GP1 – Software Rasterization Pipeline (High-Level)

INPUT **(VERTEX_IN)**

Vertices (in World Space)

Each Vertex has at least a POSITION attribute, but can be decorated with extra attributes such as COLOR, UV, ...

Every Vertex

PROJECTION STAGE
[All vertices]

[Positions]
Model(space) → NDC(space)

Multiply vertex positions with WVP matrix
World → Model Transformation
View → Camera Transformation
Projection → Projection based on AspectRatio & FOV
+ Apply Perspective Divide

[Normals & Tangents]
Model(space) → World(space)

Multiply Normals & Tangents with (3x3) World matrix
(We only want to rotate them, so cast (4x4) World matrix to a (3x3) matrix [FMatrix3] → Discards the translational component)

[ViewDirection]

Normalize(WorldPosition – CameraPosition)
WorldPosition → ModelSpace position multiplied with (4x4) World matrix

OUTPUT **(VERTEX_OUT)**

Vertices (in 'NDC' space)

Positions are Projected, Normals & Tangents are transformed, ViewDirection is calculated, other attributes are passed through

Position X & Y → NDC (Range [-1,1])
Position Z → NDC (Range [0,1])
Position W → ViewSpace (Distance from camera)

INPUT

Vertices (in 'NDC' space)

Each Vertex has at least a POSITION attribute, but can be decorated with extra attributes such as COLOR, UV, ...

Every Triangle

Every Pixel

RASTERIZATION STAGE
[Each Triangle → Iterate Pixels]

Frustum Culling (+ Clipping)
(NDC) Vertex inside Camera Frustum?
X & Y between [-1,1]
Z between [0,1]
Ignore Triangle if not in Frustum

NDC (space) → Raster/Screen (space)
See slides for formula. After this transformation your Position X & Y should be in ScreenSpace (2D raster coordinates).

Optimization → Calculate Triangle BoundingBox (BB) in Rasterspace → Only iterate the pixels covered by this BB.

PIXEL inside TRIANGLE?
Cross product between each Triangle edge and Vertex-2-Pixel vectors (2D vectors) → IF signs of all cross-product results are the same THEN pixel is inside the Triangle.

Depth Interpolation + Depth Test
Correct 'Depth Interpolation' → LAB 3

Vertex Attribute Interpolation
Correct 'Attribute Interpolation' → LAB 3

Barycentric Coordinates

Cross Product Results can be used to calculate interpolation weights for Depth & Attribute Interpolation.

+ See LAB 3 for 'correct' Depth & Vertex Interpolation

OUTPUT (Only if DepthTest succeeds)

Single VERTEX_OUT structure containing interpolated attributes (Position, Color, UV, Normal, Tangent, ViewDirection)

CULL MODE
Depends on the cross-product sign
All Negative vs All Positive ==
FrontFace vs BackFace culling

INPUT (Interpolated Vertex Attributes for Pixel)

Position : Vector4
(X & Y → ScreenSpace || Z → Interpolated Normalized Depth || W → Interpolated Camera-Space Depth)
Interpolated Color : ColorRGB
Interpolated UV : Vector2
Interpolated Normal : Vector3
Interpolated Tangent : Vector3
Interpolated ViewDirection : Vector3

Every Pixel that passed the Depth Test

PIXEL SHADING STAGE
[Calculate Color for given Pixel]

CONSTANTS (Directional Light)
LightDirection → { .577f, -.577f, .577f }
LightIntensity → 7.f
Specular Shininess → 25.f

NORMAL MAPPING (Normal Texture)
Calculate TangentSpaceAxis, multiply with sampled normal from NormalMap → Use this resulting Normal as new Normal

DIFFUSE (Diffuse Texture)
Basic Lambert Diffuse

SPECULAR (Specular + Gloss Texture)
Phong Specular (similar to RayTracer)

AMBIENT (optional)
Solid Color → RGB{ .025f, .025f, .025f }

OUTPUT → Final Pixel Color (Diffuse + Specular + Ambient)
(Normalize Color before writing to backbuffer)

* This is only a High-Level overview/guide! → Use the slides for deeper explanation & proper implementation guidelines.