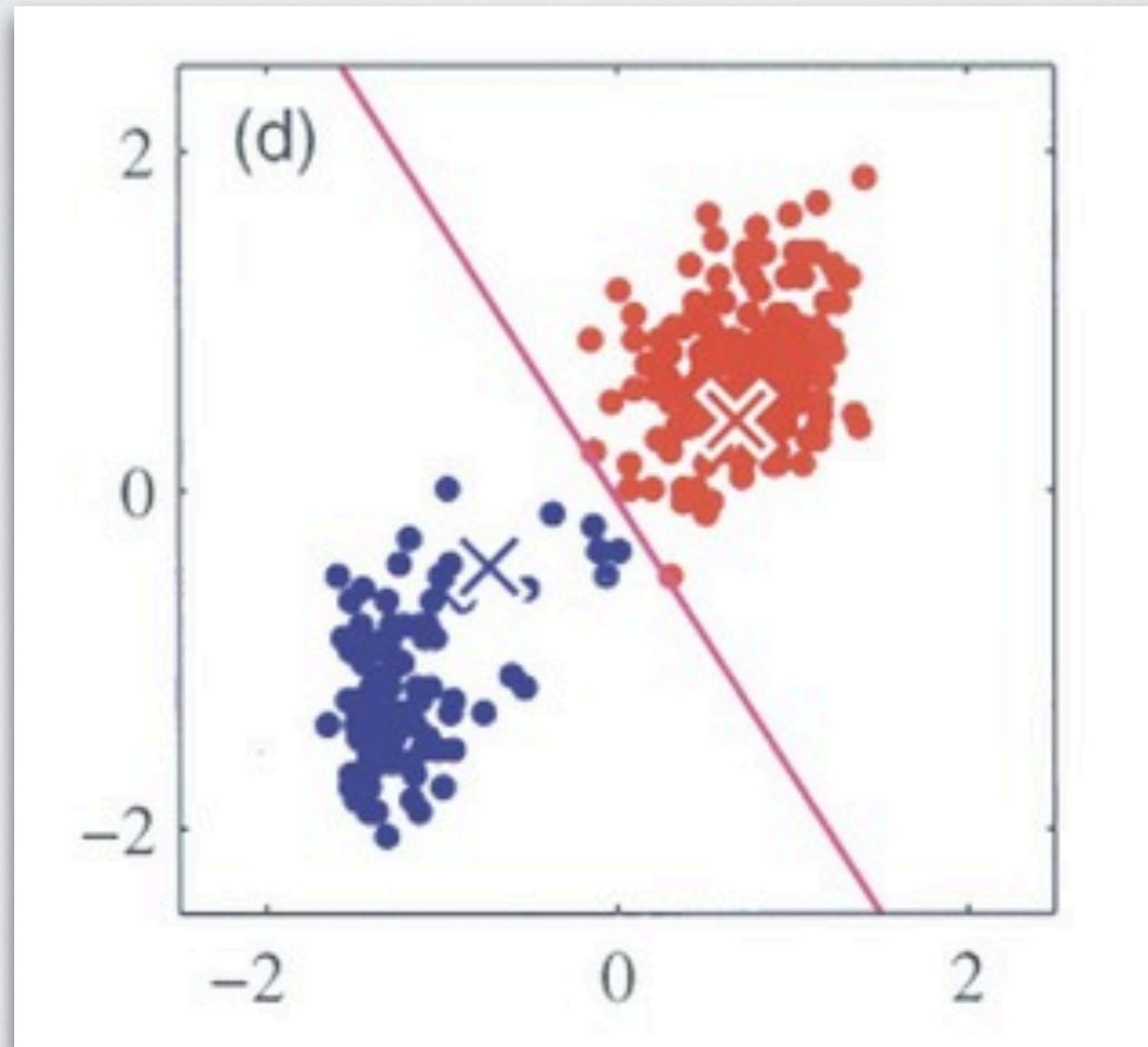
Per cluster, chose new centroid that will reduce distance to all points in the cluster.



Bishop (source)

K-MEANS COLOR COMPRESSION

Given the new centroids, reassign all points to a cluster by distance again.



K-MEANS COLOR COMPRESSION



Original



k = 16



k = 8



k = 4

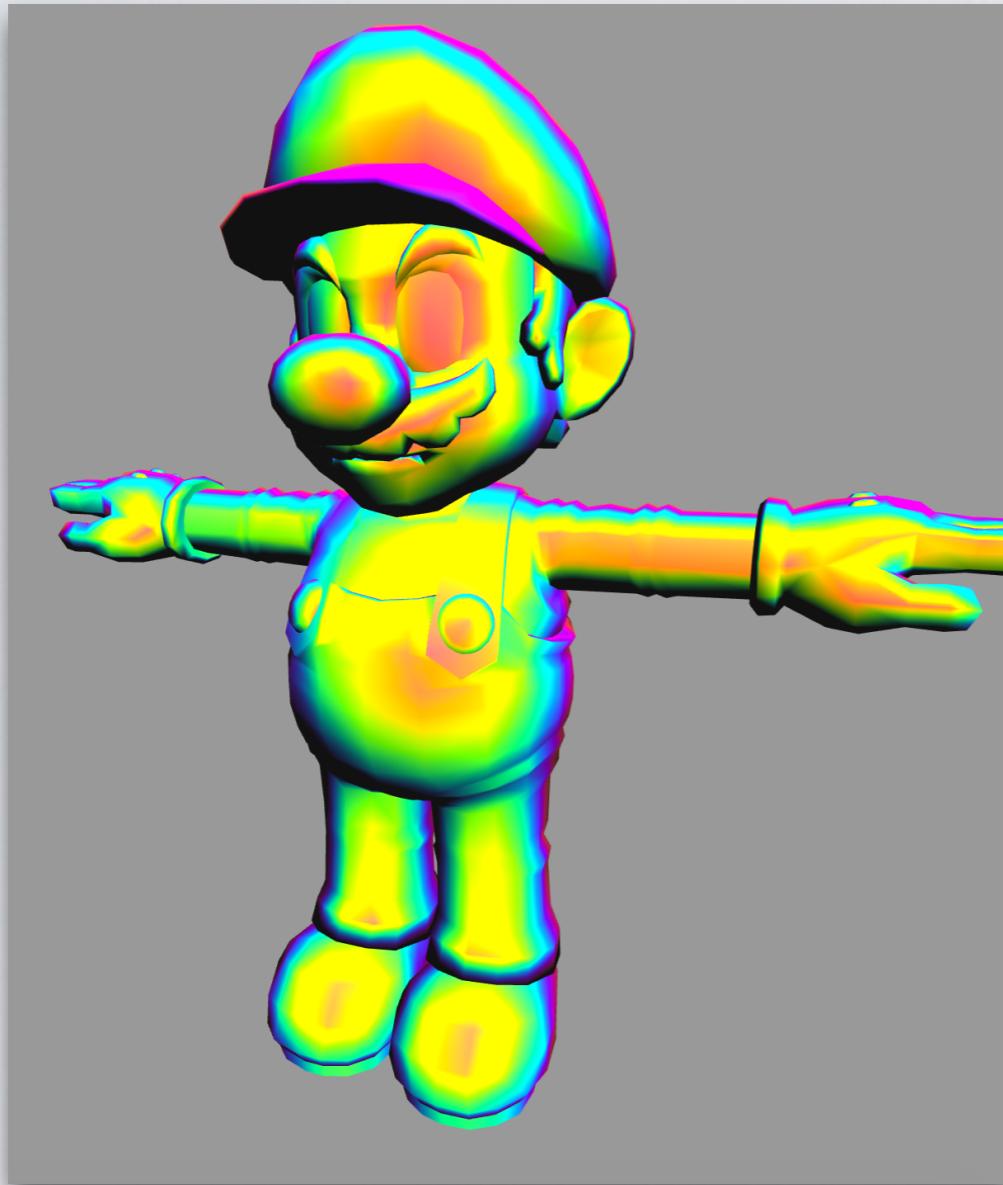
IN SUMMARY

- We introduced a lot of random shading and image processing techniques really fast!
- Graphics is full of mapping functions
 - By remapping color spaces, we can create cool effects
 - You can use any combination of existing attributes as parameters to some modification function, eg. map normals or look vector or darkness to some output space.
- Highlights
 - Gaussian blur - use distance-sensitive averaging to smooth
 - Toon shading - discretize your output color domain
 - Lit sphere - fake a reflectance model with a simple texture read
- In conclusion, there are too many cool effects to study! Take this as inspiration to experiment

REFERENCES

- Papers
 - [Lit Sphere](#)
 - [Non-photorealistic Rendering in Chinese Painting of Animals](#)
- Helpful articles
 - [Intro to color theory](#)
 - [Intro to edge/outline detection](#)
 - [Intro to gaussian blur algorithm](#) And another!
 - [IQ's color palettes](#) / [reference implementation](#) / [Unity implementation](#)
 - [Huge collection of non-realistic graphics references](#)
 - [Procedural color palette ideas](#)
- Textbook
 - [Machine Learning reference](#) (go big or go home!)

ASSIGNMENT



- Implement a variety of fun visual effects, in both world space and screen space shaders