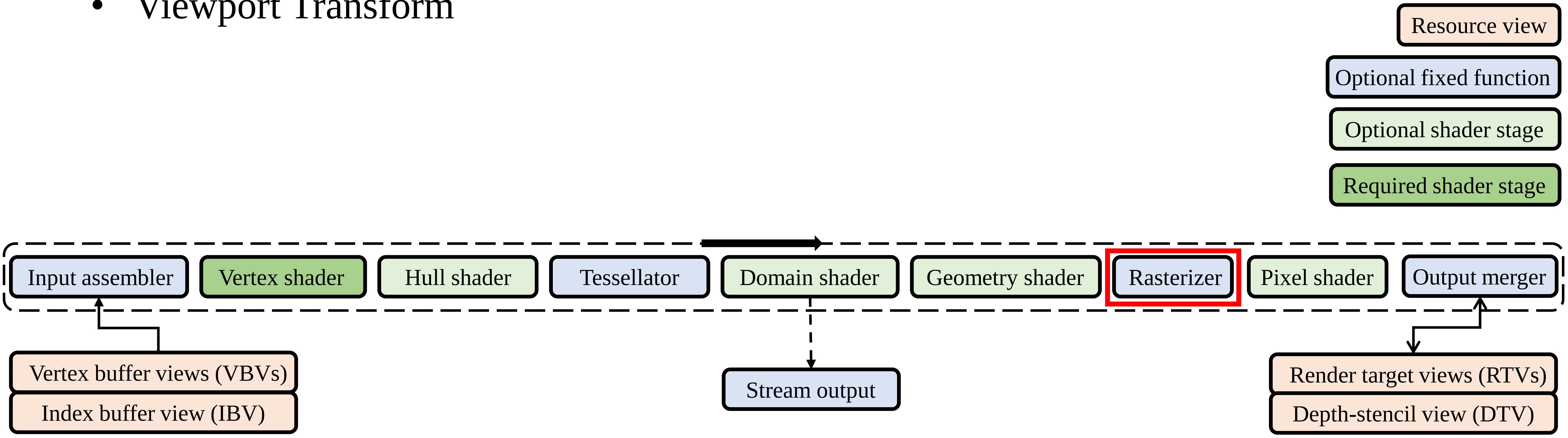
6. Rasterization

Prof. HyeongYeop Kang
siamiz@khu.ac.kr
YouTube: HKang IIIXR LAB
IIIXR LAB

Rasterizer Stage

Graphics pipeline

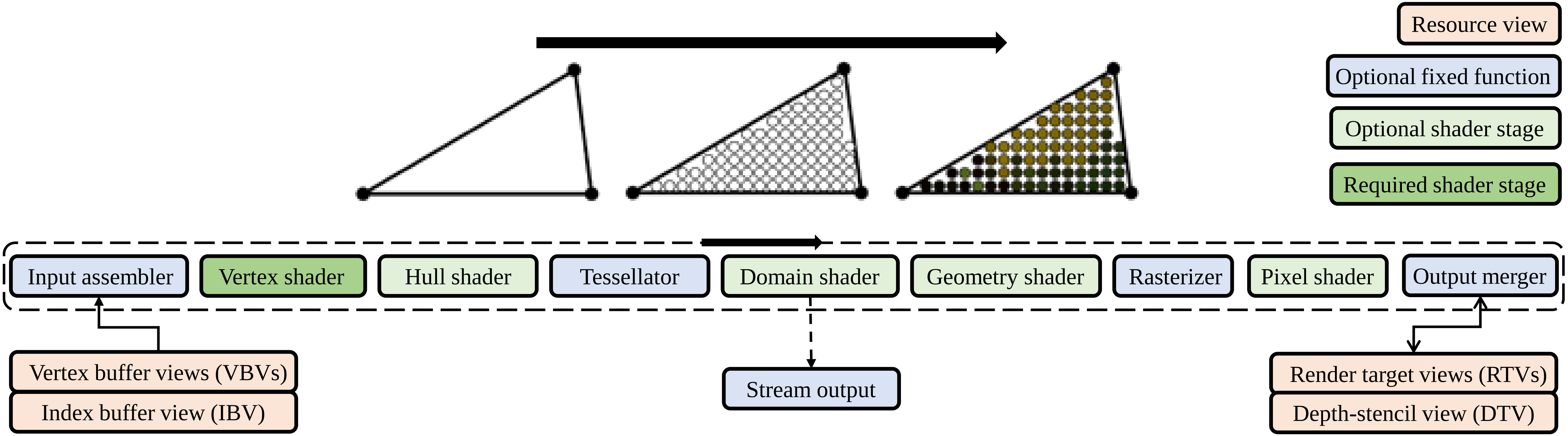
- Vertices that reach the Rasterizer stage undergo several hard-wired vertex post-processing steps.
  - Primitive Clipping
  - Perspective Division
  - Viewport Transform



Rasterizer Stage

Graphics pipeline

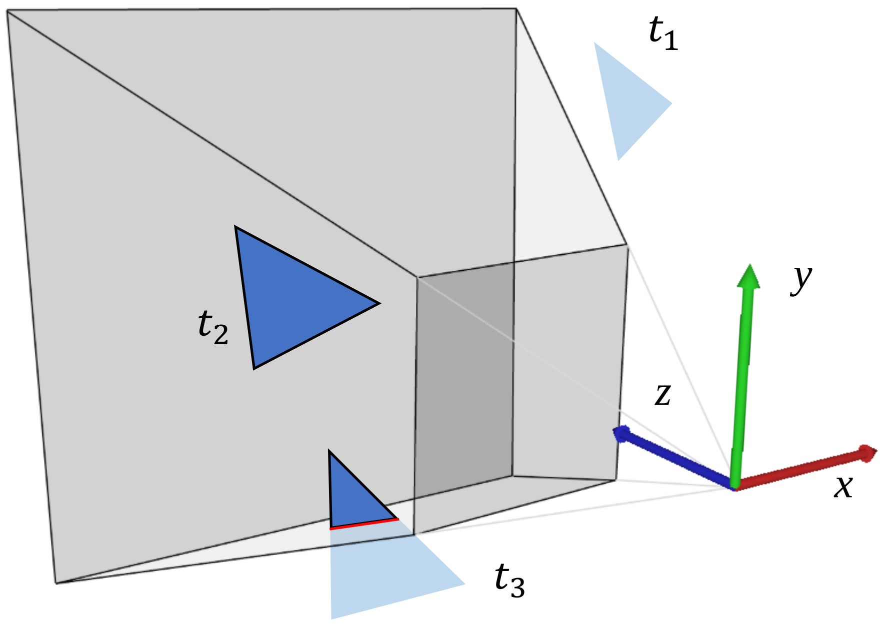
- Rasterization is the process whereby each individual *Primitive* is broken into two-dimensional image elements called *Pixels* or *Fragments*, based on the sample coverage of the primitive.
- In other words, this stage converts vector information into a raster image.



Primitive Clipping

Clipping is performed in the clip space, but the following figure presents its concept in the camera space, for the sake of intuitive understanding.

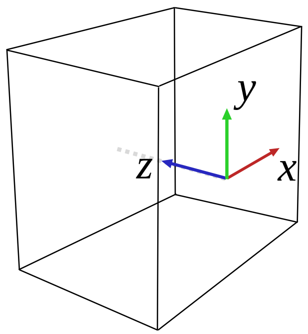
- Triangle t_1 is completely outside of the view frustum and is culled.
- Triangle t_2 is completely inside and is passed as is to the next step.
- Triangle t_3 intersects the view frustum and is thus clipped.



Primitive Clipping

The projection transform determines the clipping volume in projection space.

- The primitives outside the clipping volume are clipped.



2×2×1 sized
clipping volume

How primitives are clipped to the clipping volume depends on the basic primitive type.

- **Points**: If a point is in outside of the clipping volume, then it is discarded. If a points is bigger than one pixel, check its center of the point (SV_POSITION).
- **Lines**: If the line is partially outside of the volume, new vertex is generated and added to the end-point where the boundary of the clipping volume.
- **Triangles**: If a triangle is clipped to the viewing volume, appropriate triangles whose vertices are on the boundary of the clipping volume will be generated.

Perspective Division

Unlike affine transforms, the last column of M_{proj} is not $(0\ 0\ 0\ 1)^T$ but $(0\ 0\ 1\ 0)^T$.

- When M_{proj} is applied to $(x,y,z,1)$, the w-coordinate of the transformed vertex is $-z$.

$$M_{\text{proj}} = \begin{pmatrix} \frac{cot\frac{fovy}{2}}{aspect} & 0 & 0 & 0 \\ 0 & cot\frac{fovy}{2} & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & 1 \\ 0 & 0 & \frac{-fn}{f-n} & 0 \end{pmatrix}$$

$$(x \quad y \quad z \quad 1)\begin{pmatrix} m_{11} & 0 & 0 & 0 \\ 0 & m_{22} & 0 & 0 \\ 0 & 0 & m_{33} & 1 \\ 0 & 0 & m_{43} & 0 \end{pmatrix} = (m_{11}x \quad m_{22}y \quad m_{33}z + m_{43} \quad z)$$

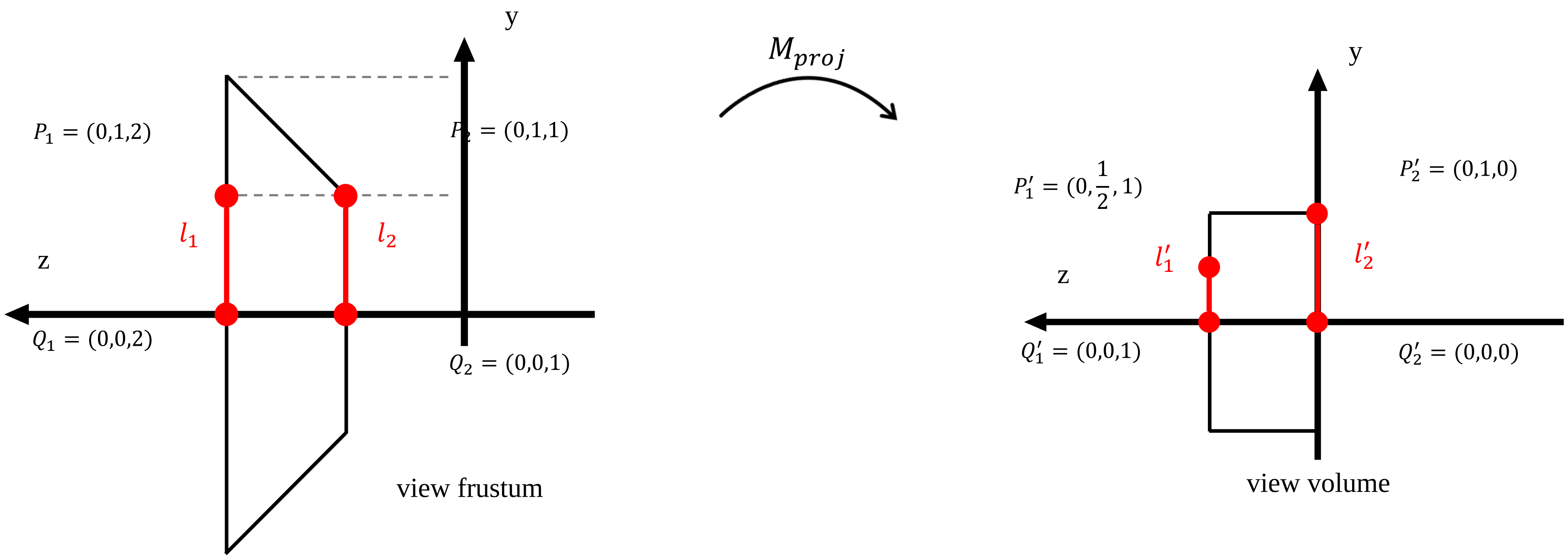
- In order to convert from the homogeneous (clip) space to the Cartesian space, each vertex should be divided by its w-coordinate (which equals $-z$).

$$(m_{11}x \quad m_{22}y \quad m_{33}z + m_{43} \quad z) = \left(\frac{m_{11}x}{z} \quad \frac{m_{22}y}{z} \quad m_{33} + \frac{m_{43}}{z} \quad 1\right)$$

Perspective Division

Note that z is a positive value representing the distance from the *xy*-plane of the camera space.

- Division by z makes distant objects smaller. It is perspective division.
- The result is said to be in NDC (normalized device coordinates).

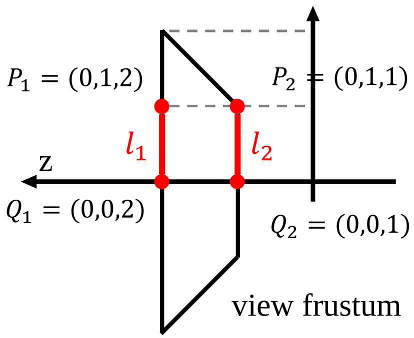


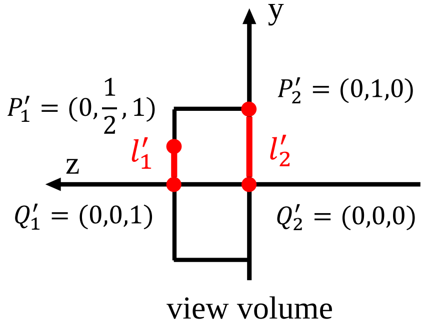
Perspective Division

Note that z is a positive value representing the distance from the *xy*-plane of the camera space.

$$M_{proj} = \begin{pmatrix} \frac{cot(\frac{fovy}{2})}{aspect} & 0 & 0 & 0 \\ 0 & cot(\frac{fovy}{2}) & 0 & 0 \\ 0 & 0 & \frac{f}{f-n} & 1 \\ 0 & 0 & -\frac{fn}{f-n} & 0 \end{pmatrix}$$

$$\boxed{\begin{aligned} fovy &= \frac{\pi}{2} \\ aspect &= 1 \\ n &= 1 \\ f &= 2 \end{aligned}} \Rightarrow M_{proj} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & -2 & 0 \end{pmatrix}$$





$$P_1 M_{proj} = (0,1,2,1)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & -2 & 0 \end{pmatrix} = (0,1,2,2) \rightarrow \left(0,\frac{1}{2},1,1\right) = P'_1$$

$$Q_1 M_{proj} = (0,0,2,1)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & -2 & 0 \end{pmatrix} = (0,0,2,2) \rightarrow (0,0,1,1) = Q'_1$$

$$P_2 M_{proj} = (0,1,1,1)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & -2 & 0 \end{pmatrix} = (0,1,0,1) = P'_2$$

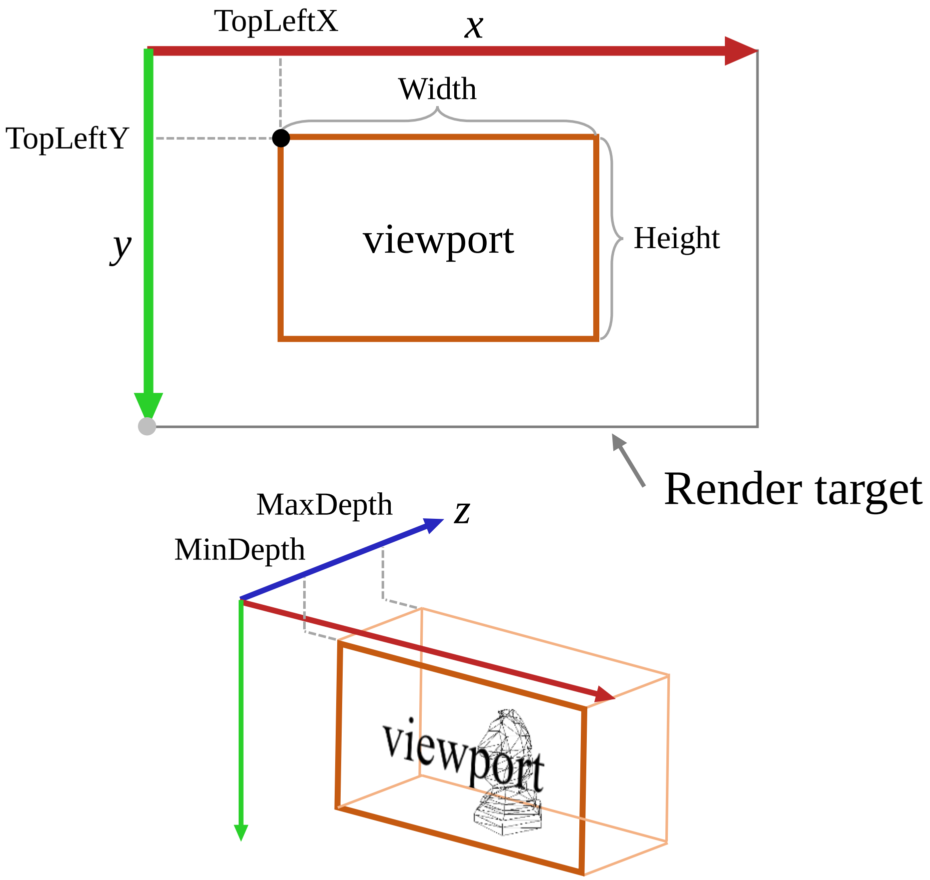
$$Q_2 M_{proj} = (0,0,1,1)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & -2 & 0 \end{pmatrix} = (0,0,0,1) = Q'_2$$

Viewport

In DirectX, an array of viewports are used to perform a rasterization stage.

- A viewport is a two-dimensional rectangle area onto which a 3D scene is projected.
- In Microsoft’s document, a viewport structure is described as follows:

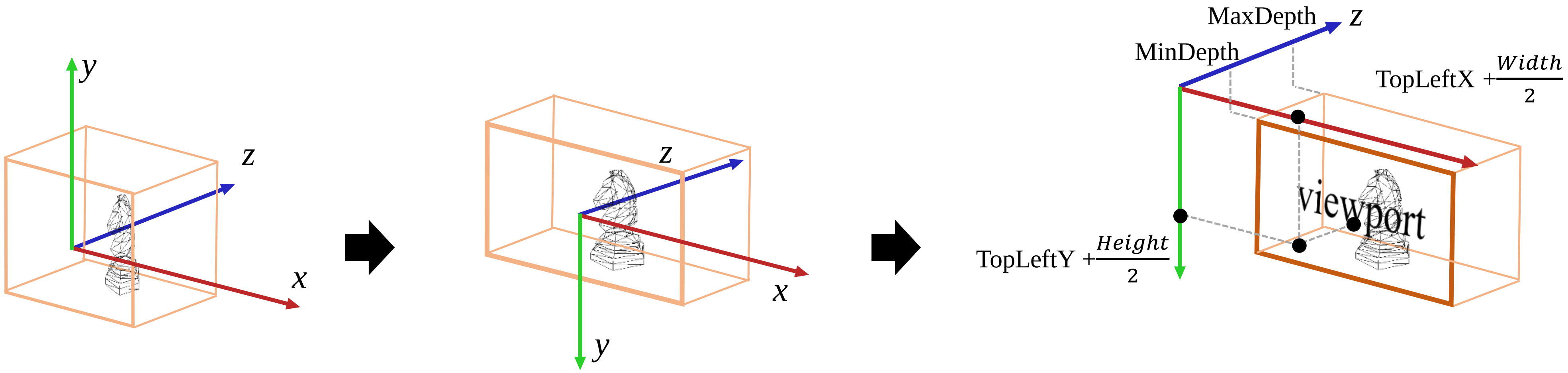
```
typedef struct D3D12_VIEWPORT {
    FLOAT TopLeftX;
    FLOAT TopLeftY;
    FLOAT Width;
    FLOAT Height;
    FLOAT MinDepth;
    FLOAT MaxDepth;
} D3D12_VIEWPORT;
```



Viewport Transform

The ‘viewport transform’ transforms vertex positions from NDC space to window space.

- It is a combination of a scaling and a translation.
- Note that the size of NDC space in DirectX is $2\times 2\times 1$ whereas that in OpenGL is $2\times 2\times 2$.



$$\begin{pmatrix} \frac{Width}{2} & 0 & 0 & 0 \\ 0 & -\frac{Height}{2} & 0 & 0 \\ 0 & 0 & \frac{MaxDepth - MinDepth}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

scaling

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \mathrm{TopLeftX} + \frac{Width}{2} & \mathrm{TopLeftY} + \frac{Height}{2} & \mathrm{MinDepth} & 1 \end{pmatrix}$$

translation

Viewport Transform

In most applications, the viewport takes up the entire window (monitor screen).

- TopLeftX = 0, TopLeftY = 0.
- MinDepth = 0, MaxDepth = 1.

$$
\begin{pmatrix}
\frac{Width}{2} & 0 & 0 & 0 \\
0 & -\frac{Height}{2} & 0 & 0 \\
0 & 0 & \frac{MaxDepth - MinDepth}{2} & 0 \\
0 & 0 & 0 & 1
\end{pmatrix}
$$

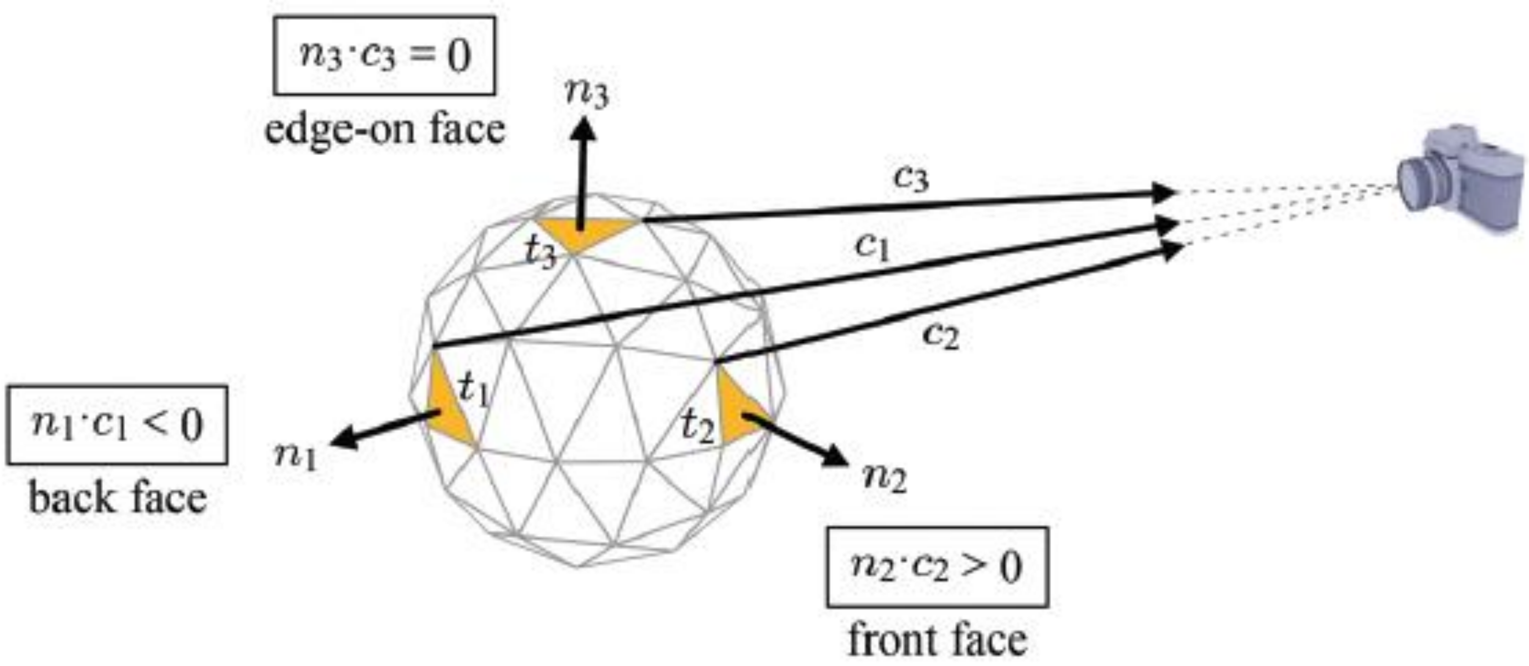
$$
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\mathrm{TopLeftX} + \frac{Width}{2} & \mathrm{TopLeftY} + \frac{Height}{2} & \mathrm{MinDepth} & 1
\end{pmatrix}
$$

$$
\rightarrow
\begin{pmatrix}
\frac{Width}{2} & 0 & 0 & 0 \\
0 & -\frac{Height}{2} & 0 & 0 \\
0 & 0 & \frac{MaxDepth - MinDepth}{2} & 0 \\
\mathrm{TopLeftX} + \frac{Width}{2} & \mathrm{TopLeftY} + \frac{Height}{2} & \mathrm{MinDepth} & 1
\end{pmatrix}
\rightarrow
\begin{pmatrix}
\frac{Width}{2} & 0 & 0 & 0 \\
0 & -\frac{Height}{2} & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 \\
\frac{Width}{2} & \frac{Height}{2} & 0 & 1
\end{pmatrix}
$$

Face Culling

Primitives have a particular facing that is defined by the order of vertices.

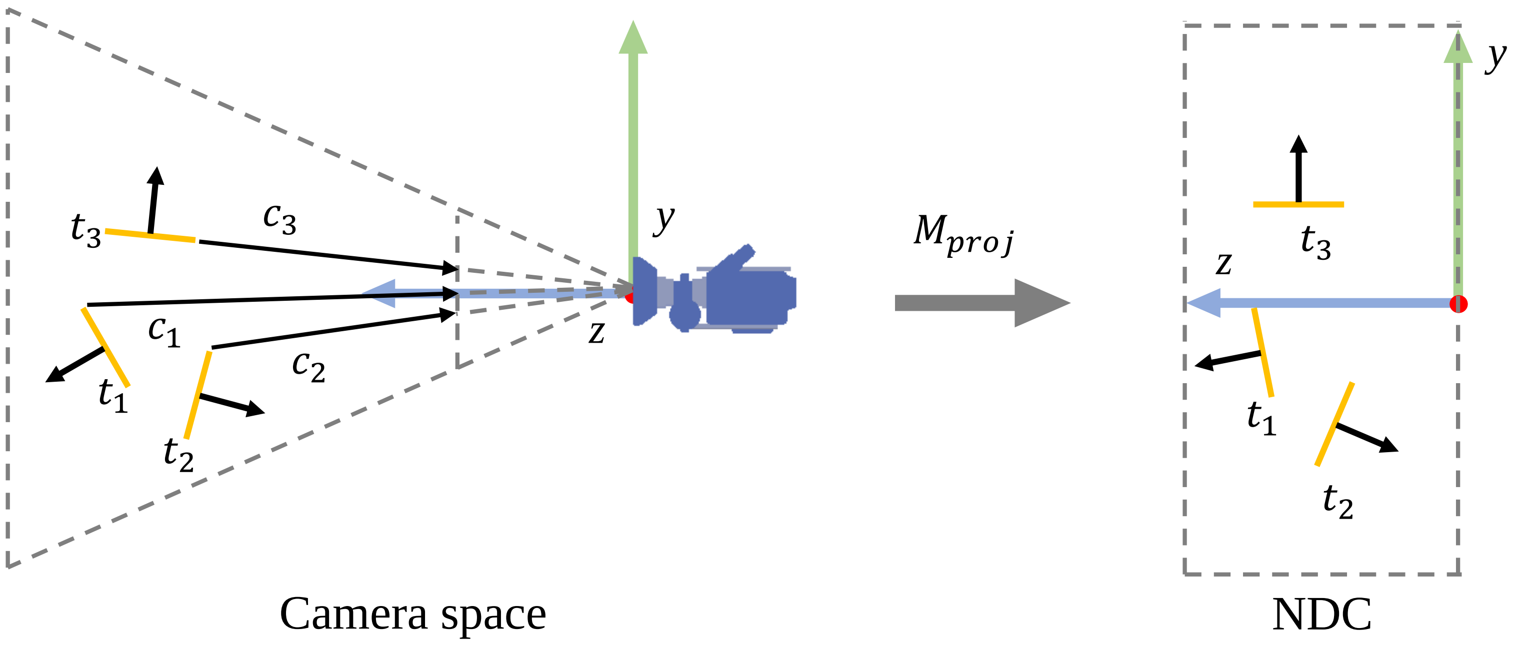
- The primitives facing away from the viewpoint of the camera are called the back faces.
- The primitives facing the camera are called the front faces.
- Face culling allows non visible primitives (back face) to be removed before expensive Rasterization and Fragment Shader operations.



Face Culling

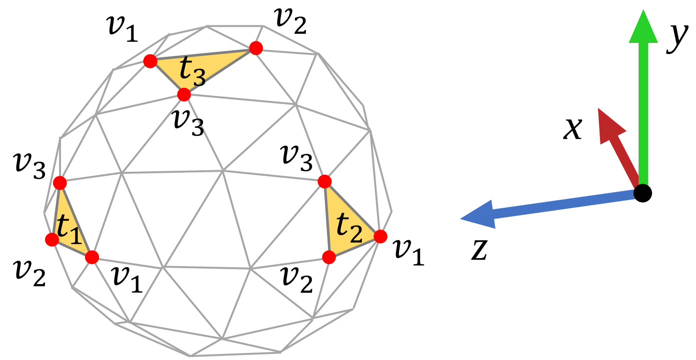
Projection line

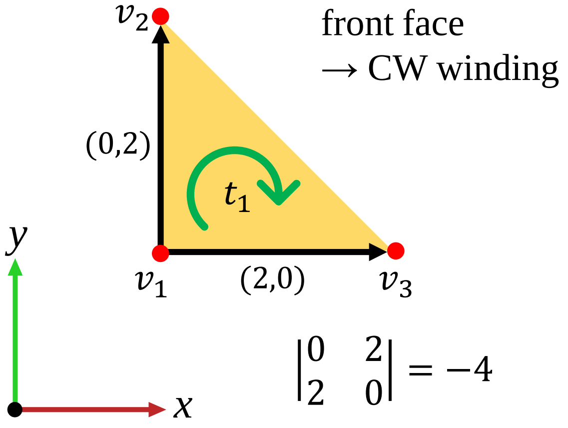
- In the below figure, c_1, c_2, and c_3 presents projection line.
- The projection transform defines a universal projection line parallel to the z-axis.

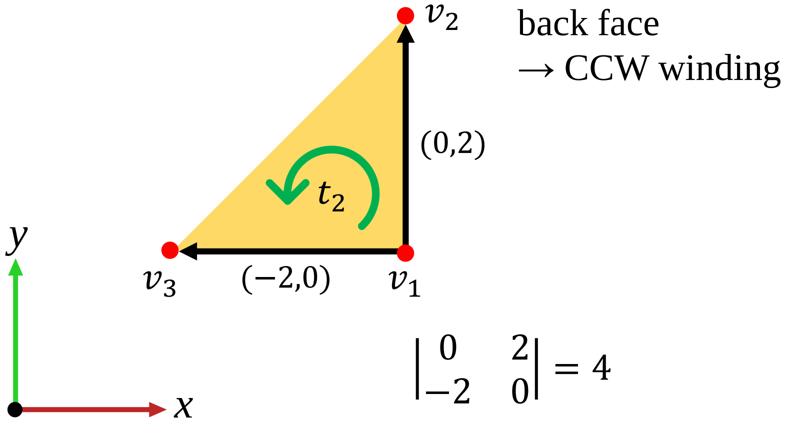


Face Culling

Let us conceptually project the triangles along the universal projection line onto the *xy*-plane. A 2D triangle with CW winding order is a front-face, and a 2D triangle with CCW winding order is a back-face.







Compute the following determinant, where the first row represents the 2D vector connecting v_1 and v_2, and the second row represents the 2D vector connecting v_1 and v_3.

- If negative, CW and so front-face.
- If it is positive, CCW and so back-face.
- If 0, edge-on face.

$$\begin{vmatrix} (x_2 - x_1) & (y_2 - y_1) \\ (x_3 - x_1) & (y_3 - y_1) \end{vmatrix}$$

Face Culling

The back faces are not always culled.

- Consider rendering a hollow translucent sphere. For the back faces to show through the front faces, no face should be culled.
- On the other hand, consider culling only the front faces of the sphere. Then the cross-section view of the sphere will be obtained.

Backface culling in DirectX

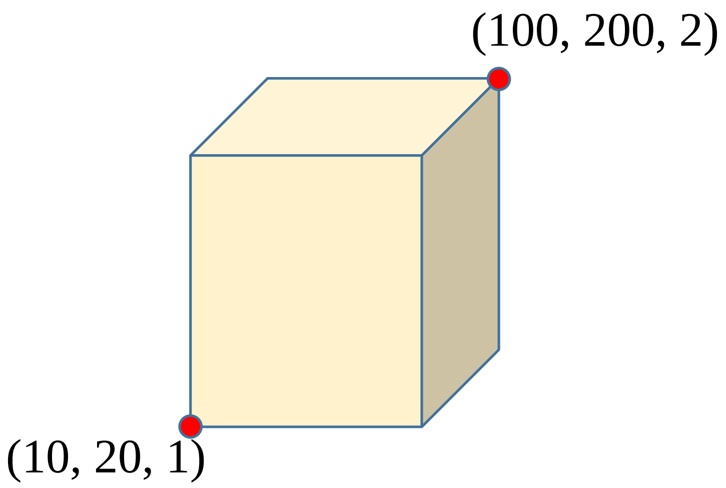
- By default, face culling is enabled.
- CULL_MODE_FRONT does not draw triangles that are front-facing.

```
typedef struct D3D12_RASTERIZER_DESC {
  ...
  D3D12_CULL_MODE          CullMode;
  ...
} D3D12_RASTERIZER_DESC;
```

```
typedef enum D3D12_CULL_MODE {
  D3D12_CULL_MODE_NONE = 1,
  D3D12_CULL_MODE_FRONT = 2,
  D3D12_CULL_MODE_BACK = 3
} ;
```

Practice

A viewport's corners are located at (10, 20, 1) and (100, 200, 2). The viewport transform is defined as a scaling followed by a translation.



(1) Write the scaling matrix.

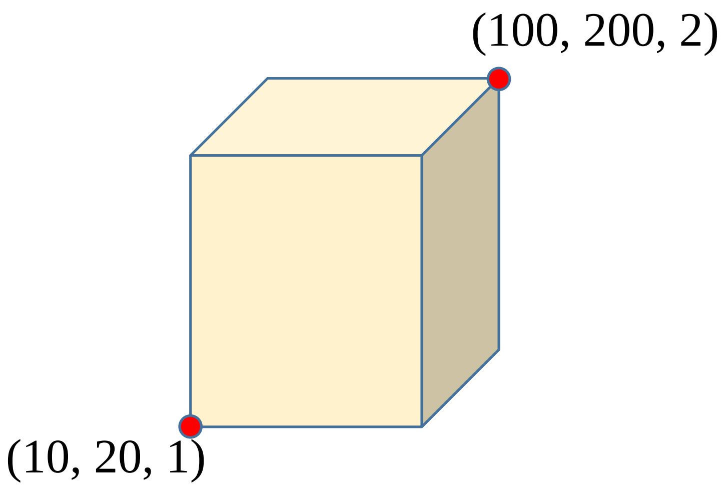
(2) Write the translation matrix.

Solution

A viewport's corners are located at (10, 20, 1) and (100, 200, 2). The viewport transform is defined as a scaling followed by a translation.

(1) Write the scaling matrix.

$$\begin{pmatrix} 45 & 0 & 0 & 0 \\ 0 & 90 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



(2) Write the translation matrix.

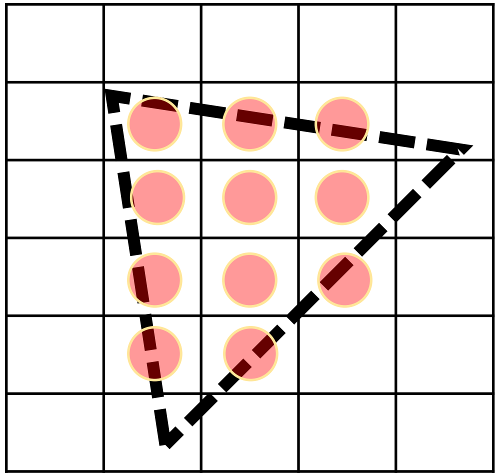
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 55 & 110 & 1.5 & 1 \end{pmatrix}$$

Rasterization

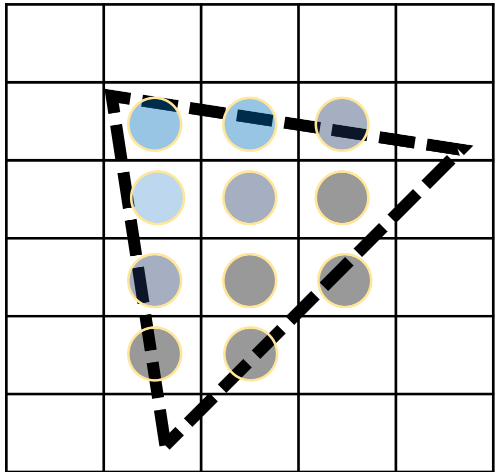
The main role of rasterization stage is to convert vector information into a raster image.

Rasterizing a primitive consists of two parts:

- Determining which square of an integer grid in window coordinates are occupied by the primitive.



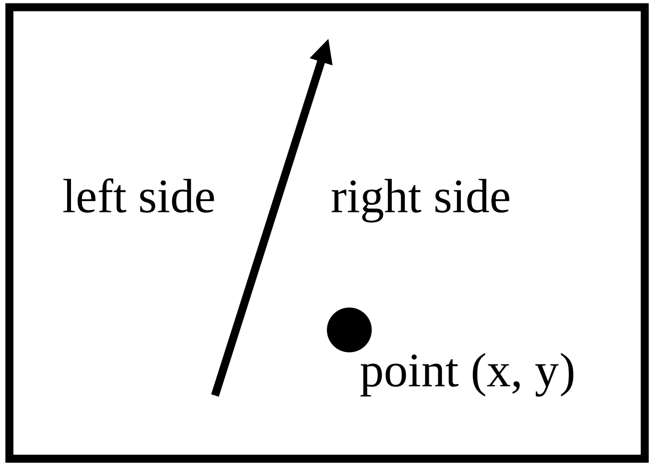
- Assigning a color and a depth value to each square of the grid.

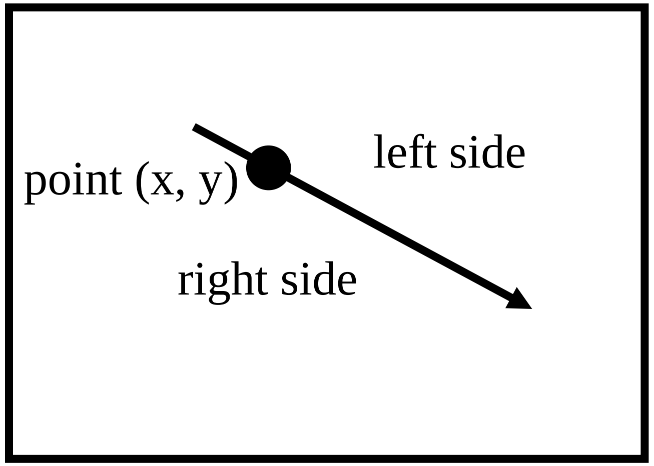


Edge Equation

Edge equation?

- Assume that there is a line splitting the 2D plane into two parts.
- Edge equation tests on which side of this line a given point is.
  - When the point is on the left side: negative number.
  - When the point is on the right side: positive number.
  - When the point is on the line: zero number.

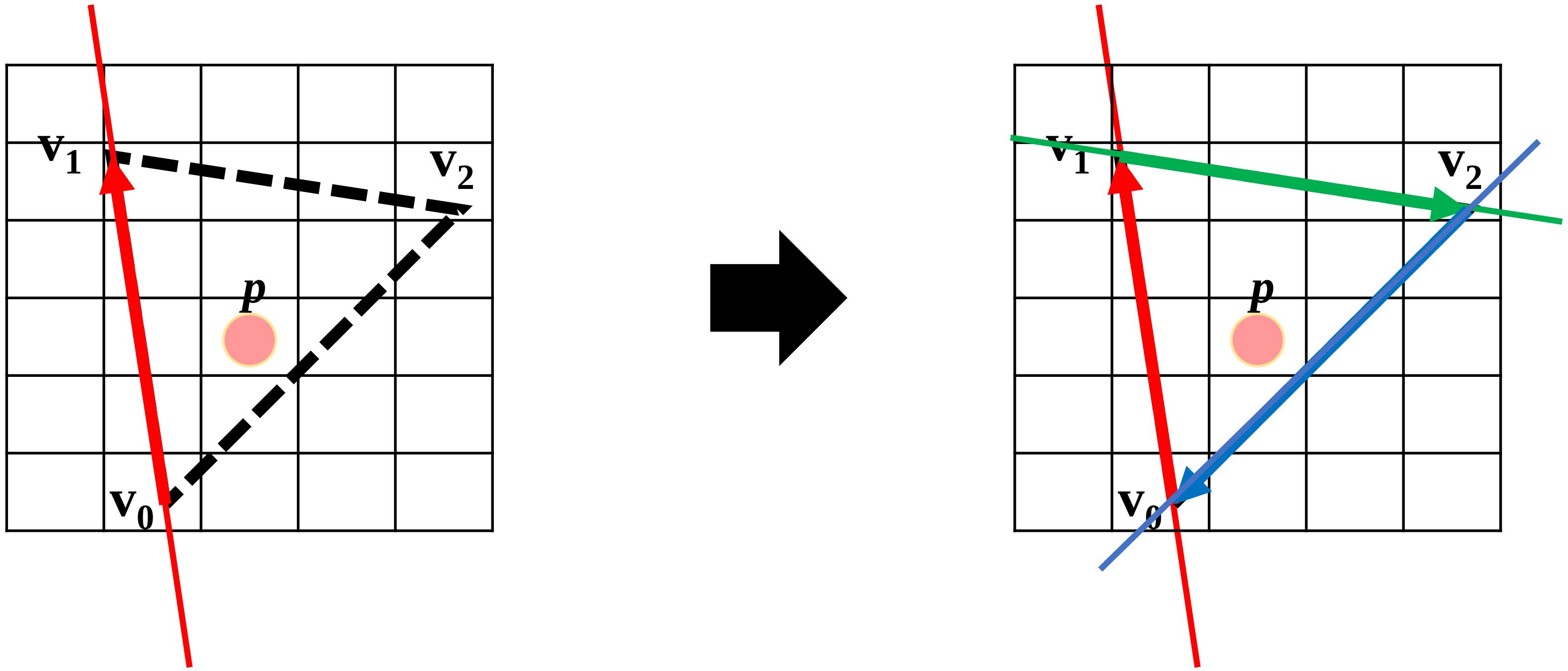




Edge Equation

Edge equation tests whether a pixel overlaps a triangle.

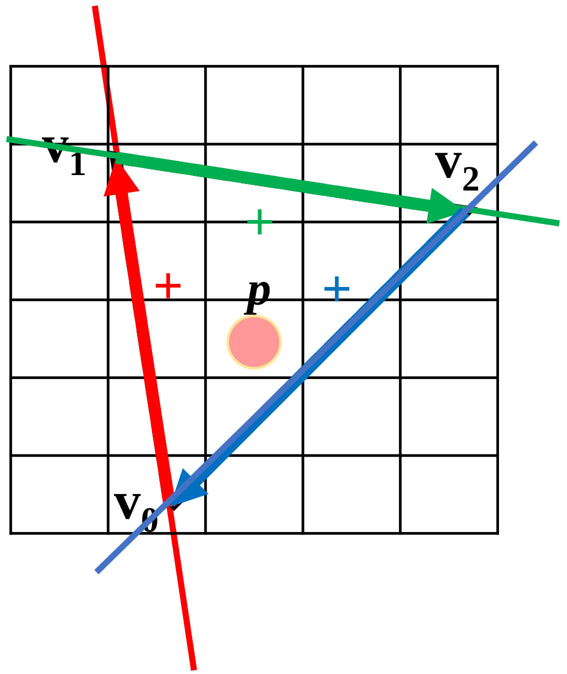
- We can see a edge of triangle as a line splitting 2D plane.
- Let's apply an edge equation to the first edge of the triangle (defined by the vertices v_0 and v_1.)
- In our example, pixel *p* is on the right side of the line.
- Let's apply the same equations to the other two edges.



Edge Equation

Edge equation tests whether a pixel overlaps a triangle.

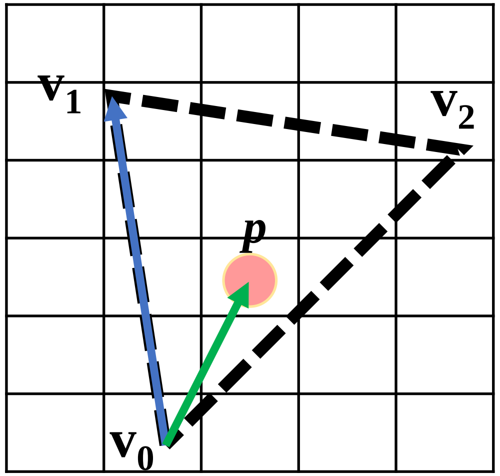
- We found that the edge function returns positive numbers with respect to all three edges of triangle.
- If a point lies within a triangle, point is on the right sides of all three edges of the triangle.
- This means that the edge function returns a positive number for all three edges.



Edge Equation

The definition of edge equation (E)

- $E(P, v_0, v_1) = (P.x - v_0.x)\cdot(v_1.y - v_0.y) - (P.y - v_0.y)\cdot(v_1.x - v_0.x)$.



This equation is equivalent to the determinant computation of 2×2 matrix defined by the components of the 2D vectors (P - v_0) and (v_1 - v_0).

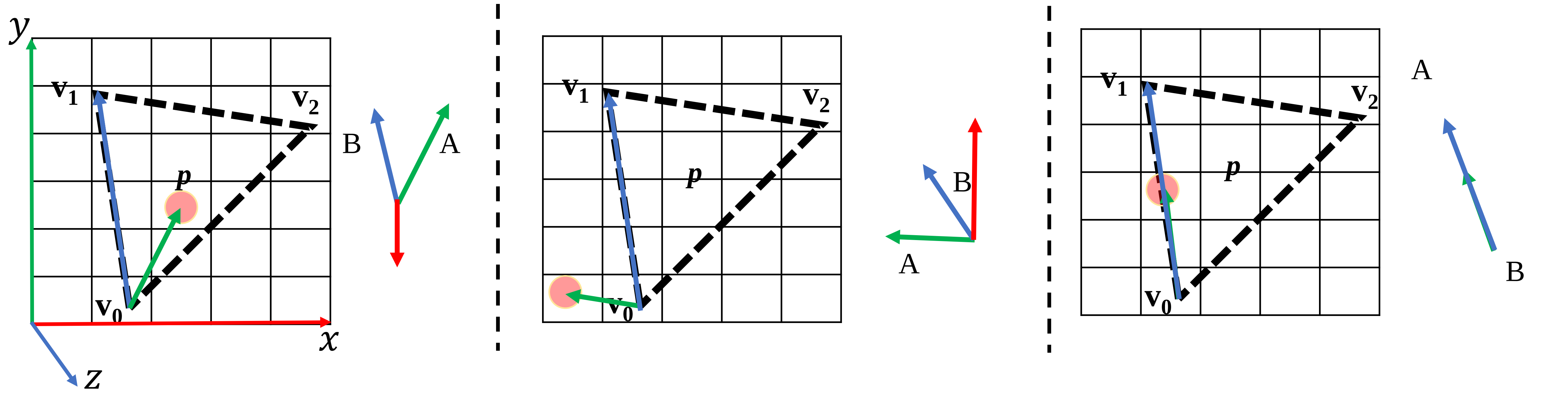
- $\begin{Vmatrix} P.x - v_0.x & P.y - v_0.y \\ v_1.x - v_0.x & v_1.y - v_0.y \end{Vmatrix}$

Edge Equation

This equation is also equivalent in mathematics to the magnitude of the cross products between the vectors A = (P - v_0) and B = (v_1 - v_0).

- Let's say that all 3D points (p, v_0, v_1) has 0 of z-value.
- Then, cross product can be computed by followings:

$$\begin{vmatrix} x & y & z \\ A.x & A.y & 0 \\ B.x & B.y & 0 \end{vmatrix} = (0, 0, A.x \cdot B.y - A.y \cdot B.x), \text{ in LHS}$$



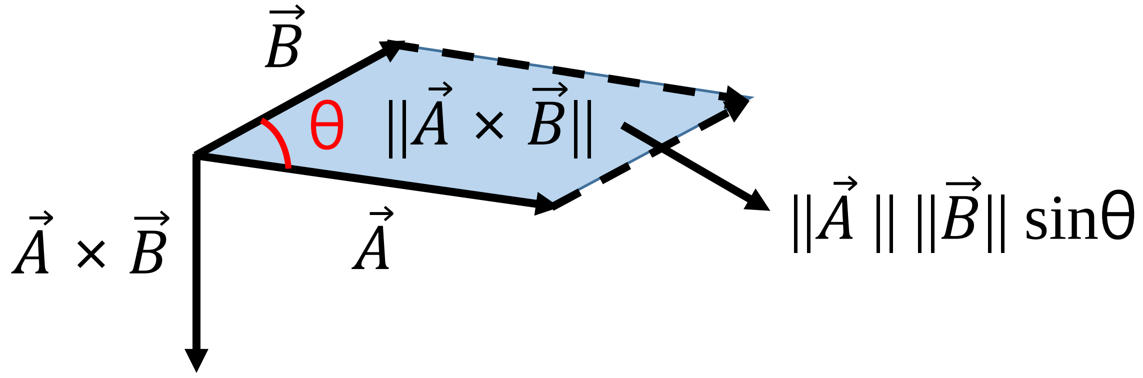
Edge Equation

This equation is also equivalent in mathematics to the magnitude of the cross products between the vectors $A = (P - v_0)$ and $B = (v_1 - v_0)$.

- From the (0, 0, A.x · B.y – A.y · B.x), we can find out the magnitude of cross product of two vectors is A.x · B.y – A.y · B.x.
- As the magnitude of cross product of two vectors can be interpreted as the area of the parallelogram as shown on the right figure, we can obtain

  $A.x \cdot B.y - A.y \cdot B.x = \|\vec{A}\|\,\|\vec{B}\| \sin\theta$

- Therefore, the edge equation returns positive value if $0 < \theta < 180$ where as it returns negative value if $180 < \theta < 360$.

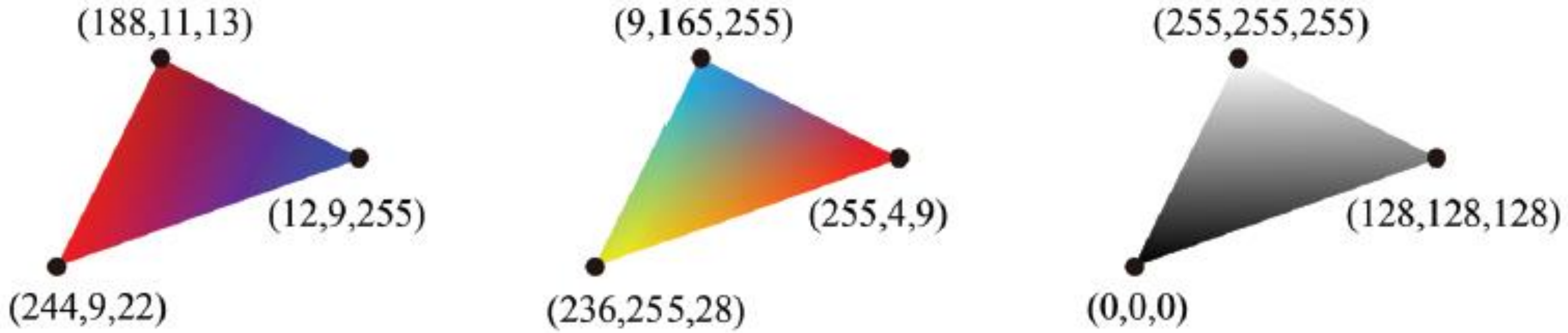


Assigning Attributes for Fragments

Attributes (color, depth, normal, etc..) of point (p) overlapping the triangle($v_0v_1v_2$) can be obtained by interpolating the attributes of triangle's vertices.

- Let's say that color of v_0, v_1, and v_2 are C_{v0}, C_{v1}, C_{v2}.

- Then, we can compute the color of p with barycentric coordinate:
  - $p = \lambda_0 * C_{v0} + \lambda_1 * C_{v1} + \lambda_2 * C_{v2}$,
  - $\lambda_0 + \lambda_1 + \lambda_2 = 1$,
  - for $p \in \triangle v_0, v_1, v_2$.

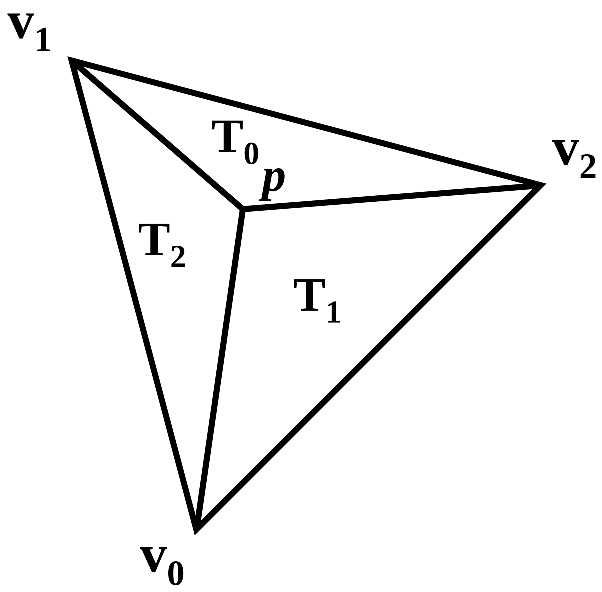


- Position, depth, normal can also be computed with this method.

Barycentric coordinate

The weights of barycentric coordinates (λ_0, λ_1, λ_2) are proportional to the areas of the triangles (T_0, T_1, T_2).

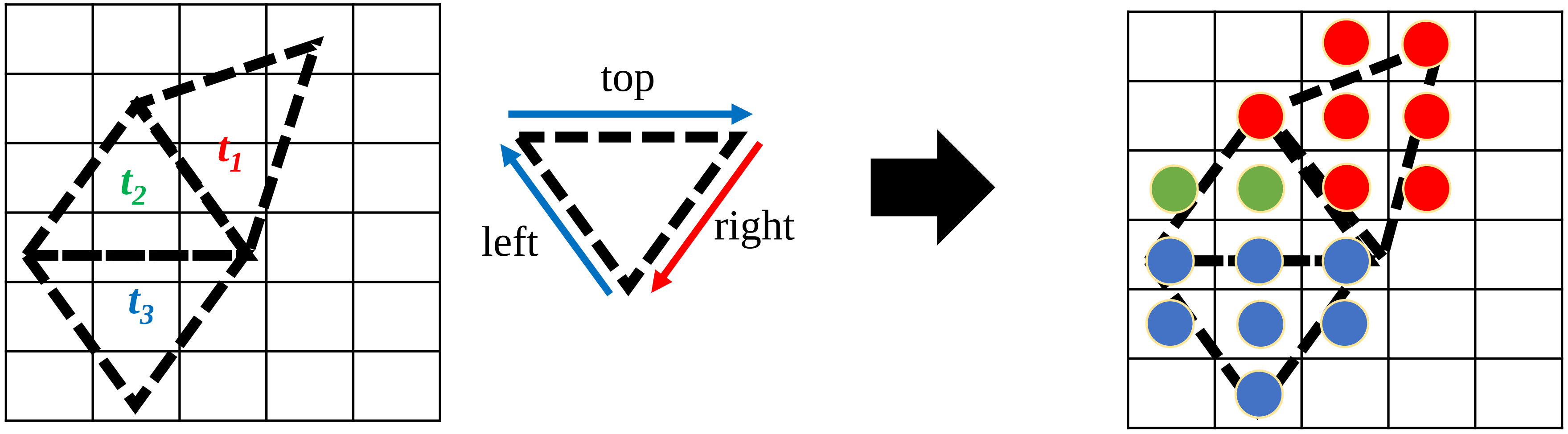
- T_0 can be computed by $0.5 * E(p, v_1, v_2)$ since edge equation returns the area of parallelogram.
- T_1 can be computed by $0.5 * E(p, v_2, v_0)$ since edge equation returns the area of parallelogram.
- T_2 can be computed by $0.5 * E(p, v_0, v_1)$ since edge equation returns the area of parallelogram.
- The area of triangle $v_0v_1v_2$ can be computed by $0.5 * E(v_2, v_0, v_1)$.
- Therefore, $\lambda_0 = \dfrac{E(p, v_1, v_2)}{E(v_2, v_0, v_1)}$, $\lambda_1 = \dfrac{E(p, v_2, v_0)}{E(v_2, v_0, v_1)}$, and $\lambda_2 = \dfrac{E(p, v_0, v_1)}{E(v_2, v_0, v_1)}$



Rasterization Rule (Top-left Rule)

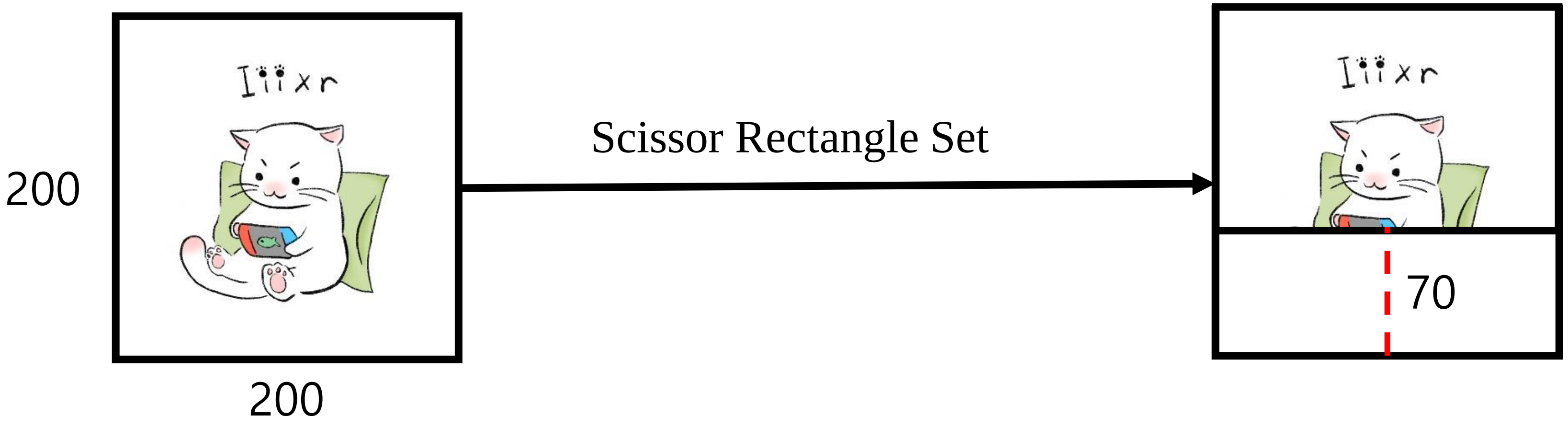
When a pixel is on the edge shared by two triangles, we have to decide to which triangle it belongs. Otherwise, it would be processed twice.

- Intuitively, a triangle may have left, right, top or bottom edges.
- In the below figure, t_1 has two left edges and one right edge, t_2 has one left edge, one right edge, and one bottom edge, and t_3 has one left edge, one right edge, and one top edge.
- Direct3D adopts the top-left rule, which declares that a pixel belongs to a triangle if it lies on the top or left edge of the triangle.



Scissor Test

The scissor test discards pixels (or fragments) outside a specified rectangle of the screen.



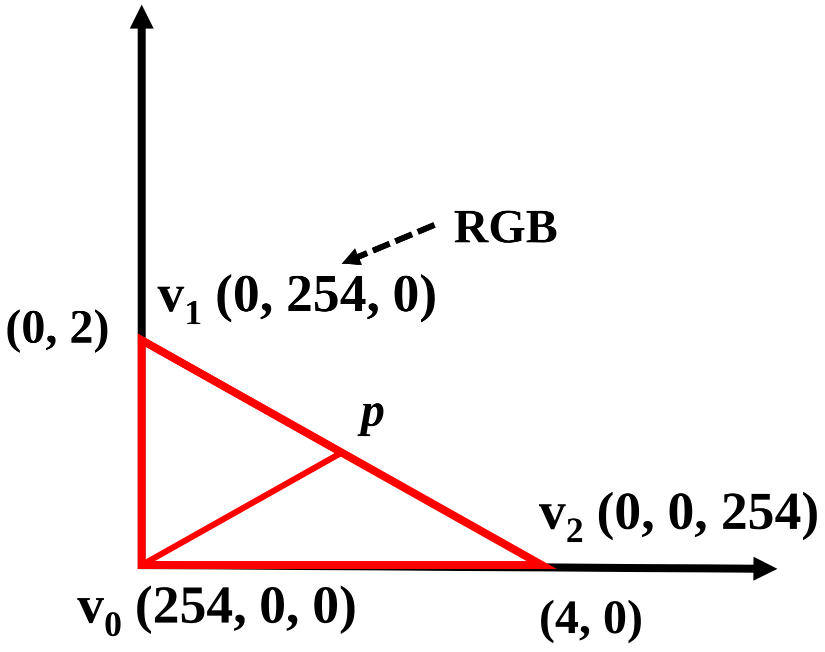
Practice

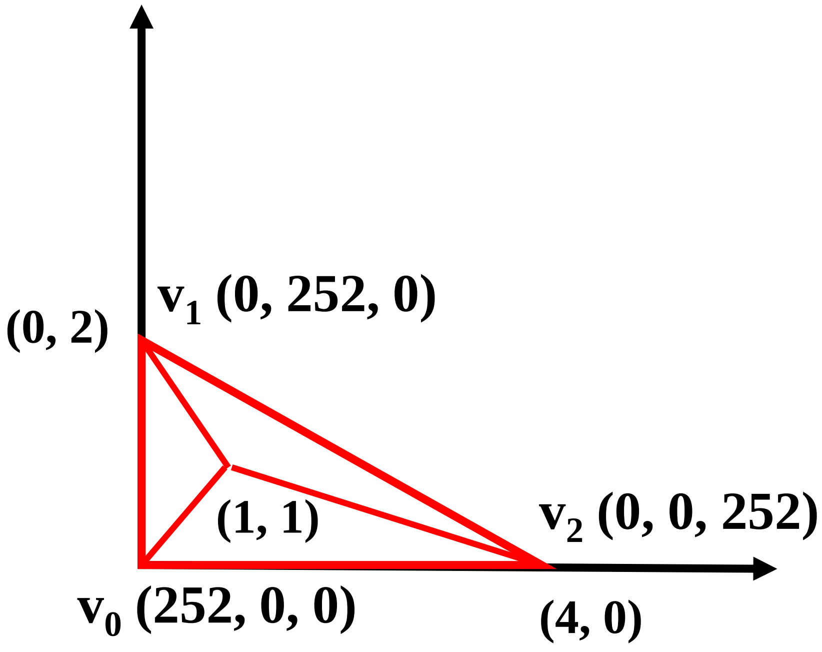
Q1. Assume that *p* lies exactly on the center of the edge defined by v_0 and v_1.

- Determine the position and color of the point *p*.

Q2. Assume that p lies (1, 1) within the triangle defined by v_0, v_1, and v_2.

- Determine the color of the point *p*.





Practice - Solution

Q1. Assume that p lies exactly on the center of the edge defined by v_0 and v_1.

- Determine the position and color of the point p.

Q2. Assume that p lies (1, 1) within the triangle defined by v_0, v_1, and v_2.

- Determine the color of the point p.