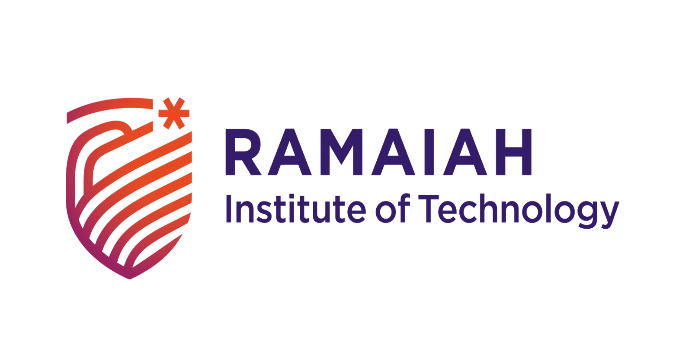
**M S RAMAIAH INSTITUTE OF TECHNOLOGY**

**(Autonomous Institute Affiliated to VTU)**

**Department of Information Science and Engineering**

****

**UNIT-5** Notes

*Submitted in partial fulfillment of the CIE for the subject*

**COMPUTER GRAPHICS**

* **Explain the gouraud shading with the help of neat sketches.**

Ans -> Gouraud Shading:

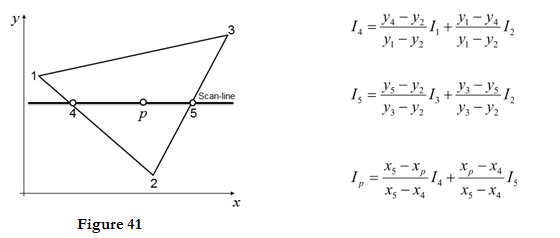
1. This was developed in the 1970s by Henri Gouraud.
2. It 's an interpolation technique.
3. Intensity levels are calculated at each vertex and interpolated across the surface.
4. Intensity values for each polygon are matched with the values of adjacent polygons along the common edges.
5. Its eliminated that the intensity discontinuities that may happen in flat shading.
6. To render a polygon, proceedings are:
   * Determine the average unit normal vector at each vertex of the polygon.
   * Apply an illumination model at each polygon vertex to obtain the light intensity at that position.
   * Linearly interpolate the vertex intensities over the projected area of the polygon
7. The average unit normal vector at V is given as:

Nv=N1+N2+N3+N4|N1+N2+N3+N4|Nv=N1+N2+N3+N4|N1+N2+N3+N4|

1. More generally as

Nv=Σni=1NiΣni=1NiNv=Σi=1nNiΣi=1nNi

1. Illumination values are linearly interpolated across each scan-line as shown in figure 41.



1. The intensities at point 4 can be interpolated from intensities 1 and 2.
2. Similarly, the intensities at point 5 can be interpolated from intensities 2 and 3.
3. Therefore the intensities of interaction points 4 and 5 are calculated from scan line.

Advantages:

i. It removes the intensity discontinuity which exists in constant shading model.

ii. It can be combined with hidden surface algorithm to fill in the visible polygons along each scan line.

Disadvantages:

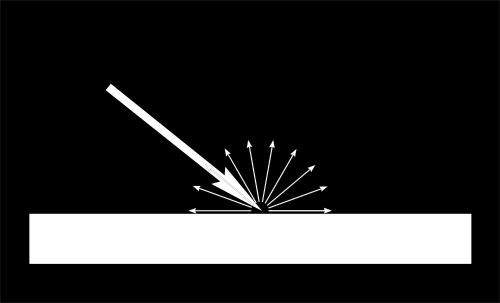
i. Gouraud shading has a problem with specular reflections.

ii. Gouraud shading can introduce anomalies known as Mach bands.

* **What do you understand by rendering? Also explain the role of normal vector in shading.**

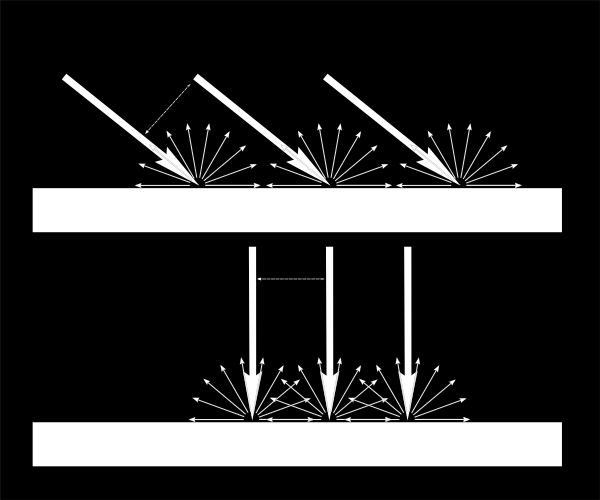
-> Rendering or image synthesis is the automatic process of generating a photorealistic or non-photorealistic image from a 2D or 3D model by means of a computer program. The resulting image is referred to as the render.

When light hits an object, an important fraction of it is reflected in all directions. This is the “diffuse component”. (We’ll see what happens with the other fraction soon)



When a certain flux of light arrives at the surface, this surface is illuminated differently according to the angle at which the light arrives.

If the light is perpendicular to the surface, it is concentrated on a small surface. If it arrives at a gazing angle, the same quantity of light spreads on a greater surface :



Each point of the surface will look darker with gazing light.

When we compute the color of a pixel, the angle between the incoming light and the surface normal matters. So, we have :

*// Cosine of the angle between the normal and the light direction,*

*// clamped above 0*

*// - light is at the vertical of the triangle -> 1*

*// - light is perpendicular to the triangle -> 0*

float cosTheta = dot( n,l );

color = LightColor \* cosTheta;

In this code, n is the surface normal and l is the unit vector that goes from the surface to the light.

* **How can we determine visible surfaces using Z Buffer algorithm?**

-> It is an image-space approach. The basic idea is to test the Z-depth of each surface to determine the closest visible surface.

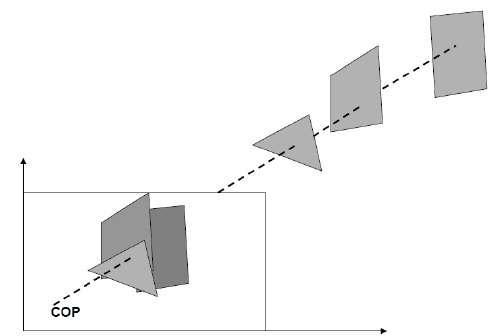
In this method each surface is processed separately one pixel position at a time across the surface. The depth values for a pixel are compared and the closest smallest surface determines the color to be displayed in the frame buffer.

It is applied very efficiently on surfaces of polygon. Surfaces can be processed in any order. To override the closer polygons from the far ones, two buffers named frame buffer and depth buffer, are used.

Depth buffer is used to store depth values for x,yx,y position, as surfaces are processed 0≤depth≤10≤depth≤1.

The frame buffer is used to store the intensity value of color value at each position x,yx,y.

The z-coordinates are usually normalized to the range [0, 1]. The 0 value for z-coordinate indicates back clipping pane and 1 value for z-coordinates indicates front clipping pane.



Algorithm

Step-1 − Set the buffer values −

Depthbuffer x,yx,y = 0

Framebuffer x,yx,y = background color

Step-2 − Process each polygon OneatatimeOneatatime

For each projected x,yx,y pixel position of a polygon, calculate depth z.

If Z > depthbuffer x,yx,y

Compute surface color,

set depthbuffer x,yx,y = z,

framebuffer x,yx,y = surfacecolor x,y

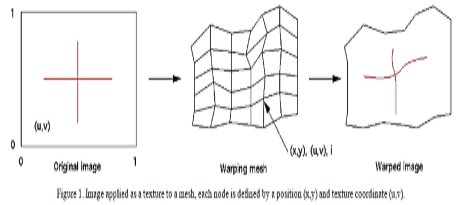
* Differentiate between z-buffer method and scan-line method. What is the visibility test made in these methods?

**Solution**: In depth buffer algorithm every pixel location on the projection plane is identified for determining the visibility of surfaces regarding this pixel. Conversely, in scan-line method all surfaces intersected through a scan line are studied for visibility. The visibility test in depth-buffer method engages the comparison of depths of surfaces regarding a pixel on the projection plane. The surface closest to the pixel position is seemed visible. The visibility test in scan-line method which compares depth computations, for each overlapping surface to find out which surface is nearest to the view-plane hence it is declared as visible

* **What is the process of performing Morphing? Discuss with the help of interpolation of polygons in rendering.**

**Ans:**

* Many projection environments require images that are not simple perspective projections that are the norm for flat screen displays
* Examples include geometry correction for cylindrical displays and some new methods of projecting into planetarium domes orupright domes intended for VR. The standard approach is to create the image in a format that contains all the required visual information and distort it (from now on referred to as "warping") to compensate for the non planar nature of the projection device or surface. For both realtime and offline warping the concept of a OpenGL texture is used, that is, the original image is considered to be a texture that is applied to a mesh defined by node positions and corresponding texture coordinates, see figure 1. The following describes a file format for storing such warping meshes, it consists of both an ascii/human readable format and a binary format.



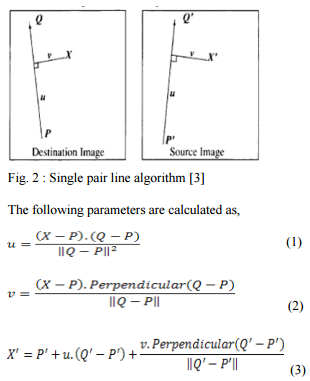
Feature Morphing

This method gives the animator a high level of control over the process. The animator interactively selects corresponding feature lines in the 2 images to be morphed.

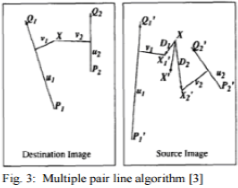
The algorithm uses lines to relate features in the source image to features in the destination image.

It is based upon fields of influence surrounding the feature lines selected. It uses reverse mapping (i.e. it goes through the destination image pixel by pixel, and samples the correct pixel from the source image) for warping the image.

A pair of lines (one defined relative to the source image, the other defined relative to the destination image) defines a mapping from one image to the other



* Where, X is the pixel co-ordinate in the destination image and X’ is the corresponding pixel co-ordinate in the source image, PQ is a line segment in the destination image and P’Q’ is the corresponding line segment in the source image, u is the position along the line, and v is the distance from the line.
* The value u goes from 0 to 1 as the pixel moves from P to Q, and is less than 0 or greater than 1 outside that range. The value for v is the perpendicular distance in pixels from the line.
* The above algorithm is for the case of a single feature line. In a normal morphing scenario, however there are multiple features in the images to be morphed and consequently multiple feature line pairs are specified



The displacement of a point in the source image is then, actually a weighted sum of the mappings due to each line pair, with the weights attributed to distance and line length. The weight assigned to each line should be strongest when the pixel is exactly on the line, and weaker the further the pixel is from it. The equation used is as follow.

weight=(lengthp(a+dist))b.

* **How would scan line coherence algorithm look like from the context of visible surface determination.**

**Ans:** It is an image space algorithm. It processes one line at a time rather than one pixel at a time. It uses the concept area of coherence. This algorithm records edge list, active edge list. The edge list or edge table contains the coordinate of two endpoints. The active edge list (AEL) should be sorted in increasing order of x.

**Step1:** Start algorithm

**Step2:** Initialize the desired data structure

1. Create a polygon table having color,edge pointers,coefficients
2. Establish edge table contains information regarding, the endpoint of edges, pointer to polygon, inverse slope.
3. Create Active edge list. This will be sorted in increasing order of x.
4. Create a flag F. It will have two values either on or off.

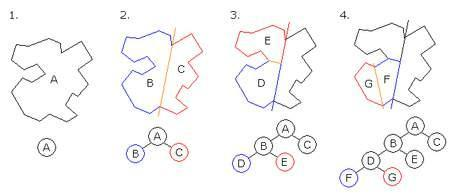
**Step3:** Perform the following steps for all scan lines

1. Enter values in Active edge list (AEL) in sorted order using y as value
2. Scan until the flag, i.e. F is on using a background color
3. When one polygon flag is on, and this is for surface S1enter color intensity as I1into refresh buffer
4. When two or image surface flag are on, sort the surfaces according to depth and use intensity value Sn for the nth surface. This surface will have least z depth value
5. Use the concept of coherence for remaining planes.

* **Explain binary space partition technique.**

**Ans:** Binary space partitioning is a 3-D graphics programming technique of dividing a scene into two recursively using hyperplanes. The technique is usually used to speed up 3-D scenes rendering , for example in games. Because the object location in a scene can be quickly specified, the renderer can fastly create the point of view of a player. BSP is also usually used for collision detection in robotics and rendering in computer-aided design and games.

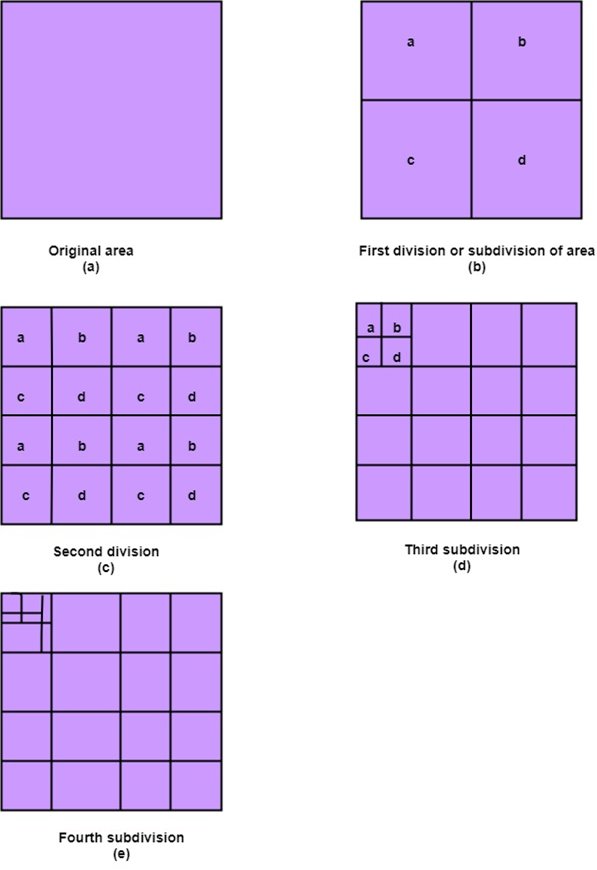
BSP trees can be traversed quickly for shadow casting and hidden surface removal. Using the Painter’s algorithm in combination with the BSP-tree gives a very optimal result that is used by the industry today.



* **How would Area Coherence algorithm look like from the context of Hidden Surface elimination , Warnock’s algorithm.**

**Ans:** Coherence is used to take advantage of the constant value of the surface of the scene. It is based on how much regularity exists in the scene. Area Coherence is used to group of pixels cover by same visible face.

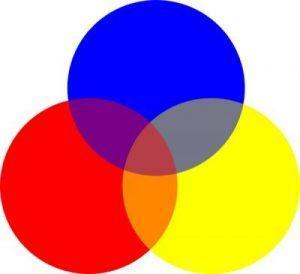
Warnock’s algorithm  is based on a divide & conquer method. It resolves the algorithm visibility . It classifies polygons in two cases i.e. trivial and non-trivial. Trivial cases are easily handled. Four equal subwindows are created for non-trivial cases. Using recursion again the windows are further subdivided until all polygons classified trivial and non trivial.



* **What are the issues and solutions associated under priority algorithm.**

**Ans.)** The Painter’s algorithm, also known as Priority Fill, is one of the simplest solutions to the visibility problem in 3D computer graphics for example to create a error-free 3D configurator. When a 3D scene is projected onto a 2D plane, it is necessary to decide which polygons are visible and which are hidden.

The algorithm may fail in some cases, including cyclic overlap. In the case of a cyclic overlap, as shown in the figure below, the circles A, B, and C overlap in such a way that it is impossible to determine which circle is above the other. In this case, the disturbing circles must be cut off to allow sorting. In 1972 Newell algorithm proposed a method for cutting such objects. Numerous methods have also been proposed in the field of computer geometry.

****

* **Explain span coherence algorithm with the help of examples.**

**Ans:** Another property of coherence that can be made use of in scan conversion is the one-dimensional form of area coherence called span coherence. It holds good upto a span.Once the span is detected , all pixels within the span can be set to the intensity value of the polygon and the next comparison can take place at the end of the span. The concept of span can be understood by taking the following simple example.

In the 3-dimensional space , each scan line produces a plane , which means each line is for a particular value of y. A plane with this value of y as a constant over different values of x and z form a plane. If one travels along this scan line , the plane intersects one/more polygons at different points. If we note the points of intersection and are sorted in the increasing order of x , we get a sort of xz algorithm and the list is given of intersections with different polygons.

* **Explain scan line algorithm and priority algorithm.**

**Ans:** Scan line algorithm is an image space algorithm. It processes one line at a time rather than one pixel at a time. It uses the concept area of coherence. This algorithm records edge list, active edge list. So accurate bookkeeping is necessary. The edge list or edge table contains the coordinate of two endpoints. The active edge list (AEL) should be sorted in increasing order of x. The AEL is dynamic, growing and shrinking.

The Painter’s algorithm, also known as Priority Fill, is one of the simplest solutions to the visibility problem in 3D computer graphics for example to create a error-free 3D configurator. When 3D scene is projected onto a 2D plane, it is eventually necessary to decide which polygons are hidden and which are visible. The term priority algorithm also called painter’s algorithm refers to the technique used by many painters to paint distant parts of a scene in front of closer parts and thus cover some areas of distant parts.

* **Use illumination model equations to explain the following-**

**a) ambient light b) diffuse reflection**

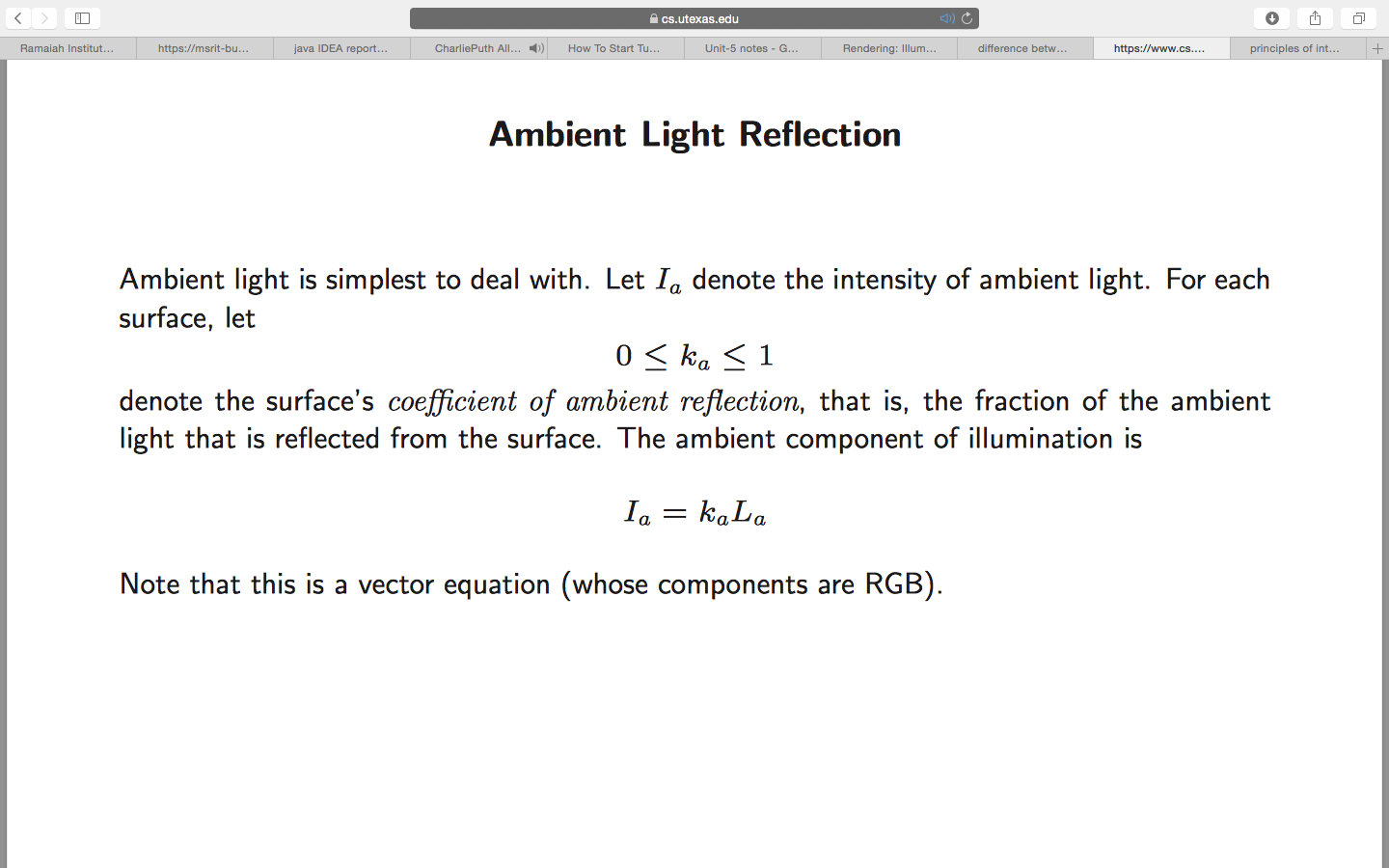
**Ans:**

a) Ambient Light: It is the light reflected by other objects that helps a nearby object to lit up and we are able to see it.

Equation : I=IaKa

* **Ia - ambient light intensity the scene-based value that remains the same for all surfaces in the scene.**
* **Ka - ambient reflection coefficient a material based property that determines how much ambient light is actually reflected.**

**This equation allows individual surface properties to “reflect” a level of ambient light.**



b)Diffuse Reflection: Surfaces that are rough/grainy tend to reflect light in all directions.

• This scattered light is called diffuse reflection.

• perfect diffuse reflection-Surfaces that reflect incident light with equal intensity in all

directions

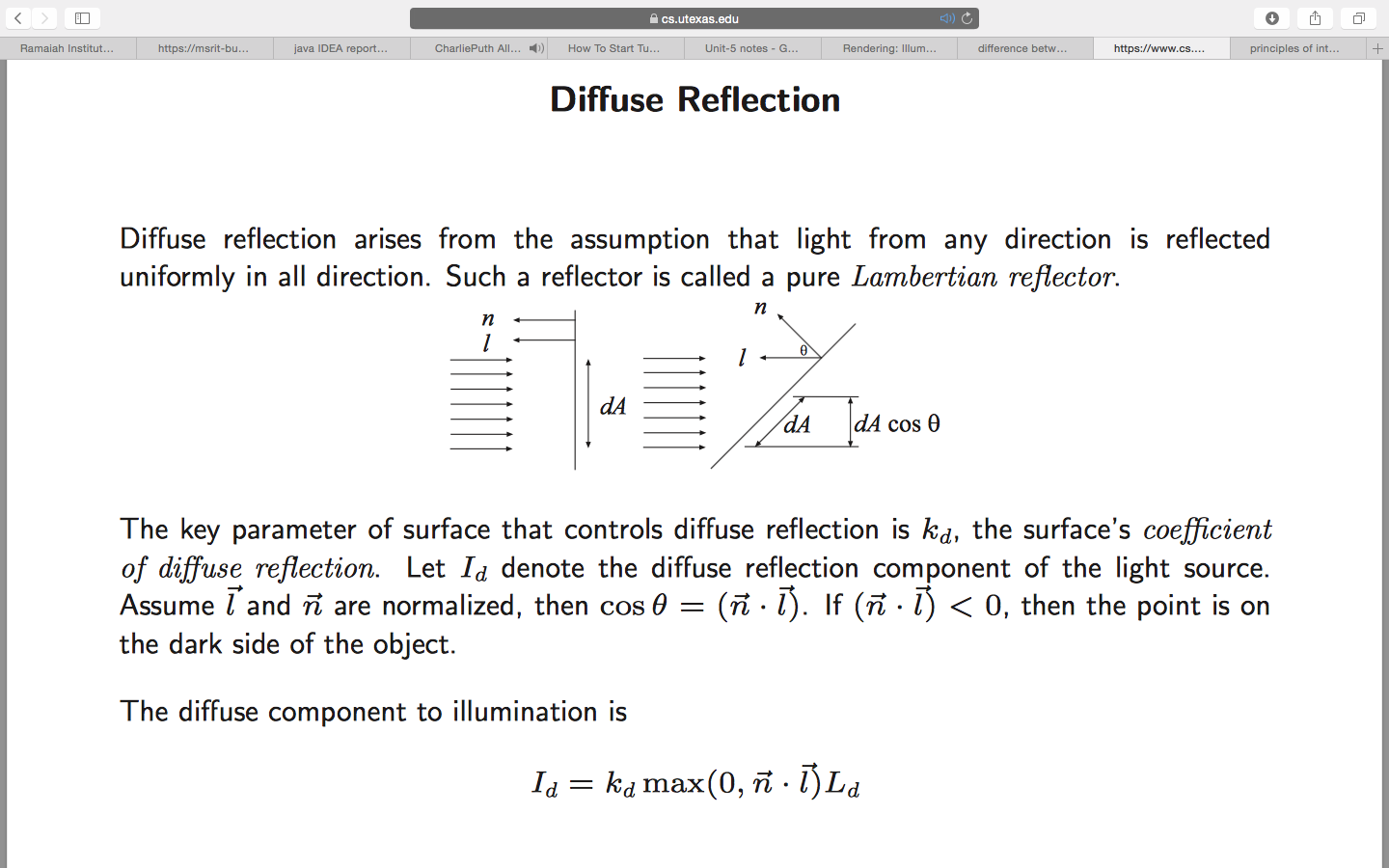
• Such surfaces are referred to as ideal diffuse reflectors.

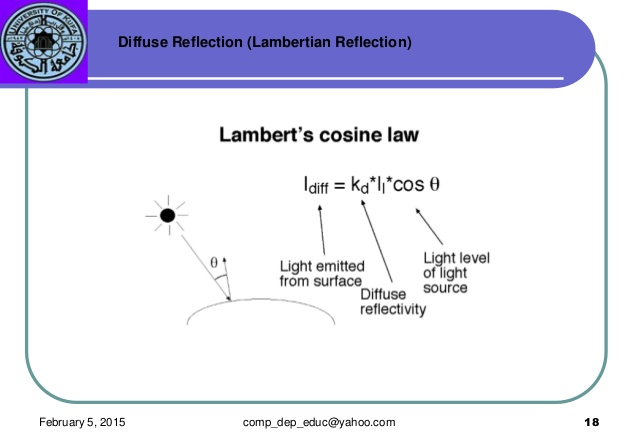
• Example- snow, movie screen.

• Lambertian surface - it appears equally bright from all the

viewing directions. Because they reflect light with equal

intensity in all direction.



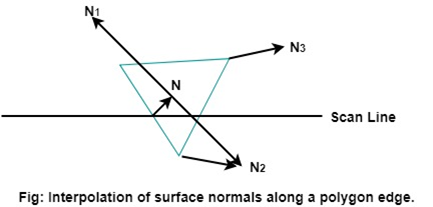


* **What is phong shading? Explain with a neat sketch**

**Ans:** Phong shading => It is a more accurate method for rendering a polygon surface is to interpolate the normal vector and then apply the illumination model to each surface point. This method developed by Phong Bui Tuong is called Phong Shading or normal vector Interpolation Shading. It displays more realistic highlights on a surface and greatly reduces the Match-band effect.

A polygon surface is rendered using Phong shading by carrying out the following steps:

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 calculate projected pixel intensities for the surface points.



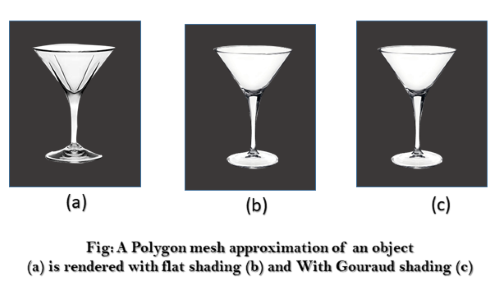


Incremental methods are used to evaluate normals between scan lines and along each scan line. At each pixel position along a scan line, the illumination model is applied to determine the surface intensity at that point.

Intensity calculations using an approximated normal vector at each point along the scan line produce more accurate results than the direct interpolation of intensities, as in Gouraud Shading. The trade-off, however, is that phong shading requires considerably more calculations.

* **Briefly explain constant shading and interpolating shading.**

**Ans:**  Constant shading: A fast and straightforward method for rendering an object with polygon surfaces is constant intensity shading, also called Flat Shading. In this method, a single intensity is calculated for each polygon. All points over the surface of the polygon are then displayed with the same intensity value. Constant Shading can be useful for quickly displaying the general appearances of the curved surface as shown in fig:



In general, flat shading of polygon facets provides an accurate rendering for an object if all of the following assumptions are valid:-

The object is a polyhedron and is not an approximation of an object with a curved surface.

All light sources illuminating the objects are sufficiently far from the surface so that N. L and the attenuation function are constant over the surface (where N is the unit normal to a surface and L is the unit direction vector to the point light source from a position on the surface).

The viewing position is sufficiently far from the surface so that V. R is constant over the surface (where V is the unit vector pointer to the viewer from the surface position and R represent a unit vector in the direction of ideal specular reflection).

Interpolated Shading- The idea of interpolative shading is to avoid computing the full lighting equation at each pixel by

interpolating quantites at the vertices of the faces.

Given vertices v1,v2,v3 we need to compute the normals for each vertex, compute the radiances for each vertex, project onto the window in device coordinates, and fill the polygon using

scan conversion.

There are two methods used for interpolative shading:

Gouraud Shading The radiance values are computed at the vertices and then linearly interpolated within each triangle.

Phong shading The normal values at each vertex are linearly interpolated within each triangle,

and the radiance is computed at each pixel.

* **Briefly explain specular reflection in the process of illuminating an object.**

**Ans:** Specular Reflection - Some portion of the object produce more light and shine/produce bright spots.

Perfect Mirror produce reflection in specular reflection direction only.

Other objects exhibit specular reflections over

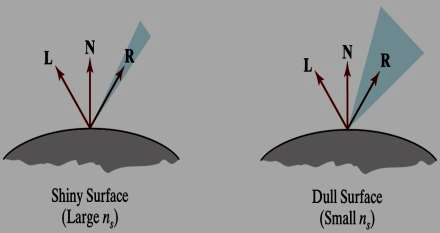
a finite range of viewing positions around

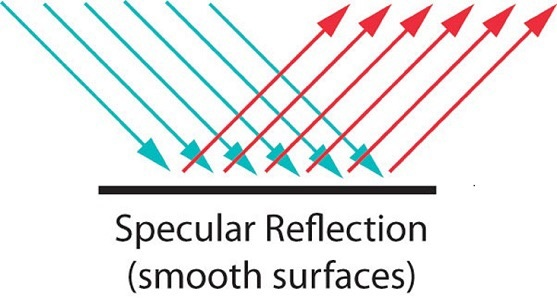
vector R.

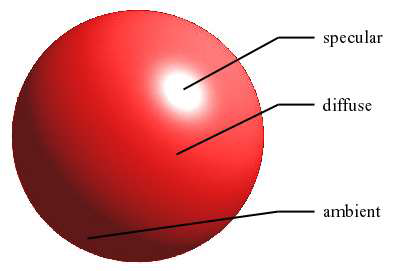
L=light source

N=normal

R=Specular Reflection







* **list the methods required for efficient visible surface algorithm.**

1. Back Face Detection Method

2. Depth Buffer Method

3. Scan line Method

4. Depth Sorting Method

**Ans:**

1)) BACK-FACE DETECTION METHOD

1. A fast and simple object-space method for identifying the back faces of a polyhedron is based on the "inside-outside" tests. A point (x, y, z) is "inside" a polygon surface with plane parameters A, B, C, and D if

**Ax+By+Cz+D<0**

2. When an inside point is along the line of sight to the surface, the polygon must be a back face (we are inside that face and cannot see the front of it from our viewing position).

3. We can simplify this test by considering the normal vector N to a polygon surface, which has Cartesian components (A, B, C). In general, if V is a vector in the viewing direction from the eye (or "camera") position, as shown in Fig, then this polygon is a backface if

**V.N>0**

4. Furthermore, if object descriptions have been converted to projection coordinates and our viewing direction is parallel to the viewing z, axis, then V = (0, 0, Vz) and

V.N=Vz.C

so that we only need to consider the sign of C, the; component of the normal vector N.

5. In a right-handed viewing system with viewing direction along the negative z, axis, the polygon is a back face if C < 0. Also, we cannot see any face who's normal has z component C=0, since our viewing direction is grazing that polygon. Thus, in general, we can label any polygon as a back face if its normal vector has a z-component value:

C<=0

6. In a Left-handed viewing system with viewing direction along the positive z, axis, the polygon is a back face if C > 0. Also, we cannot see any face who's normal has z component C=0, since our viewing direction is grazing that polygon. Thus, in general, we can label any polygon as a back face if its normal vector has a z-component value

C>=0

2. Depth Buffer Method ( also called as Z-Buffer method):

* A commonly used image-space approach to detecting visible surfaces is the depth-buffer method, which compares surface depths at each pixel position on the projection plane. This procedure is also referred to as the Z-buffer method, since object depth is usually measured from the view plane along the z axis of a viewing system.
* Each surface of a scene is processed separately, one point at a time across the surface. The method is usually applied to scenes containing only polygon surfaces, because depth values can be computed very quickly and the method is easy to implement. But the method can be applied to non planar surfaces.
* With object descriptions converted to projection coordinates, each (x, y, z) position on a polygon surface corresponds to the orthographic projection point (x, y) on the view plane. Therefore, for each pixel position (x, y) on the view plane, object depths can be compared by comparing z values. 4. Below Figure shows three surfaces at varying distances along the orthographic projection line from position (x ,y ) in a view plane taken as the xy, plane. Surface 1, is closest at this position, so its surface intensity value at (x, y) is saved. As implied by the name of this method, two buffer areas are required.

4. Below Figure shows three surfaces at varying distances along the orthographic projection line from position (x ,y ) in a view plane taken as the xy, plane. Surface 1, is closest at this position, so its surface intensity value at (x, y) is saved. As implied by the name of this method, two buffer areas are required.

5. Here two buffer’s are required: a. Depth Buffer( store’s depth value for each pixel ) b. Refresh Buffer( store’s intensity values for each pixel)

6. Initially, all positions in the depth buffer are set to 0 (minimum depth), and the refresh buffer is initialized to the background intensity. Each surface listed in the polygon tables is then processed, one scan line at a time, calculating the depth (z value) at each (x, y) pixel position.

7. The calculated depth is compared to the value previously stored in the depth buffer at that position. If the calculated depth is patter than the value stored in the depth buffer, the new depth value is stored, and the surface intensity at that position is determined and in the same xy location in the refresh buffer.

3. SCANLINE METHOD:

1. This method is extension of the scan-line algorithm for filling polygon interiors

2. This method is an example of image space method.

3. For all polygons intersecting each scan line

 Processed from left to right

 Depth calculations for each overlapping surface

 The intensity of the nearest position is entered into the refresh buffer

polygon tables:

The following polygon tables are used to store coordinate descriptions of polygons along with surfaces. Vertex Table: contains all vertices and their coordinates Edge table: contains all edge names and their coordinate endpoints Surface facet table: contains all surfaces along with their corresponding edge names.

Edge table

 Coordinate endpoints for each line

 Slope of each line

 Pointers into the polygon table

 Identify the surfaces bounded by each line Surface table

 Coefficients of the plane equation for each surface

 Intensity information for the surfaces

 Pointers into the edge table

4. DEPTH SORTING ALGORITHM (also known as Painters Algorithm):

This algorithm involves both object space and image space operations.  Image-space and object-space operations

o Sorting operations in both image and object-space

o the scan conversion of polygon surfaces in image-space

 Basic functions o Surfaces are sorted in order of decreasing depth o Surfaces are scan-converted in order, starting with the surface of greatest de pth 1. This algorithm also referred to as the painter’s algorithm In creating an oil painting

 First paints the background colors

 The most distant objects are added

 Then the nearer objects, and so forth

 Finally, the foregrounds are painted over all objects

 Each layer of paint covers up the previous layer this algorithm process the surfaces in the above order only.

2. Process Sort surfaces according to their distance from the view plane the intensities for the farthest surface are then entered into the refresh buffer Taking each succeeding surface in decreasing depth order

3. Sorting the surfaces can be done only by taking the following tests into consideration S-surface which is being tested S'-overlapping surface  Tests for each surface that overlaps with So The bounding rectangle in the xy plane for the two surfaces do not overlap (1) o Surface S is completely behind the overlapping surface(S') relative to the viewing position (2) o The overlapping surface(S') is completely in front of S relative to the viewing position (3) o The projections of the two surfaces onto the view plane do not overlap (4)

 If all the surfaces pass at least one of the tests, none of them is behind So No reordering is then necessary and S is scan converted

* **Explain principles and techniques involved in visible surface determination.**

**Ans**:

General Principles

We do not want to draw surfaces that are hidden. If we can quickly compute which surfaces are hidden, we can bypass them and draw only the surfaces that are visible.

For example, if we have a solid 6-sided cube, at most 3 of the 6 sides are visible at any one time, so at least 3 of the sides do not even need to be drawn because they are the back sides.

We also want to avoid having to draw the polygons in a particular order. We would like to tell the graphics routines to draw all the polygons in whatever order we choose and let the graphics routines determine which polygons are in front of which other polygons.

With the same cube as above, we do not want to have to compute for ourselves which order to draw the visible faces, and then tell the graphics routines to draw them in that order.

The idea is to speed up the drawing, and give the programmer an easier time, by doing some computation before drawing.

Unfortunately, these computations can take a lot of time, so special purpose hardware is often used to speed up the process.

Techniques

Two types of approaches:

* object space (object precision in our text)
* image space (image precision in our text)

**object space** algorithms do their work on the objects themselves before they are converted to pixels in the frame buffer. The resolution of the display device is irrelevant here as this calculation is done at the mathematical level of the objects

for each object a in the scene determine which parts of object a are visible (involves comparing the polygons in object a to other polygons in a and to polygons in every other object in the scene)

**image space** algorithms do their work as the objects are being converted to pixels in the frame buffer. The resolution of the display device is important here as this is done on a pixel by pixel basis.

for each pixel in the frame buffer determine which polygon is closest to the viewer at that pixel location color the pixel with the color of that polygon at that location

As in our discussion of vector vs raster graphics earlier in the term the mathematical (object space) algorithms tended to be used with the vector hardware whereas the pixel based (image space) algorithms tended to be used with the raster hardware.

When we talked about 3D transformations, we reached a point near the end when we converted the 3D (or 4D with homogeneous coordinates) to 2D by ignoring the Z values. Now we will use those Z values to determine which parts of which polygons (or lines) are in front of which parts of other polygons.

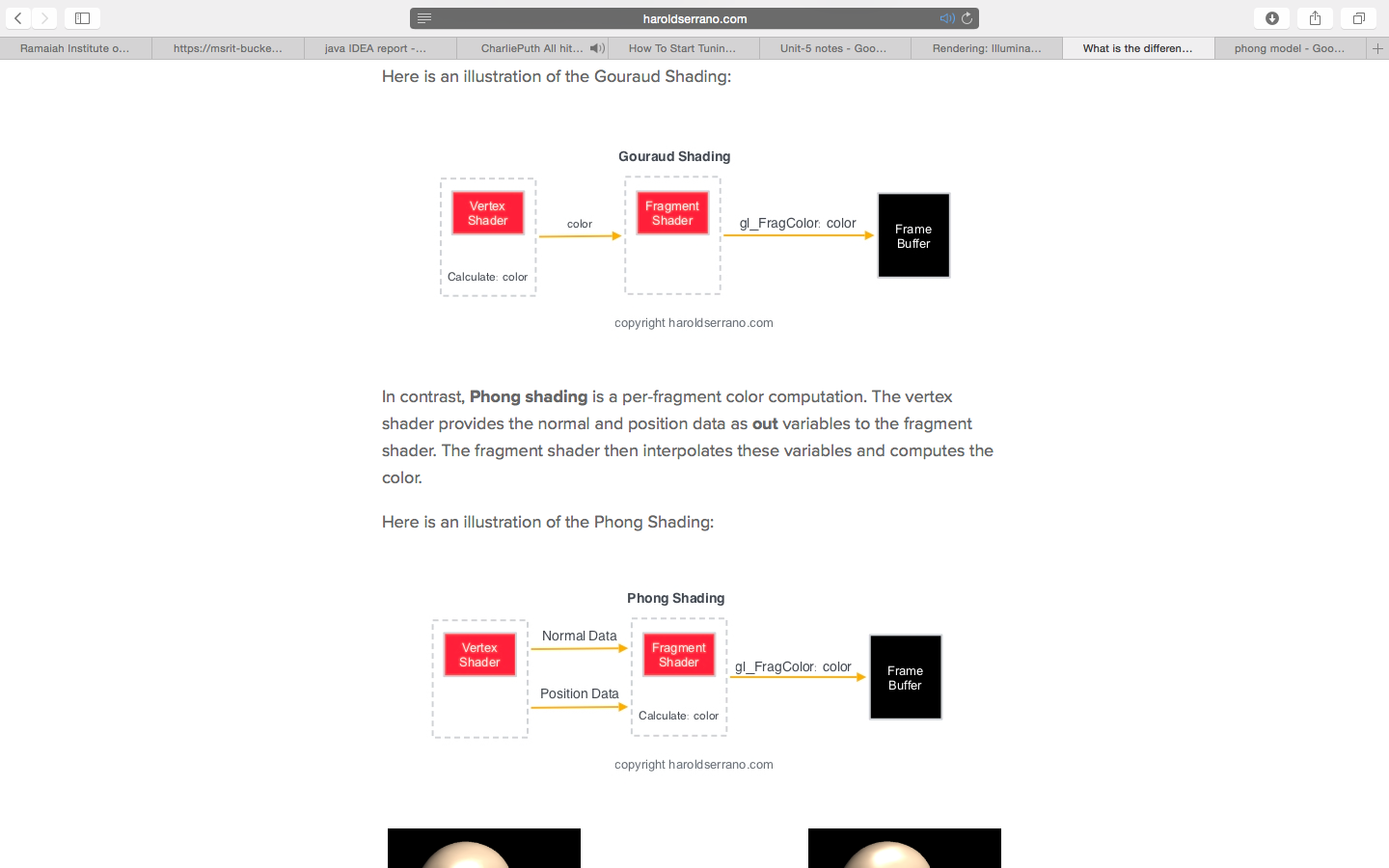
There are different levels of checking that can be done.

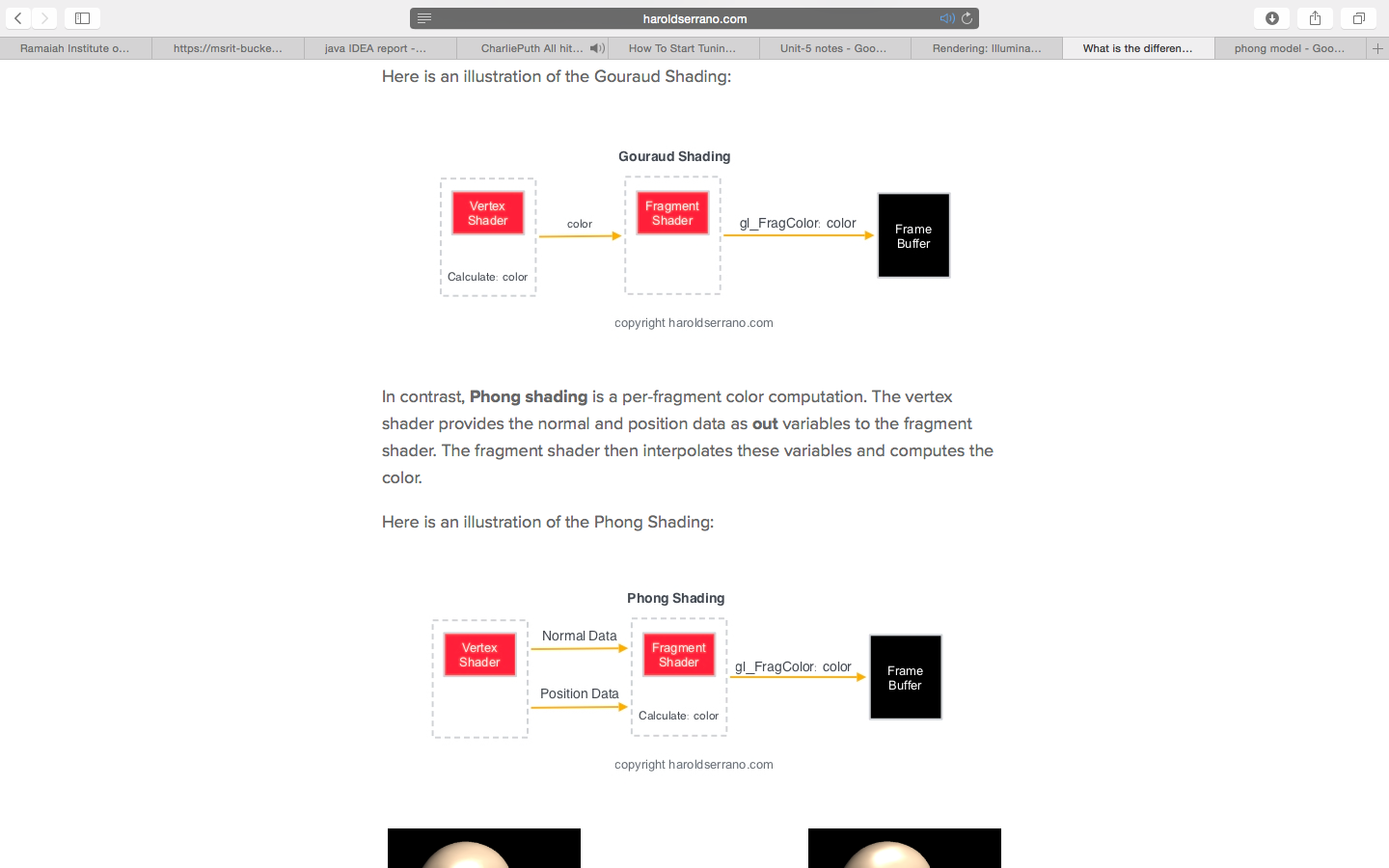
* Object
* Polygon
* part of a Polygon

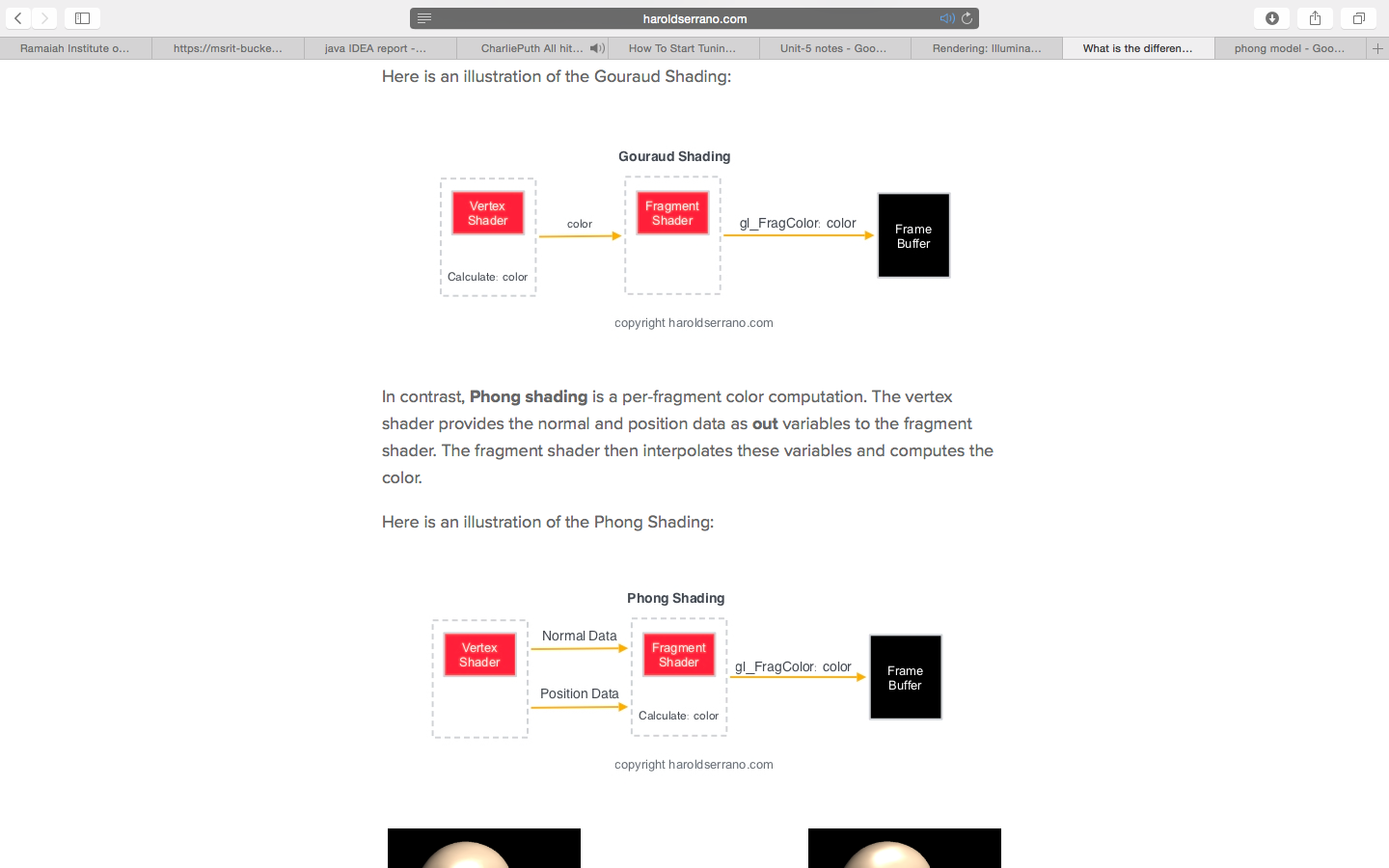
# what is the difference between gouraud shading and phong shading?

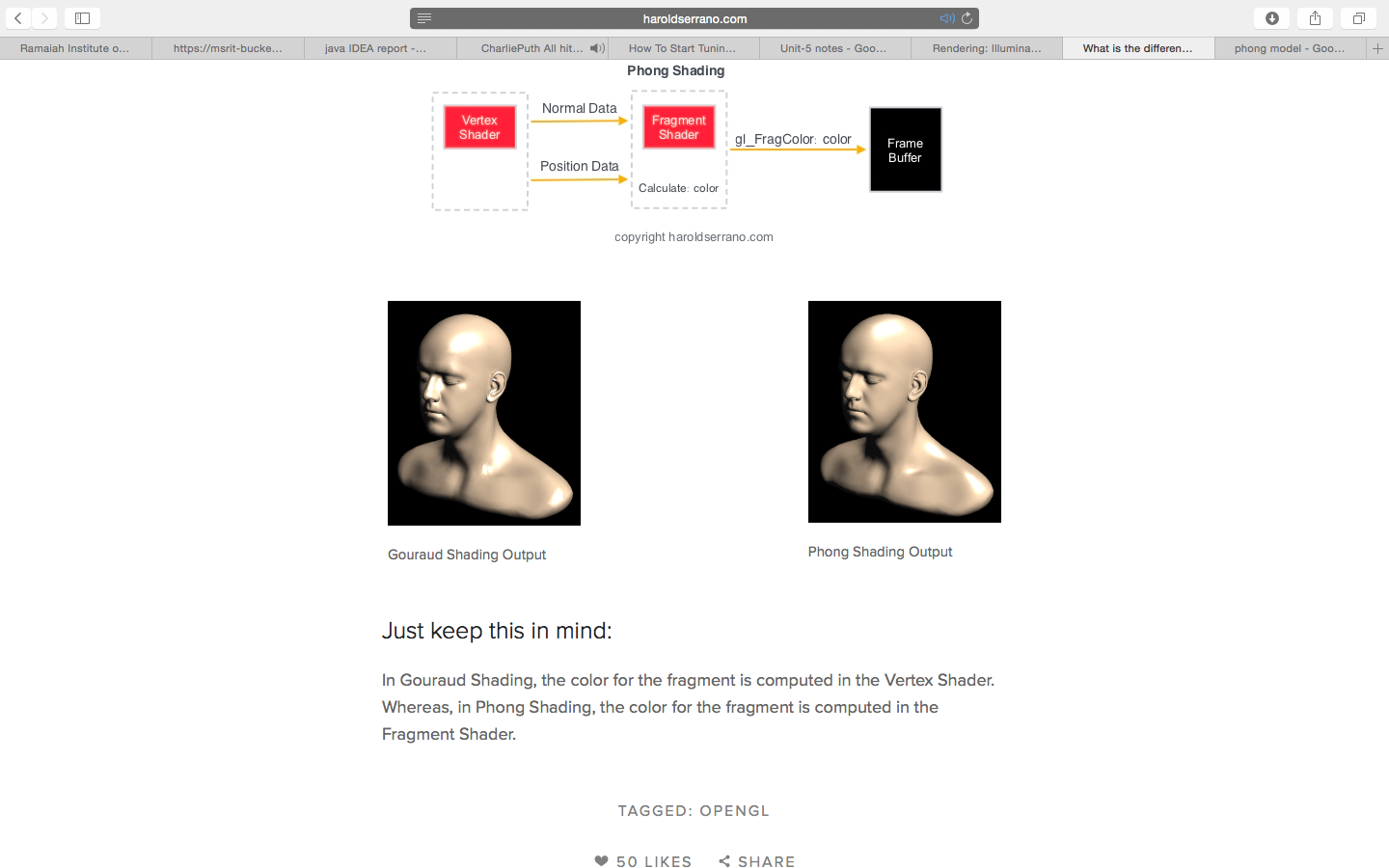
**Gouraud shading** (AKA Smooth Shading) is a per-vertex color computation. What this means is that the vertex shader must determine a color for each vertex and pass the color as an **out** variable to the fragment shader. Since this color is passed to the fragment shader as an **in** varying variable, it is interpolated across the fragments thus giving the smooth shading.

In contrast, **Phong shading** is a per-fragment color computation. The vertex shader provides the normal and position data as **out** variables to the fragment shader. The fragment shader then interpolates these variables and computes the color.

In Gouraud Shading, the color for the fragment is computed in the Vertex Shader. Whereas, in Phong Shading, the color for the fragment is computed in the Fragment Shader.





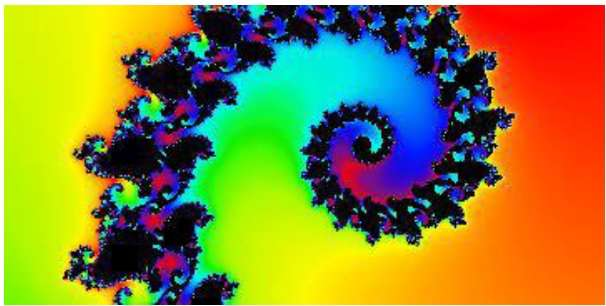


## What are Fractals?

Fractals are very complex pictures generated by a computer from a single formula. They are created using iterations. This means one formula is repeated with slightly different values over and over again, taking into account the results from the previous iteration.

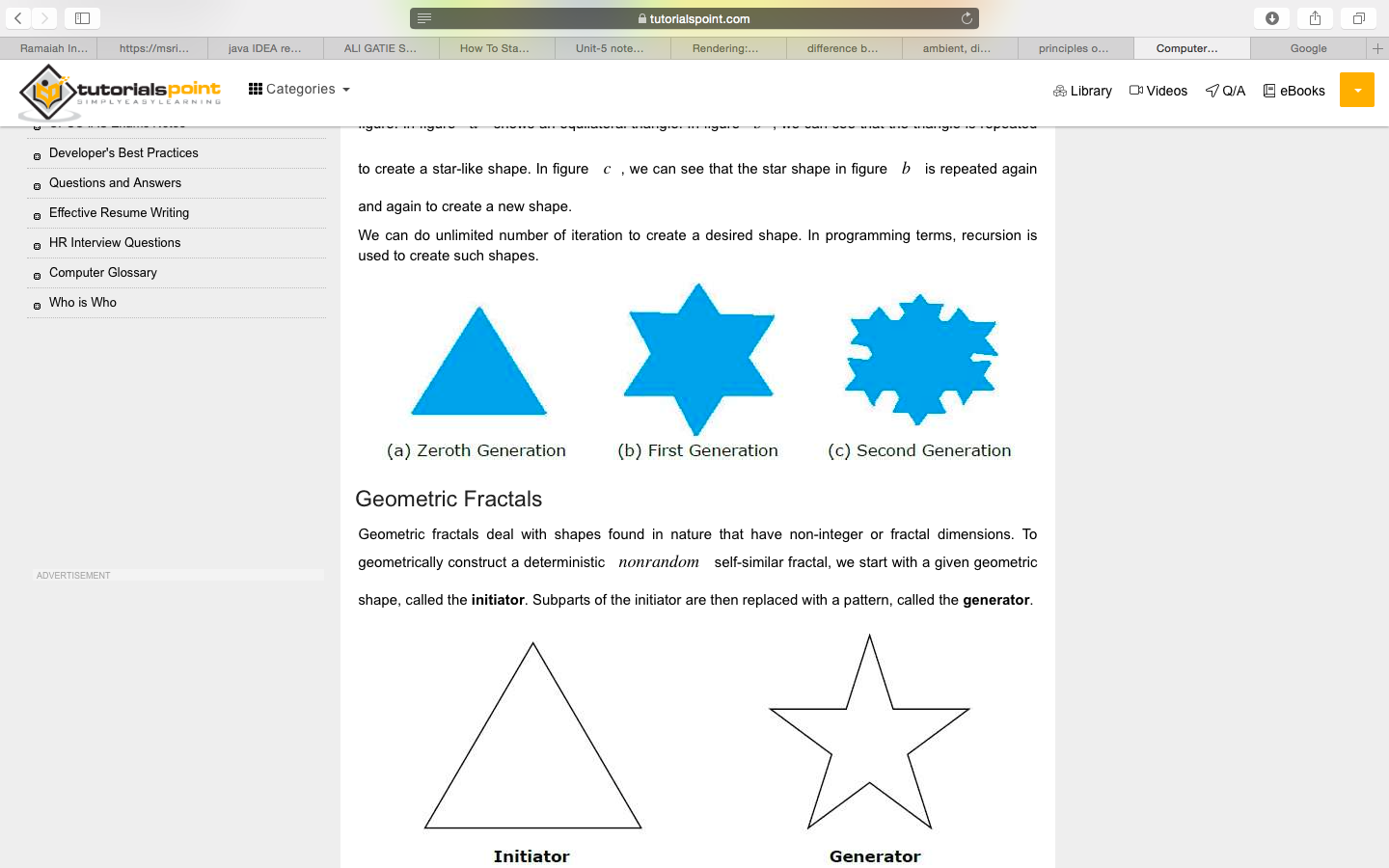
Fractals are used in many areas such as −

* Astronomy − For analyzing galaxies, rings of Saturn, etc.
* Biology/Chemistry − For depicting bacteria cultures, Chemical reactions, human anatomy, molecules, plants,
* Others − For depicting clouds, coastline and borderlines, data compression, diffusion, economy, fractal art, fractal music, landscapes, special effect, etc.

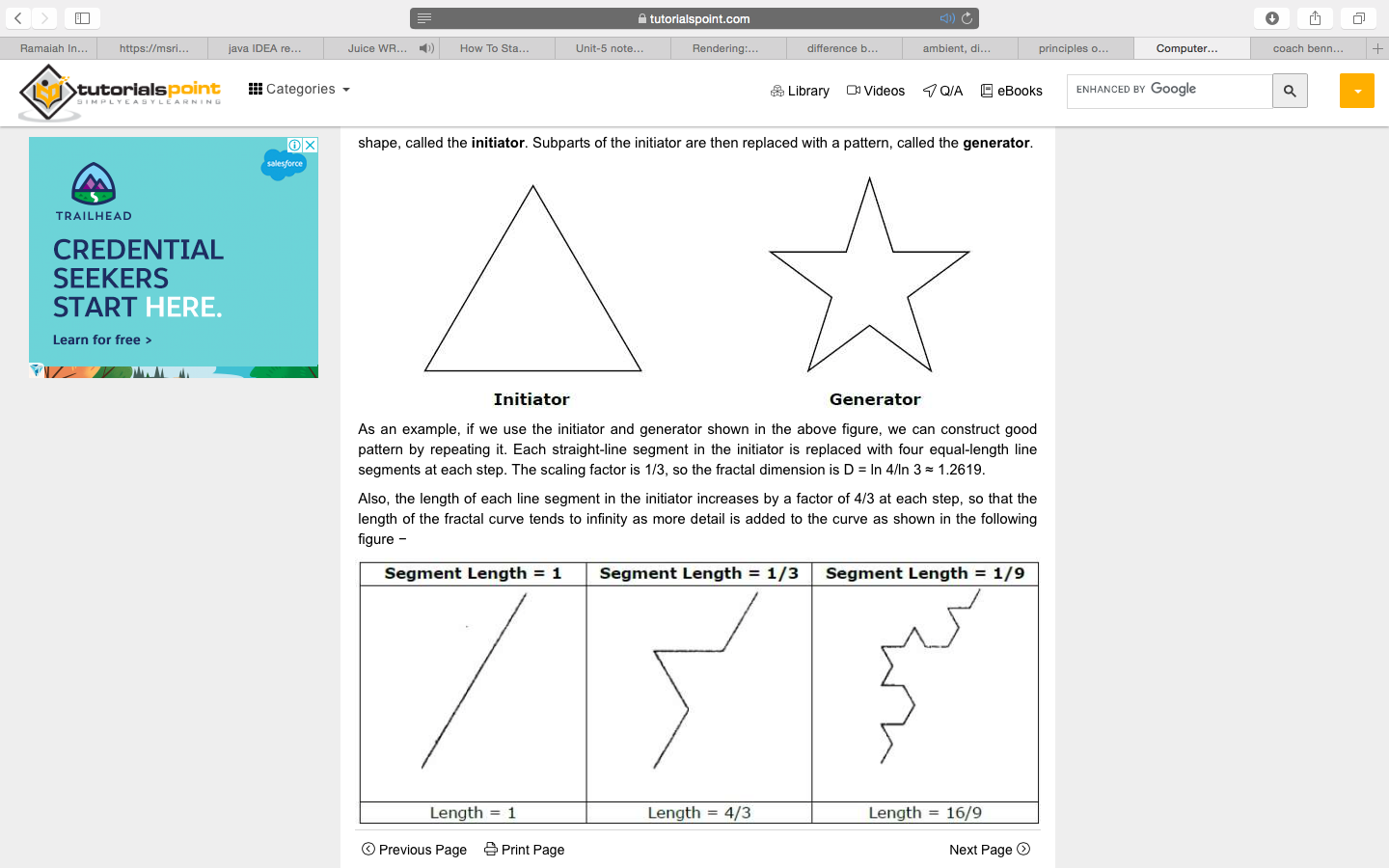


* How are fractals generated ?

Fractals can be generated by repeating the same shape over and over again as shown in the following figure. In figure a a shows an equilateral triangle. In figure b b, we can see that the triangle is repeated to create a star-like shape. In figure c c, we can see that the star shape in figure b b is repeated again and again to create a new shape.We can do unlimited number of iteration to create a desired shape. In programming terms, recursion is used to create such shapes.

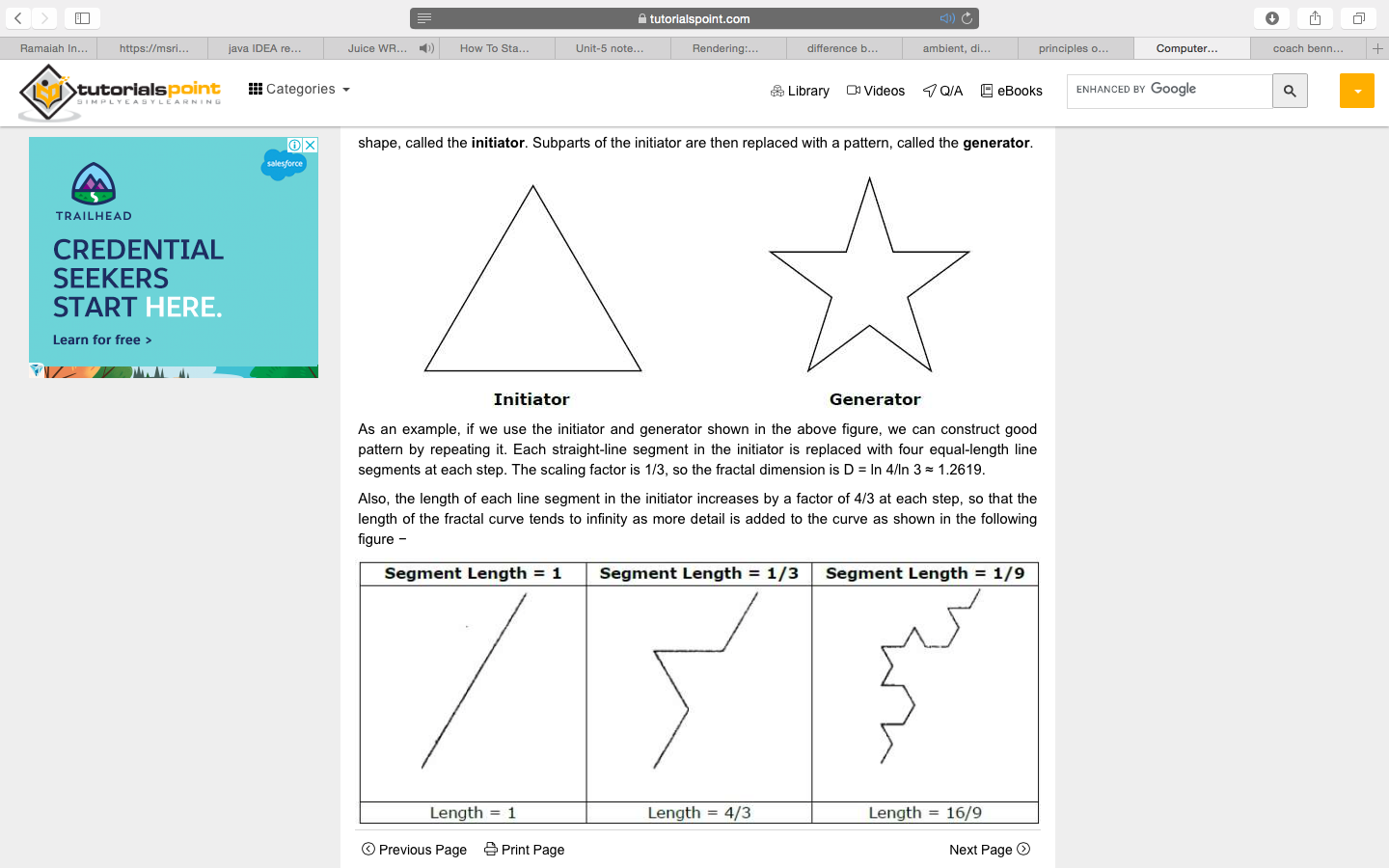


Geometric fractals deal with shapes found in nature that have non-integer or fractal dimensions. To geometrically construct a deterministic nonrandom self-similar fractal, we start with a given geometric shape, called the initiator. Subparts of the initiator are then replaced with a pattern, called the generator.



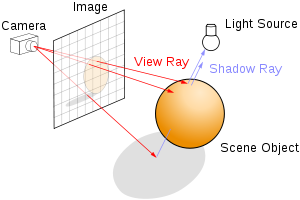
As an example, if we use the initiator and generator shown in the above figure, we can construct good pattern by repeating it. Each straight-line segment in the initiator is replaced with four equal-length line segments at each step. The scaling factor is 1/3, so the fractal dimension is D = ln 4/ln 3 ≈ 1.2619.

Also, the length of each line segment in the initiator increases by a factor of 4/3 at each step, so that the length of the fractal curve tends to infinity as more detail is added to the curve as shown in the following figure −



* what is ray tracing ?

**ray tracing** is a [rendering](https://en.wikipedia.org/wiki/Rendering_(computer_graphics)) technique for generating an [image](https://en.wikipedia.org/wiki/Digital_image) by tracing the path of [light](https://en.wikipedia.org/wiki/Light) as [pixels](https://en.wikipedia.org/wiki/Pixel) in an [image plane](https://en.wikipedia.org/wiki/Image_plane) and simulating the effects of its encounters with virtual objects. The technique is capable of producing a high degree of visual realism, more so than typical [scanline rendering](https://en.wikipedia.org/wiki/Scanline_rendering) methods, but at a greater [computational cost](https://en.wikipedia.org/wiki/Computation_time). This makes ray tracing best suited for applications where taking a relatively long time to render can be tolerated, such as in still computer-generated images, and film and television [visual effects](https://en.wikipedia.org/wiki/Visual_effects) (VFX), but more poorly suited to [real-time](https://en.wikipedia.org/wiki/Real-time_computer_graphics)applications such as [video games](https://en.wikipedia.org/wiki/Video_game), where [speed is critical](https://en.wikipedia.org/wiki/Frame_rate) in rendering each [frame](https://en.wikipedia.org/wiki/Film_frame).



### 

### 

### Effects such as reflections and [shadows](https://en.wikipedia.org/wiki/Shadow), which are difficult to simulate using other algorithms, are a natural result of the ray tracing algorithm. The computational independence of each ray makes ray tracing amenable to a basic level of computation.

A serious disadvantages of ray tracing is performance (though it can in theory be faster than traditional scanline rendering depending on scene complexity vs. number of pixels on-screen)

-ray tracing in real time was usually considered impossible on consumer hardware for nontrivial tasks. Scanline algorithms and other algorithms use data coherence to share computations between pixels, while ray tracing normally starts the process anew, treating each eye ray separately.

-it does handle interreflection and optical effects such as refraction accurately

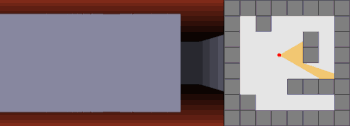
-is not necessarily the most realistic method , Methods that trace rays, but include additional techniques ([photon mapping](https://en.wikipedia.org/wiki/Photon_mapping), [path tracing](https://en.wikipedia.org/wiki/Path_tracing)), give a far more accurate simulation of real-world lighting.

* what is ray casting ?

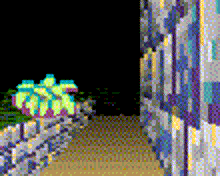
**Ray casting** is the use of ray–surface [intersection tests](https://en.wikipedia.org/wiki/Glossary_of_computer_graphics#intersection_test) to solve a variety of problems in 3D [computer graphics](https://en.wikipedia.org/wiki/Computer_graphics) and [computational geometry](https://en.wikipedia.org/wiki/Computational_geometry). The term was first used in computer graphics in a 1982 paper by Scott Roth to describe a method for rendering [constructive solid geometry](https://en.wikipedia.org/wiki/Constructive_solid_geometry) models.[[1]](https://en.wikipedia.org/wiki/Ray_casting#cite_note-1)

*Ray casting* can refer to a variety of problems and techniques:

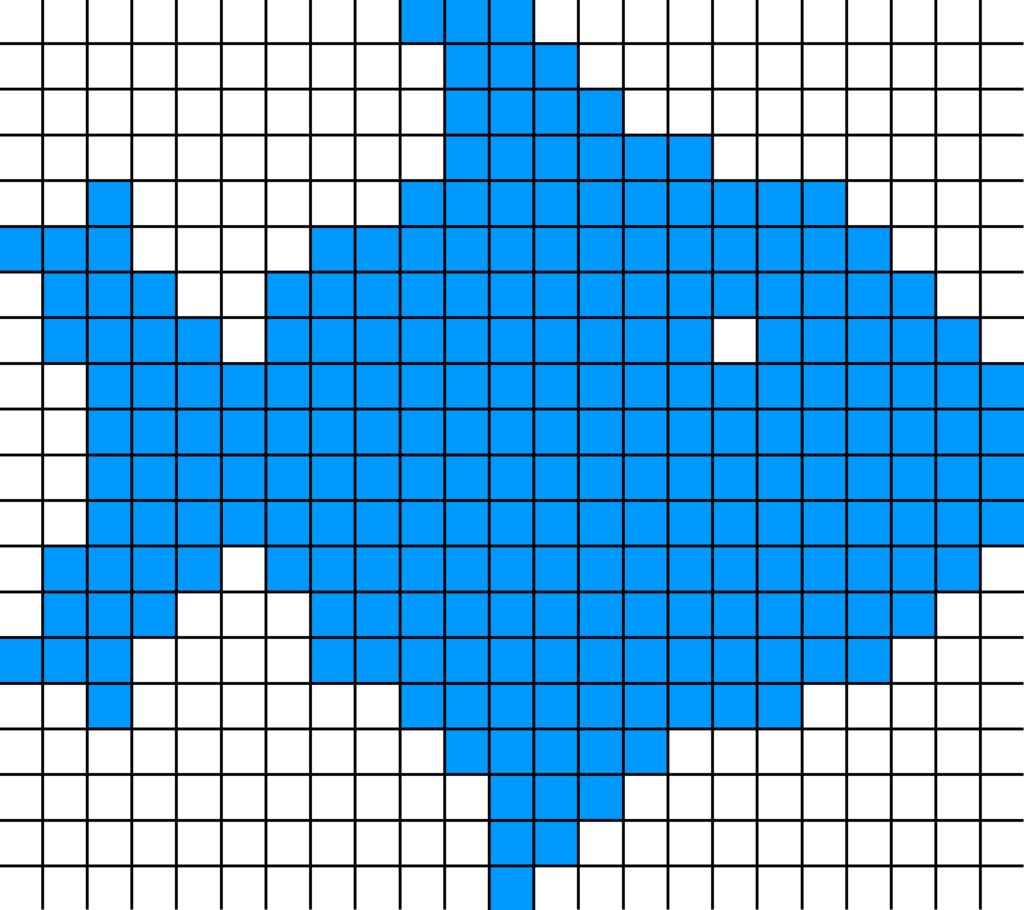
* the general problem of determining the first object intersected by a ray,[[2]](https://en.wikipedia.org/wiki/Ray_casting#cite_note-2) as in [collision detection](https://en.wikipedia.org/wiki/Collision_detection)
* a technique for [hidden surface removal](https://en.wikipedia.org/wiki/Hidden_surface_removal) based on finding the first intersection of a ray cast from the eye through each pixel of an image
* a [non-recursive](https://en.wikipedia.org/wiki/Recursion_(computer_science)) [ray tracing](https://en.wikipedia.org/wiki/Ray_tracing_(graphics)) [rendering algorithm](https://en.wikipedia.org/wiki/Rendering_algorithm) that only casts primary rays
* [volume ray casting](https://en.wikipedia.org/wiki/Volume_ray_casting)(it computes 2D images from 3D volumetric data sets (3D [scalar fields](https://en.wikipedia.org/wiki/Scalar_field)). Volume ray casting, which processes volume data), a direct [volume rendering](https://en.wikipedia.org/wiki/Volume_rendering) method in which the ray is "pushed through" the object and the 3D [scalar field](https://en.wikipedia.org/wiki/Scalar_field) of interest is sampled along the ray inside the object. No secondary rays are spawned in this method.[[3]](https://en.wikipedia.org/wiki/Ray_casting#cite_note-w-3)

here is an example for ray casting in all the directions.

-one of the major applications of this can be used in development of virtual world games



* what is rasterization ?

**Rasterisation** (or **rasterization**) is the task of taking an image described in a [vector graphics](https://en.wikipedia.org/wiki/Vector_graphics) format (shapes) and converting it into a [raster image](https://en.wikipedia.org/wiki/Raster_image) (a series of [pixels](https://en.wikipedia.org/wiki/Pixel), dots or lines, which, when displayed together, create the image which was represented via shapes).[]](https://en.wikipedia.org/wiki/Rasterisation#cite_note-Chang2007-2) The rasterised image may then be displayed on a [computer display](https://en.wikipedia.org/wiki/Computer_display), [video display](https://en.wikipedia.org/wiki/Video_display) or [printer](https://en.wikipedia.org/wiki/Computer_printer), or stored in a [bitmap](https://en.wikipedia.org/wiki/Bitmap) file format. Rasterisation may refer to the technique of drawing [3D models](https://en.wikipedia.org/wiki/3D_model_(computer_graphics)), or the conversion of 2D [rendering primitives](https://en.wikipedia.org/wiki/Rendering_primitive) such as [polygons](https://en.wikipedia.org/wiki/Polygon), [line segments](https://en.wikipedia.org/wiki/Line_segment) into a rasterized format.

to better understand the concept of ray casting ,ray tracing click here

<https://slideplayer.com/slide/9807646/>