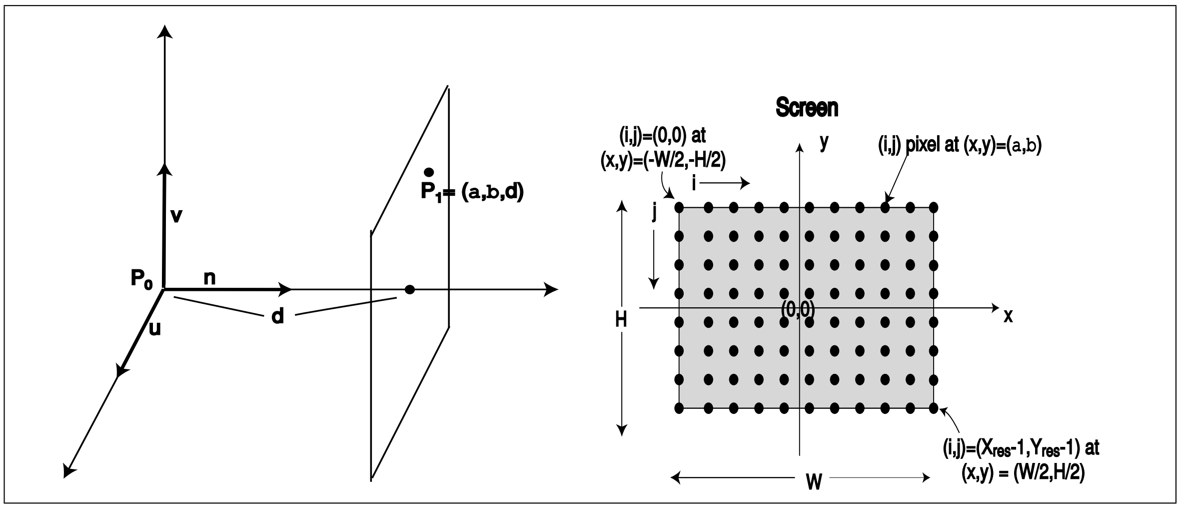
Implementing a Simple Ray Tracing

1. Inputs:

   (a) Camera/Screen Information:

$$
\begin{aligned}
P_0 &= \text{location of camera} \\
\text{VPN} &= \text{normal to view plane} \\
\text{VUP} &= \text{up direction} \\
d &= \text{distance of camera from view screen} \\
H &= \text{height of screen} \\
W &= \text{width of screen} \\
X_{res} &= \text{number of pixels per column} \\
Y_{res} &= \text{number of pixels per row}
\end{aligned}
$$



   (b) Scene Information
   $I_a = (I_{a,r}, I_{a,g}, I_{a,b})$ = RGB components of the intensity of ambient light (constant throughout scene). Note that this is a property of the light and not of the object.

   (c) Objects

      i. spheres : requires center, radius
      ii. planes : requires normal and point on plane
      iii. For each object, we need
         - $0 \leq k_a \leq 1$ = coefficient of ambient light
         - RGB color = (c_r, c_g, c_b) where $0 \leq c_{r,g,b} \leq 255$

2. Compute Screen/View unit vectors $\hat{u}$, $\hat{v}$, $\hat{n}$:

   If the screen coordinates of the i, j^{th} pixel are expressed as

$$(\alpha, \beta) \equiv \left(-\frac{W}{2} + \frac{W \cdot i}{X_{res} - 1}, -\frac{H}{2} + \frac{H \cdot j}{Y_{res} - 1}\right)$$

   then, the direction of the ray is (assuming a *left handed* coordinate system):

$$P_1 - P_0 = \alpha\hat{u} + \beta\hat{v} + d\hat{n}$$

3. Compute Pixel Color

   Loop over column i and row j (i.e. for each pixel (i, j)):

   (a) Compute Ray :

$$ray = P_0 + t\ dir = P_0 + t\frac{(\alpha\hat{u} + \beta\hat{v} + d\hat{n})}{||(\alpha\hat{u} + \beta\hat{v} + d\hat{n})||}$$

   (b) Loop over objects in world.
   Compute the intersection of object with ray (i.e. the t value). Keep track of smallest t value (this is closest object).

   (c) For the closest object:
   Determine the color that is assigned to the i,j-th pixel:

$$\text{RGB pixel color} = k_a(I_{a,r}c_r, I_{a,g}c_g, I_{a,b}c_b)$$

   where each component must be restricted to being between 0 and 255.