Tutorial 1
CS3241 Computer Graphics (AY22/23)

August 23, 2022

Wong Pei Xian

e0389023@u.nus.edu

Question 1

To be able to display **realistic** images, our display devices need to be able to produce every frequency in the visible light spectrum.

True or false? Why? What are the advantages and disadvantages?

Three-Color Theory

To be **realistic to human**

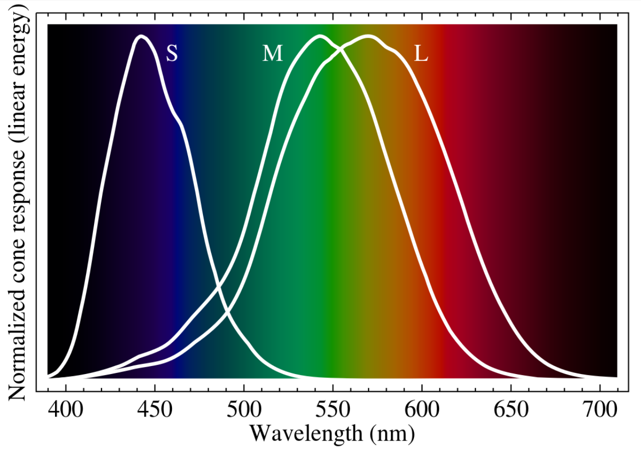
$\Rightarrow$ To be compatible with human visual system (L01, slide 35)

- Rods: Monochromatic
- **Cones**: Color sensitive to wavelengths
  - Long $\approx$ red
  - Medium $\approx$ green
  - Short $\approx$ blue

Proportion of the three gives us the sensation of different colors.

Cone sensitivity and Additive color theory

Single frequency = proportion of responses of each cone.



Additive Color Display

Pros

1. We don't have to produce light of every wavelength in vis. light spectrum for realism
2. We can see colors that are **NOT** on vis. light spectrum (e.g. **PURPLE**).

Additive Color Display

Two different RGB values can produce the same color.

Q: What's an example of this?

A: There is no definitive inverse mapping of RGB to a wavelength, as it is display dependent.

https://physics.stackexchange.com/questions/248139/can-two-different-rgb-color-triplets-give-the-same-color

Question 2

Each pixel in a frame-buffer has 8 bits for each of the R, G and B channels. How many different colors can each pixel represent? Is this enough?

32-bit color

Each of the following four channels is described in **8 bits**.

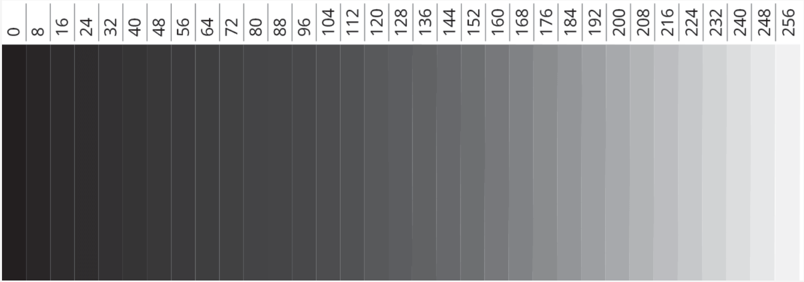
- R
- G
- B
- (not in this question) A for Alpha (transparency)

Hence color has 32 bits: $2^3 2$ values (can be represented with an `int`)!
Based on RGB only: 2^{24} = **16,777,216**. See: Color Depth

32-bit color

Is this enough?

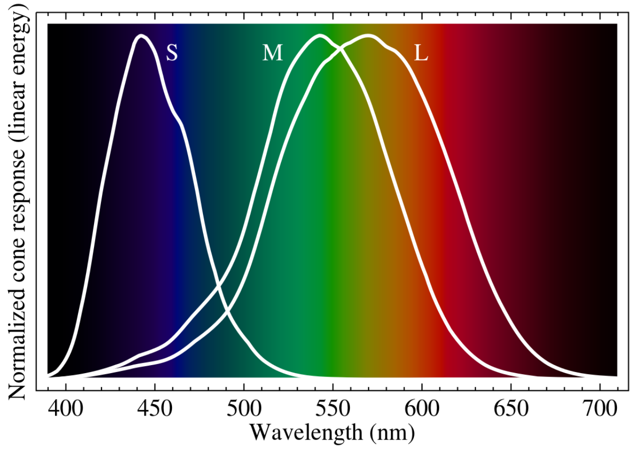
256 shades of gray: **banding artifacts**.



Use case decides if this is undesirable or not.

8-bit representation of color

On some systems, each pixel has only 8 bits (for all R, G, and B combined). How would you allocate the bits to the R, G and B primaries?



8-bit representation of color

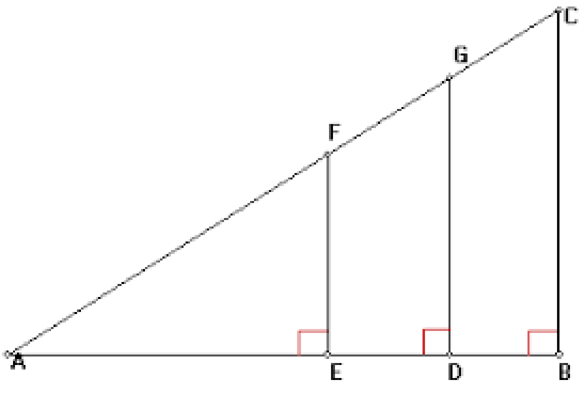
On some systems, each pixel has only 8 bits (for all R, G, and B combined). How would you allocate the bits to the R, G and B primaries?

3:3:2 for R:G:B. Our eyes are less sensitive to changes in blue, as you can see from the normalized cone response for high frequencies.

Question 3

Referring to Lecture 1 Slide 26. If an imaginary image plane is d unit distance in front of the pinhole camera, what are the coordinates of the projection (on the imaginary image plane) of the 3D point (x, y, z)?

Similar triangles



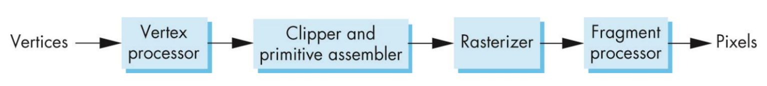
$$\frac{AE}{AD} = \frac{AF}{AG} = \frac{EF}{DG}$$

Answer

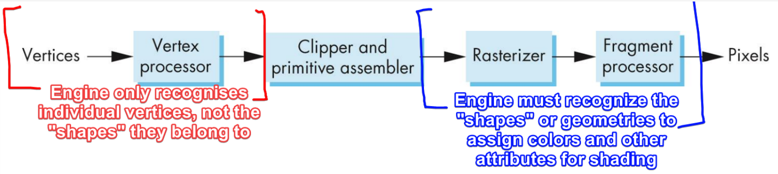
- $z_p = d$ (Why?)
- $x_p = dx/z$
- $y_p = dy/z$

Question 4

Why do we need a **primitive assembly** stage in the rendering pipeline architecture?



Primitive Assembly



Primitive: One polygonal unit

Question 5

What does the rasterization stage (rasterizer) do in the rendering pipeline architecture?

Question 5

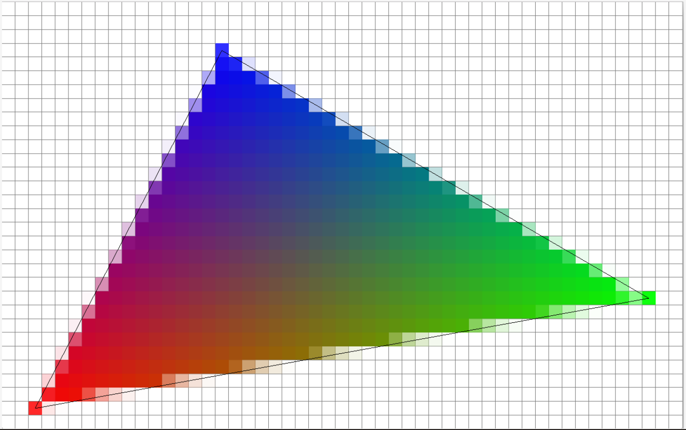
Describe what it does to a triangle that is supposed to be filled, and the three vertices have different color. Assume smooth shading is turned on.

Rasterization

(Lecture 1 Slide 40)

Assigning colors to pixels occupied by a primitive/polygon.

1. Each vertex has an attribute.
2. Each attribute is **interpolated** across the vertices.

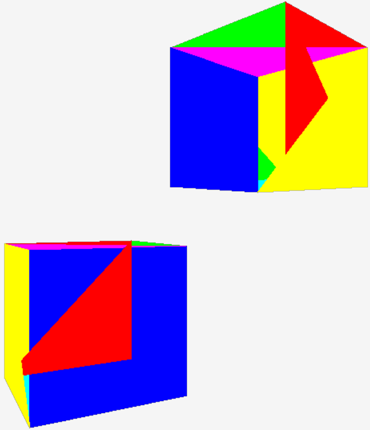


Question 6

What is hidden-surface removal? When is it not necessary?

Hidden Surface Removal

https://gabrielgambetta.com/computer-graphics-from-scratch/12-hidden-surface-removal.html



Question 7

Which of the two following program fragments is more efficient? Why?

A	B

`double v[3*N][3];`
`...`
`for ( int i = 0; i < 3*N; i+=3 )`
`{`
`    glBegin(GL_TRIANGLE);`
`        glVertex3dv( v[i] );`
`        glVertex3dv( v[i+1] );`
`        glVertex3dv( v[i+2] );`
`    glEnd();`
`}`
	`double v[3*N][3];`
	`...`
	`glBegin(GL_TRIANGLE);`
	`    for ( int i = 0; i < 3*N; i+=3 )`
	`    {`
	`        glVertex3dv( v[i] );`
	`        glVertex3dv( v[i+1] );`
	`        glVertex3dv( v[i+2] );`
	`    }`
	`glEnd();`

Can the same optimization be done for the case of `GL_POLYGON`?

Calls to `glBegin` and `glEnd`

A	B
`double v[3*N][3];`	`double v[3*N][3];`
`...`	`...`
`for ( int i = 0; i < 3*N; i+=3 )`	$\boxed{\texttt{glBegin(GL\_TRIANGLE);}}$
`{`	`for ( int i = 0; i < 3*N; i+=3 )`
$\boxed{\texttt{glBegin(GL\_TRIANGLE);}}$	`{`
`glVertex3dv( v[i] );`	`glVertex3dv( v[i] );`
`glVertex3dv( v[i+1] );`	`glVertex3dv( v[i+1] );`
`glVertex3dv( v[i+2] );`	`glVertex3dv( v[i+2] );`
$\boxed{\texttt{glEnd();}}$ N times	`}`
`}`	$\boxed{\texttt{glEnd();}}$ once!

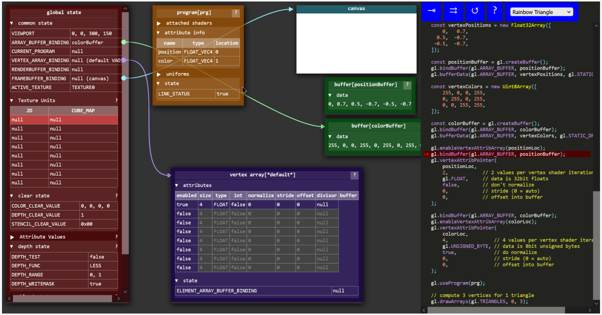
Note that OpenGL (together with mosts other graphics engines) is a **state machine**. Method B greatly reduces the number of state changes.

What about GL_POLYGON?

We can't do this with GL_POLYGON or we'll be defining one massive $3N$-vertex polygon.

GL_POLYGON
: Draws a single, convex polygon. Vertices 1 through N define this polygon.

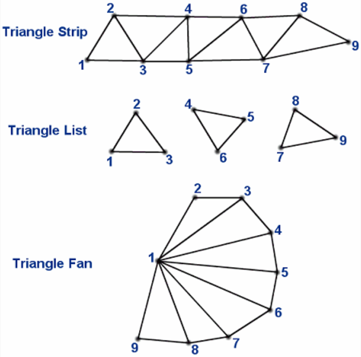
WebGL State Machine visualizer/debugger tool



https://webglfundamentals.org/webgl/lessons/resources/webgl-state-diagram.html

Question 8

OpenGL supports the GL_TRIANGLES primitive type. Why do you think that OpenGL also supports GL_TRIANGLE_FAN and GL_TRIANGLE_STRIP?



Comparison

Type	Vertices	Triangles
GL_TRIANGLES	3n	n
GL_TRIANGLE_FAN	n + 2	n
GL_TRIANGLE_STRIP	n + 2	n

Question 9

Devise a test to check whether a polygon in 3D space is planar.

Question 10

Devise a test to check whether a polygon on the x-y plane is convex.

Thanks!

https://github.com/