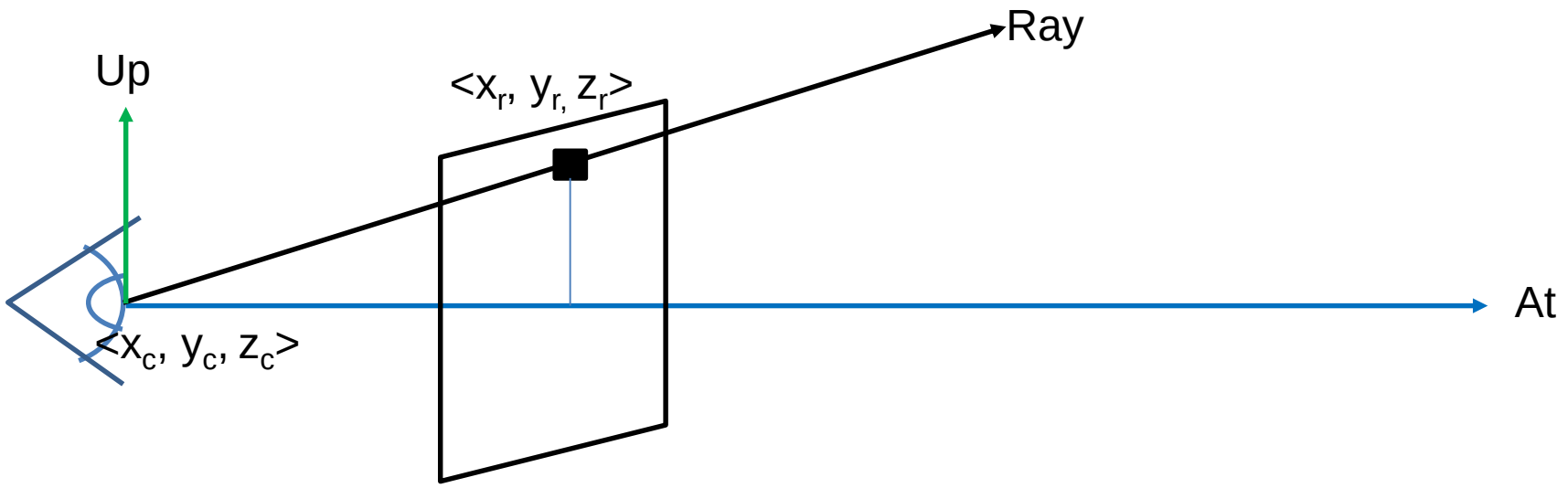
Ray Tracing

Creating Rays

You can treat the plane defined by the 2 dimensional set of pixels (raster) as to coincide with the camera's near clip plane

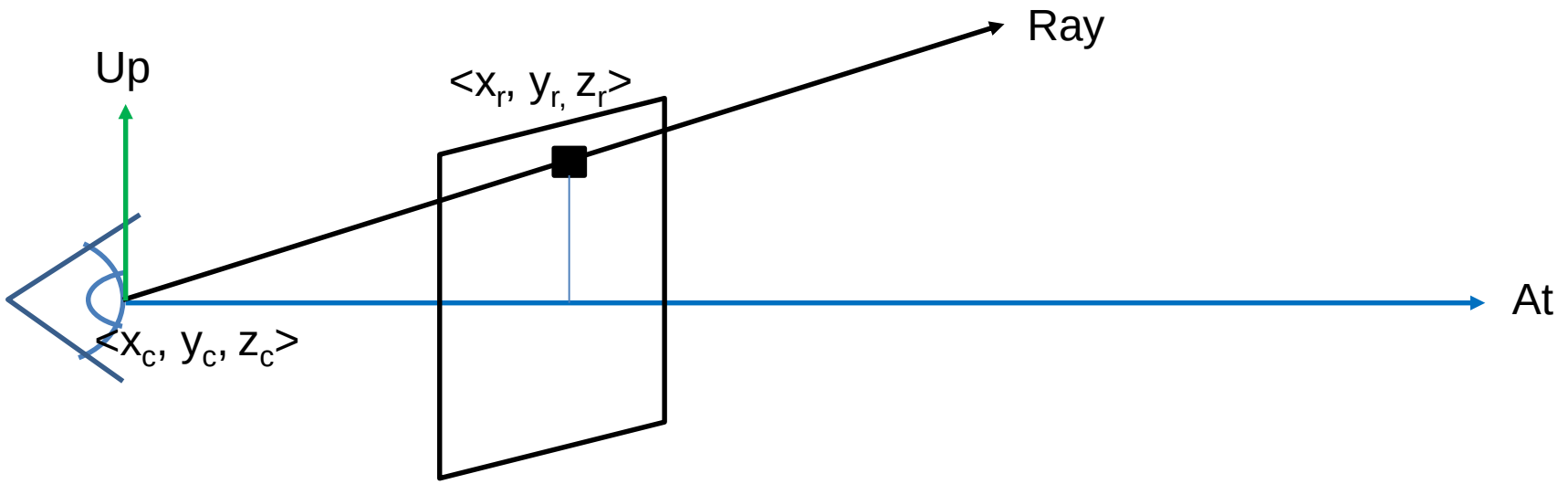


Creating Rays

A point on the raster can be defined as:

$$\langle x_r, y_r, z_r \rangle = \text{Cam.Pos} + \text{Cam.At} + rel_X * \text{Cam.Right} + rel_Y * \text{Cam.Up}$$

where Cam.Pos is $\langle x_c, y_c, z_c \rangle$



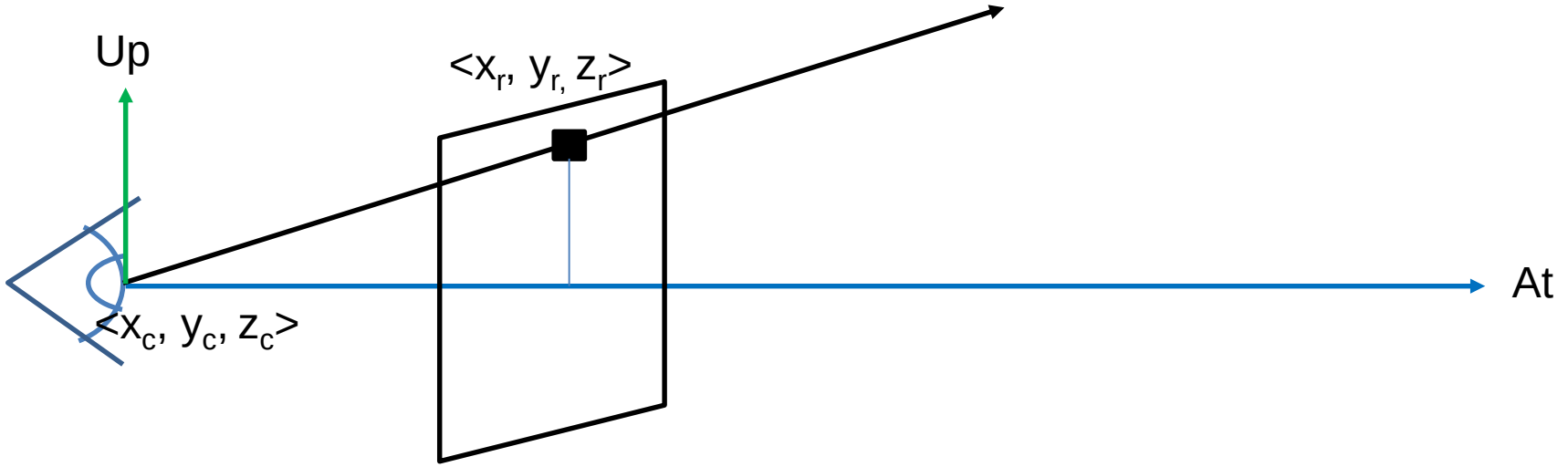
Creating Rays

The ray vector is defined as:

$$RayVec = \langle x_r, y_r, z_r \rangle - \langle x_c, y_c, z_c \rangle$$

or

$$RayVec = Cam.At + rel_X * Cam.Right + rel_Y * Cam.Up$$



rel_X and rel_Y

$min_X = -||Cam.At||*tan(\theta_x)$
$max_X = ||Cam.At||*tan(\theta_x)$

$min_y = -||Cam.At||*tan(\theta_y)$
$max_y = ||Cam.At||*tan(\theta_y)$

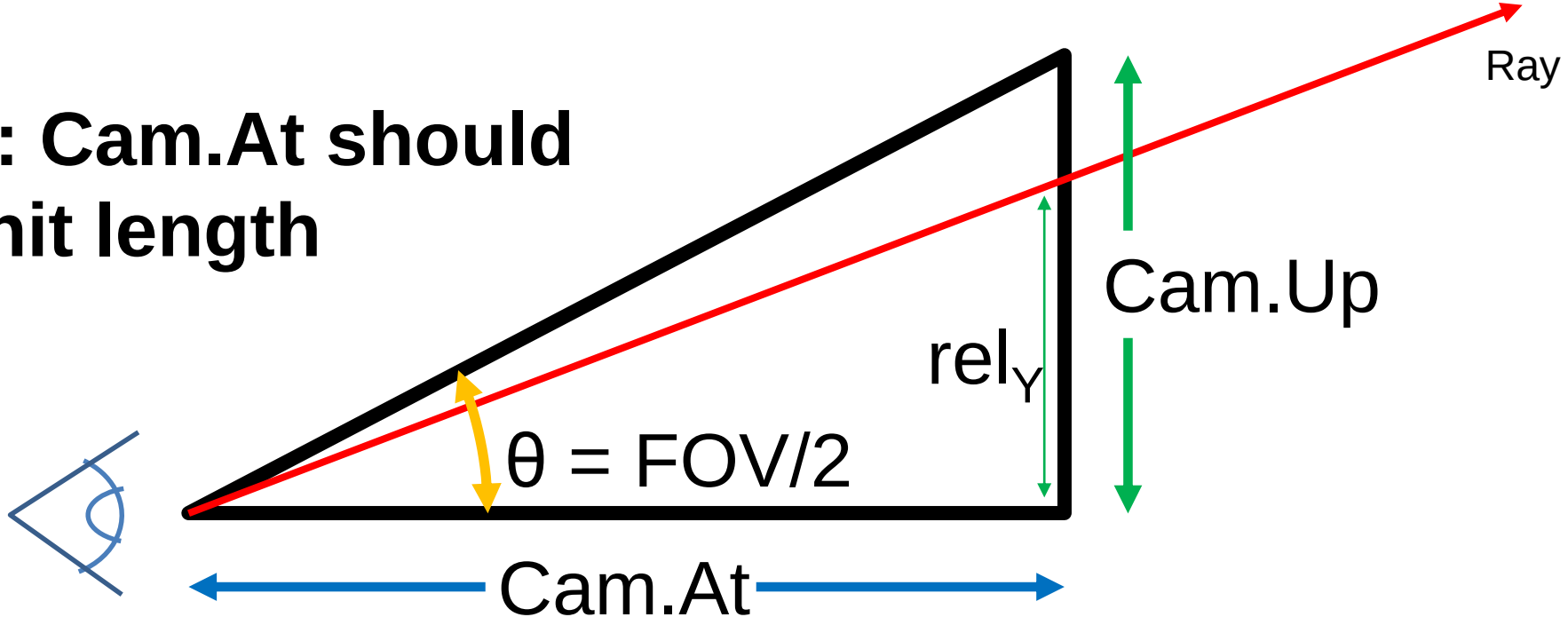
For N pixels wide:

$rel_X(i) = lerp(\frac{i}{N-1}, min_X, max_X)$

For N pixels high:

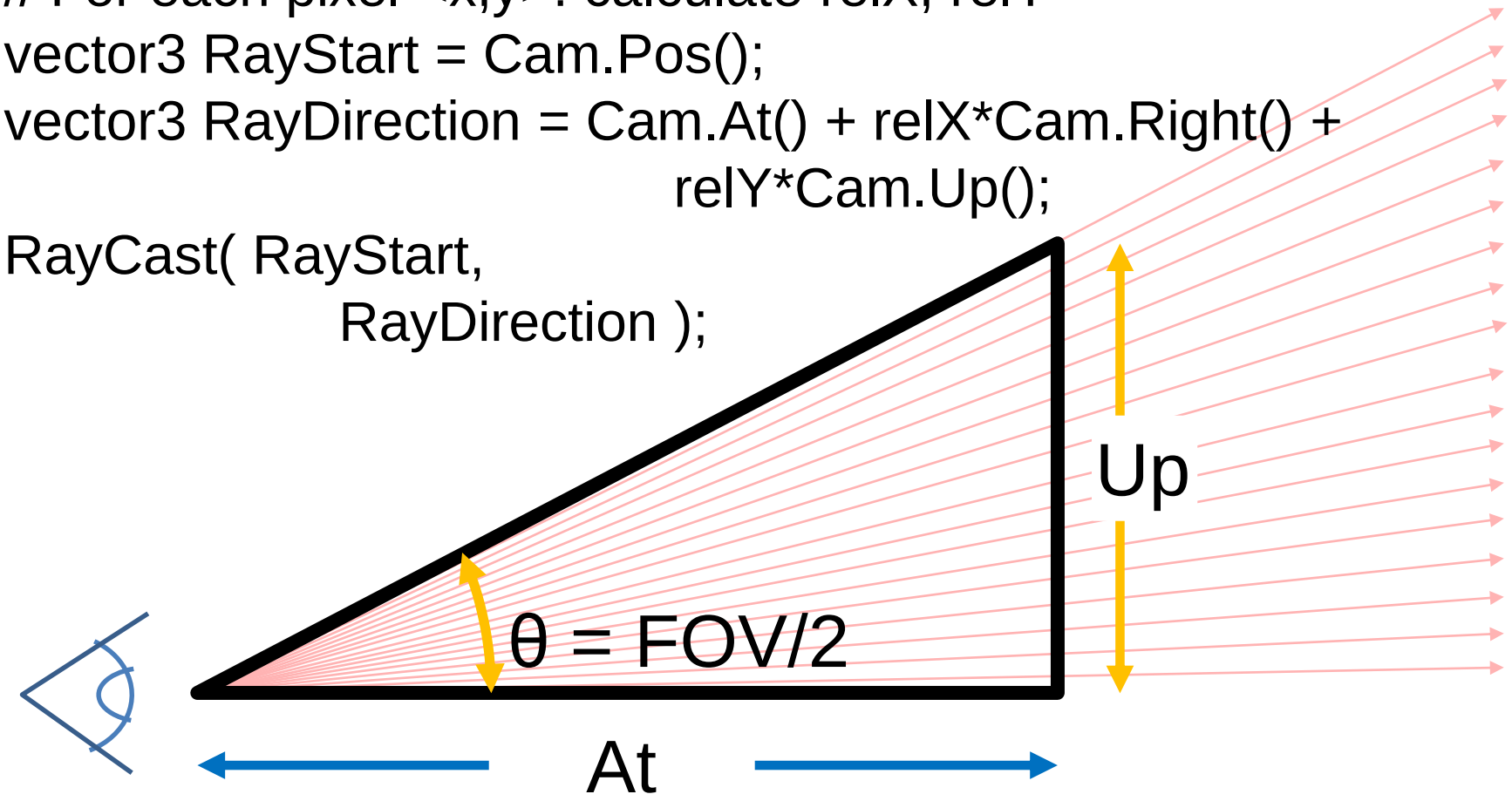
$rel_Y(i) = lerp(\frac{i}{N-1}, min_Y, max_Y)$

Note: Cam.At should be unit length



Ray Casting

```
for( int y = 0; y < height; y++ )
{
    for( int x = 0; x < width; x++ )
    {
        // For each pixel <x,y>: calculate relX, relY
        vector3 RayStart = Cam.Pos();
        vector3 RayDirection = Cam.At() + relX*Cam.Right() +
                                          relY*Cam.Up();
        RayCast( RayStart,
                 RayDirection );
    }
}
```



Ray Tracing

```
foreach(pixel)
{
        vec3 RayVec = GenerateRay( Camera, pixel );
        pixel.colour = Trace( Camera.Pos, RayVec );
}

// Rough pseudo code for raytracing
colour Trace( vec3 RayStart, vec3 RayVec, Object ObjectList[] )
{
    Object *ReturnObject = NULL;
    float ClosestIntersectionParam = ClosestIntersection( RayStart, RayVec, ObjectList, &ReturnObject );
    if( EPSILON < ClosestIntersectionParam )
    {
        colour returnColour = colour( 0, 0, 0 );   // Black
        vec3 IntersectionPos = RayStart + ClosestIntersectionParam * RayVec;
        vec3 InterNormal = ReturnObject.Normal( IntersectionPos );
        foreach( Light in LightList[] )
            if( !IsInShadow( IntersectionPos, Light.Pos, ObjectList )
                returnColour  += Illumination( IntersectionPos, InterNormal, ReturnObject.Material, Light );
        vec3 ReflectVec = Reflect( RayVec, InterNormal );
        colour reflectColour = Trace( IntersectionPos, ReflectVec, ObjectList );
        returnColour = Blend( returnColour, reflectColour, ReturnObject.Material );
        return returnColour;
    }
    return GetBackgroundColour( RayStart, RayVec );
}
```