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.

I. Is this True or False? Why?

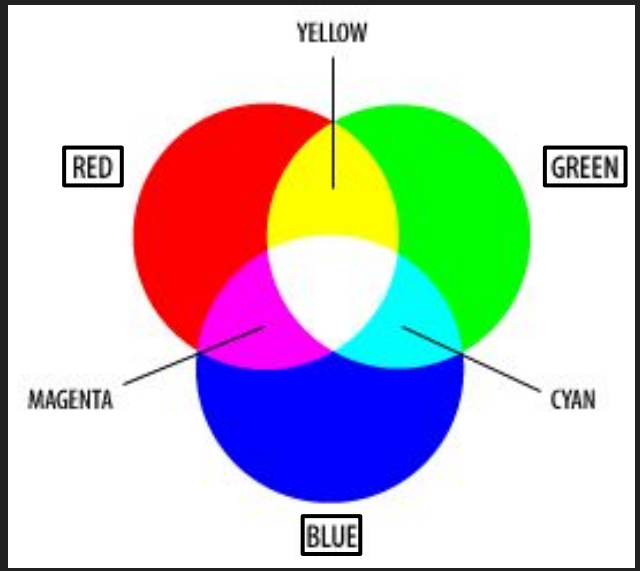
II. What are the Advantages and Disadvantages (of not producing only every frequency in the visible spectrum)?

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.

I. Is this True or **False**? Why?

- **Three-Color Theory** (Lecture 1, Slide 34)
- A single-frequency light and a multiple-frequency light can **excite the 3 cones in the same way**, resulting in the same perception.
- e.g. a mixture of pure “red” and pure “green” light can excite the cones in the same way as a pure “yellow” light does, and produce the same sensation of “yellow”.

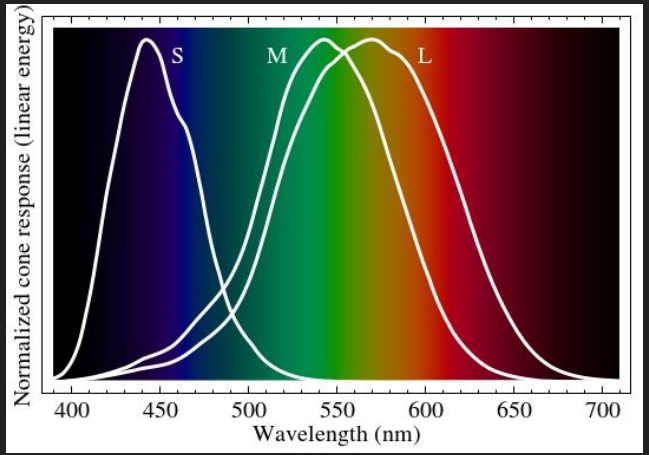


Question 1

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

I. Is this True or **False**? Why?

- Take note that scientifically the cone **“**colours**”** are not direct mappings (hence the quotation marks).
- Cones are actually labeled according to the ordering of the **wavelengths** of the peaks of their spectral sensitivities: Short (S), Medium (M), and Long (L) cone types.



Read More: https://en.wikipedia.org/wiki/Color_vision#Physiology_of_color_perception

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.

II. What are the **Advantages** and Disadvantages (of not producing only every frequency in the visible spectrum for display devices)?

- There is no need to build such display devices (so save money and resources).
- Allows us to display colours like **Purple**, which cannot be produced by any single-frequency light and is **not in the visible spectrum**. Purple does not have own wavelength of light, so it's a non-spectral color.

Read More: https://helenair.com/lifestyles/primary-colors-the-truth-about-purple/article_b2080202-4a1f-11e0-8d90-001cc4c03286.html

Question 1

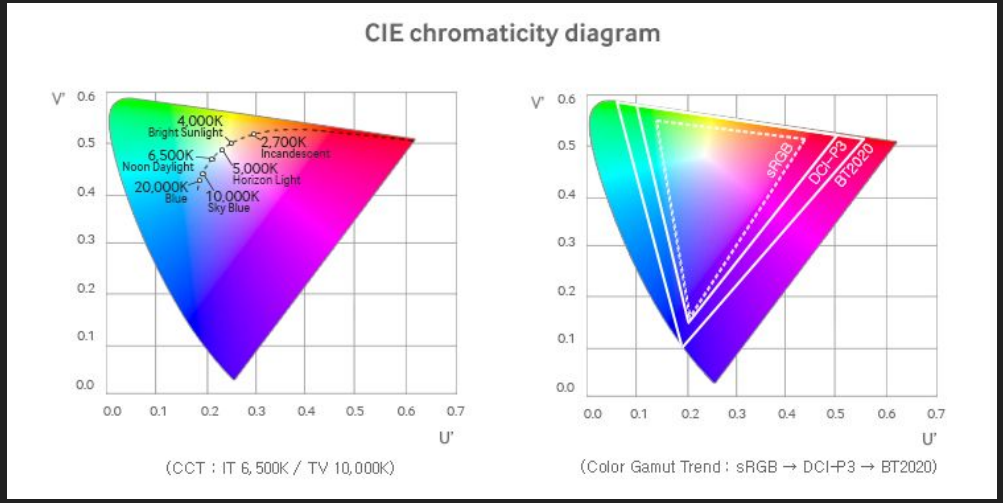
To be able to display realistic images, our display devices need to be able to produce every frequency in the visible light spectrum.

II. What are the Advantages and **Disadvantages** (of not producing only every frequency in the visible spectrum for display devices)?

- There *shouldn't* be any. Generally speaking, many different combinations of many different frequencies of light can produce the same color and we **cannot distinguish** them apart.

Read More: https://helenair.com/lifestyles/primary-colors-the-truth-about-purple/article_b2080202-4a1f-11e0-8d90-001cc4c03286.html

F.Y.I.: **Color Space/Gamut**



- Shows the range of colors a device can display.
- 3 color systems often have a gamut in the shape of a triangle.

Read More: https://en.m.wikipedia.org/wiki/CIE_1931_color_space

Question 2

I. 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?

II. 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?

Question 2

I. 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?

- $2^{8+8+8} = 2^{24} =$ 16,777,216 different colours.
- Sometimes this is not enough. For example, only **256 shades of gray** can be represented, and if we have these 256 shades displayed across the screen, we can still see the **banding effects**.

Question 2

II. 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?

- Most systems allocate **3:3:2** for R:G:B because our eyes are actually **less sensitive** to the changes in the **blue component**.
- You see more details in green, then red, and then blue,

Read More: https://nfggames.com/games/NTSC/visual.shtm

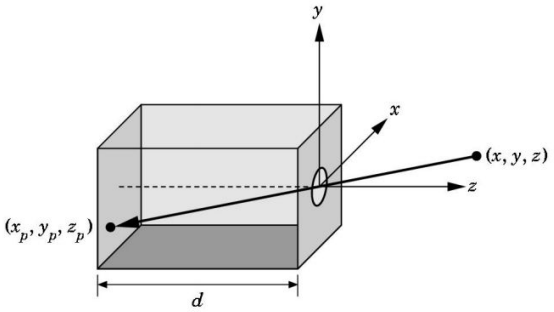
Question 3

I. Referring to Lecture 1 Slide 36. 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)?

II. If the camera's center of projection is not located at the origin, and the camera is pointed in an arbitrary orientation, the calculation of the projection becomes very messy. How would you make it less messy?

Question 3

I. Referring to Lecture 1 Slide 36. 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)?

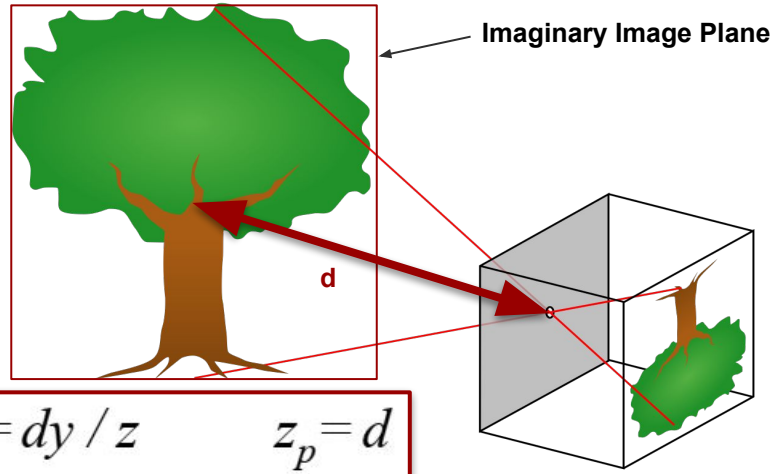


Use trigonometry to find projection of point at (x, y, z)

$x_p = -dx / z$ $y_p = -dy / z$ $z = -d$

These are equations of sim

$$x_p = dx / z \qquad y_p = dy / z \qquad z_p = d$$



Question 3

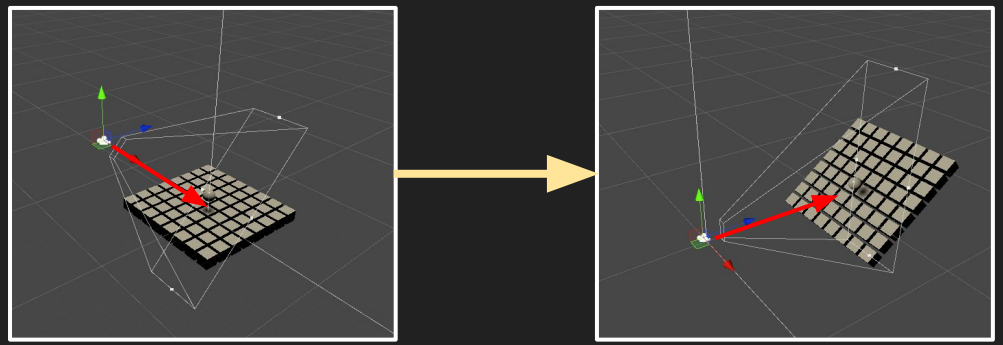
II. If the camera's center of projection is not located at the origin and the camera is pointed in an arbitrary orientation, the calculation of the projection becomes very messy. How would you make it less messy?

- The Camera has its own coordinate frame and the 3D Point's coordinates are expressed in the world coordinate frame (these frames are **separate**).
- In the example given, these two frames **coincide exactly** so it is easy to calculate the coordinates of the projection of the 3D Point. If the Camera's center of projection is not located at the origin and the camera is pointed in an arbitrary orientation, that results in the **unalignment** of the two coordinate frame.

Question 3

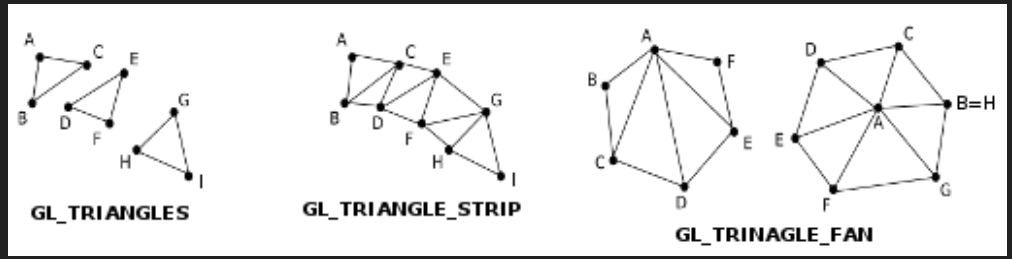
II. If the camera's center of projection is not located at the origin and the camera is pointed in an arbitrary orientation, the calculation of the projection becomes very messy. How would you make it less messy?

- We should do a **3D Rotation** and **Translation** to align the two coordinate Frames, i.e. a **3D Transformation** to express the coordinates of the 3D point with respect to the camera coordinate frame.



Question 4

- Why do we need a primitive assembly stage in the rendering pipeline architecture?
  - Each primitive is defined by its vertices only, and these **vertices are processed independently in the vertex processor** without regards to the type of the primitive that they define.
  - We need the primitive assembly stage to **recall the primitive type** and **collect its processed vertices**, so that we can **reconstruct the primitive** for clipping and rasterization.



Question 5

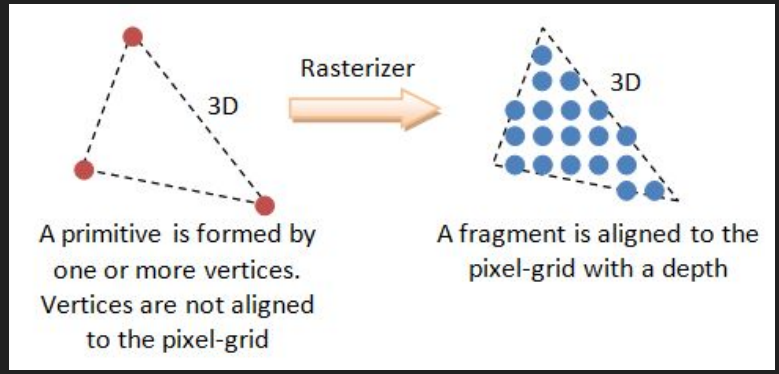
I. What does the rasterization stage (rasterizer) do in the rendering pipeline architecture?

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

Question 5

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

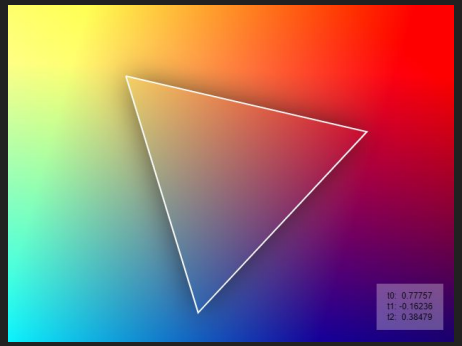
- "**Turns on**" the pixels of each primitive that are projected on the **image plane**, producing a set of **fragments** for each primitive.
- Each primitive is raster-scanned to obtain a set of fragments enclosed within the primitive.



Question 5

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

- For a filled triangle, whose vertices have been projected to window space, the rasterizer produces **fragments for the interior of the triangle**, and the colors of these pixels are the results of **smooth interpolation** of the colors at the vertices.



Try It Out: https://observablehq.com/@toja/triangular-color-interpolation

Question 6

- What is a GLUT display callback function? Give example events for which the display callback function should be called.
  - A GLUT display callback function is a **user-defined function** registered using `glutDisplayFunc()`. This user-defined function will be executed whenever GLUT determines that the window should be **refreshed**.
  - Example events in which we should refresh the drawing:
    - when the window is **first created**,
    - when the window is **resized**,
    - when the window is **restored from a minimized state**,
    - when the window is **uncovered** (another window is no longer blocking it),
    - when the **scene is updated** and has to be **redrawn**.
  - **Note**: you can also call `glutPostRedisplay()`.

Question 7

- Which of the two following program fragments is more efficient? Why? Can the same optimization be done for the case of **GL_POLYGON**?

A

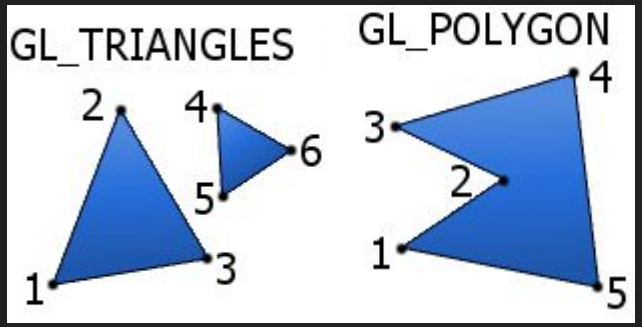
```
double v[3*N][3];
for (int i = 0; i < 3*N; i+=3) {
   glBegin(GL_TRIANGLE);
      glVertex3dv(v[i]);
      glVertex3dv(v[i+1]);
      glVertex3dv(v[i+2]);
   glEnd();
}
```

B

```
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();
```

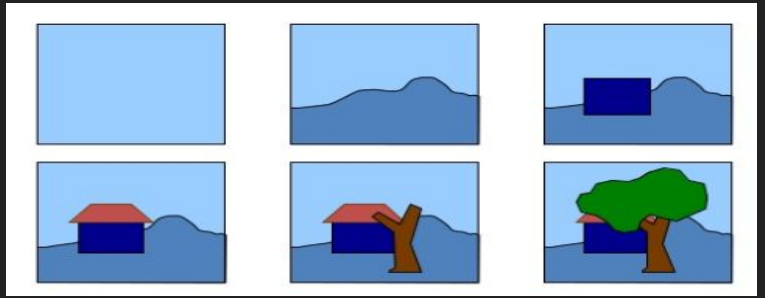
Question 7

- Which of the two following program fragments is more efficient? Why? Can the same optimization be done for the case of **`GL_POLYGON`**?
  - The same optimization cannot be used for **`GL_POLYGON`** because each **`glEnd()`** is used to indicate the **end of the list of vertices of the polygon**, and this indirectly specify the number of vertices the polygon has.



Question 8

- What is Hidden-Surface Removal (HSR)? When is it not necessary?



Question 9

Definition of a Planar Polygon: A polygon that has all its vertices on the same plane.

- Devise a test to check whether a polygon in **3D space** is **Planar**.

Question 10

Definition of a Convex Polygon: A polygon that has all interior angles less than 180°.

- Devise a test to check whether a polygon on the **x-y plane** is **Convex**.
  1. Given an polygon of size N with vertices [$v_1..v_N$].
  2. Assuming that any 3 consecutive vertices are **not collinear**.
  3. Find the **normal vectors** of triangles with consecutive vertices in the polygon.
     i.e. ($v_1, v_2, v_3 \rightarrow n_1$), ($v_2, v_3, v_4 \rightarrow n_2$), …, ($v_{(N-1)}, v_{(N)}, v_{(1)} \rightarrow n_{(N-1)}$)
  4. The polygon is Convex if and only if **all the normal vectors are in the same direction**.

Try It Out: https://www.mathopenref.com/polygonconvex.html