This book covers two main topics of computer graphics, 1) Raytracing 2) Rasterization.

Chapter 1

It introduces the Coordinate System (x,y,z) and color models.

CMYK color model is Subtractive, RGB is additive color model.

Most images use color depth of 8 bits per channel that gives 24 bits per pixel. It means each pixel can represent 2^24 different colors.

For color manipulation, you can multiply each channel by a constant value to increase brightness. You can add two chanels to mix/generate a different color channel.

Usually we put objects in a coordinate system, where x and z are horizontal, y axis is vertical.

Part 1 Raytracing

Chapter 2 : Bacie Raytracing

Each scene can be broken down to small pixel regions. We need to color each region with the right color.

We need position and orientation of the camera, and determine field of view.

https://gabrielgambetta.com/computer-graphics-from-scratch/02-basic-raytracing.html

Chapter 3 : Light

It covers three different light source: **point** lights**, directional** lights and **ambient** light.

We can classify objects intor two broad objects, "**matte**" and "**shiny**" objects.

Modeling Diffuse Reflection for "matte" objects.

Modeling Specular Reflection for "shiny" objects.

Chapter 4 : Shadow and Reflections

We need to model how light interacts with the scene: 1) objects cast shadows and 2) objects reflect on other objects.

If the intersection point is in shadow, we ignore the illumination from the light.

We can use *recursion limit* to control the depth of reflections. Generally speaking, maximum 3 levels are enough.

Chapter 5 : Extending the Raytracer

We have lights and objects in the scene. We should be able to place the camera anywhere in the scene.

To generate arbitrary camera positioning, we only need to change the value for O. (I think O means objects)

Performance optimizations for : 1) parallelization 2) cache immutables 3) shadow optimizations 4) bounding volume hierarchy (bvh) 5) subsampling .

Constructive solid geometry : allow rendering the results of set operators between objects without ever having to explicitly compute the results.

Render transparency : 1) similar to reflection 2) need to blend the color with the local and reflected colors

Supersampling : look for accuracy instead of performance

Part 2 Rasterization

Chapter 6 : lines

It shows the equations to draw line with slope.
It shows the linear interpolation function.

Chapter 7 : filled triangles

Use drawline() function to draw the wireframe triangles.
Use horizontal segments to draw filled triangle.

Chapter 8 : Shaded triangles

Shaded triangle: a triangle filled with a color gradient

It works on edge shading first, then interior shading.

Chapter 9 : Perspective Projection

This chapter answers a question, how to render 3d scenes.
We need to figure out how to turn 3d scene coordinate into 2d canvas coordinates.

A simple perspective projection: camera sees P far away through a pont of P' on a viewport.

Project equation is used to find the P' on the projection plane.

Chapter 10 : Describing and Rendering a Scene

We can use a generic structure to represent objects made of triangles.
We can use wireframe triangle to render a cube.
We can apply model transform (scaling factor, rotation, translation to a specific point) to place an object to a different location with a specific rotation.
It needs three matrices for the perspective projection and viewport-to-canvas mapping,2)for the instance transform, 3) for camera transform.

Chapter 11 : clipping

Previous chapters work on the perspective projection for points that are in front of the camera. This chapter discusses the techniques to identify dentify points triangles,entrire objects that are behind camera.

Not to render anything behind the projection plan Z = d. Use *clipping plane* to classify point as inside or outside of the *clipping volume.* Only render the parts of the scene that are inside the clipping volume.

Define places to clip to exactly what should be visiblel on the viewpoint.

Clip triangles : 1) accepted 2) discarded 3) clip.

Chapter 12 : hidden surface removal

The rest chapters focus on improving visual quality of the rendered scene.

How to render solid objects?
Painter's Algorithm : sort the triangles according to their distance to the camera. Draw backgrounds first, then cover parts of them with foreground objects.

Limitations: sometimes a correct ordering of triangles doesn't exist at all.
Use depth buffering: solve it at the pixel level, store the z coordinate of every pixel on the canvas. Using 1/z instead of z for correct linear interpolation.

To run things faster, Back Face Culing is applied. It avoids computing pixels that will never be visible.
We need to classify triangle as "front" or "back", given they point toward or away the camera. Then, we discard (cull) all the back faces, saving valuable computation time.

Chapter 13 : shading

This chapter covers how to add lights to the scene and how to illuminate the objects.
Shading deals with techniques to extend light effects on a discrete sets of points to entire obj.
Illumination refers to the math and algos to compute the light effect on a single point.

Flat shading : pick any point in a triangle, compute the illumination use it to shade the whole triangle (make curved obj look flat).

Gouraud Shading : compute illumination at its three vertices of a triangle.
It has drawbacks : the closer the point light is to a face, the darker it looks.

Phong Shading : calculate illumination at every pixel fo the triangle.

Rendering by rasterizer is faster than rendering by raytracer.
Using raytracing producing better image than using rasterizer.

Chapter 14 : textures

How to use rasterizer to render real-world objects?
Use textures to add visual details to the surface of our objects.

First, we need an image (texture) to paint on the triangles.

We use u and v for the texture coordinates.

- We can use attribute mapping to interpolate the values of new coordinate system (u,v) across the face of the triangle
- Compute (tx, ty)
- fetch the texel
- Apply shading, paint the pixel with the resulting color

Problem: linear interpolation doesn't produce the expected perspective-correct results
Solution: use u/z, v/z, rather than u, v

Bilinear Filtering
Issues: blocky , the triangle has more pixels than the texture has texels, each texel is mapped to many consecutive pixels (nearest neighbor filtering)

Solution : bilinear filtering, linear interpolation twice, once in each dimension.

Issues :   what about number of texels > pixel ?
Mipmapping: precompute texture at different scales,
Trilinear filtering: select the closest match the relative size of the texture and the square

Chapter 15 : extending the rasterizer

Previous chapter presents texture mapping which gives finer-grained appearance of a surface. However, texture mapping doesn't change the shape of triangle (still flat).

Solution: *Normal mapping*, which paints the texture using normal vectors, instead of colors.

Bump mapping: add subtle details.

Environment mapping: we wan to show objects reflect one another.
Solution : Cube mapping which places a camera in the middle of the room, render the scene 6 times (up,down,left, right, front back) , keep 6 renders as texture.

Shadows

- Stencil shadows (well defined edges, hard shadows)
    - Render the scene in several passes

(Noted that, a complicated problem can be broken down into simple problem, solve the small problems multiple times with different directions)

- - Creating shadow volumes
        - Cast light from front faces to a big distance beyond the scene
    - Counting Shadow Volumes - ray intersections
        - Count the shadow interaction (entry +1, leave -1) , zero means light, non zero means shadow
    - Setting up stencil buffer
        - Rasterizer needs to keep the counters without actual computing
        - Render scene by ambient light, no shadow
        - Count the times the ray leaves a shadow volume
        - Count the times the ray enters a shadow volume
        - Use single light render the scene
        - Repeat this process for every light
        - Final render, add them together pixel by pixel

    - Shadow mapping
        - Render shadows with less defined edges , soft shadows.