

Problema 4.1.

Supongamos que queremos visualizar una secuencia de frames, en los cuales la cámara va cambiando. Para ellos queremos escribir el código de una función que fija la matriz de vista en el cauce. La función acepta como parámetro un valor real t , que es el tiempo en segundos transcurrido desde el inicio de la animación. Suponemos que la animación dura s segundos en total.

En ese tiempo el observador de cámara se desplaza con un movimiento uniforme desde un punto de coordenadas de mundo \mathbf{o}_0 (para $t = 0$) hasta un punto destino \mathbf{o}_1 (para $t = 1$). Además el punto de atención de la cámara también se desplaza desde \mathbf{a}_0 hasta \mathbf{a}_1 . Durante toda la animación, el vector VUP es $(0, 1, 0)$.

Escribe el pseudo-código de la citada función.

// Variables estáticas o globales

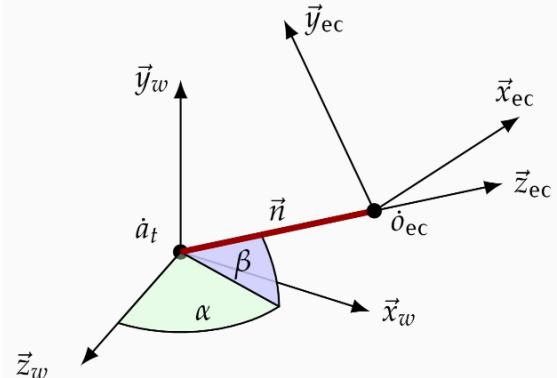
`float s = ...;`

`Tupla3f o0 = ...;`

`Tupla3f o1 = ...;`

`Tupla3f a0 = ...;`

`Tupla3f a1 = ...;`



nord desplazar (float t) {

$$\mathbf{o}_{ec} = (1 - \frac{t}{s})\mathbf{o}_0 + \frac{t}{s}\mathbf{o}_1$$

$$\mathbf{a}_t = (1 - \frac{t}{s})\mathbf{a}_0 + \frac{t}{s}\mathbf{a}_1$$

$$\mathbf{n} = \mathbf{o}_{ec} - \mathbf{a}_t$$

Cálculo del marco de vista.

A partir de esos parámetros se obtiene se calculan los versores del marco de vista:

$$\hat{\mathbf{z}}_{ec} = \frac{\vec{u} \times \vec{n}}{\|\vec{u} \times \vec{n}\|} \quad (\text{eje Z paralelo a VPN, normalizado})$$

$$\hat{\mathbf{x}}_{ec} = \frac{\vec{u} \times \hat{\mathbf{z}}_{ec}}{\|\vec{u} \times \hat{\mathbf{z}}_{ec}\|} \quad (\text{eje X perpendicular a VPN y VUP, normalizado})$$

$$\hat{\mathbf{y}}_{ec} = \hat{\mathbf{z}}_{ec} \times \hat{\mathbf{x}}_{ec} \quad (\text{eje Y perpendicular a los otros dos})$$

Para que este cálculo pueda hacerse, los vectores \vec{u} y \vec{n} no pueden ser nulos ni paralelos, de forma que siempre $\|\vec{u} \times \vec{n}\| > 0$.

actualizar
ejes marco
de vista } } $\left\{ \begin{array}{l} \mathbf{u} = (0, 1, 0) \\ \mathbf{z}_{ec} = \frac{\mathbf{n}}{\|\mathbf{n}\|} \quad \mathbf{x}_{ec} = \frac{\mathbf{u} \times \mathbf{n}}{\|\mathbf{u} \times \mathbf{n}\|} \quad \mathbf{y}_{ec} = \mathbf{z}_{ec} \times \mathbf{x}_{ec} \end{array} \right.$

matrices actualizadas = false;

}

Cálculo del marco de vista.

El marco de referencia de vista \mathcal{V} , se define a partir de los siguientes parámetros

\mathbf{o}_{ec} = es el punto del espacio foco de la proyección, donde estaría situado el observador ficticio que contempla la escena (*projection reference point, PRP*)

\vec{n} = vector libre perpendicular al *plano de visión* (*plano ficticio* donde se proyecta la imagen perpendicular al eje óptico de la cámara virtual). (*view plane normal, VPN*).

\vec{a} = punto en el eje óptico, también llamado *punto de atención o look-at point*.

\vec{u} = es un vector libre que indica una dirección que el observador ve proyectada en vertical en la imagen (apuntando hacia arriba) (*view-up vector, VUP*)

Problema 4.2.

Una posibilidad para hacer selección en mallas de triángulos es usar cálculo de intersecciones entre un rayo (una semirrecta que pasa por el centro de un pixel) y cada uno de los triángulos de la malla. Diseña un algoritmo en pseudo-código para el cálculo de intersecciones entre un rayo y un triángulo:

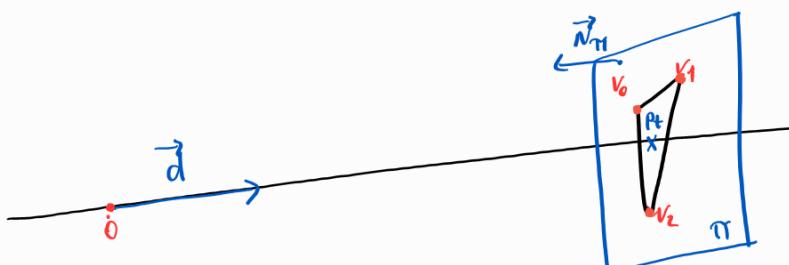
- ▶ El rayo tiene como origen o extremo el punto cuyas coordenadas del mundo es la tupla \mathbf{o} , y como vector de dirección la tupla \mathbf{d} (la suponemos normalizada).
- ▶ Las coordenadas del mundo de los vértices del triángulo son $\mathbf{v}_0, \mathbf{v}_1$ y \mathbf{v}_2 .
- ▶ El algoritmo debe de indicar si hay intersección o no, y, en caso de que la haya, calcular las coordenadas del mundo del punto de intersección.

Problema 4.2. (continuación)

Ten en cuenta que habrá intersección si y solo si se cumplen cada una de estas dos condiciones:

1. El rayo interseca con el plano que contiene al triángulo, es decir, existe $t > 0$ tal que el punto $\mathbf{p}_t \equiv \mathbf{o} + t\mathbf{d}$ está en dicho plano. Equivale a decir que el vector $\mathbf{p}_t - \mathbf{v}_0$ es perpendicular a la normal al plano.
2. El punto \mathbf{p}_t citado arriba está dentro del triángulo. Es decir, hay dos valores reales no negativos a y b (con $0 \leq a + b \leq 1$) tales que el vector $\mathbf{p}_t - \mathbf{v}_0$ es igual a $a(\mathbf{v}_1 - \mathbf{v}_0) + b(\mathbf{v}_2 - \mathbf{v}_0)$.

(a los tres valores a, b y $c \equiv 1 - b - a$ se les llama *coordenadas baricéntricas* de \mathbf{p}_t en el triángulo, se usan en ray-tracing).



Explicación:

- 1) La recta y el plano no intersecan si, y solo si, $\vec{d} \perp \vec{N}_{\pi} \Rightarrow$ intersecan si, y solo si, $\vec{d} \nparallel \vec{N}_{\pi} \Rightarrow$ intersecan si, y solo si $\vec{d} \cdot \vec{N}_{\pi} \neq 0$.
- 2) Calculamos el punto de intersección $\mathbf{p}_t = \mathbf{o} + t\mathbf{d}$. Si $t < 0$, el rayo no interseca con el plano del triángulo; es decir, solo seguimos si $t \geq 0$.
- 3) Calculamos las coordenadas baricéntricas de $\mathbf{p}_t - \mathbf{v}_0 = a(\mathbf{v}_1 - \mathbf{v}_0) + b(\mathbf{v}_2 - \mathbf{v}_0)$ y si $0 \leq a+b \leq 1$; $a, b \geq 0$, devolvemos true.

* Para calcular el punto intersección, el plano es el conjunto de puntos x que satisfacen $(x - v_0) \cdot \vec{N}_n = 0$. Llegando $p_t = o + td$ y sustituyendo:

$$\begin{aligned} ((o + td) - v_0) \cdot \vec{N}_n &= 0 \\ \Downarrow \\ (o^0 + td^0 - v_0^0) \cdot N_n^0 + (o^1 + td^1 - v_0^1) \cdot N_n^1 + (o^2 + td^2 - v_0^2) \cdot N_n^2 &= 0 \\ \Downarrow \\ t = \frac{-N_n^0 o^0 + N_n^0 v_0^0 - N_n^1 o^1 + N_n^1 v_0^1 - N_n^2 o^2 + N_n^2 v_0^2}{N_n^0 d^0 + N_n^1 d^1 + N_n^2 d^2} \end{aligned}$$

* Cálculo de las coordenadas báxicentricas:

$$\begin{aligned} (p_t - v_0) &= a(v_i - v_0) + b(v_e - v_0), a, b \in \mathbb{R} \\ \Downarrow \\ p_t - v_0^0 &= a(v_i^0 - v_0^0) + b(v_e^0 - v_0^0) \quad \left. \begin{array}{l} a = \frac{p_t - v_0^0 - b(v_i^0 - v_0^0)}{v_i^0 - v_0^0} \\ p_t - v_0^1 = a(v_i^1 - v_0^1) + b(v_e^1 - v_0^1) \end{array} \right\} \\ p_t - v_0^1 &= a(v_i^1 - v_0^1) + b(v_e^1 - v_0^1) \end{aligned}$$

← tiene que ser redundante pues $p_t - v_0$ está en el plano y hay solución para a, b

sustituyendo: $p_t - v_0^1 = \frac{p_t - v_0^0 - b(v_i^0 - v_0^0)}{v_i^0 - v_0^0} (v_i^1 - v_0^1) + b(v_e^1 - v_0^1)$

$$\Rightarrow b = \frac{(p_t - v_0^1)(v_i^0 - v_0^0) - (v_i^1 - v_0^1)(p_t - v_0^0 - b(v_i^0 - v_0^0))}{(v_i^0 v_0^0)(v_i^1 - v_0^1)}$$

$$a = \frac{p_t - v_0^0 - \left[\frac{(p_t - v_0^1)(v_i^0 - v_0^0) - (v_i^1 - v_0^1)(p_t - v_0^0 - b(v_i^0 - v_0^0))}{(v_i^0 v_0^0)(v_i^1 - v_0^1)} \right] (v_i^0 - v_0^0)}{v_i^0 - v_0^0}$$

Ya podemos hacer el pseudocódigo:

bool interseca(o, d, v₀, v₁, v₂) {

$$u = v_1 - v_0;$$

$$v = v_2 - v_0;$$

$$N \cdot \pi = \|u \times v\|$$

if (d · N · π = 0)

return false;

$$t = \frac{-N^0 o^0 + N^0 v^0 - N^1 o^1 + N^1 v^1 - N^2 o^2 + N^2 v^2}{N^0 d^0 + N^1 d^1 + N^2 d^2};$$

if (t < 0)

return false;

$$p_t = o + t d;$$

$$b = \frac{(p_t - v_0^1)(v_1^0 - v_0^0) - (v_1^1 - v_0^1)(p_t - v_0^0 - b(v_1^0 - v_0^0))}{(v_1^0 - v_0^0)(v_1^1 - v_0^1)};$$

$$a = \frac{p_t - v_0^0 - b(v_1^0 - v_0^0)}{v_1^0 - v_0^0} ;$$

if (a < 0 || b < 0 || a + b < 0 || a + b > 1)

return false;

return true;

}

Problema 4.3.

Para implementar la selección usando intersecciones es necesario calcular el rayo que tiene como origen el observador y pasa por centro del pixel donde se ha hecho click. Escribe el pseudo-código del algoritmo que calcula el rayo a partir de las coordenadas del pixel donde se ha hecho click:

- ▶ Tenemos una vista perspectiva, y conocemos los 6 valores l, r, t, b, n, f usados para construir la matriz de proyección.
- ▶ También conocemos el marco de coordenadas de vista, es decir, las tuplas $\mathbf{x}_{ec}, \mathbf{y}_{ec}$ y \mathbf{o}_{ec} con los versores y la tupla \mathbf{o}_{ec} con el punto origen (todos en coordenadas del mundo).
- ▶ El viewport tiene w columnas y f filas de pixels. Se ha hecho click en el pixel de coordenadas enteras x_p e y_p

El algoritmo debe producir como salida las tuplas \mathbf{o} y \mathbf{d} (normalizado) que definen el rayo.