Computer Graphics

Unit 4

Light, Colour, Shading and Hidden Surfaces

Mapping of Course Outcomes: CO5

CO5: Understand the concepts of color models, lighting, shading models and hidden surface elimination.

(Duration: 7 Hours)

Contents

- **Colour models:** Properties of Light, CIE chromaticity Diagram, RGB, HSV, CMY.
- **Illumination Models:** Ambient Light, Diffuse reflection, Specular Reflection, and the Phong model, Combined diffuse and Specular reflections with multiple light sources, warn model
- **Shading Algorithms:** Halftone, Gauraud and Phong Shading.
- **Hidden Surfaces:** Introduction, Back face detection and removal, Algorithms: Depth buffer (z), Depth sorts (Painter), Area subdivision (Warnock)

Color models

- The purpose of color model (color space or color system) is to facilitate specification of colors in some standard way.
- A color model provides a coordinate system and a subspace in it where each color is represented by a single point.
- There are 3 hardware oriented color models:

1) **RGB model:** Used for color CRT monitor
2) **YIQ model:** Used for broadcast TV color system
3) **CMY model:** Used for color printing devices

- There are another class of color models which are depend on hue, saturation and brightness. They are: **HSV, HLS, HVC** models

Properties of light

Light

Light is fundamental for *color vision*

Unless there is a source of light, there is nothing to see!

What do we see?

We do not see objects, but the light that has been *reflected by* or *transmitted through* the objects

Light and EM waves

Light is an electromagnetic wave

If its wavelength is comprised between 400 and 700 nm (*visible spectrum*), the wave can be detected by the human eye and is called *monochromatic light*

- A light produced by a sun or any light source emits all frequencies within the visible range.
- When this light is incident upon an object, some frequencies are absorbed and some are reflected by the object.
- The combination of reflected frequencies decides the color of the object.
- If lower frequencies (or higher wavelengths) are predominant in the reflected frequencies, the object color is red. (Since red color has lowest frequency or highest wavelength than other colors)
- Thus dominant frequency decides the color of object.
- The dominant frequency is also called as **hue** or simply the **color.**
- There are two more properties which describe various characteristics of light. These are: **brightness** and **saturation (purity).**
- The brightness refers to the intensity of the perceived light.
- The saturation describes the purity of the color. There is no white, black, or gray present in a color that has high purity.

- When purity and dominant frequency are used collectively to describe the color characteristics, are referred as **chromaticity.**
- We can combine two colors with suitable intensities to produce a range of other color.
- The two colors that are combined to produce white color are called as **complementary colors** of each other.
- Red-Cyan, Green-Magenta, Blue-Yellow are the complementary color pairs.
- Typically, the color model use combination of three colors to produce wide range of colors. This range of colors is known as **color gamut** for that model. The basic colors used to produce color gamut in a particular model are called as **primary colors** for that model.
- Primary colors for RGB models are: red, green and blue
- Primary colors for CMY models are: cyan, magenta and yellow

Primary and Secondary Colors

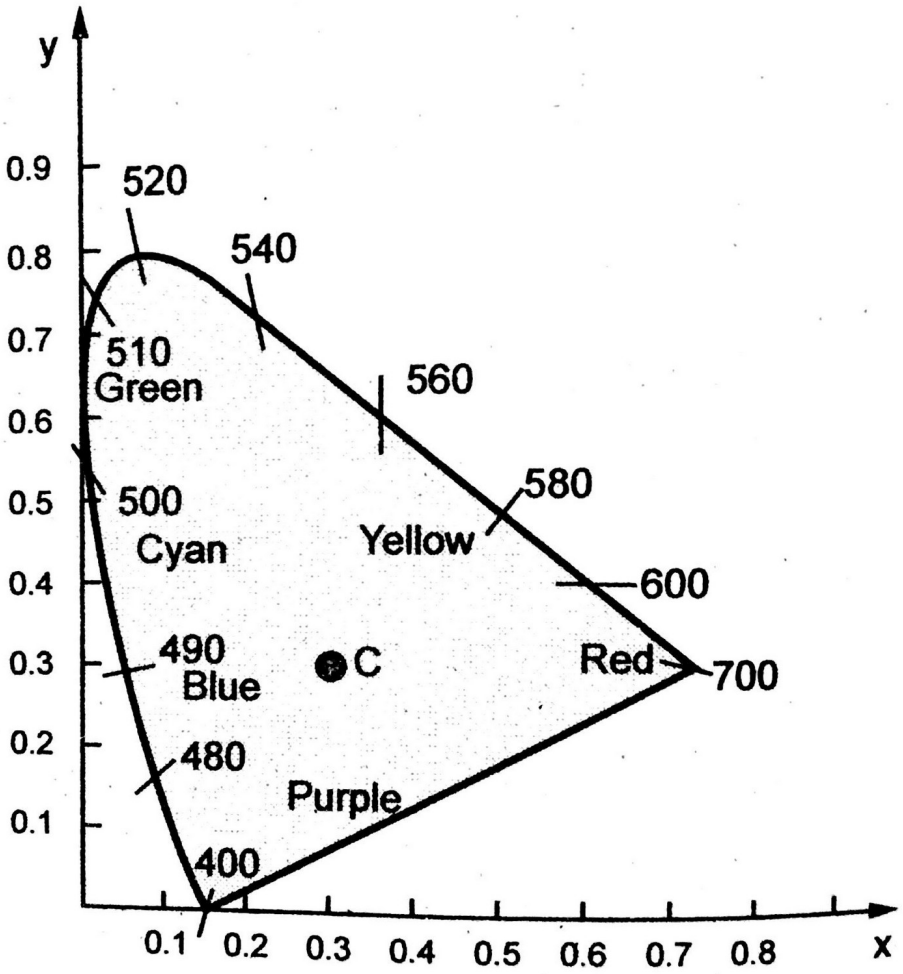
Due to the different absorption curves of the cones, colors are seen as variable combinations of the so-called primary colors: *red*, *green*, and *blue*

Their wavelengths were standardized by the CIE in 1931: *red*=700 nm, *green*=546.1 nm, and *blue*=435.8 nm

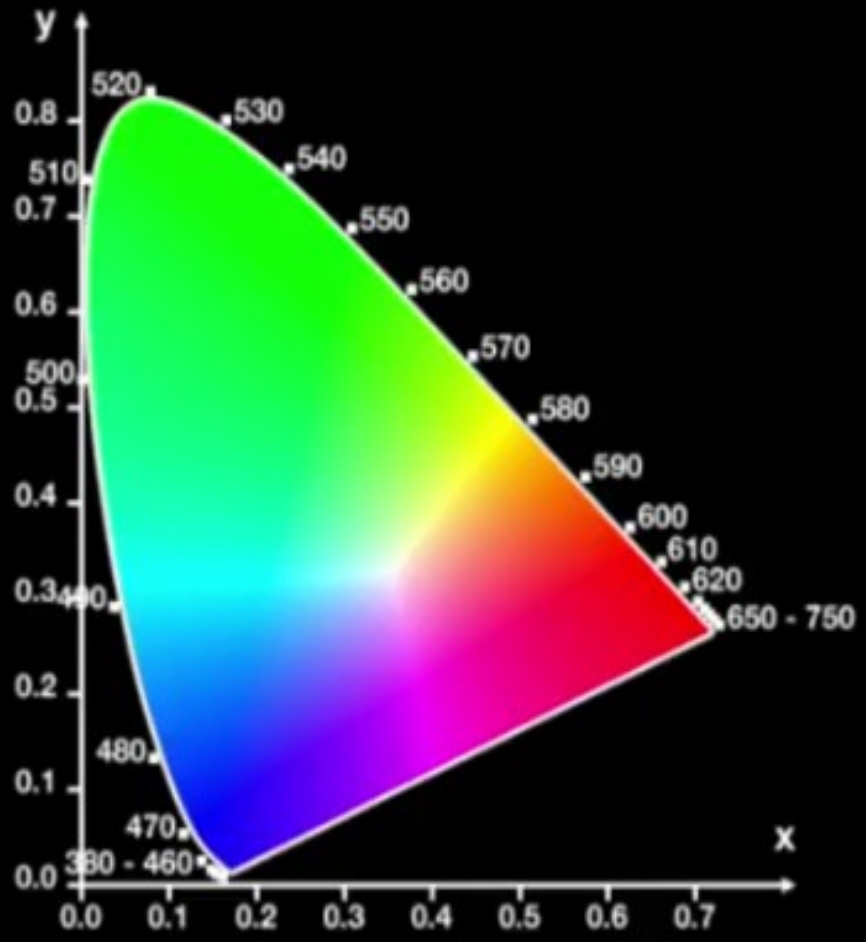
The primary colors can be added to produce the secondary colors of light, *magenta* (R+B), *cyan* (G+B), and *yellow* (R+G)

CIE Chromaticity Diagram

- In 1931, the CIE introduced the Chromaticity diagram that represents the mapping of human color perception in terms of two CIE parameters x and y.
- The Commission Internationale de l'Éclairage (CIE).
- The points on the boundary are the pure colors in the electromagnetic spectrum, labeled according to wavelength in nanometers from the red end (700nm) to the violet end (400nm) of the spectrum.
- The Line joining the red and violet spectral is called purple line, which is not the part of the spectrum.



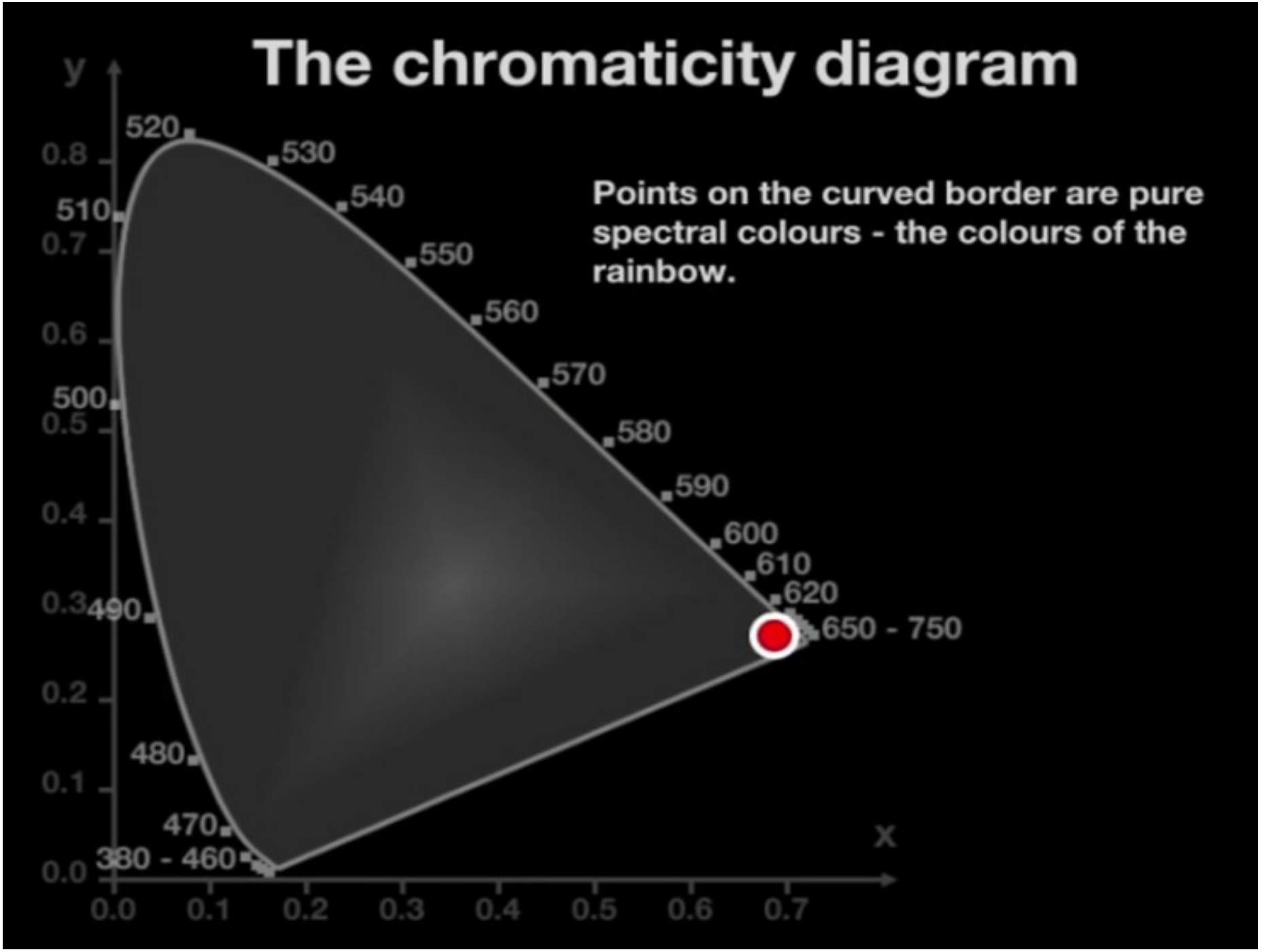
The chromaticity diagram

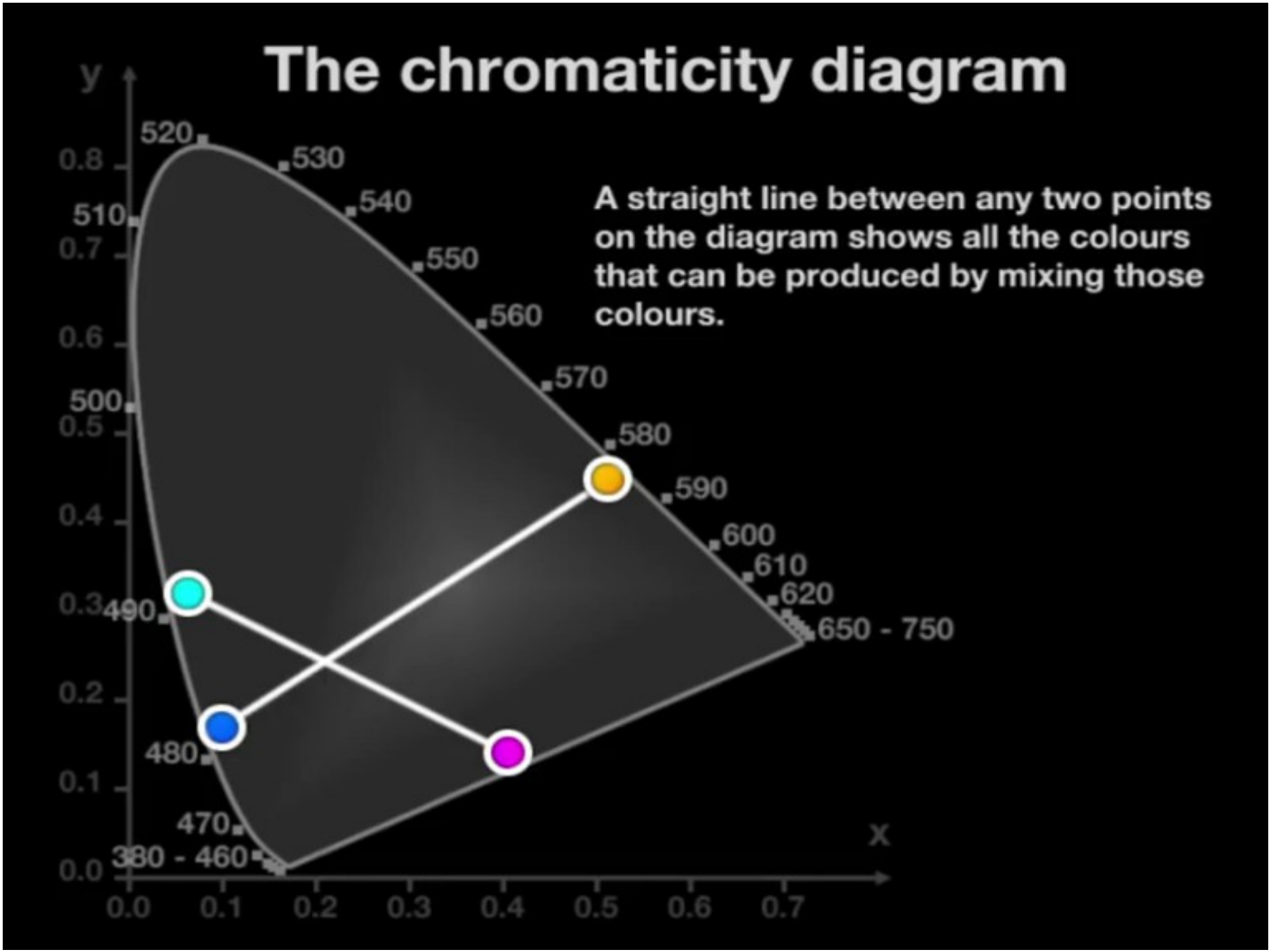


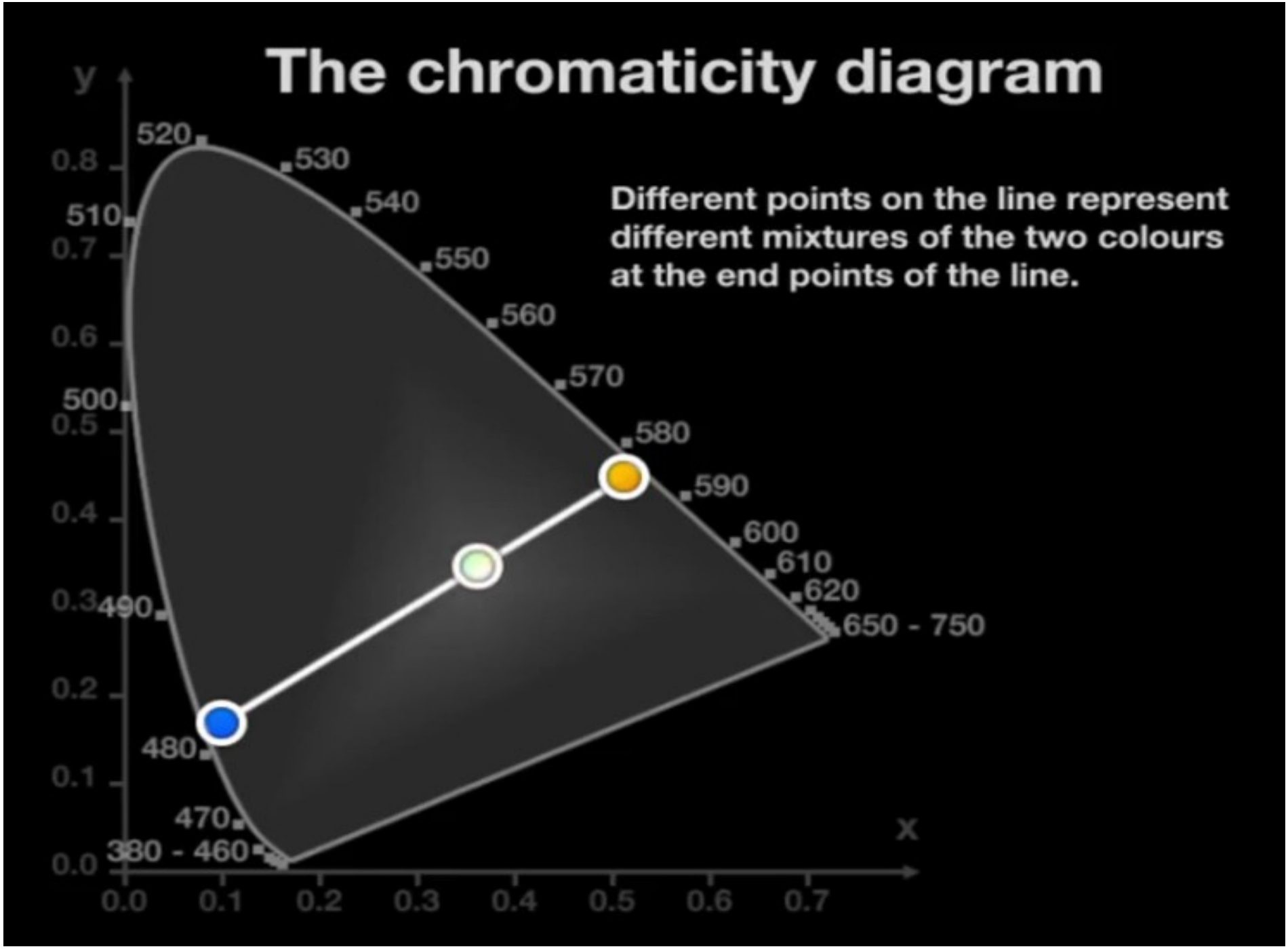
The chromaticity diagram is a graph which shows ALL possible colours.

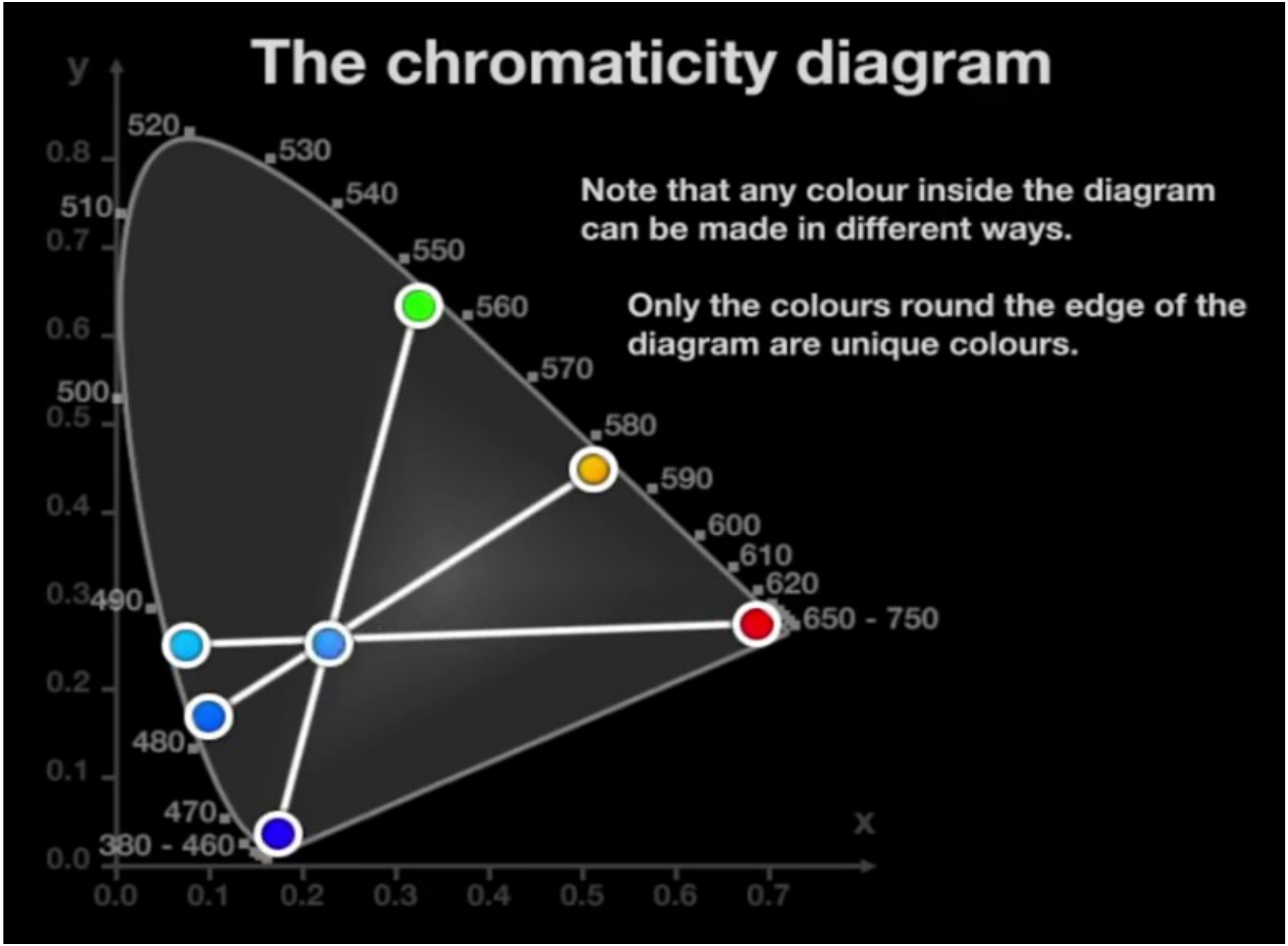
Each colour is defined by a pair of numerical co-ordinates - the chromaticity co-ordinate.

We can use the chromaticity diagram to see how different colours of light mix together

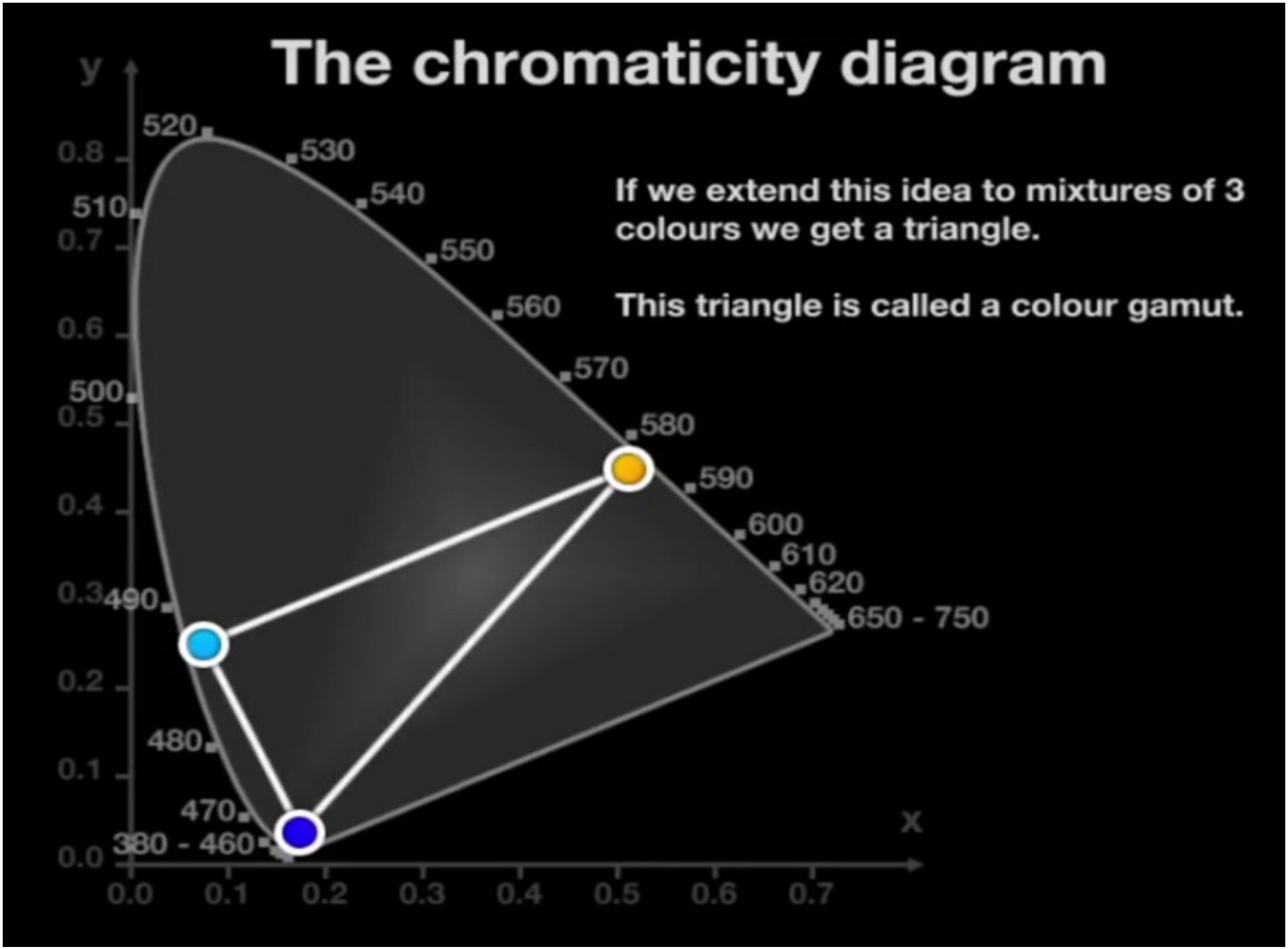

The chromaticity diagram
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Points on the curved border are pure spectral colours - the colours of the rainbow.


The chromaticity diagram
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
A straight line between any two points on the diagram shows all the colours that can be produced by mixing those colours.


The chromaticity diagram
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Different points on the line represent different mixtures of the two colours at the end points of the line.

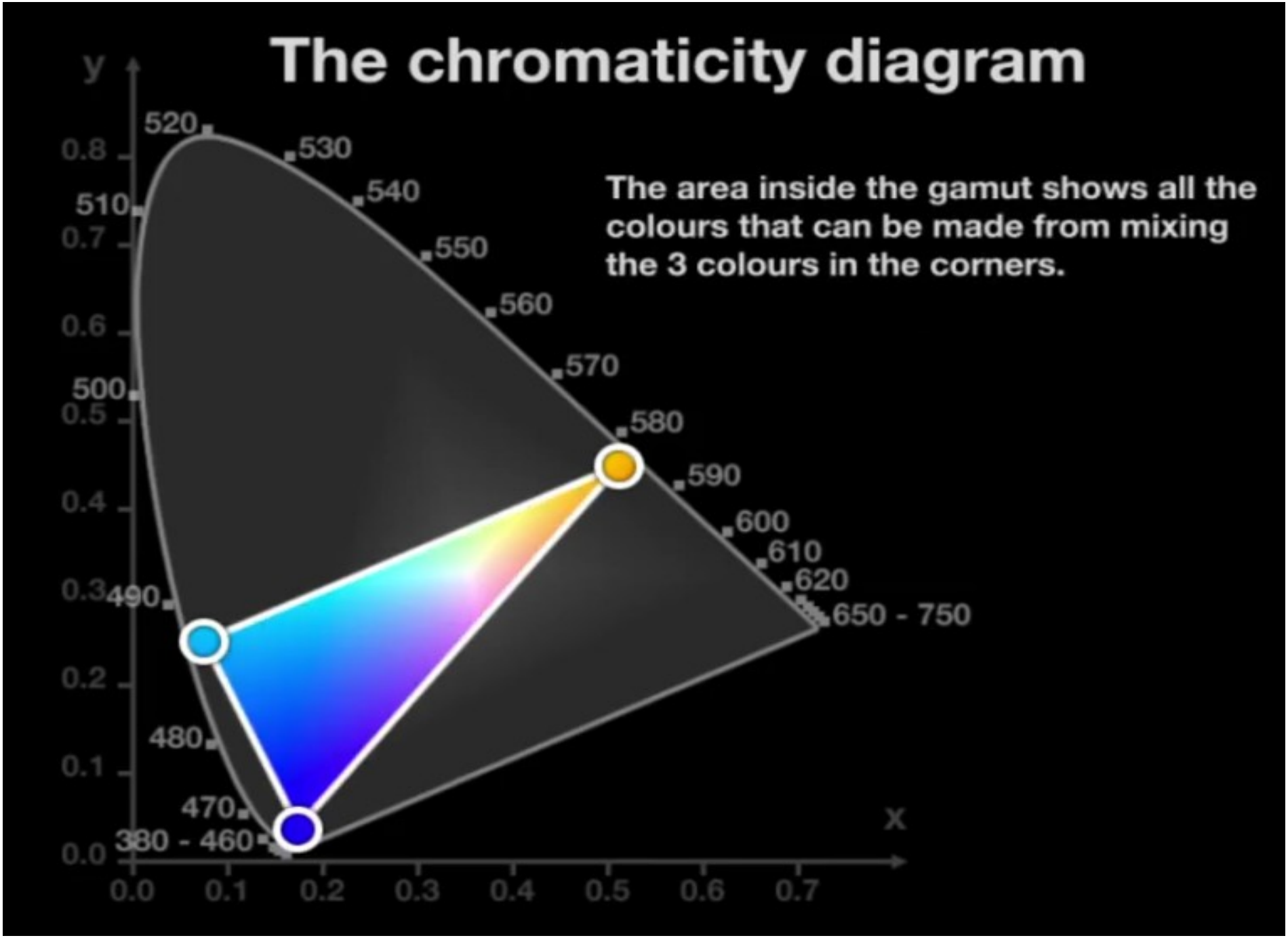

The chromaticity diagram
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Note that any colour inside the diagram can be made in different ways.
Only the colours round the edge of the diagram are unique colours.

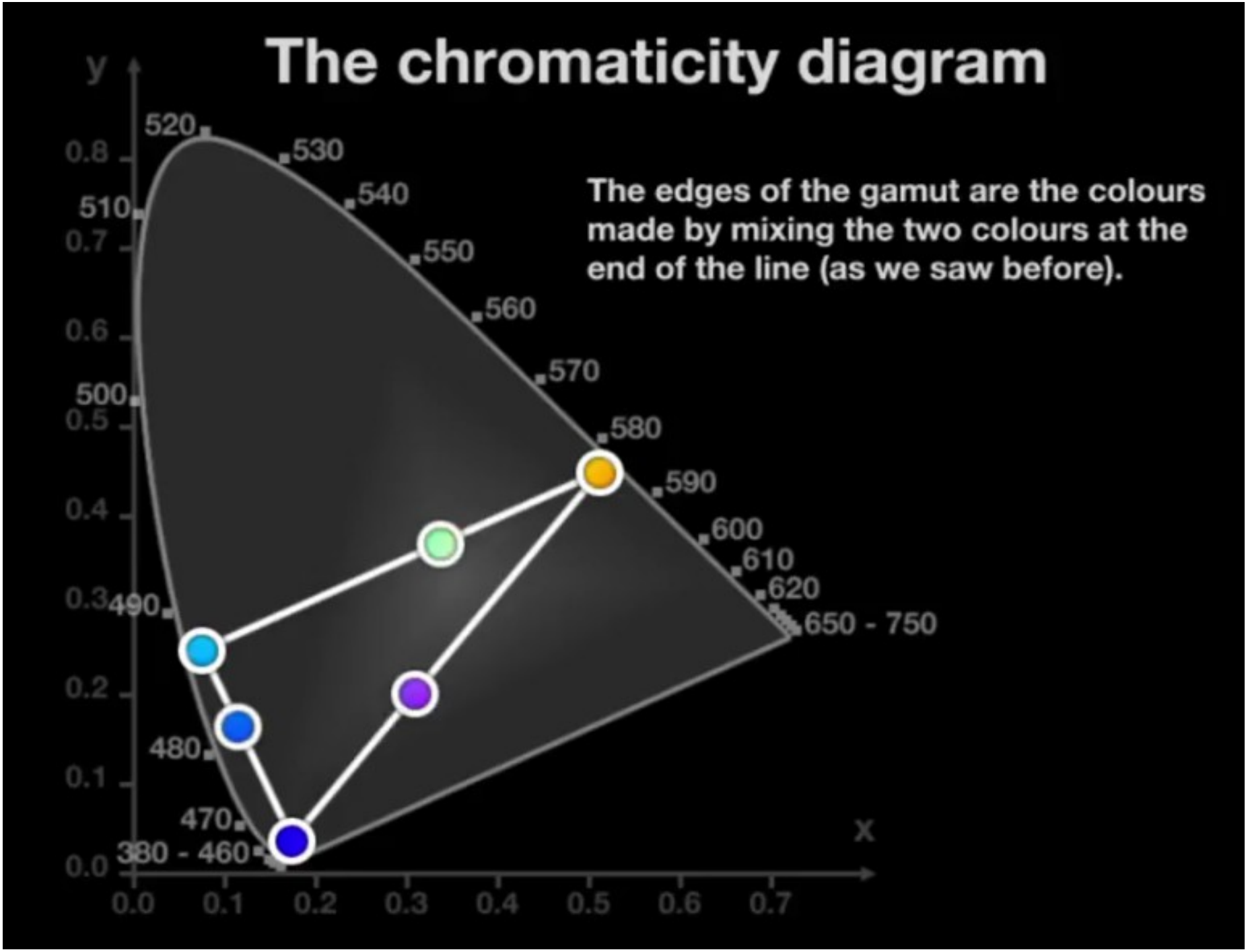
The chromaticity diagram



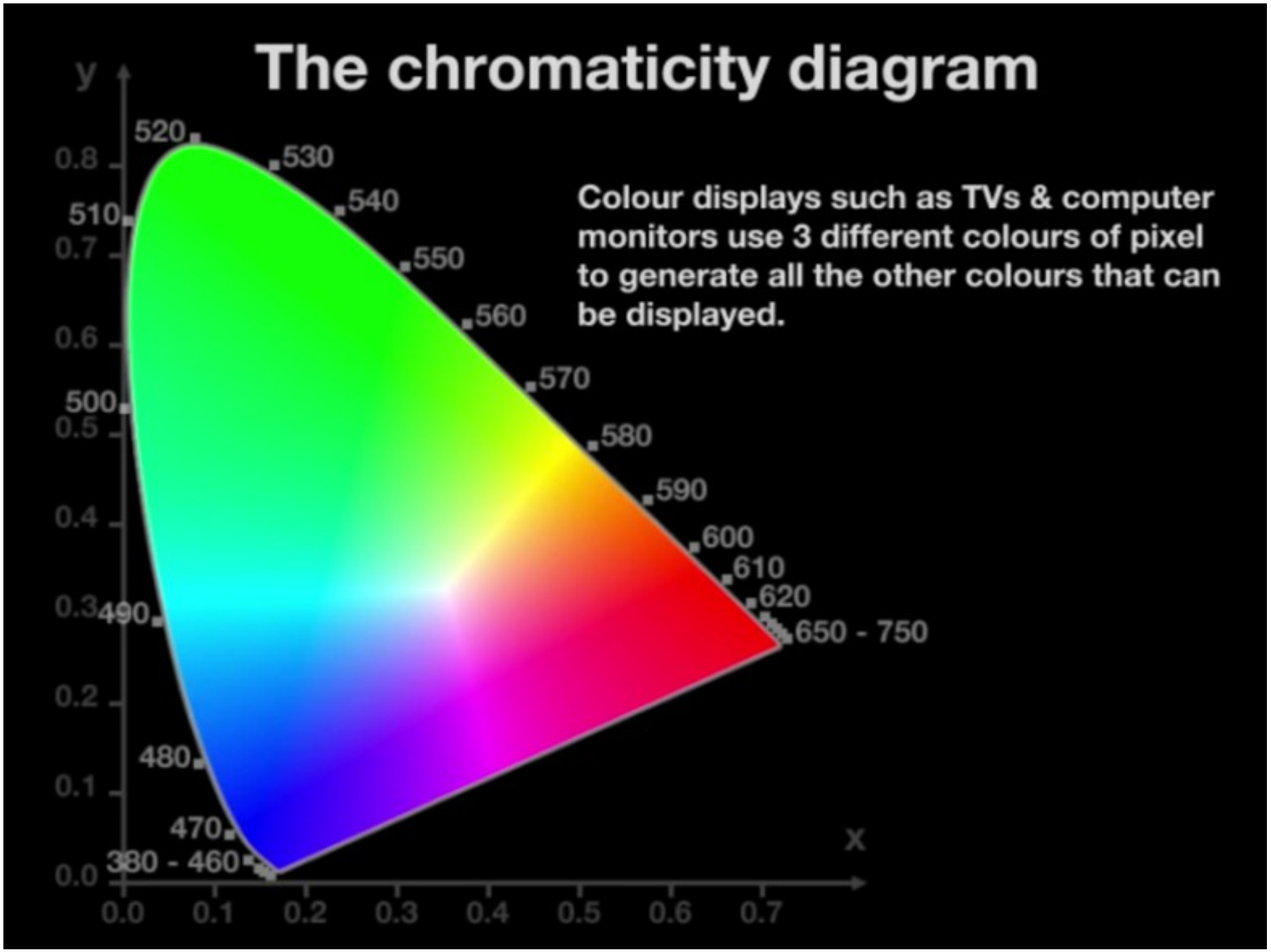
If we extend this idea to mixtures of 3 colours we get a triangle.

This triangle is called a colour gamut.

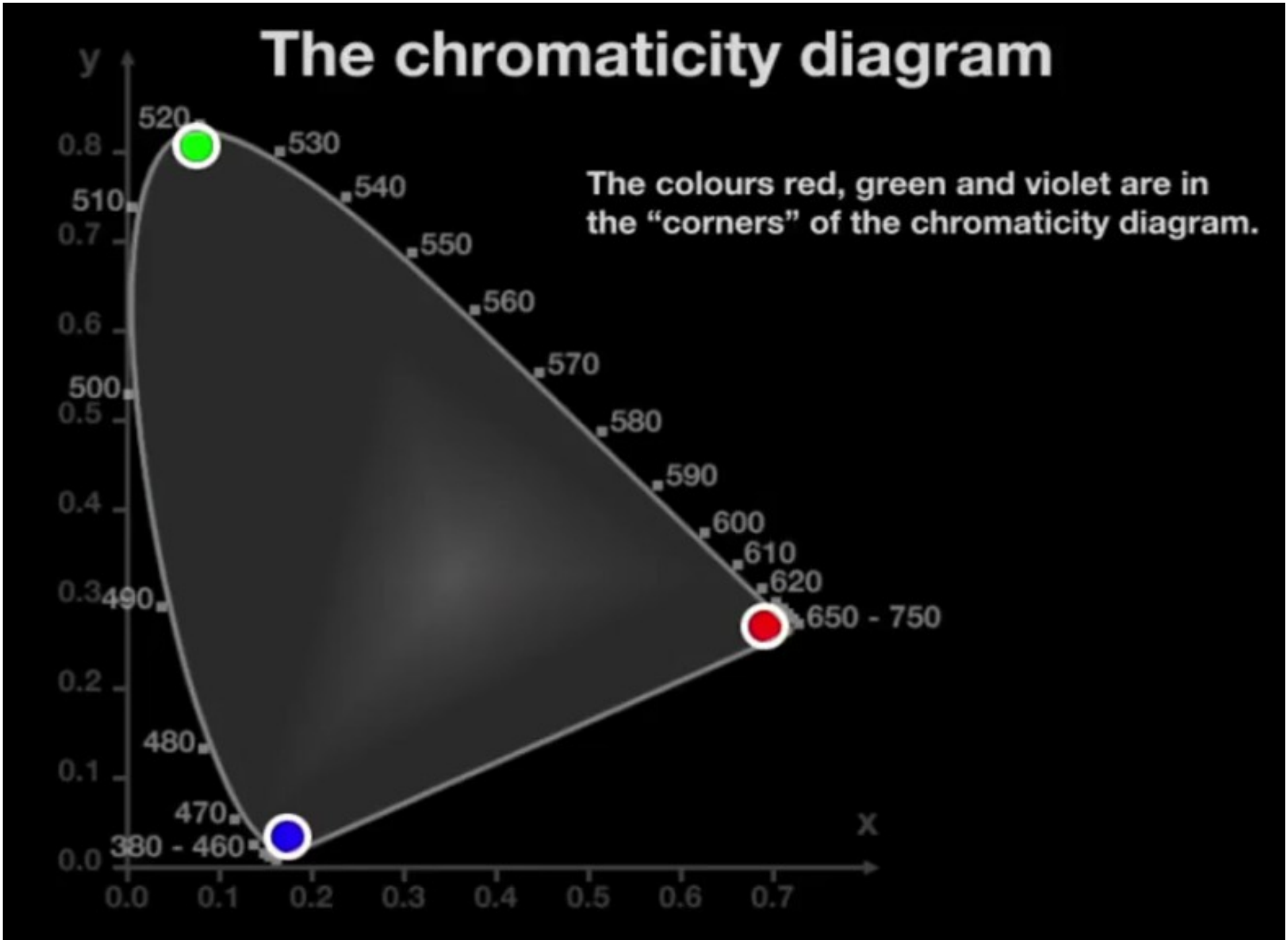

The chromaticity diagram
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
The area inside the gamut shows all the colours that can be made from mixing the 3 colours in the corners.

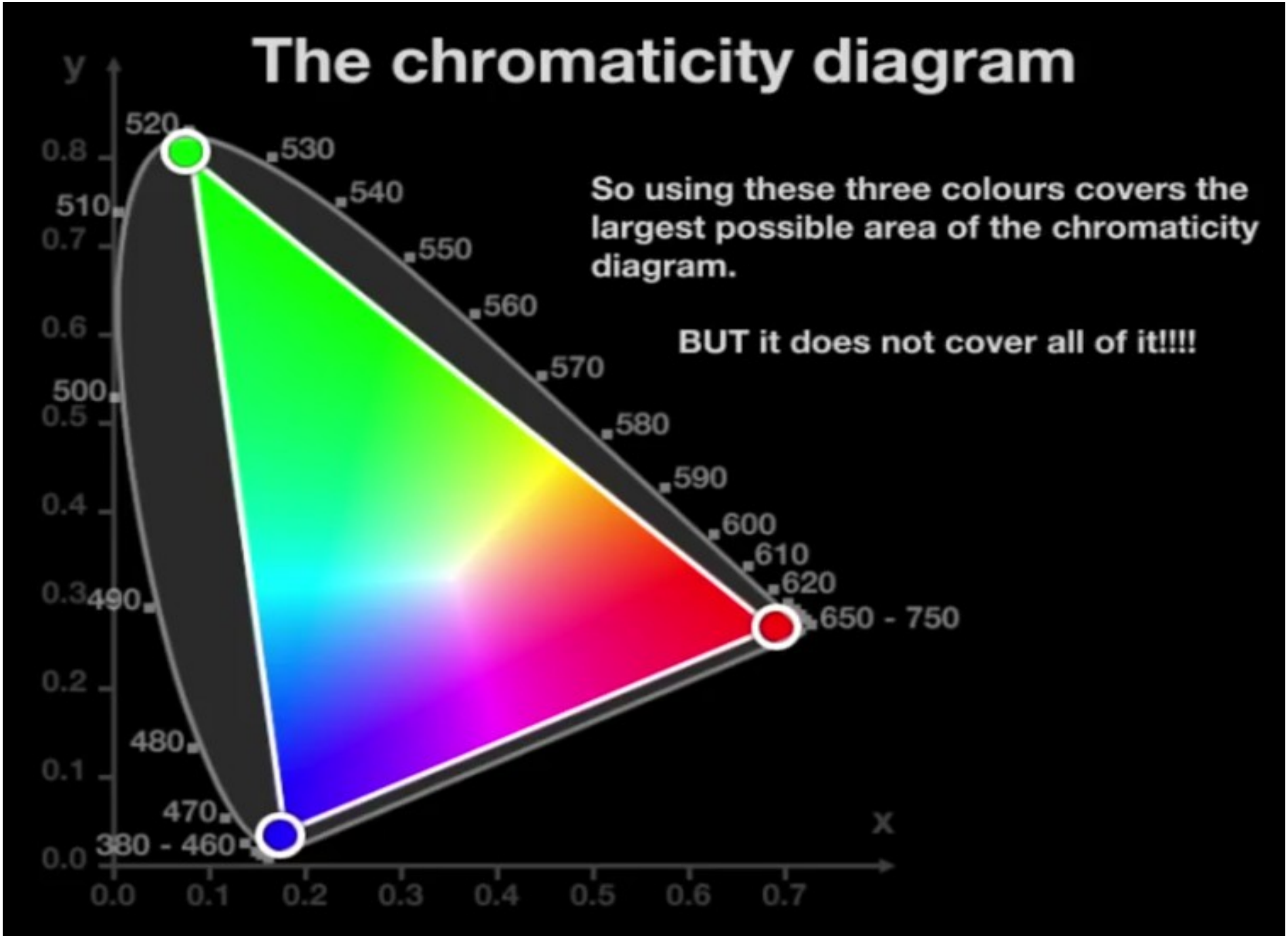

The chromaticity diagram
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
The edges of the gamut are the colours made by mixing the two colours at the end of the line (as we saw before).

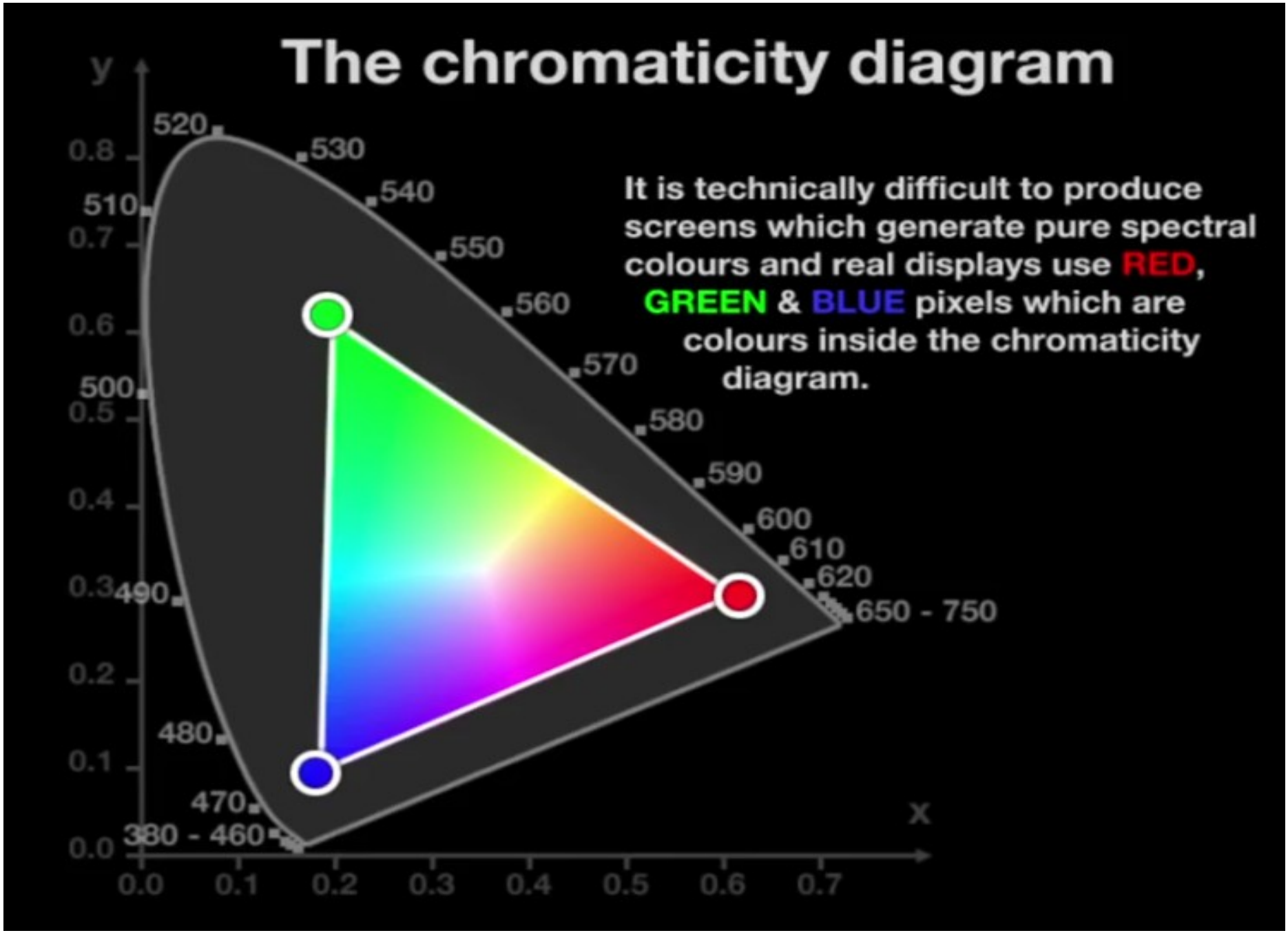
The chromaticity diagram

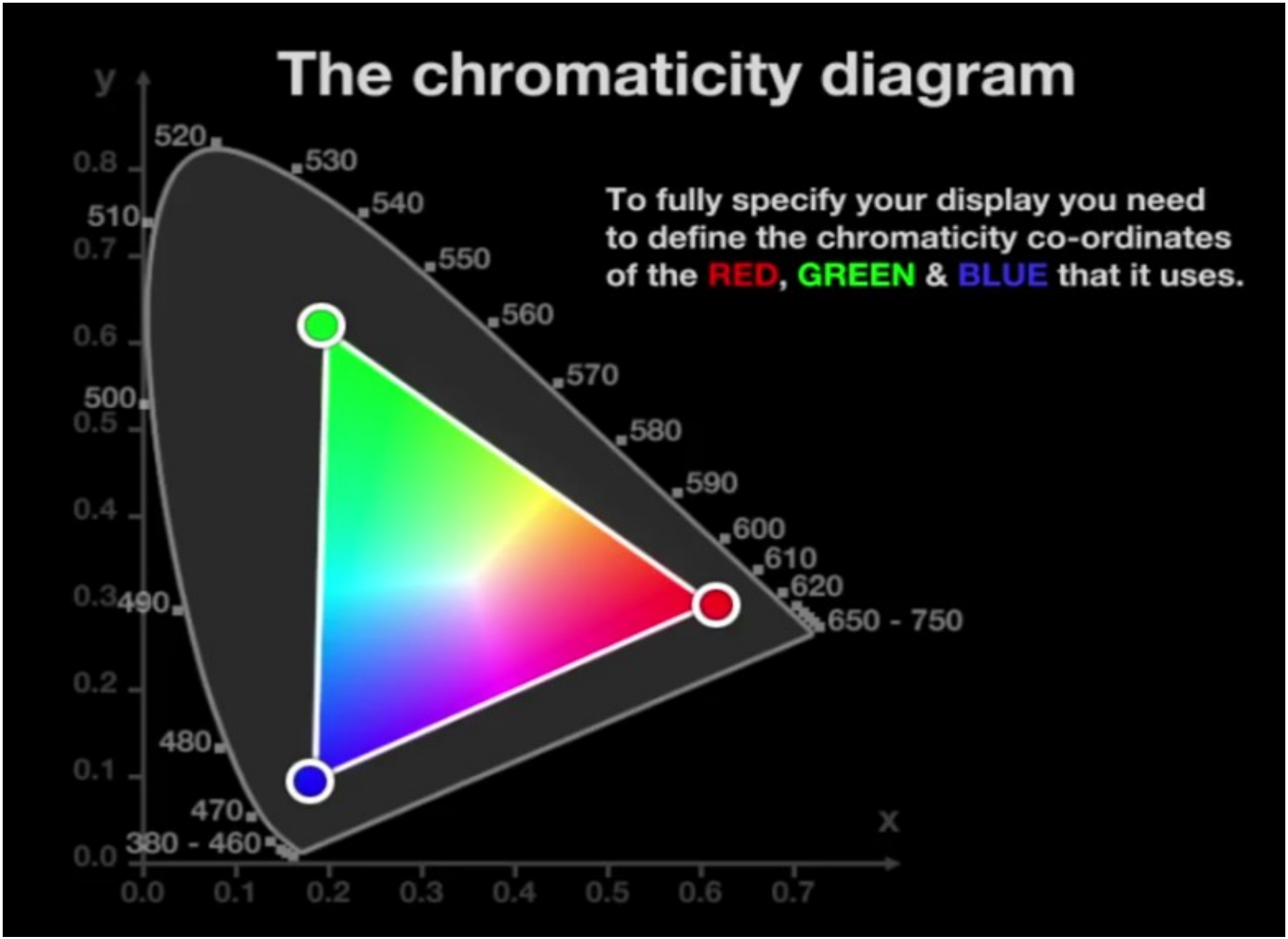


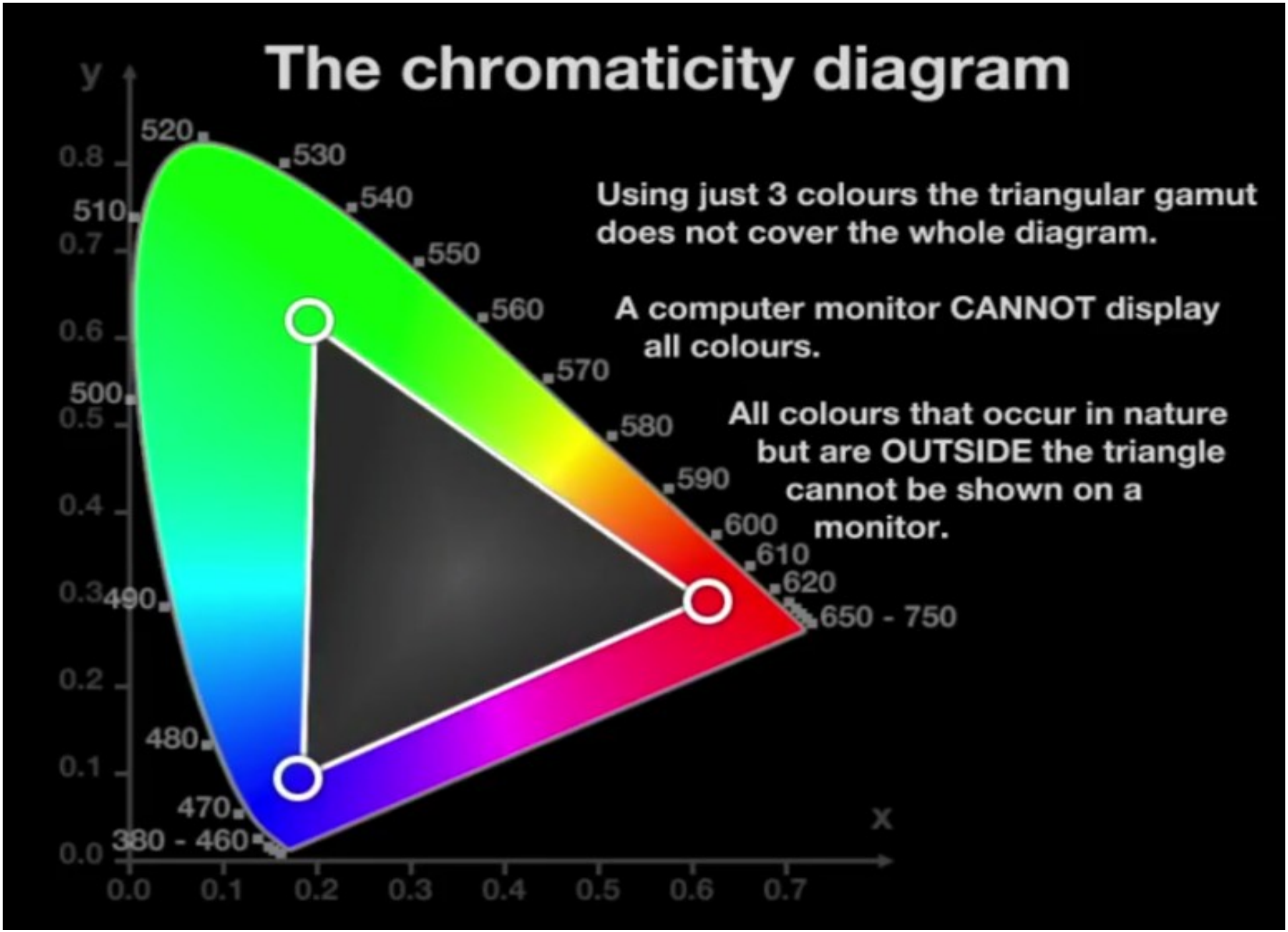
Colour displays such as TVs & computer monitors use 3 different colours of pixel to generate all the other colours that can be displayed.

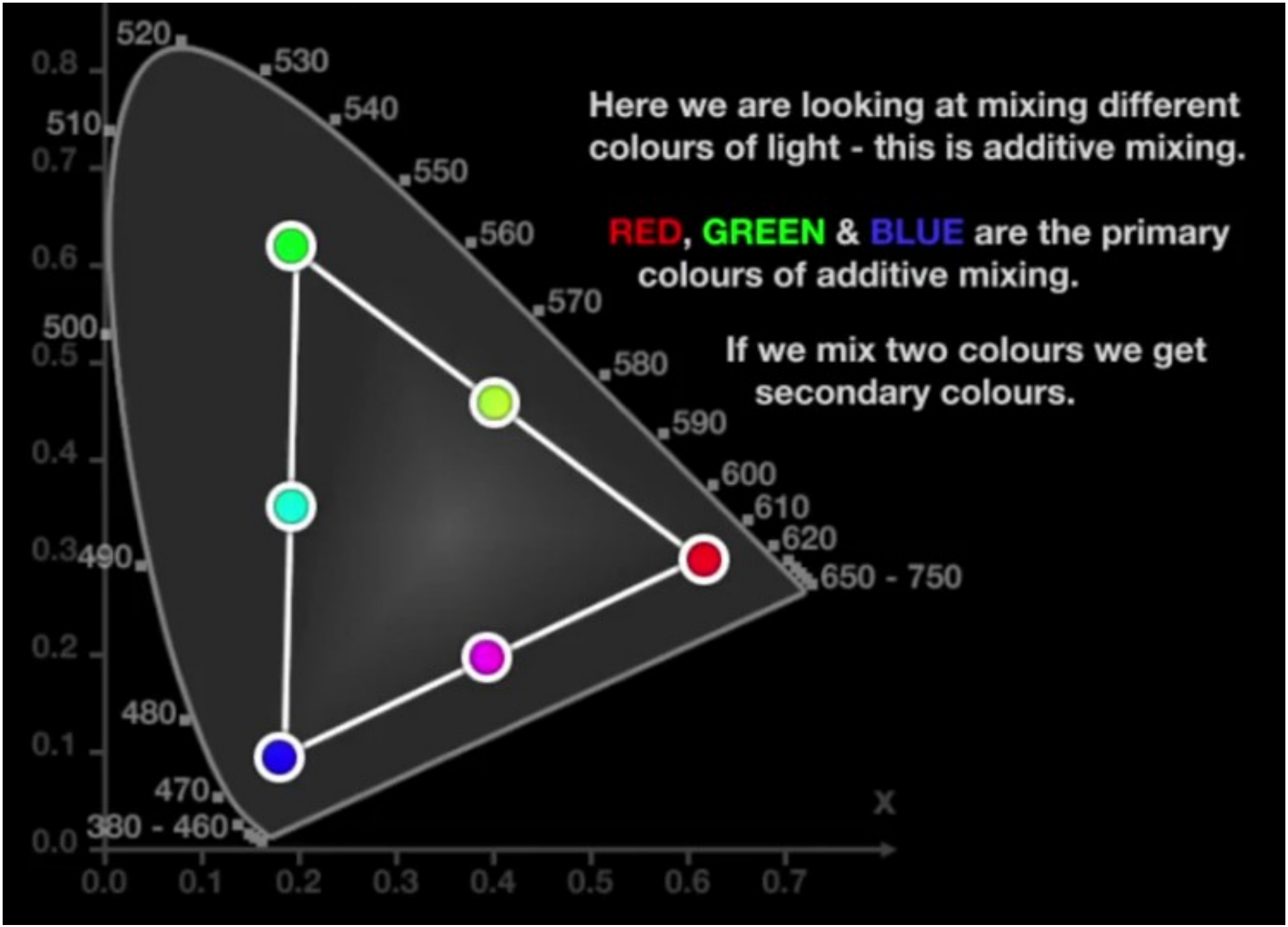

The chromaticity diagram
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
The colours red, green and violet are in the “corners” of the chromaticity diagram.

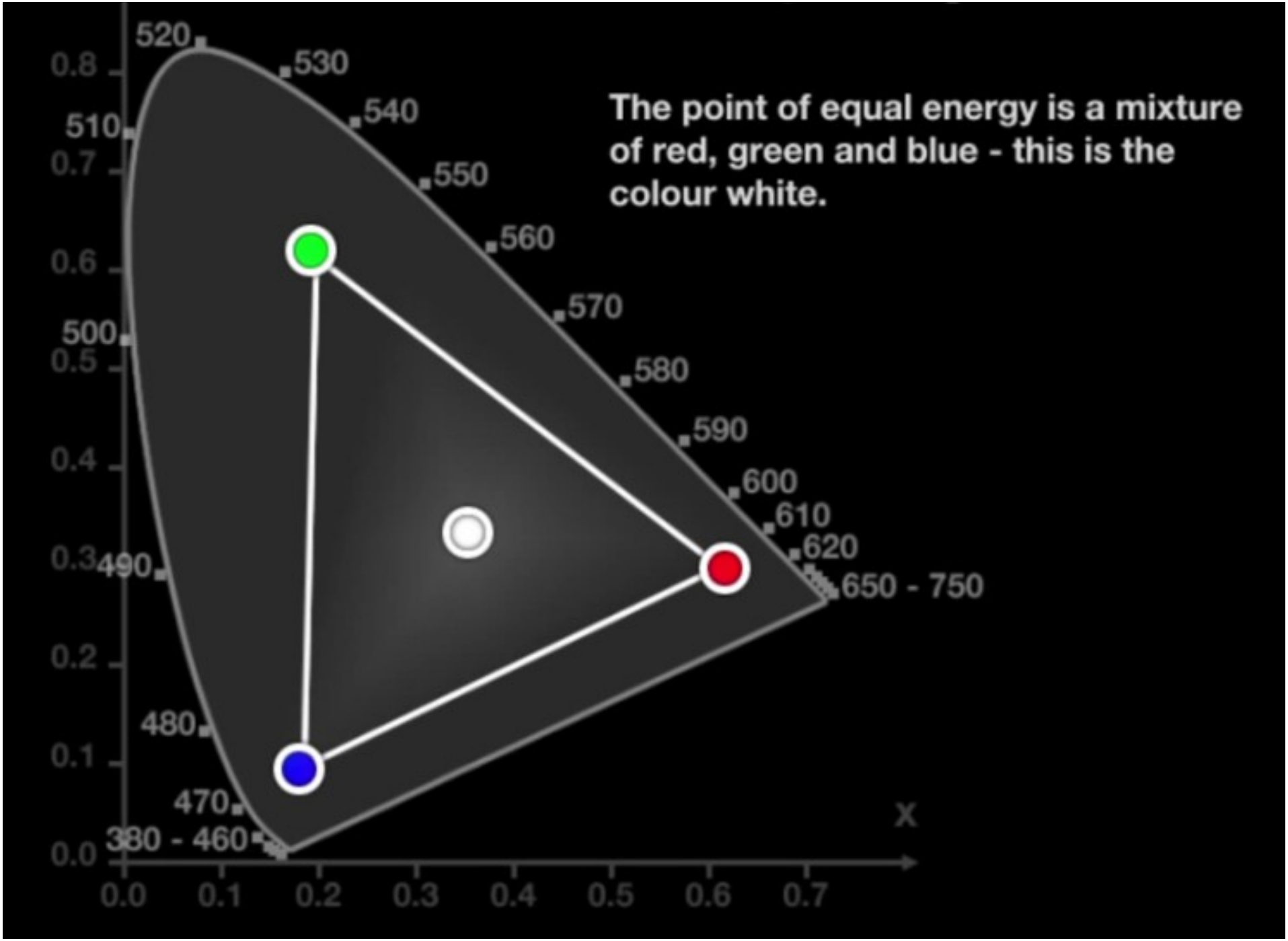

The chromaticity diagram
So using these three colours covers the largest possible area of the chromaticity diagram.
BUT it does not cover all of it!!!!
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


The chromaticity diagram
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
It is technically difficult to produce screens which generate pure spectral colours and real displays use RED, GREEN & BLUE pixels which are colours inside the chromaticity diagram.


The chromaticity diagram
To fully specify your display you need to define the chromaticity co-ordinates of the RED, GREEN & BLUE that it uses.
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7


The chromaticity diagram
y
x
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Using just 3 colours the triangular gamut does not cover the whole diagram.
A computer monitor CANNOT display all colours.
All colours that occur in nature but are OUTSIDE the triangle cannot be shown on a monitor.


520
530
540
550
560
570
580
590
600
610
620
650 - 750
490
480
470
380 - 460
0.8
510
0.7
0.6
500
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
X
Here we are looking at mixing different colours of light - this is additive mixing.
RED, GREEN & BLUE are the primary colours of additive mixing.
If we mix two colours we get secondary colours.


The point of equal energy is a mixture of red, green and blue - this is the colour white.
520
530
540
550
560
570
580
590
600
610
620
650 - 750
510
500
490
480
470
380 - 460
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
X

- The CIE chromaticity diagram is useful in many ways:

1. It identifies complementary colors
2. It allows to define color gamuts or color ranges , that show the effect of adding colors together
3. It allows us to measure the dominant wavelength of any color in chromaticity diagram.

1) To identify complementary colors

- The Fig. 10.3.3 represents the complementary colours on the chromaticity diagram.
- The straight line joining colours represented by points D and E passes through point C (represents white light).
- This means that when we mix proper amounts of the two colours D and E in Fig. 10.3.3, we can obtain white light. Therefore, colours D and E are complementary colours and with point C on the chromaticity diagram we can identify the complement colour of the known colour.

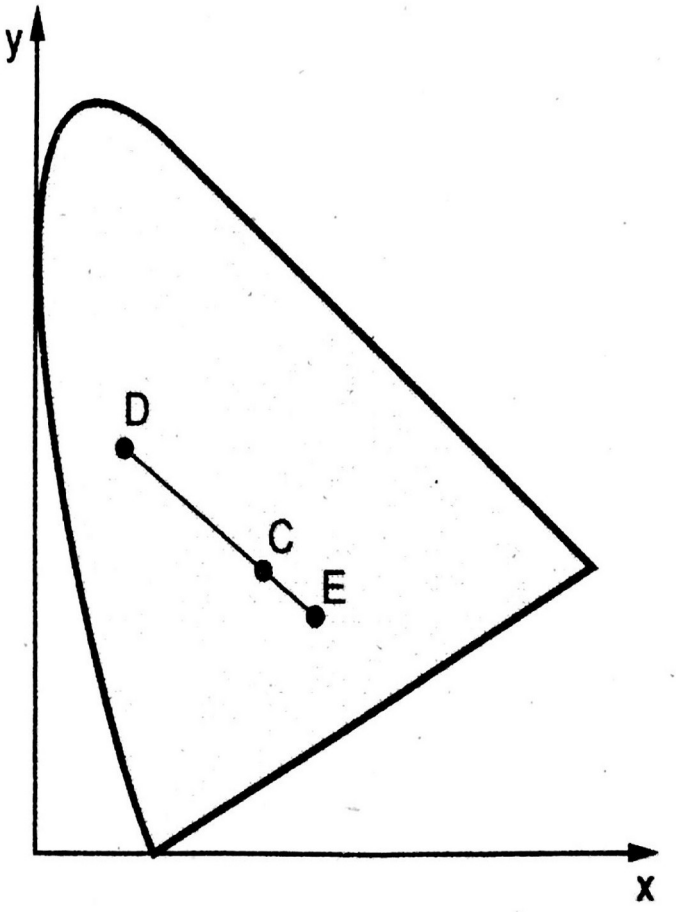


Fig. 10.3.3 Complementary colours on chromaticity diagram

2) To define color gamuts or color ranges

- Colour gamuts are represented on the chromaticity diagram as a straight line or as a polygon.
- Any two colours say A and B can be added to produce any colour along their connecting line by mixing their appropriate amounts.
- The colour gamut for three points, such as D, E and F in Fig. 10.3.4, is a triangle with three colour points as vertices.
- The triangle DEF in Fig. 10.3.4, shows that three primaries can only generate colours inside or on the bounding edges of the triangle.
- The chromaticity diagram is also useful to determine the dominant wavelength of a colour.

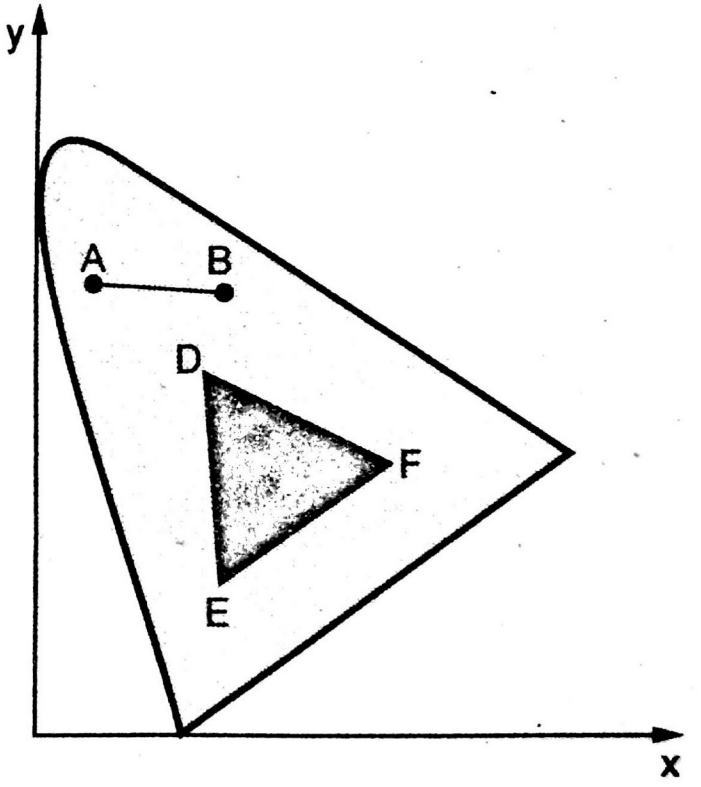


Fig. 10.3.4 Definition of colour gamuts on the chromaticity diagram

3) To identify the dominant wavelength of any color

- For colour point D in the Fig. 10.3.5, we can draw a straight line from C through D to intersect the spectral curve at point E.
- The colour D can then be represented as a combination of white light C and the spectral colour E. Thus, the dominant wavelength of D is E.
- This method for determining dominant wavelength will not work for colour points that are between C and the purple line because the purple line is not a part of spectrum.

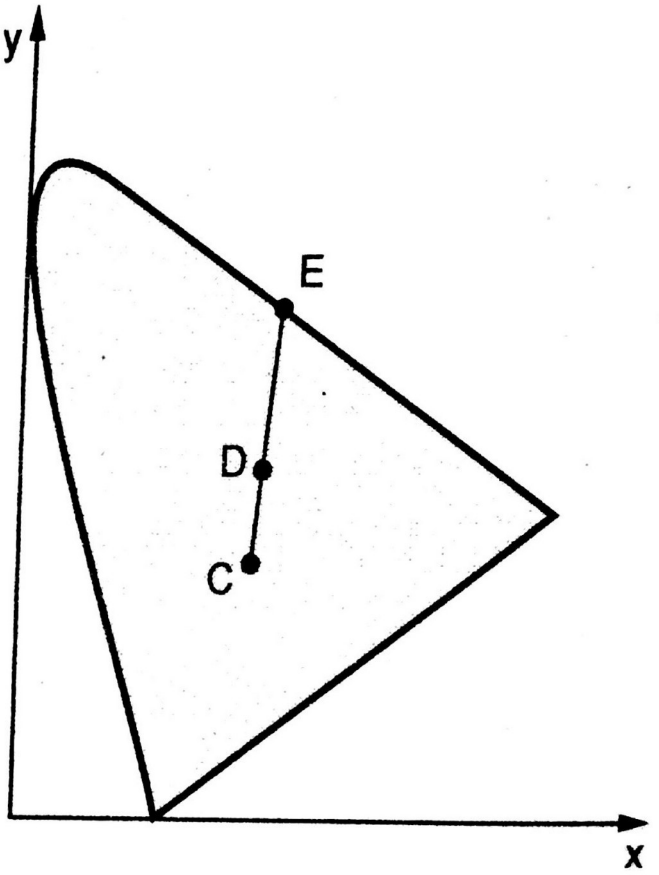


Fig. 10.3.5 Determination of dominant wavelength on the chromaticity diagram

Color Models

- The purpose of color model (color space or color system) is to facilitate specification of colors in some standard way.
- A color model provides a coordinate system and a subspace in it where each color is represented by a single point.
- There are 3 hardware oriented color models:

1) **RGB:** Used for color CRT monitor
2) **YIQ:** Used for broadcast TV color system
3) **CMY:** Used for color printing devices

- There are another class of color models which are depend on hue, saturation and brightness. They are: **HSV, HLS, HVC** models

RGB Model

- The red, green and blue (RGB) color model used for color CRT monitor.
- In this model, primary colors - red, green and blue are added together to get the resultant color. So this is **additive model.**
- We can represent this color model with the unit cube defined on R, G and B axes.
- The vertices of the cube on the axes represent the primary colors.
- The remaining vertices represent the complementary color for each of the primary colors.
- Main diagonal of cube have equal amount of each primary so represents grey levels. The end at the origin of the diagonal represents black (0,0,0) and other end represents white (1,1,1).
- Thus each color point within the bounds of cube is represented by triple (R,G,B) where R,G and B are assigned in the range from 0 to 1.

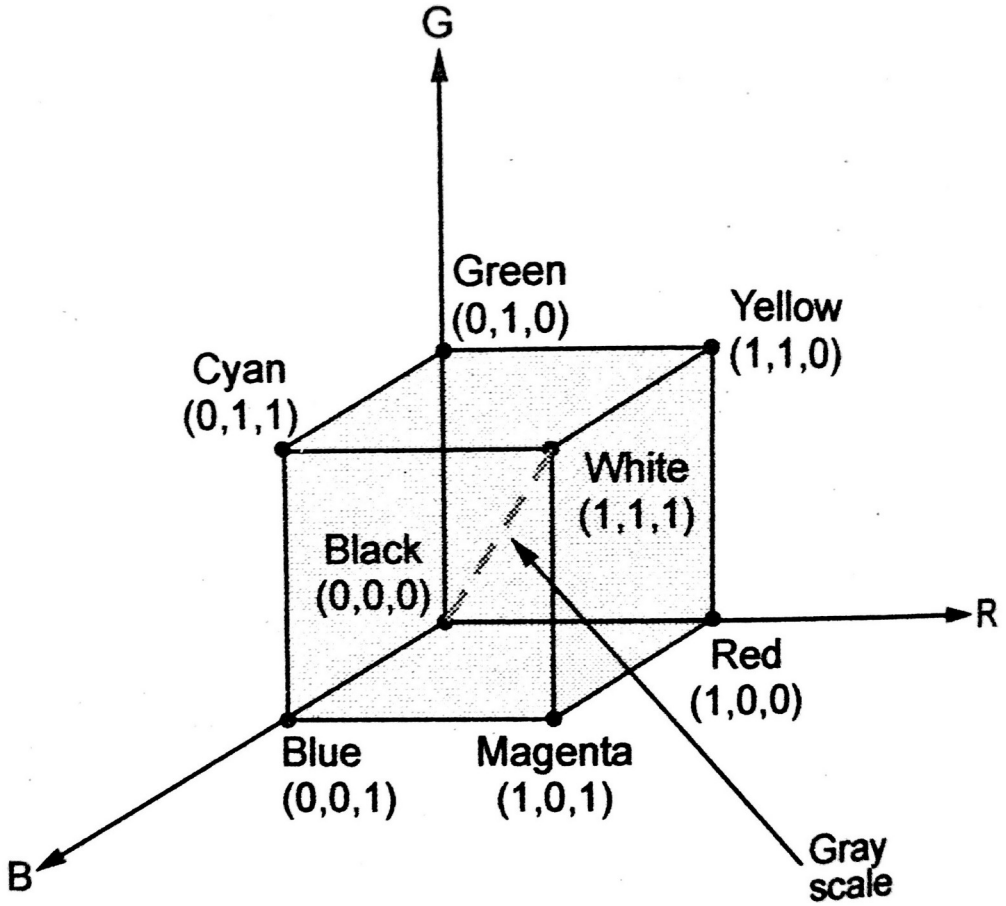
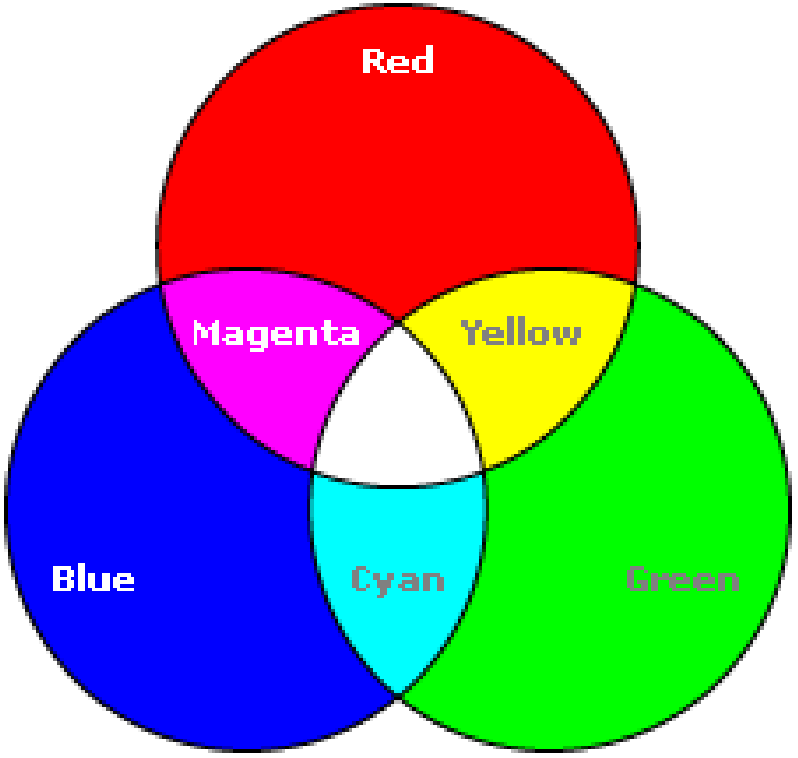
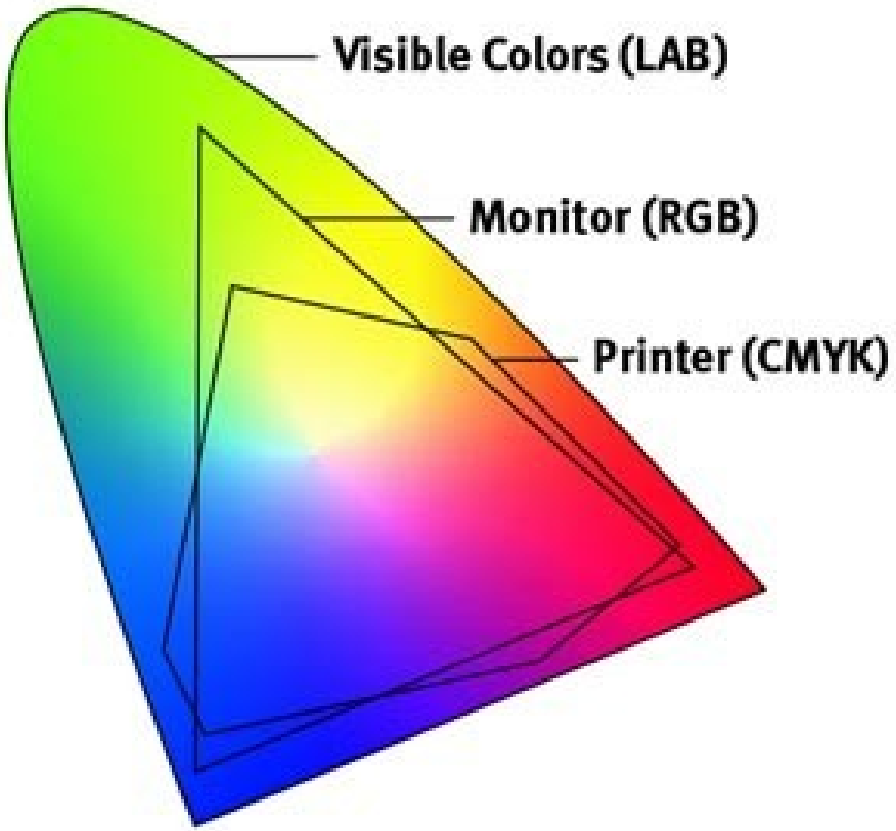


Fig. 10.4.1 The RGB cube



- Color gamut for RGB model Vs CMYK model in CIE Chromaticity diagram is given as:



CMY Model

- In CMY model cyan, magenta and yellow are used as a primary colors.
- This model is used for describing color output to hard-copy devices (e.g. printers, plotters etc.)
- The subset of cartesian co-ordinate system for CMY model is similar to that of RBG model. Difference is axes are made up of cyan, magenta and yellow colors. White instead of black is at origin.
- Colors are specified by what is removed or subtracted from white rather than what is added to blackness.
- Cyan can be formed by adding green and blue light. Therefore, when white light is reflected from cyan colored ink, the reflected light does not have red component. That is, red color is absorbed or subtracted by the ink.

- Fig. 10.6.1 shows the cube representation for CMY model.
- As shown in the Fig. 10.6.1, point (1,1,1) represents black, because all components of the incident light are subtracted.
- The point (0, 0, 0), the origin represents white light.
- The main diagonal represents equal amount of primary colours, thus the gray colours.
- A combination of cyan and yellow produces green light, because the red and blue components of the incident light are absorbed.
- Other colour combinations are obtained by a similar subtractive process.
- It is possible to get CMY representation from RGB representation as follows,

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

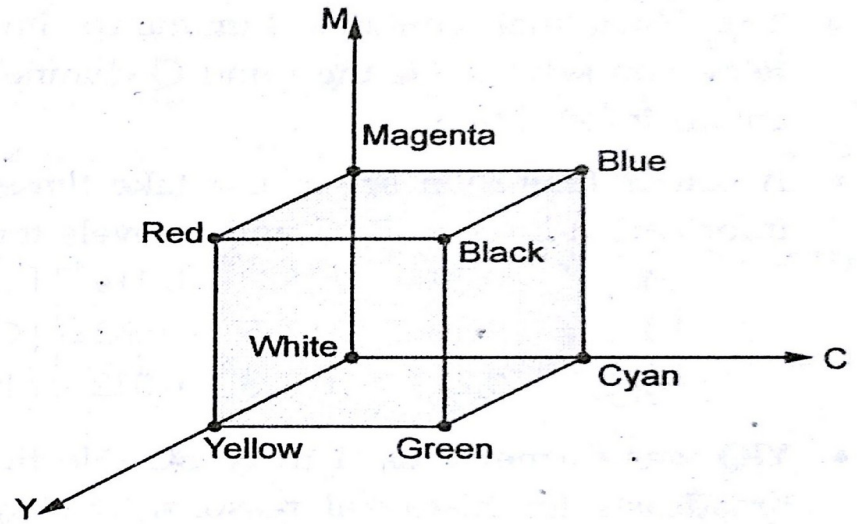


Fig. 10.6.1 The CMY cube

- The unit column vector is the RGB representation for white and the CMY representation for black. The conversion from RGB to CMY is then can be given as,

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

- The CMY color space is **subtractive.** Therefore, white is at (0.0, 0.0, 0.0) and black is at (1.0, 1.0, 1.0). If you start with white and subtract no colors, you get white. If you start with white and subtract all colors equally, you get black.

CMYK Model:

- The CMYK color space is a variation on the CMY model. It adds black (**C**yan, **M**agenta, **Y**ellow, and blac**K**).
- The CMYK color space closes the gap between theory and practice. In theory (i.e. in CMY model), if you mix C, M and Y equally then you will get black color. However, experience with various types of inks and papers has shown that when equal components of cyan, magenta, and yellow inks are mixed, the result is usually a dark brown, not black. Adding black ink to the mix solves this problem.
- The CMY and CMYK colors models are used in printing devices.

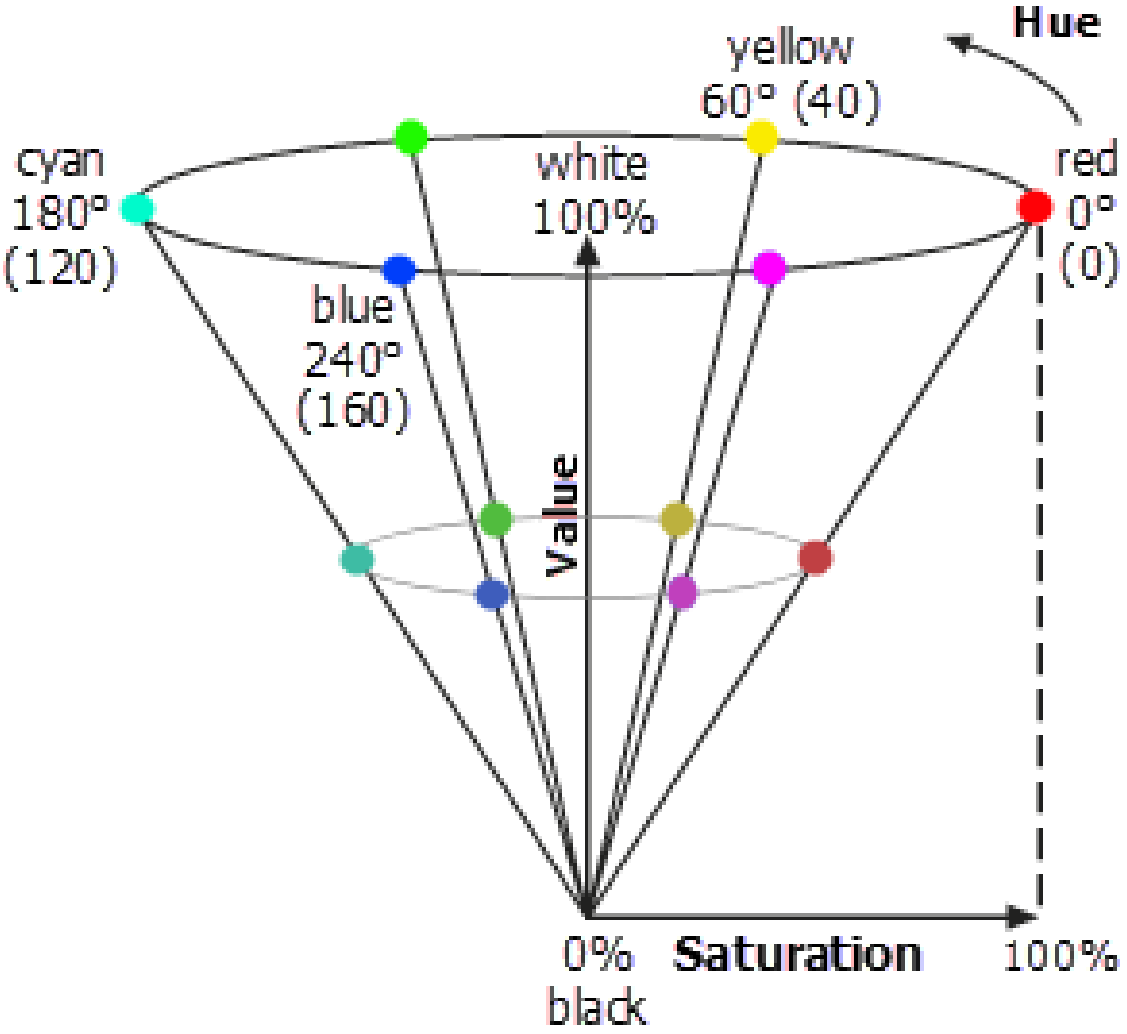
HSV Model

- HSV is named as such for three values: hue, saturation, and value.
- This color space describes colors (hue) in terms of their shade (saturation or amount of gray) and their brightness value.
- The HSV color wheel is depicted as a cone or cylinder, but always with these three components:

  Hue: Hue is the color portion of the color model, and is expressed as a number from 0 to 360 degrees.

  Saturation: Saturation is the amount of gray in the color. Saturation is viewed on a range from just 0-1 (or 0-100 percent), where 0 is gray and 1 is a primary color.

  Value (or Brightness): Value works describes the brightness or intensity of the color, from 0-100 percent, where 0 is completely black and 100 is the brightest and reveals the most color.


Hue
yellow
60° (40)
cyan
180°
(120)
white
100%
red
0°
(0)
blue
240°
(160)
Value
0%
black
Saturation
100%

- The HSV color space is widely used to generate high quality computer graphics. In simple terms, it is used to select various different colors needed for a particular picture. An HSV color wheel is used to select the desired color. A user can select the particular color needed for the picture from the color wheel.
- The HSV model describes colors similarly to how the human eye tends to perceive color.

Illumination model

- An **illumination model** (or **lighting model** or **shading model**) is used to calculate the intensity of light that we should see at a given point on the surface of an object.

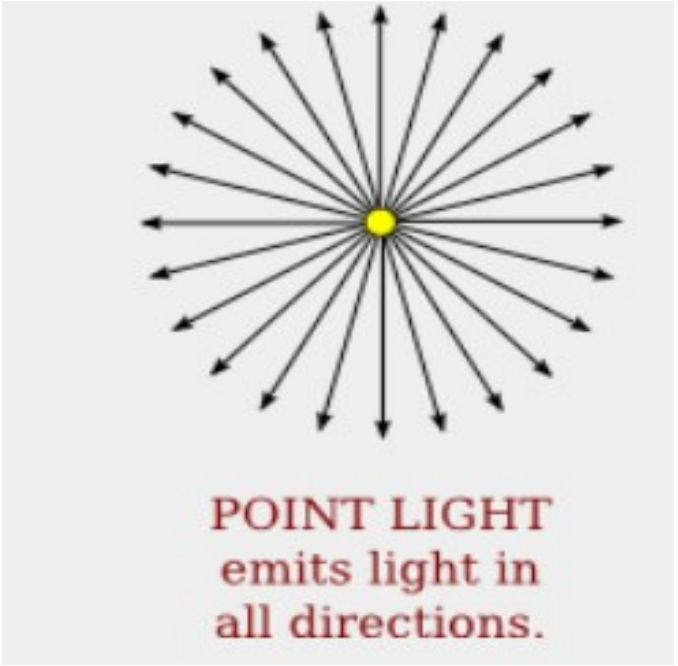
- **Surface rendering** means a procedure for applying a lighting model to obtain pixel intensities for all the projected surface positions in a scene.
- A surface-rendering algorithm uses the intensity calculations from an illumination model to determine the light intensity for all projected pixel positions for the various surfaces in a scene.
- Surface rendering can be performed by applying the illumination model to every visible surface point.

Light sources

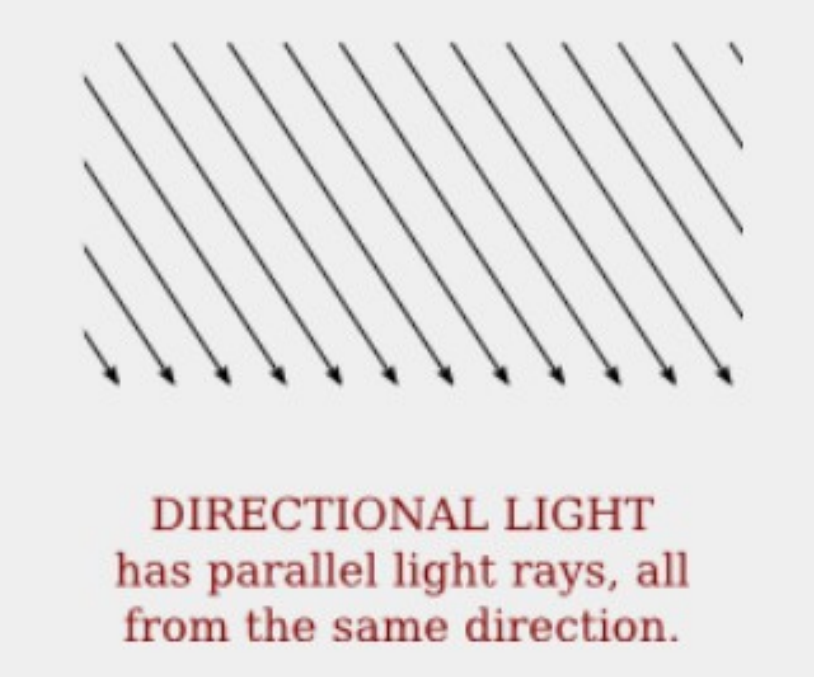
- What the human eye sees is a result of light coming off of an object or other light source and striking receptors in the eye.
- In order to understand and model this process, it is necessary to understand different light sources (point source, distributed or directional light source and spotlight) and the ways that different materials reflect those light sources (Ambient, diffuse reflection and specular reflection).

- **Types of light sources:**

1) **Point light -** a light that gives off equal amounts of light in all directions. E.g. A bare bulb hanging from a cord is essentially a point light.



2) Directional light (0r Distributed light) - produced by a light source an infinite distance from the scene. All of the light rays emanating from the light strike the polygons in the scene from a single parallel direction, and with equal intensity everywhere. E.g. Sunlight.



3) Spotlight - light that radiates light in a cone with more light in the center of the cone, gradually tapering off towards the sides of the cone. E.g. a flashlight, car headlight etc

- When light is incident on an opaque surface, part of it is reflected and part is absorbed. (Transparent objects are the objects through which light can pass easily. Translucent objects are the objects through which light can pass only partially. Opaque objects are the objects through which light can't pass.)
- The amount of incident light reflected by a surface depends on the type of material. Shiny materials reflect more of the incident light, and dull surfaces absorb more of the incident light.
- For an illuminated transparent surface, some of the incident light will be reflected and some will be transmitted through the material

- Illumination consists of 3 components:

  1) Ambient light

  2) Diffuse reflection

  3) Specular reflection

1) Ambient light

- It is the combination of light reflections from various surfaces to produce a uniform illumination called the ambient light.
- It simulates light that has been reflected so many times from so many surfaces it appears to come equally from all directions.
- Its intensity is constant over polygon's surface.
- Its intensity is not affected by the position or orientation of the polygon in the world.
- The position of viewer is not important.
- e.g.

➔ Ambient light is the light that enters a room and bounces multiple times around the room before lighting a particular object.

➔ moonlight

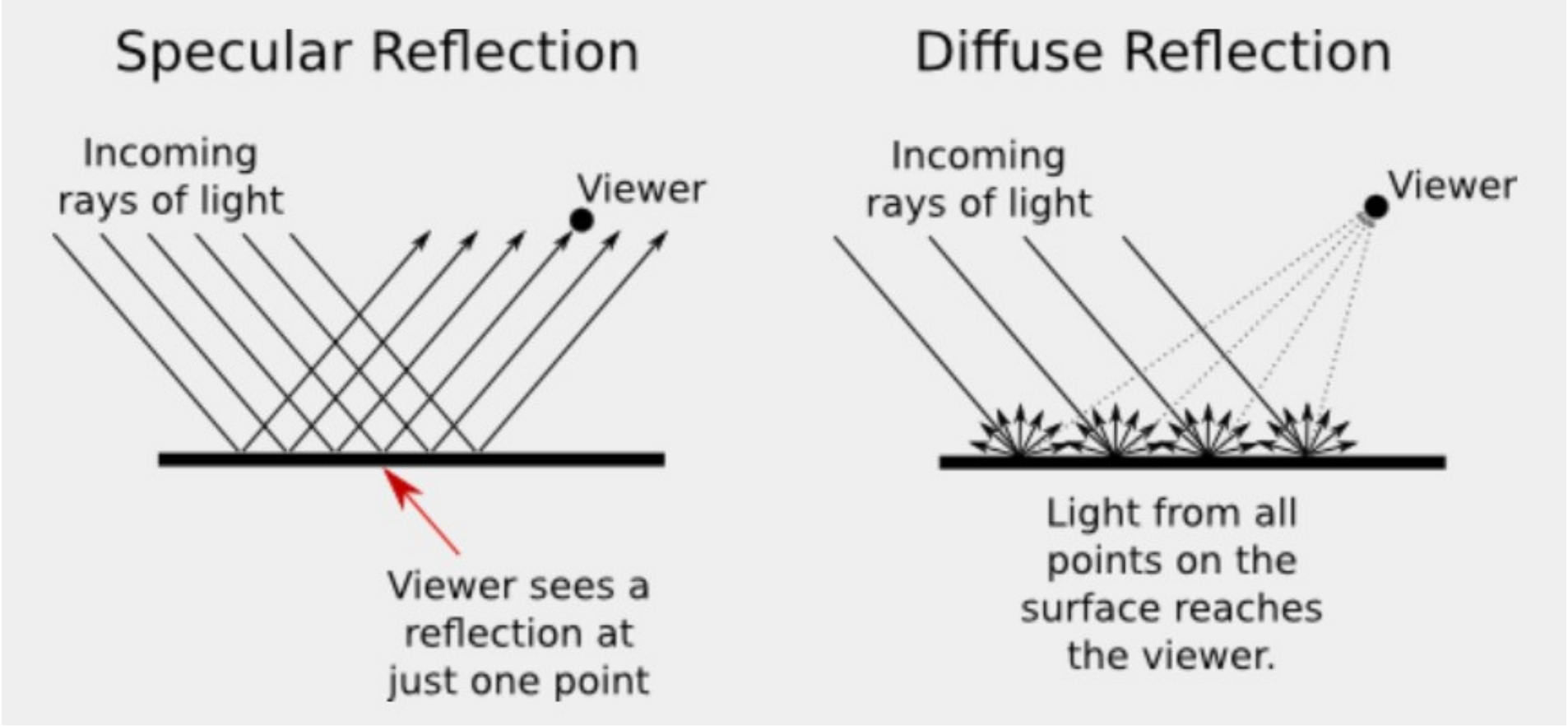
2) Diffuse reflection (Lambertian Reflection)

- Diffuse light represents direct light hitting a surface.
- The Diffuse Light contribution is dependent on the incident angle. For example, light hitting a surface at a 90 degree angle (angle of incident=0) contributes more than light hitting the same surface at a 5 degrees.
- Using a point light:

- ✔ comes from a specific direction
- ✔ reflects off of dull surfaces (lambertian surfaces)
- ✔ light reflected with equal intensity in all directions
- ✔ brightness depends on theta (angle of incidence) - angle between surface normal (N) and the direction to the light source (S)
- ✔ position of viewer is not important

3) Specular reflection

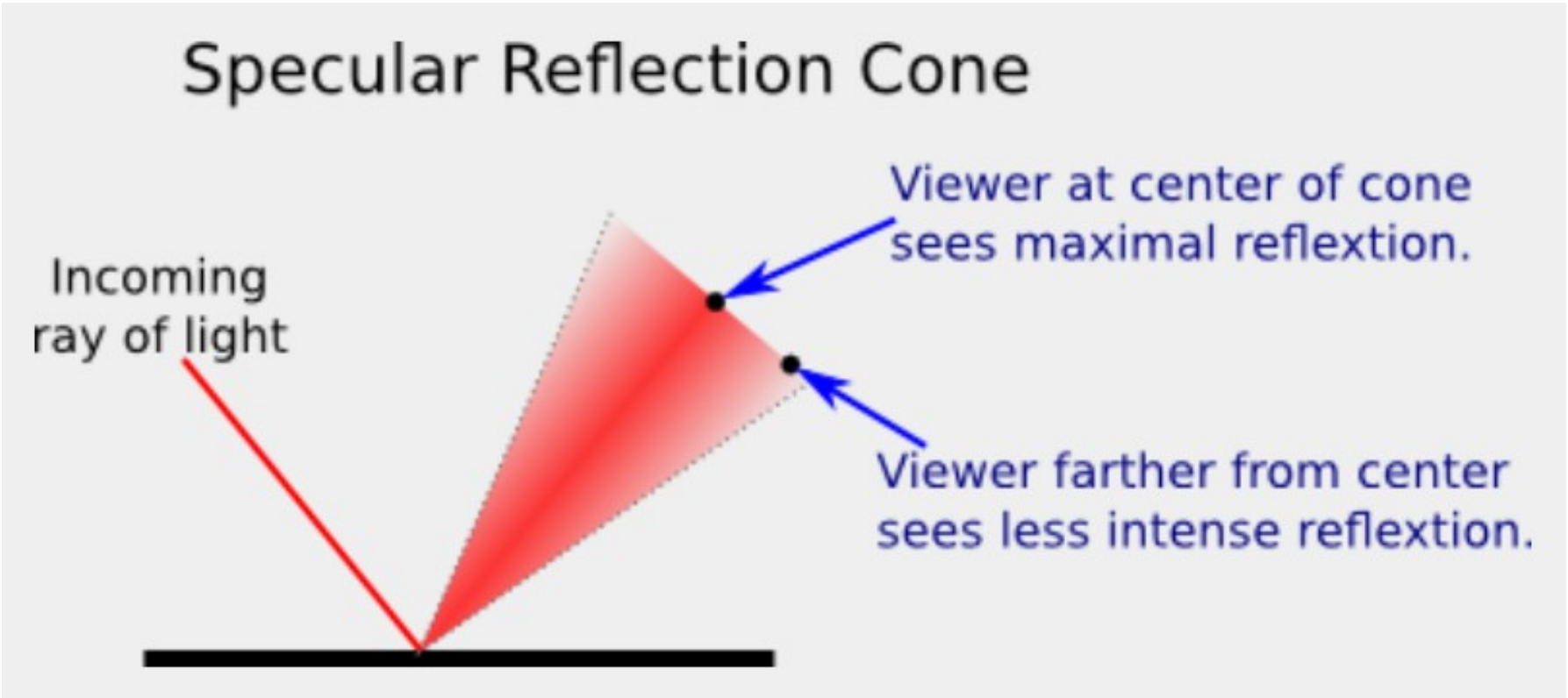
- Specular light is the white highlight reflection seen on smooth, shinny objects. Specular light is dependent on the direction of the light, the surface normal and the viewer location.
- This highlighting effect is more pronounced on shiny surfaces than on dull
- Mirror is perfect specular surface. Apple, Shiny metal or plastic has high specular component .
- Position of the viewer is important in specular reflection.


Specular Reflection
Incoming
rays of light
Viewer
Viewer sees a
reflection at
just one point
Diffuse Reflection
Incoming
rays of light
Viewer
Light from all
points on the
surface reaches
the viewer.

Ambient

Ambient + Diffuse

Ambient + Diffuse + Specular


Specular Reflection Cone
Viewer at center of cone
sees maximal reflextion.
Incoming
ray of light
Viewer farther from center
sees less intense reflextion.

Warn Model

- So far, we have modeled light sources as points radiating uniformly in all directions. The Warn model improves this by letting a light source be aimed in a certain direction.
- Light sources are modeled as points on a reflecting surface, using the Phong model for the surface points.
- Then the intensity in different directions is controlled by selecting values for the Phong exponent. (m in the Phong model: $\cos^{m}\alpha$)
- The Warn model provides a method for simulating studio lighting effects by controlling light intensity in different directions.

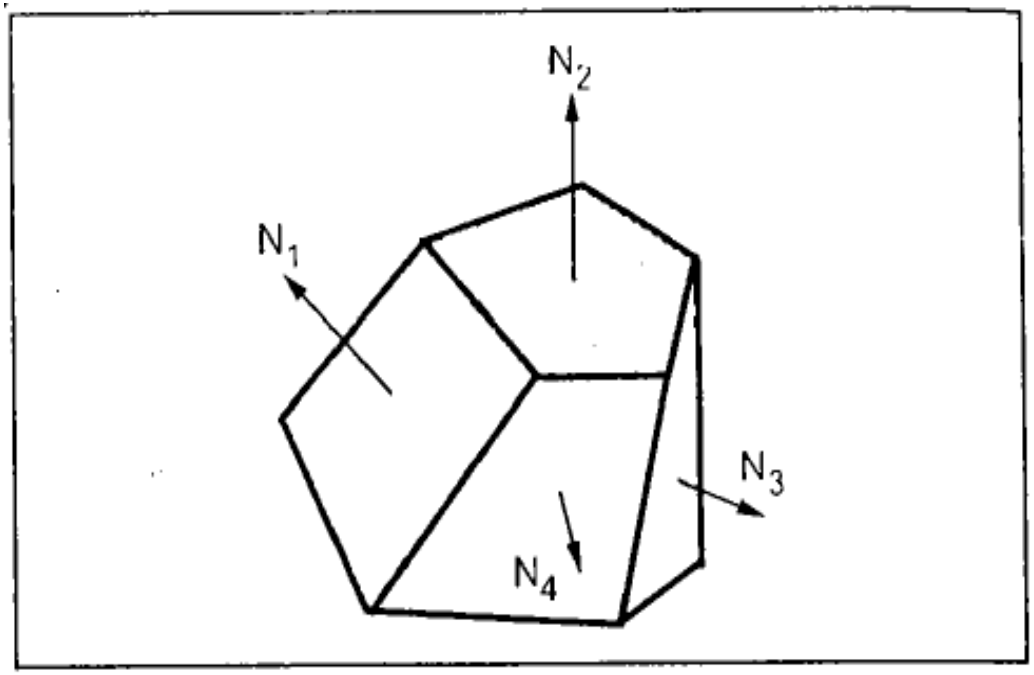
Shading algorithms

- We can shade any surface by calculating the surface normal at each visible point and applying the desired illumination model at that point.
- But this method is expensive. Following are more efficient shading methods for surfaces defined by polygons:

1. **Constant-Intensity Shading**
2. **Gouraud shading**
3. **Phong shading**
4. **Halftone shading**

1. Constant-Intensity Shading

- This is fast and simplest method for polygon shading which is also known as **faceted shading** or **flat shading.**
- In this method, illumination model is applied only once for each polygon to determine single intensity value.
- The entire polygon is then displayed with the single intensity value.



Polygons and their surface normals

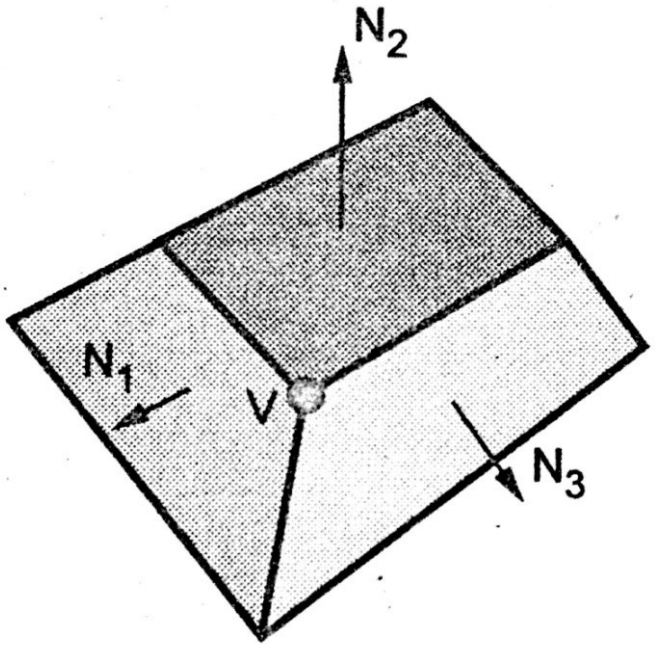
2. Gouraud shading

- In this method, the intensity interpolation technique developed by Gouraud is used, hence the name - Gouraud shading.
- Following are the steps of Gouraud Shading:

1. Determine the average unit normal vector at each polygon vertex.
2. Apply an illumination model to each polygon vertex to determine the vertex intensity.
3. Linearly interpolate the vertex intensities over the surface of the polygon.

1. Calculate average unit normal vector at each polygon vertex.

- We can obtain a normal vector at each polygon vertex by averaging the surface normals of all polygons sharing that vertex.
- As shown in figure, there are 3 surface normals N1, N2, N3 of polygons sharing vertex V.



Calculation of normal vector at polygon vertex V

normal vector at vertex V is given as

$$N_V = \frac{N_1 + N_2 + N_3}{|N_1 + N_2 + N_3|}$$

- In general, for any vertex V, we can obtain the unit vertex normal by following equation: (where n is the number of surface normals of polygons sharing the vertex V)

$$N_V = \frac{\sum_{i=1}^{n} N_i}{\left| \sum_{i=1}^{n} N_i \right|}$$

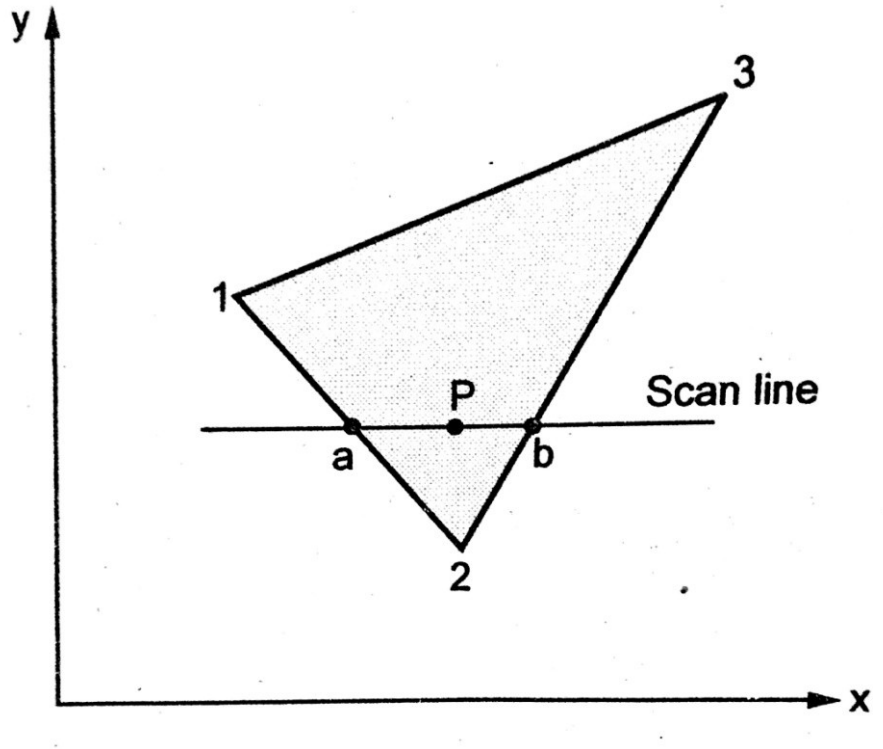
2) To find vertex intensities:

- Once we have vertex normals, their vertex intensities can be determined by applying illumination model to each polygon vertex.

- For each scan line, the intensity at the intersection of the scan line with the polygon edge is linearly interpolated from the intensities at the edge endpoints.

3) Linear interpolation of the vertex intensities over the surface of the polygon:

- Each polygon is shaded by linear interpolating of vertex intensities along each edge and then between edges along each scan line.



For example in figure, the polygon edge with endpoint vertices 1 and 2 is intersected by the scan line at point 'a'. The intensity at point 'a' can be interpolated from intensities I_1 and I_2 as,

$$I_a = \frac{y_a - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_a}{y_1 - y_2} I_2$$

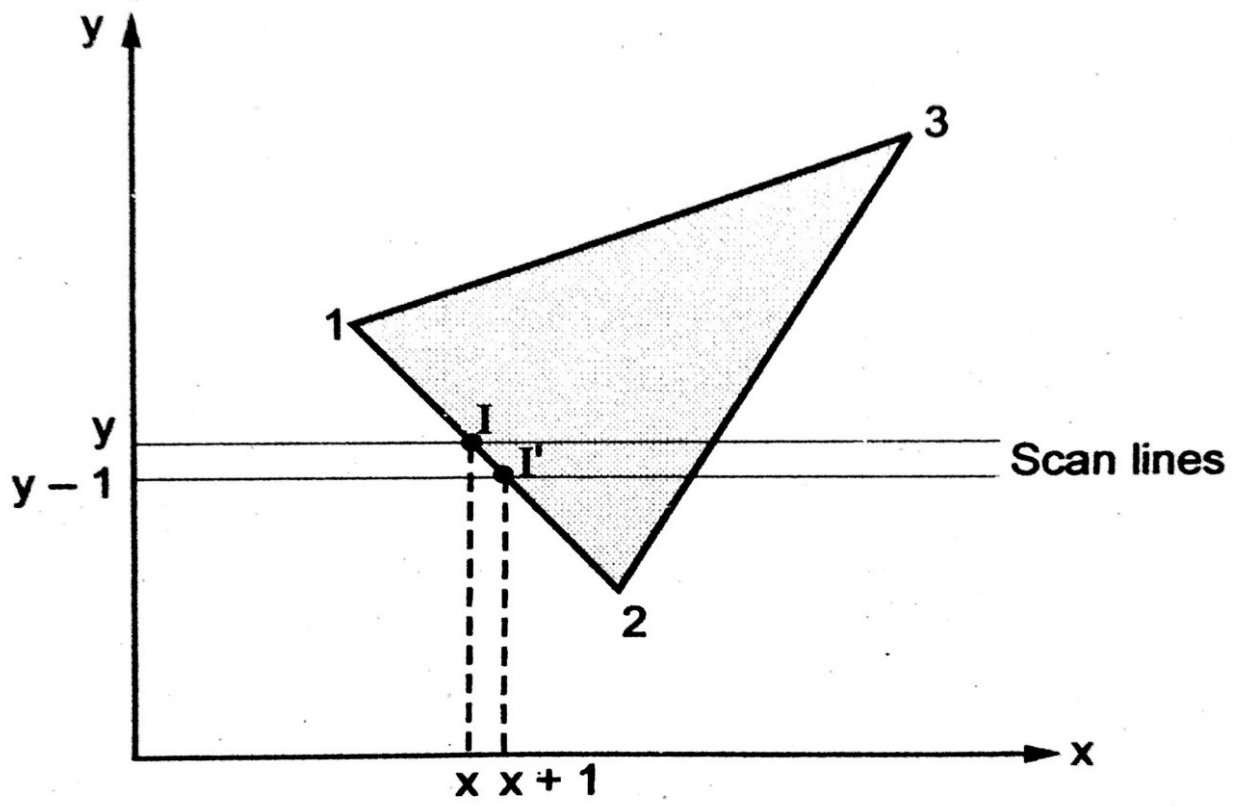
- Similarly, we can interpolate the intensity value for right intersection (point b) from intensity values I_2 and I_3 as,

$$I_b = \frac{y_a - y_2}{y_3 - y_2} I_3 + \frac{y_3 - y_b}{y_3 - y_2} I_2$$

- Once the intensities of intersection points a and b are calculated for a scan line, the intensity of an interior point (such as P in Fig. 11.9.3) can be determined as,

$$I_p = \frac{x_b - x_P}{x_b - x_a} I_a + \frac{x_P - x_a}{x_b - x_a} I_b$$

- To eliminate the repetitive calculations, incremental calculations are used to obtain the successive edge intensity values between the scan lines and to obtain successive intensity along a scan line.



Calculation of incremental interpolation of intensity values along a polygon edge for successive scan lines

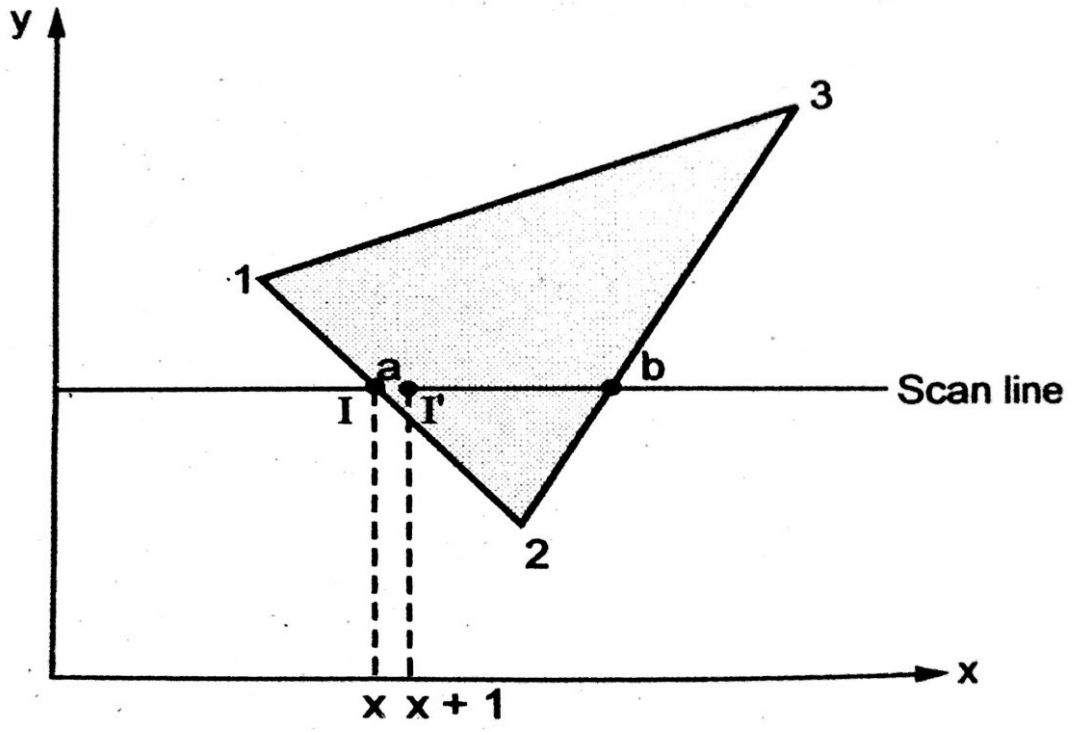
If the intensity at edge position (x,y) is interpolated as I, then we can obtain the intensity along this edge for the next scan line, y-1 as I':

$$I = \frac{y - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y}{y_1 - y_2} I_2$$

$$I' = I + \frac{I_2 - I_1}{y_1 - y_2}$$

- Similarly, we can obtain intensities at successive horizontal pixel positions along each scan line as:

$$I' = I + \frac{I_b - I_a}{x_b - x_a}$$



Calculation of incremental interpolation of intensity values

3. Phong Shading

- Phong Shading (also known as **normal-vector interpolation shading**) interpolates the surface normal vector N, instead of the intensity.
- Following steps are performed to display polygon surface using Phong shading:

1. Determine the average unit normal vector at each polygon vertex.
2. Linearly interpolate the vertex normals over the surface of the polygon.
3. Apply an illumination model along each scan line to determine projected pixel intensities for the surface points.

1) Determine the average unit normal vector at each polygon vertex:

Same as the first step in Gouraud shading.

2) Linearly interpolate the vertex normals over the surface of the polygon:

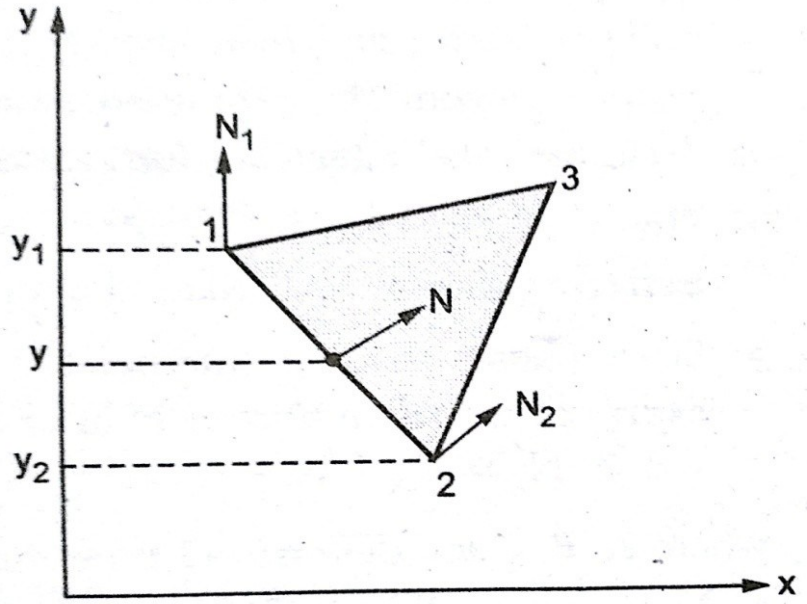


Fig. 11.9.6 Calculation of interpolation of surface normals along a polygon edge

- The normal vector N for scan line intersection point along the edge between vertices 1 and 2 can be obtained by vertically interpolating between edge endpoint normals as:

$$N = \frac{y - y_2}{y_1 - y_2} N_1 + \frac{y_1 - y}{y_1 - y_2} N_2$$

- Like Gouraud shading, here also we can use incremental methods to evaluate normals between scan lines and along each individual scan line.

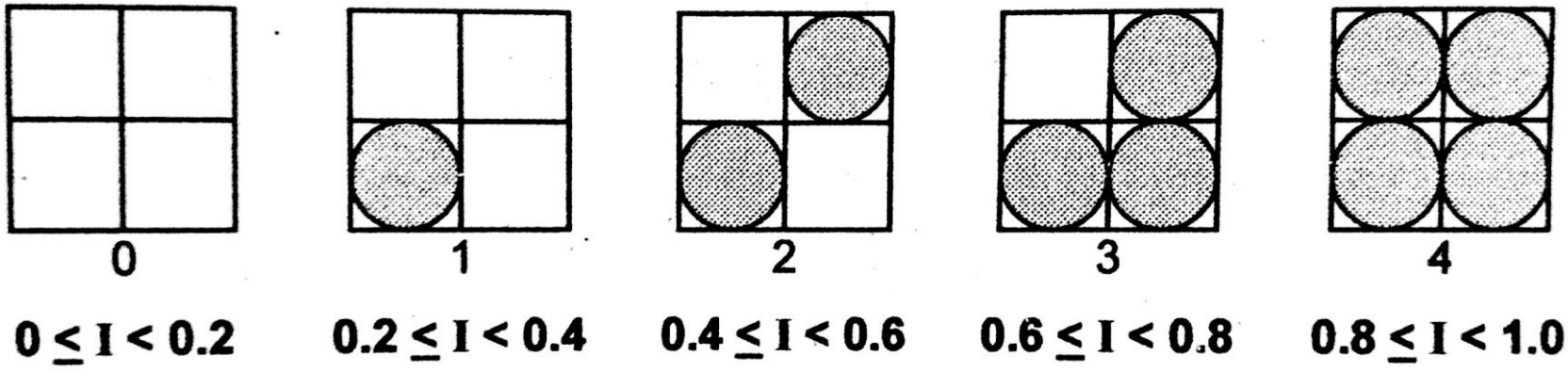
3) Apply an illumination model along each scan line to determine projected pixel intensities for the surface points:

- Once the surface normals are evaluated, the surface intensity at that point is determined by applying the illumination model.

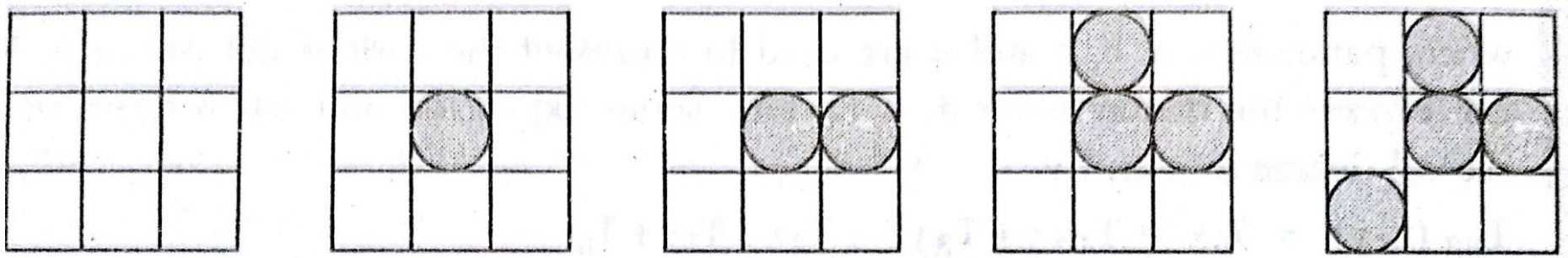
4. Halftone shading

- Many displays and hardcopy devices are bilevel. They can only produce two intensity levels (black and white).
- Halftoning is a process of increasing the number of intensities by taking combine intensity of multiple pixels in to account.
- If we view from sufficient large viewing distance, the human visual system has a tendency to average brightness over small areas, so the black dots and their white background merge and are perceived as an intermediate shade of grey.
- Halftone reproductions are usually done in pixel regions such as 2×2 or 3×3 pixels. These regions are called **halftone pattern** or **pixel pattern.**

- Following are the halftone patterns that create number of intensity levels:



2× 2 pixel patterns for creating five intensity levels



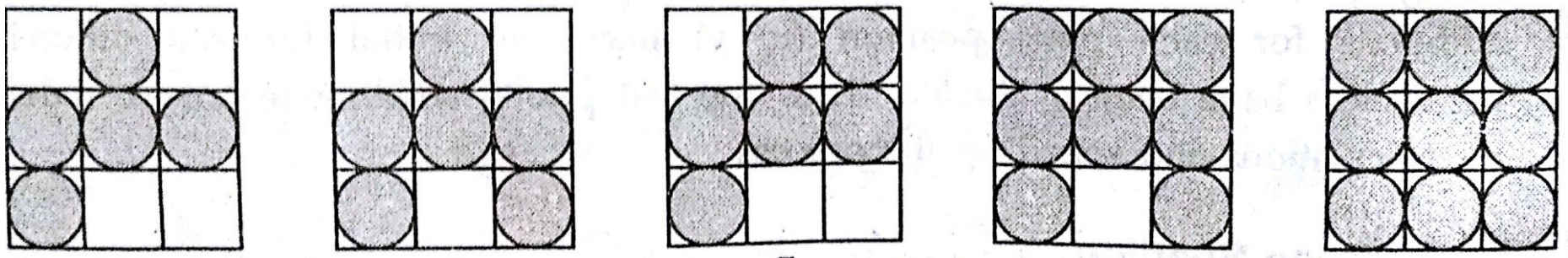
0 $0 \leq I < 0.1$

1 $0.1 \leq I < 0.2$

2 $0.2 \leq I < 0.3$

3 $0.3 \leq I < 0.4$

4 $0.4 \leq I < 0.5$



5 $0.5 \leq I < 0.6$

6 $0.6 \leq I < 0.7$

7 $0.7 \leq I < 0.8$

8 $0.8 \leq I < 0.9$

9 $0.9 \leq I < 1.0$

3× 3 pixel patterns for creating ten intensity levels

- This process is used in black and white photographs and publishing of newspapers, magazines and books.
- Newspaper photographs simulate a greyscale, despite the fact that they have been printed using only black ink.
- A newspaper picture is, in fact, made up of a pattern of tiny black dots of varying size.

Halftone: Dot size

Hidden Surfaces

- When we view a picture containing non-transparent objects and surfaces, then we cannot see those objects which are behind from objects closer to eye.
- We must remove these hidden surfaces to get a realistic screen image.
- **Surface determination** (also known as **hidden surface removal (HSR)**, **occlusion culling (OC)** or **visible surface determination (VSD)**) is the process used to determine which surfaces and parts of surfaces are not visible from a certain viewpoint.
- There are two approaches for removing hidden surface problems −

1)Object-Space method

2)Image-space method

1) **Object-space methods:** compares objects & parts of objects to each other within the scene definition to determine which surfaces, as a whole, we should label as visible.

2) **Image-space methods:** visibility is decided point by point at each pixel position on the projection plane.

- **Hidden Surface Removal consists of two steps:**

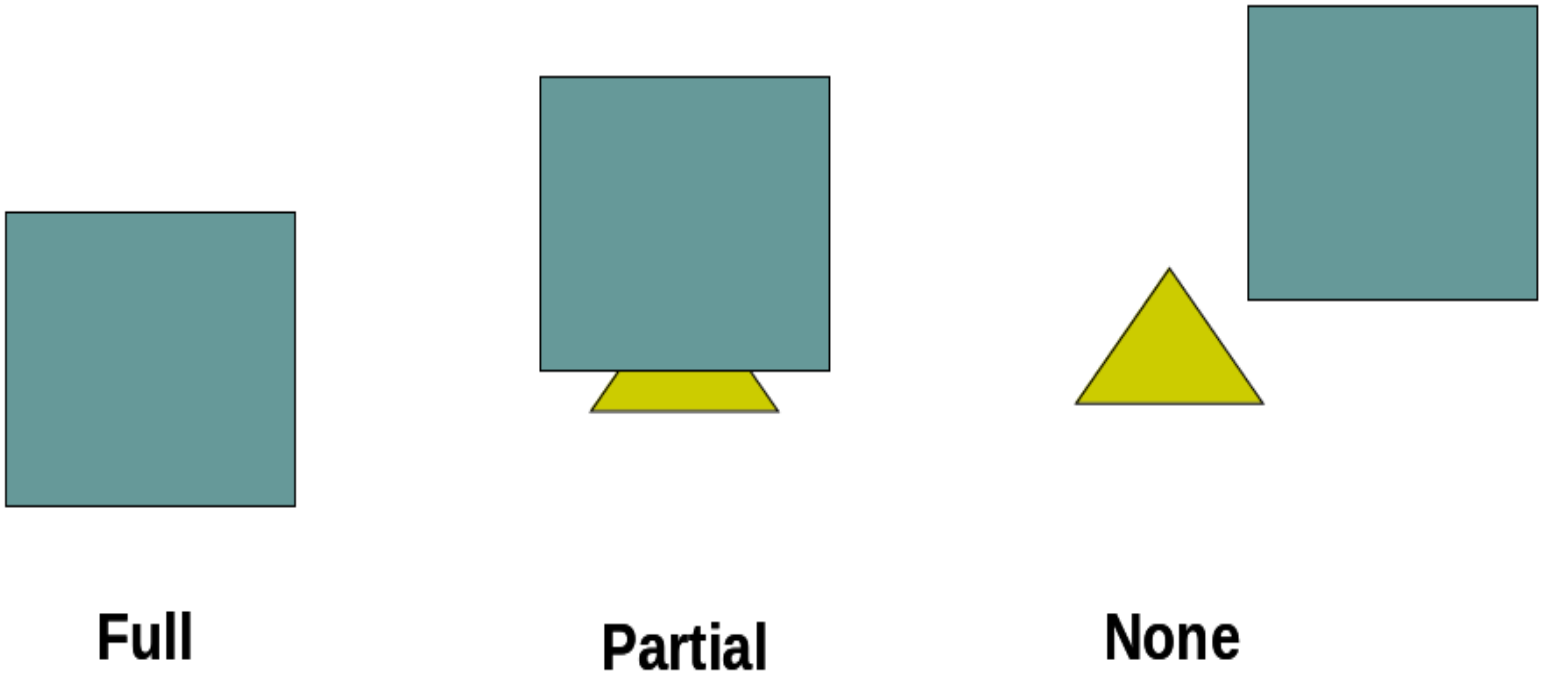
i. **Backface culling:** Polygons facing away from the viewer are removed

ii. **Occlusion:** Polygons farther away are obscured by closer polygons (i.e. closer objects occludes the others). Full or partially occluded portions of the polygons are removed.

- Why should we remove these polygons?

To avoid unnecessary expensive operations on these polygons later.

Occlusion: Full, Partial, None



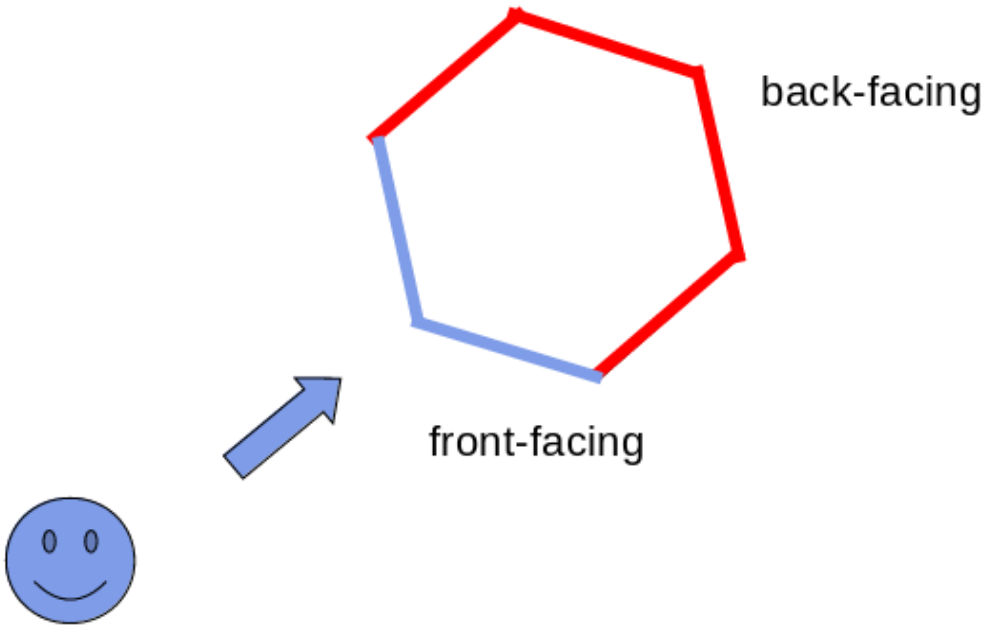
- The rectangle is closer than the triangle (Rectangle is occluding the triangle). Hence it should appear in front of the triangle

Back-Face Detection and removal (Back-Face Culling)

- Back-Face Culling avoids drawing polygons facing away from the viewer.
- It is object space method.
- The polygon normal of a …

➔ **front-facing** polygon points **towards** the viewer

➔ **back-facing** polygon points **away** from the viewer



- The equation for plane surface can be expressed as −

$$\mathbf{Ax + By + Cz + D = 0}$$

Where (x, y, z) is any point on the plane, and the coefficients A, B, C, and D are constants describing the spatial properties of the plane.

- For any point (x, y, z) with parameters A, B, C, and D, we can say that −

  $Ax + By + Cz + D \neq 0$ means the point is not on the plane.

  $Ax + By + Cz + D < 0$ means the point is inside the surface.

  $Ax + By + Cz + D > 0$ means the point is outside the surface.

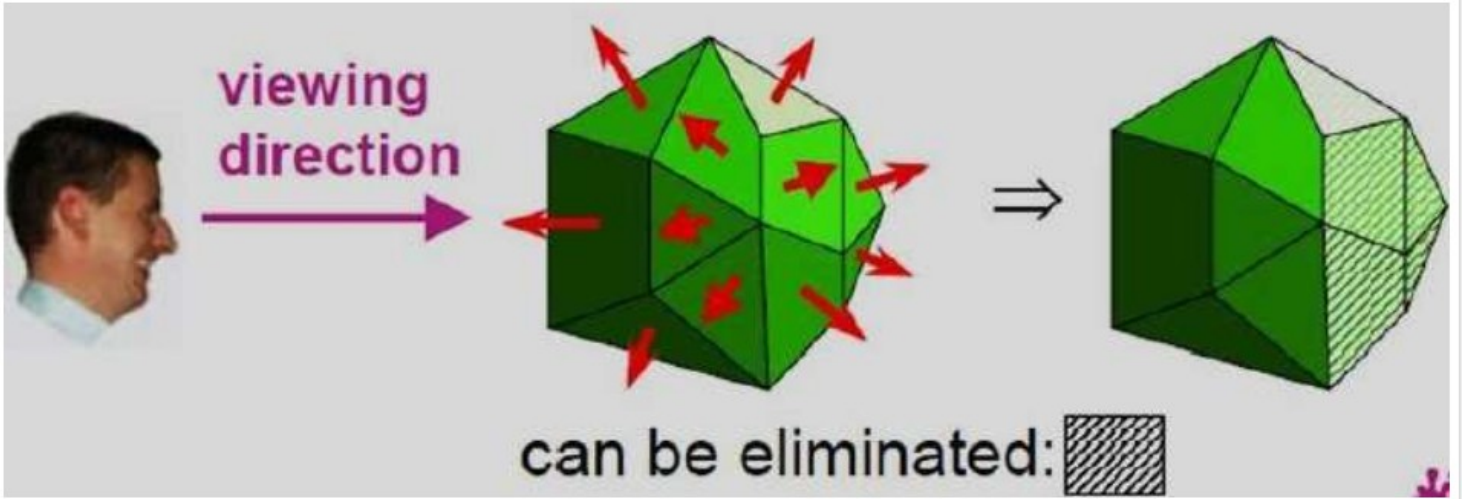
- Normal of the surface is given by:

$$\mathbf{N = (A,B,C)}$$

- If V is a vector in the viewing direction from the eye (or "camera") position and N is the surface normal, then this polygon is a back face if

$$\mathbf{V \cdot N >= 0}$$

(Above condition can be considered as: $VN\cos\theta >= 0$ where θ is angle between V and N)



Depth buffer (z-buffer) Algorithm

- It is an image space approach.
- It is proposed by Catmull in 1974.
- Easy to implement.
- Each surface is processed separately one pixel position at a time across the surface.
- The depth values for a pixel are compared and the closest (smallest z) surface determines the color to be displayed in the frame buffer.
- Surfaces are processed in any order.
- The z-coordinates (depth values) are usually normalized to the range [0,1].
- Two buffers named **frame buffer** and **depth buffer**, are used.

- **Depth buffer** is used to store depth values for (x, y) position, as surfaces are processed ($0 \leq \text{depth} \leq 1$).
- The **frame buffer** is used to store the intensity value of color value at each position (x, y).

Depth-Buffer Algorithm

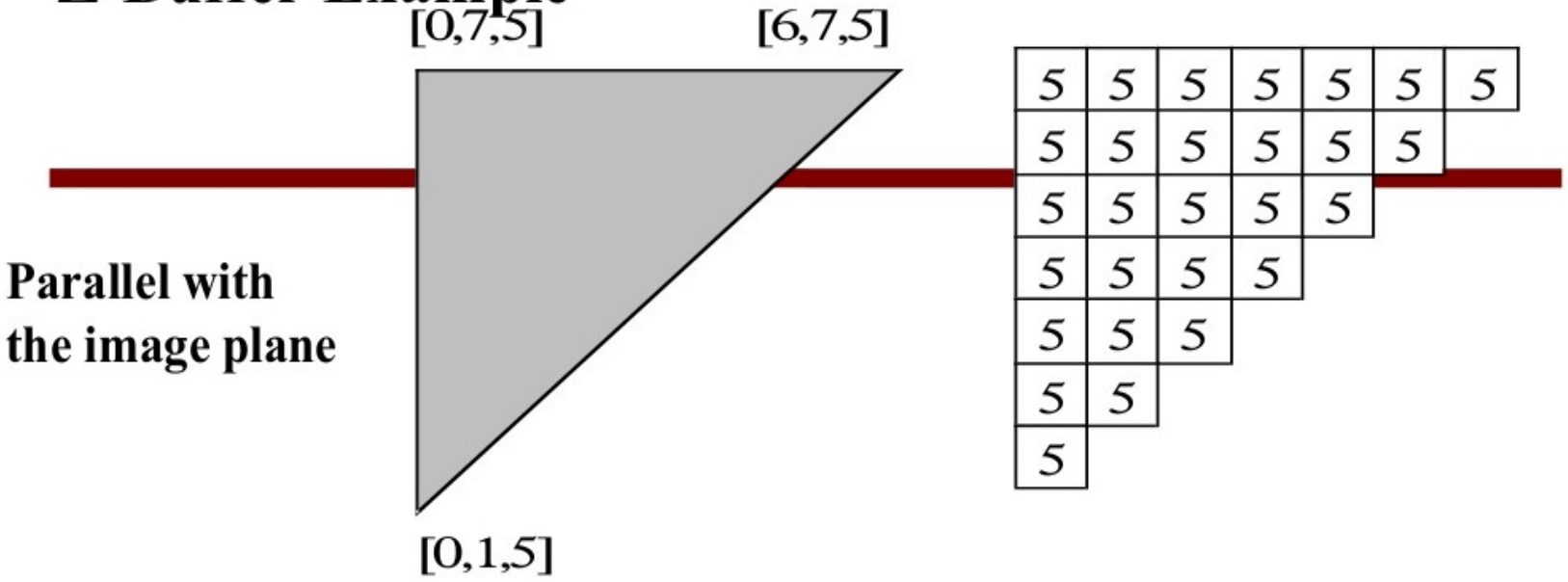
- Initialize the depth buffer and frame buffer so that for all buffer positions (x,y),

  depthBuff (x,y) = 1.0, frameBuff (x,y) =bgColor

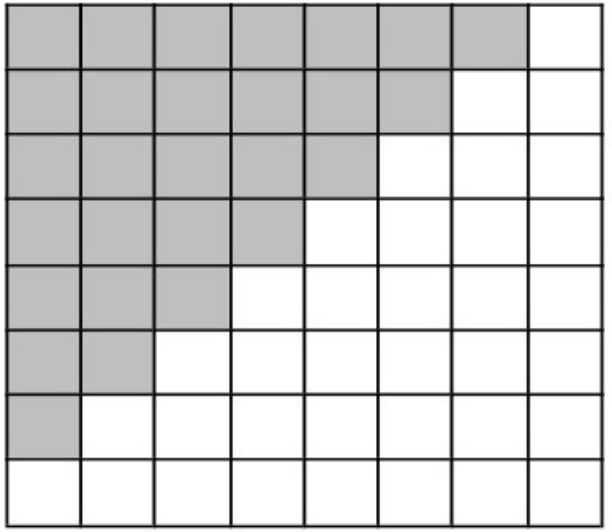
- Process each polygon in a scene, one at a time
  - For each projected (x,y) pixel position of a polygon, calculate the depth z.
  - If $z <$ depthBuff (x,y), compute the surface color at that position and set

  depthBuff (x,y) = z, frameBuff (x,y) = surfCol (x,y)

Z-Buffer Example



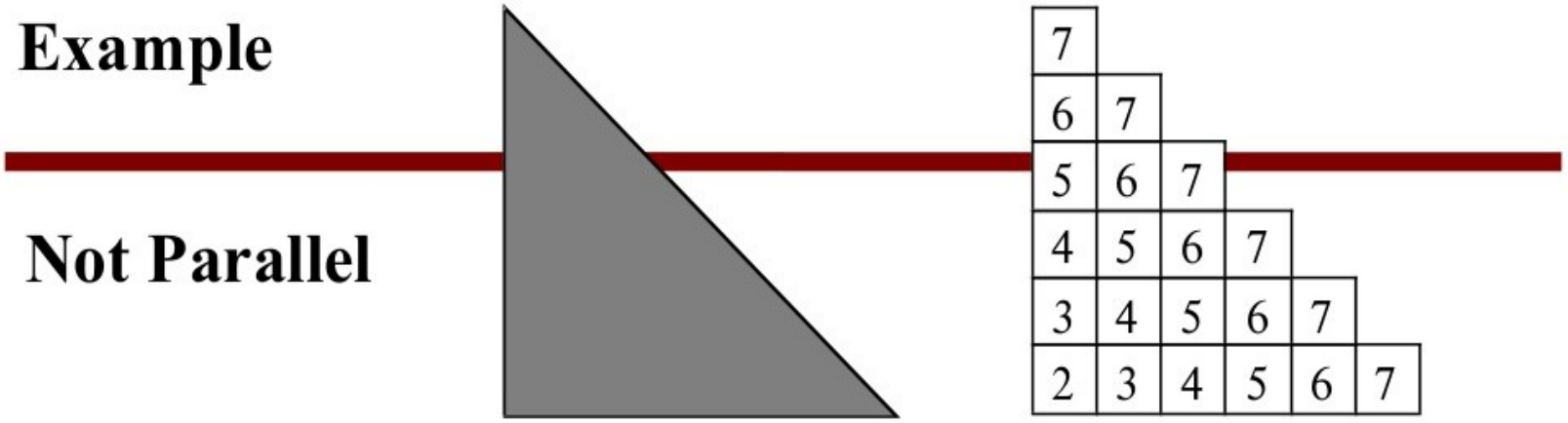
5	5	5	5	5	5	5	∞
5	5	5	5	5	5	∞	∞
5	5	5	5	5	∞	∞	∞
5	5	5	5	∞	∞	∞	∞
5	5	5	∞	∞	∞	∞	∞
5	5	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞



Z-Buffer Example

Not Parallel

[0,6,7]

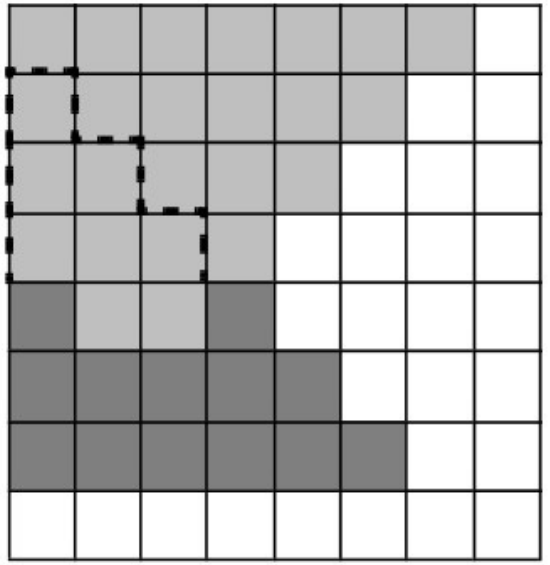


7					
6	7				
5	6	7			
4	5	6	7		
3	4	5	6	7	
2	3	4	5	6	7

[0,1,2]

[5,1,7]

5	5	5	5	5	5	5	∞
5	5	5	5	5	5	∞	∞
5	5	5	5	5	∞	∞	∞
5	5	5	5	∞	∞	∞	∞
4	5	5	7	∞	∞	∞	∞
3	4	5	6	7	∞	∞	∞
2	3	4	5	6	7	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞



- We can calculate depth of the polygon at each pixel position (x,y).
- For calculation of depth (z), we use polygon surface equation as:

$$z = \frac{-Ax - By - D}{C}$$

- For any scan line adjacent horizontal *x* positions or vertical *y* positions differ by 1 unit.
- The depth value of the next position (x+1,y) on the scan line can be obtained using

$$z' = \frac{-A(x+1) - By - D}{C}$$

$$= z - \frac{A}{C}$$

Area subdivision (Warnock) Algorithm

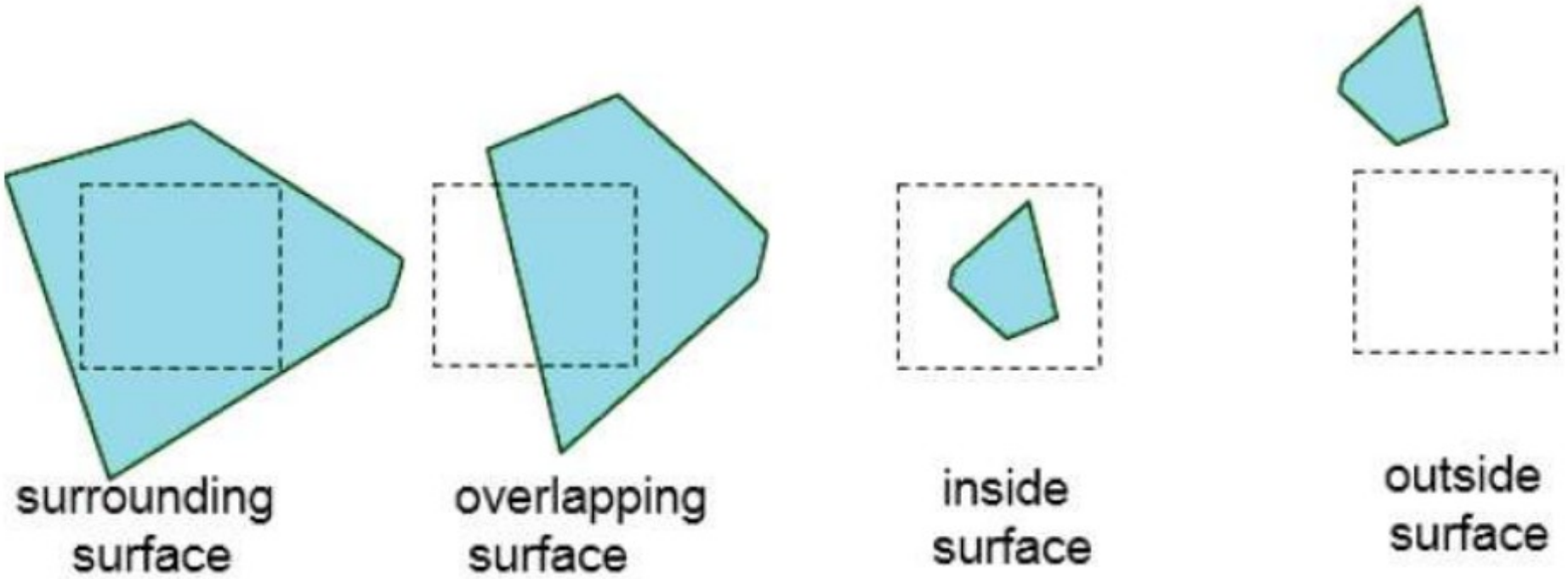
- The Warnock algorithm is a hidden surface algorithm developed by John Warnock.
- The total viewing area is successively divided into smaller and smaller rectangles until each small area is simple, I.e. it is a single pixel, or is covered wholly by a part of a single visible surface or no surface at all.
- We first classify each of the surfaces, according to their relations with the area:

➔ **Surrounding surface -** a single surface completely encloses the area

➔ **Intersecting** or **Overlapping surface -** a single surface that is partly inside and partly outside the area

➔ **Inside** or **contained surface -** a single surface that is completely inside the area

➔ **Outside** or **disjoint surface -** a single surface that is completely outside the area.


surrounding surface
overlapping surface
inside surface
outside surface

Algorithm:

1. Initialize the area to be the whole screen.

2. Generate list of polygons by sorting them with their z values.

3. Remove polygons which are outside the area (Don't include disjoint polygons in the list because they are not visible).

4. Identify relationship of each polygon (surface) with area.

5. Execute visibility decision analysis:

   a)Fill area with background color if all polygons are disjoint,

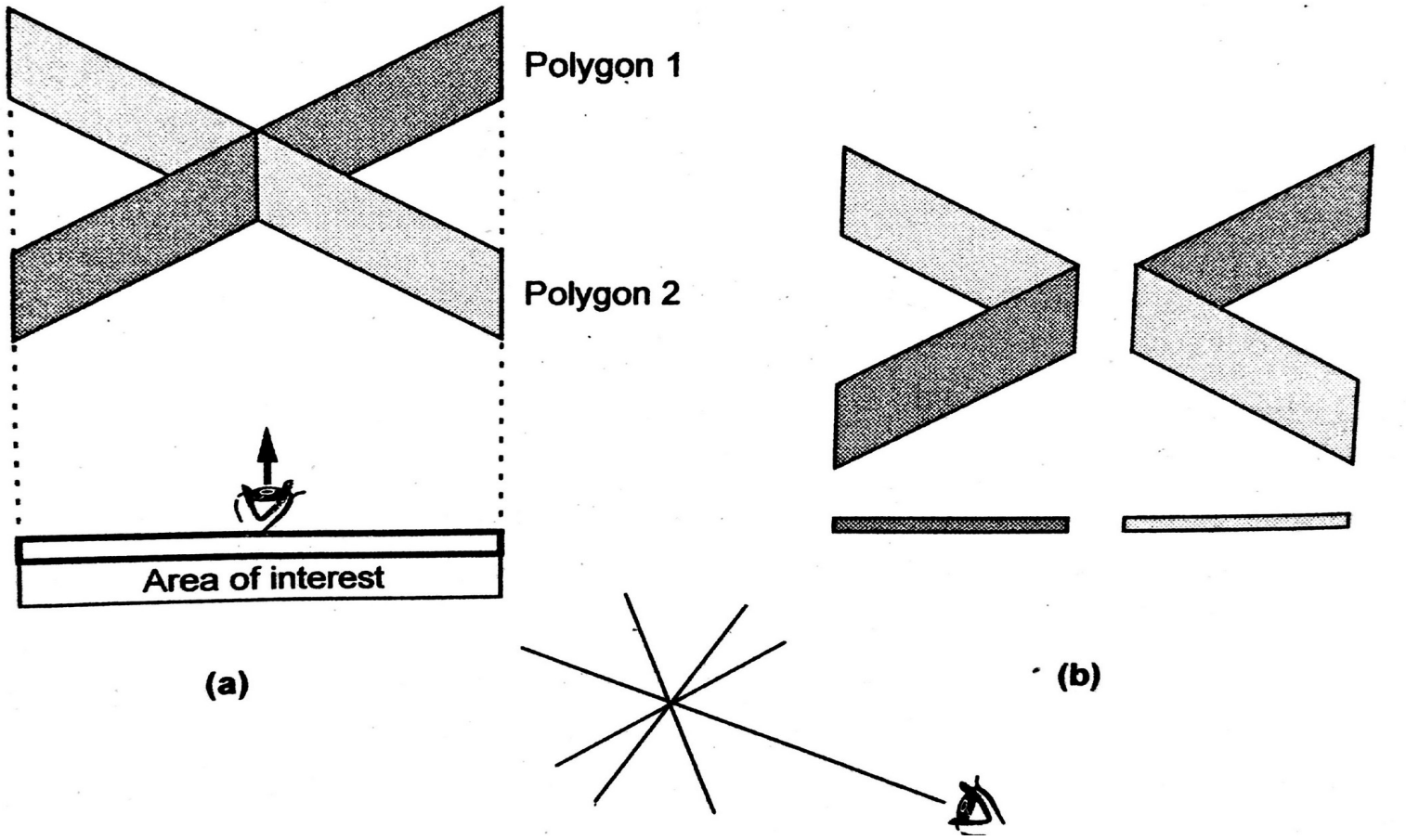
   b) If there is only one intersecting polygon or only one contained polygon, fill entire area with background color and fill part of polygon contained in area with color of polygon,

   c) If there is a single surrounding polygon, but no intersecting or contained polygon then fill area with color of surrounding polygon.

   d) If surrounding polygon is closer to the viewpoint than all other polygons, so that all other polygons are hidden by it, fill the area with the colour of the surrounding polygon.

   e) If area is the pixel (x,y) and neither of (a) to (d) applies, calculate z- coordinate at pixel (x,y) of all polygons in the list. The pixel is then set to colour of the polygon which is closer to the viewpoint.

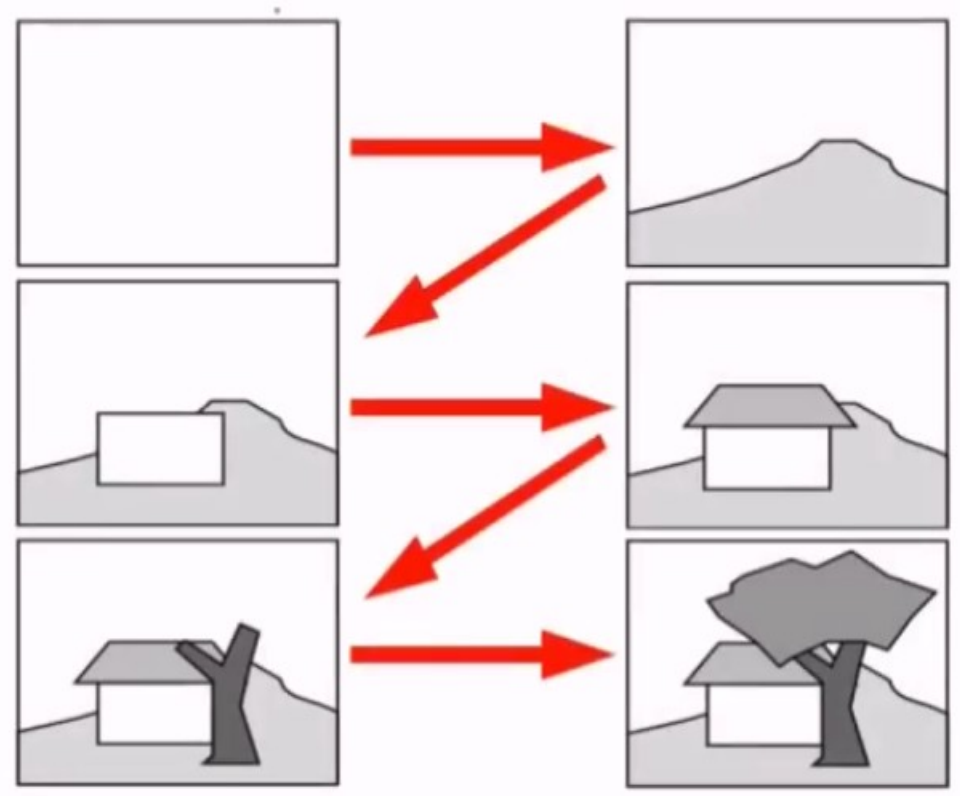
6. If none of above is correct then subdivide the area and Go to Step 2.



- As shown in above figure (a), we can not make any decision about which polygon is in front of other (because none of the condition in step 5 in Warnock's algorithm is true). So we go for subdividing the area.
- After subdividing area of interest (as in figure(b)), polygon 1 is ahead of the polygon 2 in left area and polygon 2 is ahead of polygon 1 in right area. Now we can fill these two areas with corresponding colours of the polygons.

Depth sorts (Painter's) Algorithm

- Main Idea
  - A painter creates a picture by drawing background scene elements before foreground ones
- Requirements
  - Draw polygons in back-to-front order
  - Need to **sort** the polygons by depth order to get a correct image



- The depth-sorting method performs two basic functions −
- First, the surfaces are sorted in order of decreasing depth.
- Second, the surfaces are scan-converted in order, starting with the surface of greatest depth.

Key Idea

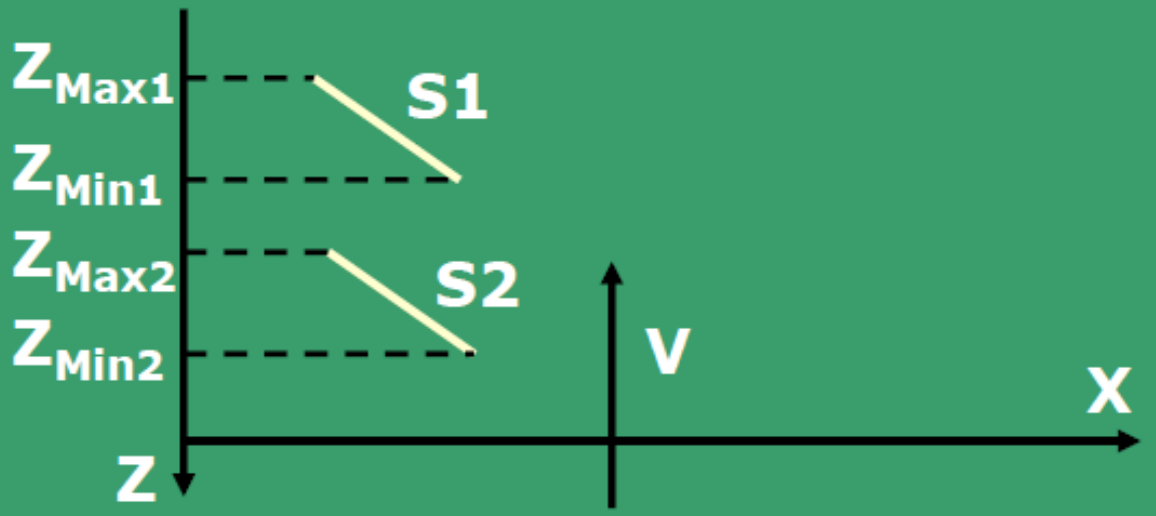
Paint the polygons in the frame buffer in order of decreasing distance from the viewpoint.

Broad Steps:

- Surfaces are sorted in increasing order of DEPTH.
- Resolve ambiguities when polygons overlap (in DEPTH), splitting polygons if necessary.
- Surfaces are scan converted in order, starting with the surface of greatest DEPTH.

Test 0: Check for Z-extents overlap

Since, $Z_{Min1} > Z_{Max2}$ no overlap occurs.
S1 is first scan converted, and then S2.
This goes on, as long as no overlap occurs.



If depth overlap occurs, additional comparisons are necessary to reorder the surfaces.

The following set of ***tests*** are used to ensure that no re-ordering of surfaces is necessary:

Four(4) Tests in Painter's algorithm

1. The boundary rectangles in the X-Y plane for the two surfaces do not overlap.

2. Surface S is completely behind the overlapping surface relative to the viewing position.

3. The overlapping surface is completely in front of S relative to the viewing position.

4. The projections of the two surfaces onto the view plane do not overlap.

Tests must be performed in order, as specified.

Condition to RE-ORDER the surfaces:

If any one of the 4 tests is TRUE, we proceed to the next overlapping surface. i.e. If all overlapping surfaces pass at least one of the tests, none of them is behind S.

No re-ordering is required, and then S is scan converted.

RE-ORDERing is required if all the four tests fail.

Test-1: Check for X and Y-extent Overlap

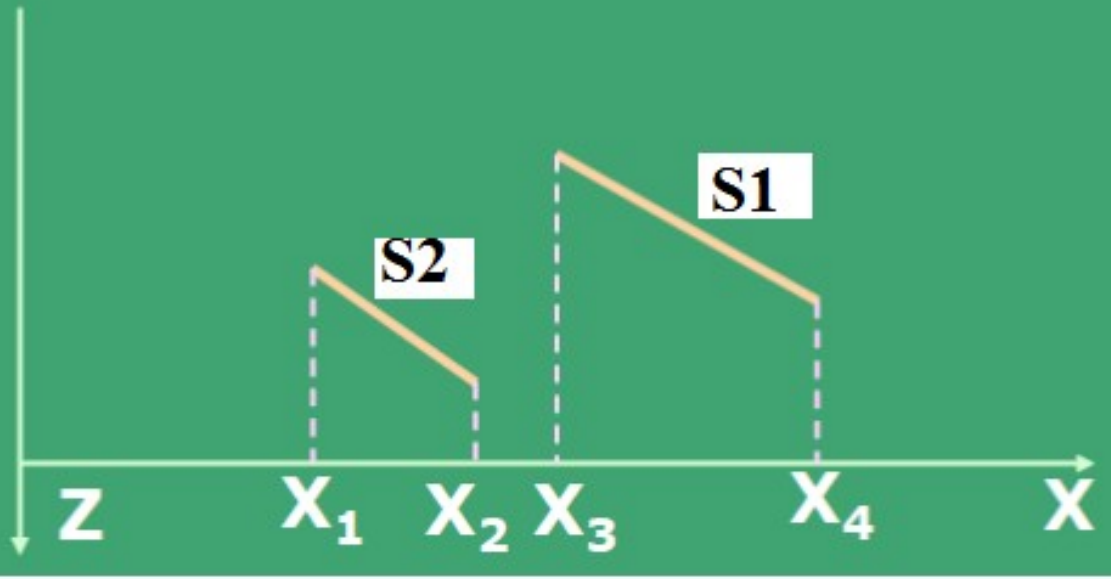
TEST #1:

The boundary rectangles in the X-Y plane for the two surfaces do not overlap.

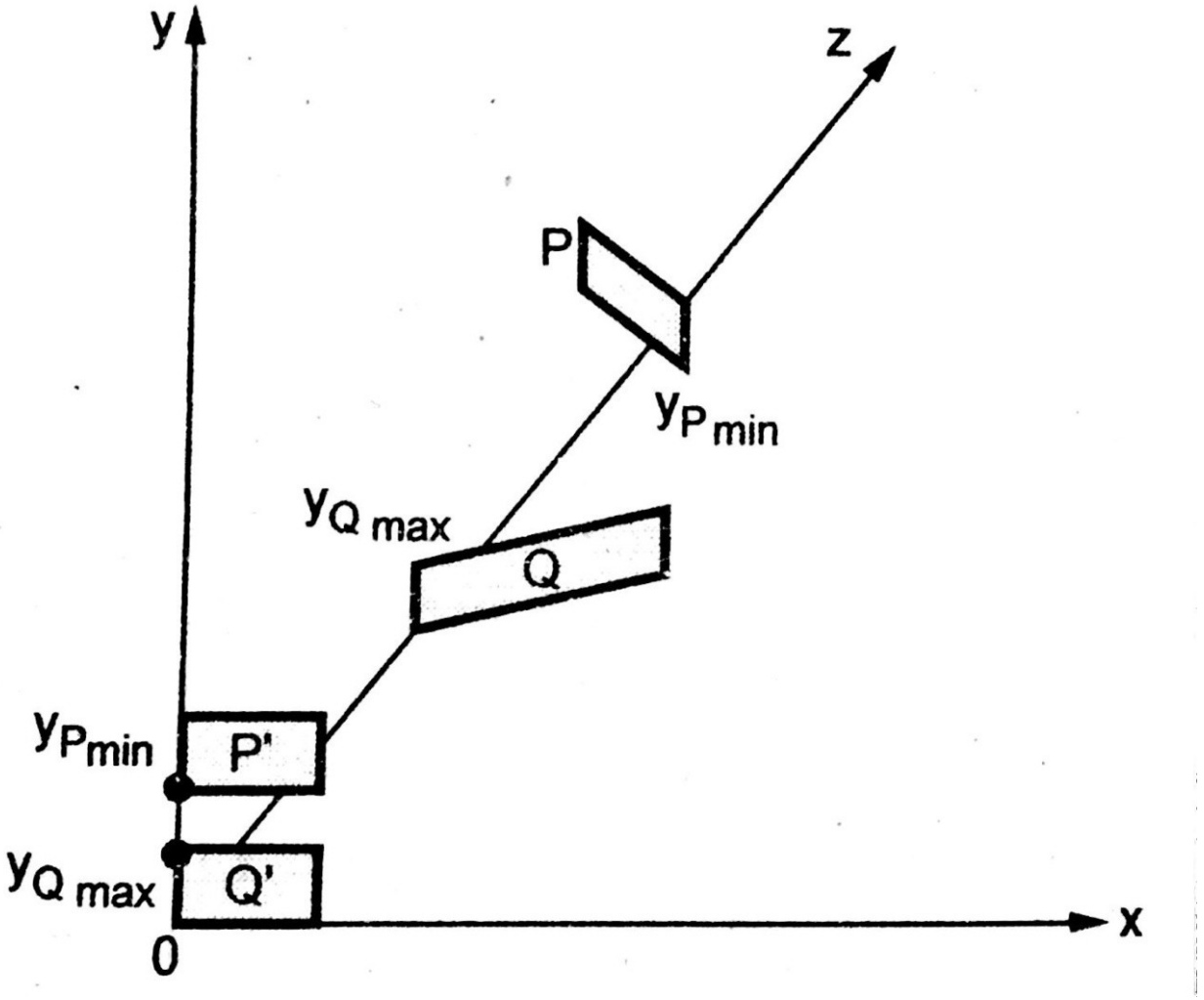
We have depth overlap, but no overlap in X-direction.

Hence, Test #1 is passed, scan convert S1 and then S2

If we have X overlap, check for the Next test



Check for Y-extent overlap



TEST #2:

Surface S is completely behind the overlapping surface relative to the viewing position.

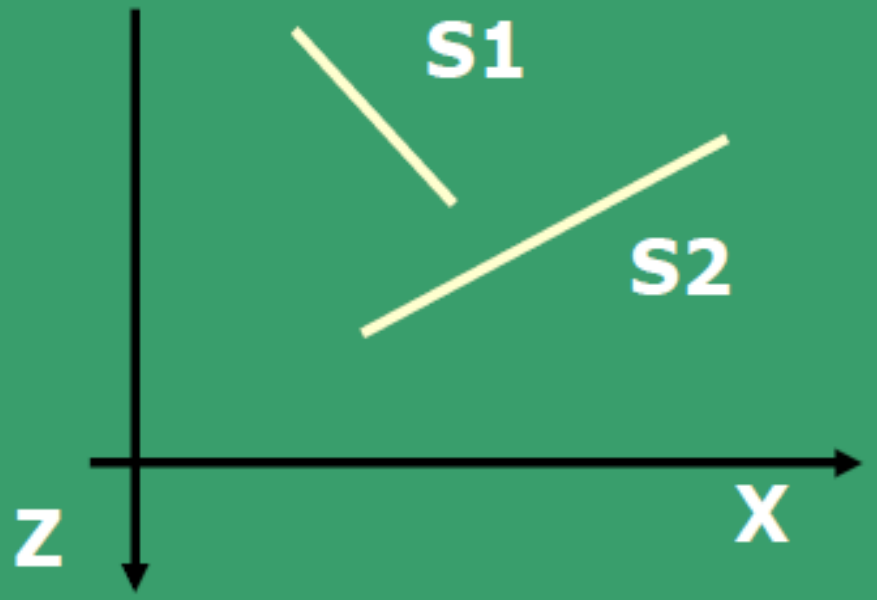


Fig. 1. S1 is completely behind/inside the overlapping surface S2

TEST #3:

The overlapping surface is completely in front of S relative to the viewing position.

S1 is not completely behind S2.
So, Test #2 fails.

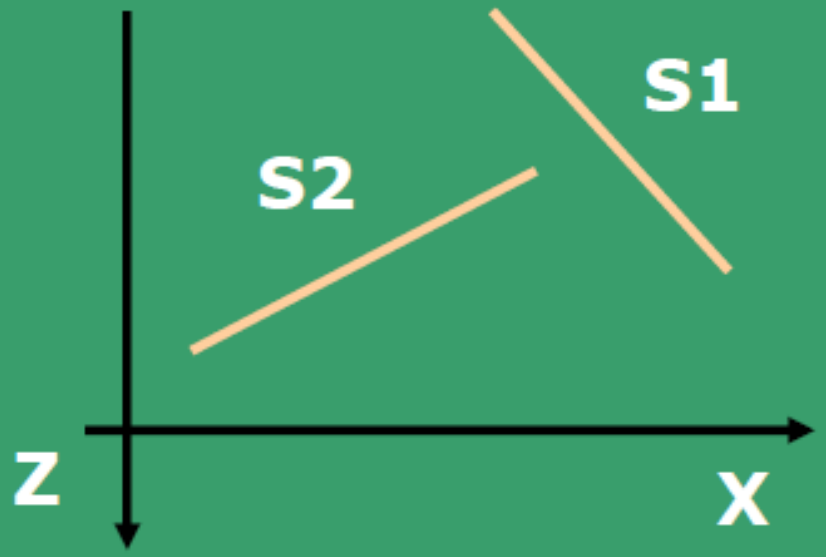


Fig. 2. Overlapping surface S2 is completely front/outside S1.

In Fig. 2, S2 is in front of S1, but S1 is not completely inside S2 – Test #2 it not TRUE or FAILS, although Test #3 is TRUE.

How to check these conditions?

i) Set the plane equation of S2, such that the surface S2 is towards the viewing position.

ii) Substitute the coordinates of all vertices of S1 into the plane equation of S2 and check for the sign.

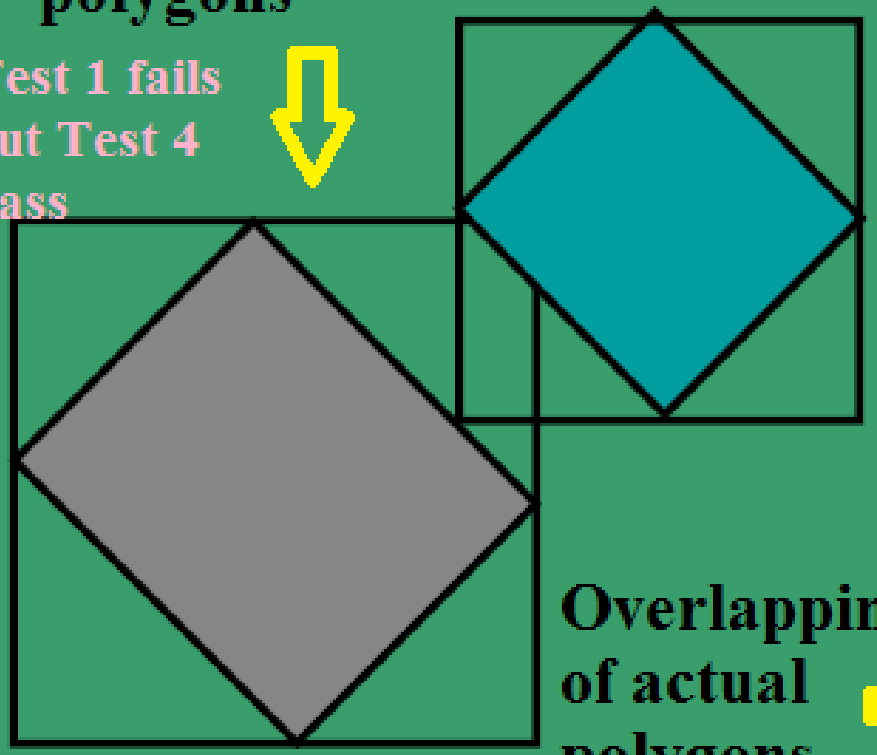
iii) If all vertices of S1 are ***inside*** S2, then S1 is ***behind*** S2. (Fig. 1).

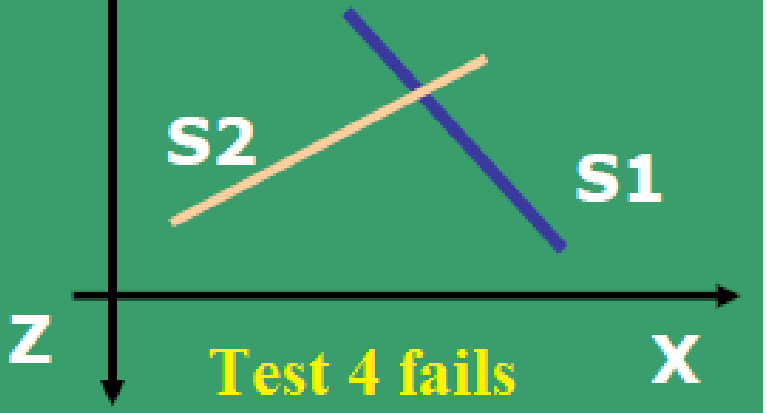
iv) If all vertices of S1 are ***outside*** S2, S1 is in ***front*** of S2.

TEST #4:
The projections of the two surfaces onto the view plane do not overlap.

Overlapping of boundary rectangles but not actual polygons

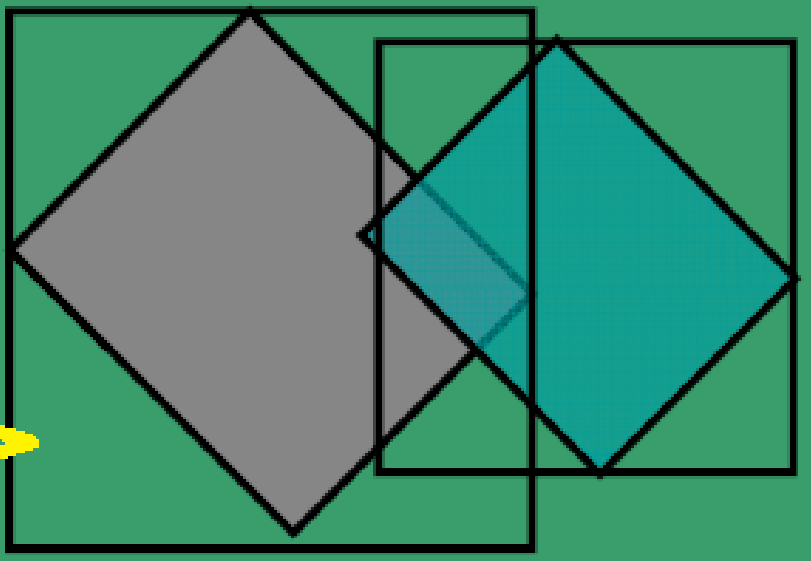
Test 1 fails but Test 4 pass





Test 4 fails

Overlapping of actual polygons



- If Test 4 also fails (means all test from 0-4 fail) then re-orderring of polygons is required.