

### Problema 2.1.

Supongamos que queremos codificar una esfera de radio  $1/2$  y centro en el origen de dos formas:

- ▶ Por enumeración espacial, dividiendo el cubo que engloba a la esfera en celdas, de forma que haya  $k$  celdas por lado del cubo, todas ellas son cubos de  $1/k$  de ancho. Cada celda ocupa un bit de memoria (si su centro está en la esfera, se guarda un 1, en otro caso un 0).
- ▶ Usando un modelo de fronteras (una malla indexada de triángulos), en el cual se usa una rejilla de triángulos y aristas que siguen los meridianos y paralelos, habiendo en cada meridiano y en cada paralelo un total de  $k$  vértices (se guarda únicamente la tabla de vértices y la de triángulos).

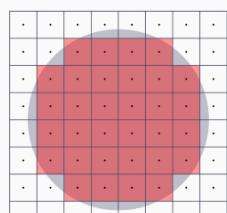
### Problema 2.1. (continuación)

Asumiendo que un **float** y un **int** ocupan 4 bytes cada uno, contesta a estas cuestiones:

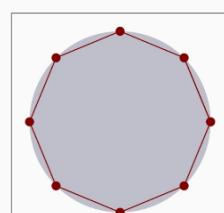
- ▶ Expresa el tamaño de ambas representaciones en bytes como una función de  $k$ .
- ▶ Suponiendo que  $k = 16$  calcula cuantos KB de memoria ocupa cada estructura.
- ▶ Haz lo mismo asumiendo ahora que  $k = 1024$  (expresa los resultados en MB)

Compara los tamaños de ambas representaciones en ambos casos ( $k = 16$  y  $k = 1024$ ).

Recordamos:



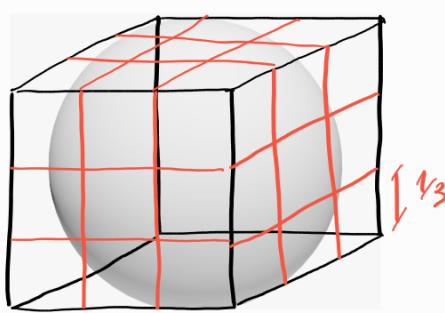
Enumeración espacial



Modelos de fronteras

Nuestra situación es, con por ejemplo  $K=3$ :

.) Enumeración espacial:

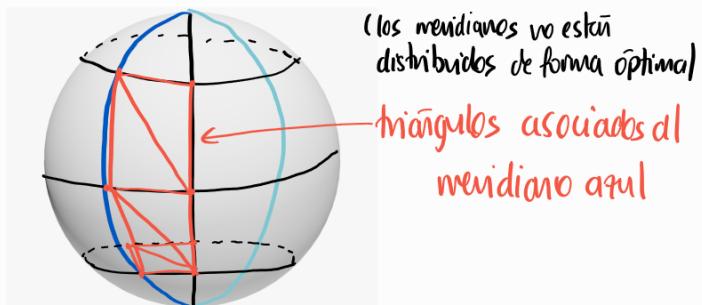


Como cada voxel tiene volumen  $\frac{1}{K^3} u^3$  y necesitamos encerrar  $1 u^3 \Rightarrow$  Necesitamos  $K^3$  cubitos. Concluimos:

$$\boxed{\text{Tamaño}(K) = K^3 \text{ bits}}$$

- Si  $K=16$ , necesitamos  $T(16) = 16^3 \text{ bits} = 16^3 \text{ bits} \cdot \frac{1 \text{ KB}}{2^{10} \text{ B}} \cdot \frac{1 \text{ B}}{8 \text{ bits}} = 0.5 \text{ KB}$
- Si  $K=1024$ , necesitamos  $T(1024) = 1024^3 \text{ bits} = 1024^3 \text{ bits} \cdot \frac{1 \text{ MB}}{2^{20} \text{ B}} \cdot \frac{1 \text{ B}}{8 \text{ bits}} = 128 \text{ MB}$

• I Modelo de fronteras:



Para que el enunciado tenga sentido y concuerde con lo dicho por el profesor en clase, entendemos por meridianos semicircunferencias (y paralelos las circunferencias completas). Se cortan en un único punto cada meridiano y paralelo  $\Rightarrow K$  vértices en cada meridiano y paralelo  $\Rightarrow K$  meridianos,  $K$  paralelos.

Como cada meridiano tiene  $K$  vértices  $\Rightarrow K \cdot K = K^2$  vértices en total (los de los meridianos los cubren todos).

Ahora, para cada meridiano, tenemos "suyos"  $2 \cdot K$  triángulos (2 para cada vértice) y de nuevo con todos completaríamos. Como hay  $K$  meridianos, tenemos un total de  $K \cdot 2 \cdot K = 2K^2$  triángulos.

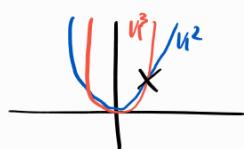
Concluimos:

$$\text{Tamaño}(K) = 2K^2(3 \cdot 4 \text{ B}) + K^2(3 \cdot 4 \text{ B}) = \boxed{36K^2 \text{ B}}$$

- Si  $K=16$ , necesitamos  $T(16) = 9216 \text{ B} \cdot \frac{\text{KB}}{2^{10} \text{ B}} = \boxed{9 \text{ KB}}$

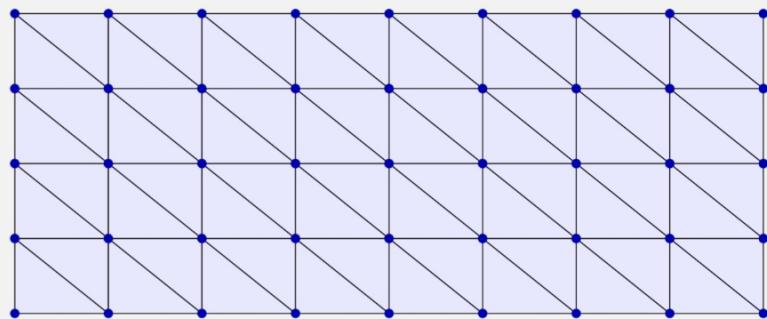
- Si  $K=1024$ , necesitamos  $T(1024) = 36 \cdot 1024^2 \cdot \frac{\text{MB}}{2^{20} \text{ B}} = \boxed{36 \text{ MB}}$

La interpretación de los resultados es que con numeración espacial, la complejidad es线性ica y con modelo de fronteras la complejidad es cuadrática. Entonces, para  $K$  pequeños se necesita menos memoria, pero a partir del punto de corte termina ocupando más memoria. Lo vemos con  $K=16$  vs.  $K=1024$ .



## Problema 2.2.

Considera una malla indexada (tabla de vértices y caras, esta última con índices de vértices) con topología de rejilla como la de la figura, en la cual hay  $n$  columnas de pares de triángulos y  $m$  filas (es decir, hay  $n+1$  filas de vértices y  $m+1$  columnas de vértices, con  $n, m > 0$ , en el ejemplo concreto de la figura,  $n = 8$  y  $m = 4$ ).



## Problema 2.2. (continuación)

En relación a este tipo de mallas, responde a estas dos cuestiones:

- Supongamos que un **float** ocupa 4 bytes (igual a un **int**) ¿que tamaño en memoria ocupa la malla completa, en bytes ? (tener en cuenta únicamente el tamaño de la tabla de vértices y triángulos, suponiendo que se almacenan usando los tipos **float** e **int**, respectivamente). Expresa el tamaño como una función de  $m$  y  $n$ .
- Escribe el tamaño en KB suponiendo que  $m = n = 128$ .
- Supongamos que  $m$  y  $n$  son ambos grandes (es decir, asumimos que  $1/n$  y  $1/m$  son prácticamente 0). deduce que relación hay entre el número de caras  $n_C$  y el número de vértices  $n_V$  en este tipo de mallas.

El objeto está en un espacio 3D (preguntado al profe).

$$\begin{aligned}
 \text{a)} \quad \text{Tamaño}(m, n) &= (n+1)(m+1) \text{ vértices} \cdot \frac{3 \cdot 4 \text{ B}}{1 \text{ vértice}} + n \cdot m \cdot 2 \text{ triángulos} \cdot \frac{3 \cdot 4 \text{ B}}{1 \text{ triángulo}} = \\
 &= 12(nm+n+m+1) + 24nm \text{ Bytes} \\
 &= \boxed{36mn + 12(n+m+1)} \text{ Bytes}
 \end{aligned}$$

$$\text{b)} \quad \text{Tamaño}(128, 128) = 592908 \cdot \frac{1 \text{ KB}}{2^{10} \text{ B}} = \boxed{57961 \text{ KB}}$$

$$\text{c)} \quad n_C = n \cdot m \cdot 2, \quad n_V = (n+1)(m+1). \quad \frac{n_C}{n_V} = \frac{n \cdot m \cdot 2}{n \cdot m + n + m + 1} = \frac{n \cdot m \cdot 2}{n(m+1) + m + 1} \xrightarrow{n \rightarrow \infty} \frac{m \cdot 2}{m+1} \xrightarrow{m \rightarrow \infty} \boxed{2}$$

Por tanto, termina habiendo dos triángulos por cada vértice.

### Problema 2.3.

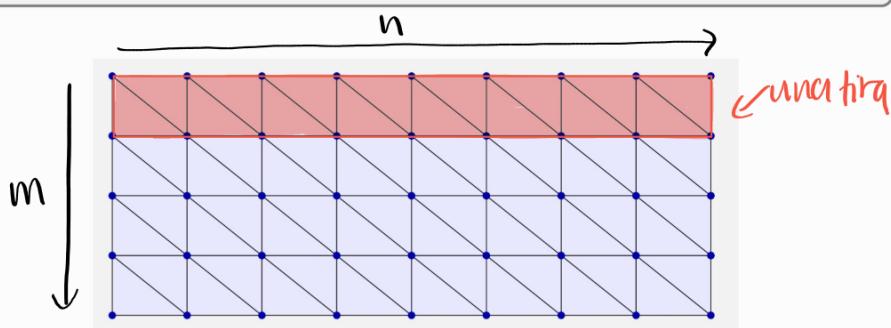
Imagina de nuevo una malla como la del problema anterior, supongamos que usamos una representación como tiras de triángulos, de forma que cada fila de triángulos (con  $2n$  triángulos) se almacena en una tira, habiendo un total de  $m$  tiras.

La tabla de punteros a tiras tiene un entero (el número de tiras) y  $m$  punteros, cada puntero suponemos que tiene 8 bytes de tamaño. De nuevo, asume que las coordenadas son de tipo **float** (4 bytes).

Responde a estas cuestiones:

### Problema 2.3. (continuación)

- (a) Indica que cantidad de memoria ocupa esta representación, en estos dos casos:
  - (a.1) Como función de  $n$  y  $m$ , en bytes.
  - (a.2) Suponiendo  $m = n = 128$ , en KB.
- (b) Para  $m$  y  $n$  grandes (es decir, cuando  $1/n$  y  $1/m$  son casi nulos), describe que relación hay entre el tamaño en memoria de la malla indexada del problema anterior y el tamaño de la malla almacenada como tiras de triángulos.
- (c) Si suponemos que la transformación de cada vértice se hace en un tiempo constante igual a la unidad, describe que relación hay entre los tiempos de procesamiento de vértices para esta malla cuando se representa como una malla indexada y como tiras de triángulos.



Una malla de tiras de triángulos se puede representar con una instancia de **MallaTT**, que a su veces tiene una o varias tiras (en **TiraTri**):

```
struct TiraTri // Tira de triángulos (coords. de vértices)
{
    unsigned long num_tri, // número de triángulos en esta tira
    std::vector<Tupla3f> ver; // coords. de vert. (num_tri+2 entradas)
    GLuint nombre_vao = 0; // VAO (creado en la visu.)
};

class MallaTT : public Objeto3D // Malla compuesta de tiras de triángulos
{
protected:
    std::vector<TiraTri> tiras; // vector de tiras
    .... int num_tiras;
public:
    virtual void visualizarGL( ContextoVis & cv );
};
```

El objeto está en un espacio 3D (preguntado al profe).

a) Cada tira ocupa en memoria un total de  $n_t + 2$  vértices (3 vértices para el primer triángulo y 1 por cada triángulo más). Por tanto:

$$\text{Tamaño}_m = \underbrace{(2n_t + 2) \cdot 3 \cdot 4}_{\substack{\text{vértices} \\ \text{por tira}}} + 4 + \underbrace{m \cdot 8}_{\substack{\text{versize} \\ (\text{uint})}} \text{ Bytes} = \boxed{4m(6(n_t + 1) + 3) + 4} \text{ Bytes}$$

int en el  $\rightarrow$  Tabla de punteros  
 número de tiras

$$\text{Tamaño}(128, 128) = 397828 \text{ B} \cdot \frac{1 \text{ KB}}{2^{10} \text{ B}} = \boxed{388150 \text{ KB}}$$

b)

$$\frac{\text{Tamaño}_m}{\text{Tamaño}_T} = \frac{36mn + 12(n+mt+1)}{4m(6(n+1)+3)+4} \xrightarrow{n \rightarrow \infty} \frac{36mn + 12}{24m} \xrightarrow{m \rightarrow \infty} \boxed{15}$$

Por tanto, la memoria para la malla indexada termina siendo aproximadamente 15.

c)

$$\frac{\text{Procesamiento}_m}{\text{Procesamiento}_T} = \frac{n_{vt} \cdot \text{constante } \alpha}{n_{vt} \cdot \text{constante } \alpha} = \frac{n_{vt}}{n_{vt}} = \frac{(n-1)(m-1)}{(2n+2)m} \xrightarrow{n, m \rightarrow \infty} \boxed{\frac{1}{2}}$$

Por tanto, el tiempo de procesamiento con la representación de malla indexada termina siendo la mitad que con tiras de triángulo.

### Problema 2.4.

Supongamos una malla cerrada, simplemente conexa (topológicamente equivalente a una esfera), cuyas caras son triángulos y cuyas aristas son todas adyacentes a exactamente dos caras (la malla es un *poliedro* simplemente conexo de caras triangulares). Considera el número de vértices  $n_V$ , el número de aristas  $n_A$  y el número de caras  $n_C$  en este tipo de mallas.

Demuestra que cualquiera de esos números determina a los otros dos, en concreto, demuestra que se cumplen estas dos igualdades:

$$\begin{aligned} n_A &= 3(n_V - 2) \\ n_C &= 2(n_V - 2) \end{aligned}$$

(nótese que, al igual que en el problema anterior, sigue siendo cierto que el número de caras es aproximadamente el doble que el de vértices).



Por un lado, cada cara tiene tres aristas, esto es, es adyacente a 3 aristas. Ahora, cada arista es adyacente a dos caras. Como coincide el número de adyacencias cara-arista con arista-cara:

$$3n_C = 2n_A \Leftrightarrow n_A = \frac{3n_C}{2}$$

↑                      ↗  
 total de adyacencias    total de adyacencias  
 con aristas para las caras    con caras para las aristas

Utilizaremos la fórmula de Euler,  $n_V - n_A + n_C = 2$  y lo tendremos.

Si no queremos usarla, esta es la forma de proceder (ito mensaje profesor):

"Ese ejercicio es difícil, aunque se puede demostrar sin la fórmula de euler, para ello consideramos un tetraedro, que es la malla cerrada más sencilla posible, y vemos que las igualdades se cumplen. Después vemos que si en una malla cualquiera se cumplen las igualdades, al insertar un vértice nuevo y unirlo a otros tres de un mismo triángulo (eliminando el triángulo antiguo e insertando tres triángulos nuevos), entonces la igualdad se sigue cumpliendo después de la inserción. De forma que por inducción se puede probar, ya que las igualdades se cumplen con el tetraedro, y añadiendo vértices y triángulos a un tetraedro podemos obtener una malla con topología igual a otra cualquier malla cerrada"

### Problema 2.5.

En una malla indexada, queremos añadir a la estructura de datos una tabla de aristas. Será un vector `ari`, que en cada entrada tendrá una tupla de tipo `Tupla2i` (contiene dos `int`) con los índices en la tabla de vértices de los dos vértices en los extremos de la arista. El orden en el que aparecen los vértices en una arista es indiferente, pero cada arista debe aparecer una sola vez.

Escribe el código de una función C++ para crear y calcular la tabla de aristas a partir de la tabla de triángulos. Intenta encontrar una solución con la mínima complejidad en tiempo y memoria posible. Suponer que el número de vértices adyacentes a uno cualquiera de ellos es como mucho un valor constante  $k > 0$ , valor que no depende del número total de vértices, que llamamos  $n$ .

### Problema 2.5. (continuación)

Considerar dos casos:

1. Los triángulos se dan con orientación *no coherente*: esto quiere decir que si un triángulo está formado por los vértices  $i, j, k$ , estos tres índices pueden aparecer en cualquier orden en la correspondiente entrada de la tabla de triángulos (puede aparecer como  $i, j, k$  o como  $i, k, j$ , o como  $k, j, i$ , etc....)
2. Los triángulos se dan con orientación *coherente*: esto quiere decir que si dos triángulos comparten una arista entre los vértices  $i$  y  $j$ , entonces en uno de los triángulos la arista aparece como  $(i, j)$  y en el otro aparece como  $(j, i)$  (decimos que en el triángulo  $a, b, c$  aparecen las tres aristas  $(a, b)$ ,  $(b, c)$  y  $(c, a)$ ). Además, asumimos que la malla es *cerrada*, es decir, que cada arista es compartida por exactamente dos triángulos.

### Problema 2.6.

Escribe el pseudo-código de la función para calcular el área total de una malla indexada de triángulos, a partir de la tabla de vértices y de triángulos. Será una función que acepta un puntero a una **MallaInd** y devuelve un número real (asumir que se dispone del tipo **Tupla3f** y de los operadores usuales de tuplas o vectores, es decir suma +, resta -, producto escalar ·, producto vectorial ×, módulo || ||, etc ...).

```
float calcularArea(const MallaInd malla) {
    float total=0;

    for(int i=0; i<triangulos.size(); i++){
        Tupla3f u= vertices[triangulos[i][1]] - vertices[triangulos[i][0]];
        Tupla3f v= vertices[triangulos[i][2]] - vertices[triangulos[i][1]];
        total+= sqrt((u.cross(v)).lengthSq()) / 2;
    }

    return total;
}
```

**Problema 2.7.**

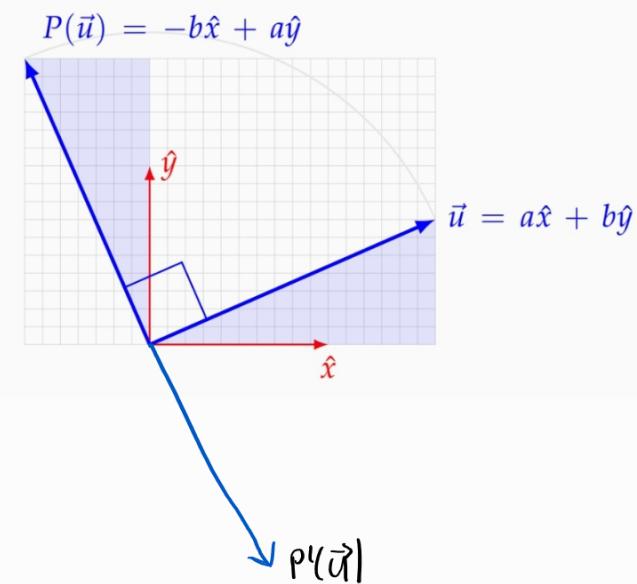
Demuestra que  $\vec{u}$  y  $P(\vec{u})$  son siempre perpendiculares según la definición anterior (es decir, siempre  $\vec{u} \cdot P(\vec{u}) = 0$ ). -

$$\vec{u} \cdot P(\vec{u}) = (a\hat{x} + b\hat{y}) \cdot (-b\hat{x} + a\hat{y}) = a(-b) + ba = 0.$$

**Problema 2.8.**

Describe como se podría definir una rotación hacia la derecha (en el sentido de las agujas del reloj) en lugar de a izquierdas.

$$\rho'(\vec{u}) = b\hat{x} + a\hat{y}$$



### Problema 2.9.

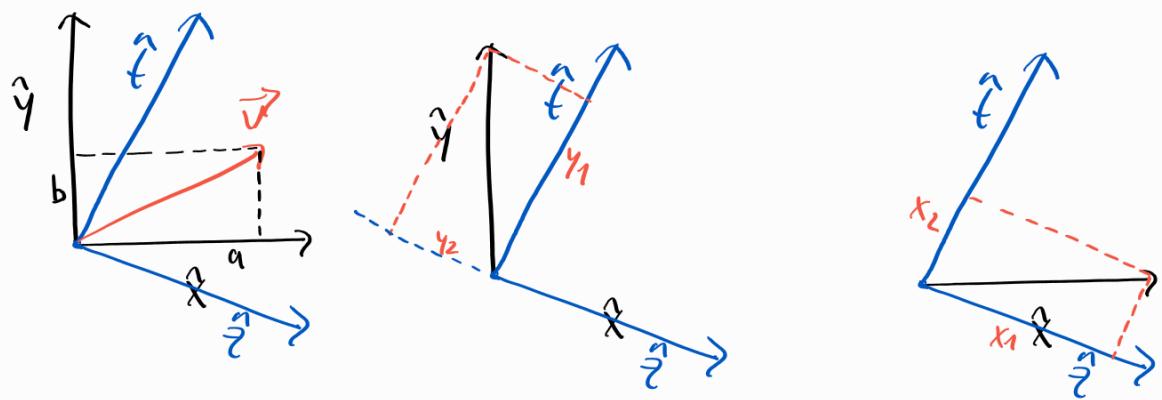
Demuestra que la transformación afín  $P$  (cuando se aplica a vectores, no a puntos) no depende del marco cartesiano  $C$  con respecto al cual expresamos las coordenadas  $(a, b)$  (en el caso de aplicarla a puntos, la rotación de  $90^\circ$  es entorno al punto origen  $o$  de  $C$ ).

Sean  $C = [\hat{x}, \hat{y}, \hat{o}]$  y  $C' = [\hat{x}', \hat{t}, \hat{o}']$  dos marcos cartesianos cuales sean. Sea  $\vec{v}$  un vector cualquiera  $\vec{v} = C(a, b) = C'(a', b')$ . Tenemos que ver que  $P(\vec{v}) = C(-b, a) = C'(-b', a')$ .  $C(-b, a) = -bx + ay$ . Sean  $\hat{x} = C'(x_1, x_2)$ ,  $\hat{y} = C'(y_1, y_2)$ . Entonces,

- $C(a, b) = C(a', b') \Leftrightarrow ax + by = a'\hat{x} + b'\hat{t} \Rightarrow a(x_1\hat{x} + x_2\hat{t}) + b(y_1\hat{x} + y_2\hat{t}) = (ax_1 + by_1)\hat{x} + (ax_2 + by_2)\hat{t} = a'\hat{x} + b'\hat{t} \Rightarrow \begin{cases} ax_1 + by_1 = a' \\ ax_2 + by_2 = b' \end{cases}$
- $C(-b, a) = -bx + ay = -b(x_1\hat{x} + x_2\hat{t}) + a(y_1\hat{x} + y_2\hat{t}) = (-bx_1 + ay_1)\hat{x} + (-bx_2 + ay_2)\hat{t} = C'(-bx_1 + ay_1, -bx_2 + ay_2) \stackrel{?}{=} C'(-b', a')$ .

Ya solo queda ver que  $-b' = -bx_1 + ay_1$ ;  $a' = -bx_2 + ay_2$ .

$$\begin{array}{ccc}
 & \uparrow & \uparrow \\
 a x_2 + b y_2 = -b x_1 + a y_1 & & a x_1 + b y_1 = -b x_2 + a y_2 \\
 \uparrow & & \uparrow \\
 b(y_2 + x_1) = a(y_1 - x_2) & & b(y_1 + x_2) = a(y_2 - x_1) \\
 \Downarrow \leftarrow \text{si } \neq 0 & & \Downarrow \leftarrow \text{si } \neq 0 \\
 a = \frac{b(y_2 + x_1)}{y_1 - x_2} & & a = \frac{b(y_1 + x_2)}{y_2 - x_1} \\
 & \uparrow & \\
 \frac{b(y_2 + x_1)}{y_1 - x_2} & = & \frac{b(y_1 + x_2)}{y_2 - x_1} \\
 & \uparrow & \\
 y_2^2 - x_1^2 & = & y_1^2 - x_2^2 \\
 & \uparrow & \\
 y_2^2 - x_1^2 & = & y_2^2 - x_1^2
 \end{array}$$



No me sale.