28. ¿Podría afirmar desde el punto de vista topológico que el cubo de la Ilustración 39 se parece mucho más a una esfera que un donut?

Si, ya que el cubo no tiene el hueco que un donut tendría y al cubo si se le van añadiendo vertices mas se parecería a una esfera

29. De las dos aproximaciones mostradas (glBegin/glEnd y glDrawArrays), ¿cuál cree que es más eficiente en términos de tiempo de procesamiento? ¿y en cuanto a transferencia CPU-bus-GPU?

Es mas eficiente glDrawArrays ya que con glBegin/glEnd se envían los vértices de uno en uno mientras que con glDrawArrays se usa un puntero al vector de vertices, y en cuanto a transferencia CPU-bus-GPU igual.

30. Documente en sus apuntes el formato de archivo STL. ¿Es eficiente?

El formato de archivo STL define la geometria de los objetos 3D. Es eficiente ya que no incluye información acerca del color, las texturas o las propiedades físicas del objeto. Ademas el formato STL especifica representaciones ASCII y binarias, los archivos binarios son muy compactos.

31. ¿Cuántas llamadas a glVertex se realizan para una tira de n triángulos?

3n si los triangulos estan separados, si varios triangulos repiten vertices solo hay que llamar a glVertex por cada vertice

32. Defina una estructura de datos que permita almacenar tiras de triángulos de una malla 3D.

Ir guardando los vertices en tuplas de 3 elementos:
(v0,v1,v2)
(v3,v2,v4)

33. Escriba un código en C que calcule el área total de una malla de triángulos almacenada como lista de triángulos y vértices.
34. Defina la estructura de datos en C que almacene la estructura de aristas aladas. Escriba una función en C que la rellene a partir de una lista de triángulos y vértices.

35. ¿Se podría evitar tener el campo semiaristaopuesta ? En caso afirmativo, ¿cómo? Si no, ¿por qué?

Si, las aristas pares tendrían su opuesta en la siguiente arista y las impares en la siguiente.

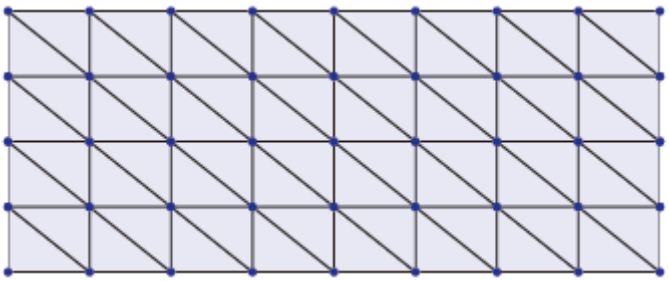
36. Defina la estructura de datos en C que almacene la estructura de semiaristas aladas.

```
typedef unsigned int Index;

struct Point3D{
	float x;
	float y;
	float z;
}

struct vertice{
	Point3D p;
```

```
	Index sa;
}

struct cara{
	Index sa;
}

struct semiAristas{
	Index v;
	Index c;
	Index saS;
	Index saA;
	Index saO;
}

class MallasAA{
	std::vector<vertice> v;
	std::vector<cara> c;
	std::vector<semiAristas> sa;
}
```

37. Escriba un código en C que averigüe de forma eficiente el número medio de caras por vértice usando una malla almacenada en la estructura de semiaristas aladas definida en el ejercicio anterior.

38. Considera una malla como la de la figura, en la cual hay n columnas y m filas de (pares de) triángulos



- Expresa el número de vértices en función de n y m.

(n+1)*(m+1)

- Suponiendo que Natural y Real ocupan 4 bytes, calcular el espacio en disco utilizando:
1. Lista de triángulos
2. Lista de triángulos y vértices
3. Aristas aladas
4. Semiaristas aladas

39. Escriba un código en C que calcule la normal de un triángulo, dados sus tres vértices.
40. Escriba un código en C que, dada una malla de triángulos, calcule la normal en los vértices, si:
- La malla está definida como lista de caras y vértices,
- La malla está definida con una estructura de semiaristas aladas.

41. ¿Tiene influencia en el valor de la normal en el vértice el área de los triángulos que lo comparten?

No, ya que el valor de la normal en el vértice se calcula a través de las caras que comparten dicho vértice, y aunque el triangulo aumente y la distancia aumente también, el valor de la normal del vértice no cambia.

42. Escribe el código para crear una copia de una malla de triángulos, de forma que en la copia los vértices estén cada uno desplazado en la dirección de su normal una distancia d, siendo d un parámetro Real de la función.

```
class Malla{
	std::vector<xyz> v;
	std::vector<cara> c;
	std::vector<xyz> n;
}

Malla::Malla (Malla &m, float d){
	xyz tmp;
	this -> c = m ->  c;
	this -> n = m ->  n;

	for(int i=0; i<m-> v.size(); i++){
		tmp= m-> v[i];
		tmp.x += m-> n[i].x * d;
		tmp.y += m-> n[i].y * d;
		tmp.z += m-> n[i].z * d;
		this->push_back(tmp);
}
```

43. Escribe el código para visualizar una malla con los triángulos de un color sólido y las aristas en modo alambre, a la vez. ¿Qué problemas se pueden encontrar?

44. Calcule los nuevos valores de la posición en el espacio del punto [3,0,0] aplicándole las rotaciones del ejercicio anterior, por separado, y además tras las siguientes secuencias:
- R x [90] R y [-90] R z [200]:

Rotamos en x:

$$[p'x, p'y, p'z] = [3,0,0]\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(90) & \sin(90)\\ 0 & -\sin(90) & \cos(90)\end{bmatrix} = [3,0,0]$$

Rotamos en y:

$$[p'x, p'y, p'z] = [3,0,0]\begin{bmatrix}\cos(-90) & 0 & -\sin(-90)\\ 0 & 1 & 0\\ \sin(-90) & 0 & \cos(-90)\end{bmatrix} = [0,0,3]$$

Rotamos en z:

$$[p'x, p'y, p'z] = [0,0,3]\begin{bmatrix}\cos(200) & \sin(200) & 0\\ -\sin(200) & \cos(200) & 0\\ 0 & 0 & 1\end{bmatrix} = [0,0,3]$$

Posicion final: [0,0,3]

- R z [200] R y [-90] R x [90]

Rotamos en z:

[p′x, p′y, p′z] = [3cos(200),3sin(200),0]

Rotamos en y:

[p′x, p′y, p′z] = [0,3sin(200),3cos(200)]

Rotamos en x:

[p′x, p′y, p′z] = [0,-3cos(200),3sin(200)]

Posicion final: [0,2,819077862,−1,02606043]

45. Para comprobar la no conmutatividad de las transformaciones geométricas, calcule los nuevos valores de la posición en el espacio del punto p= [3,0,0] aplicándole las siguientes secuencias:

- P'=(T 3 •T 2 •T 1)p
- P'=(T 1 •T 2 •T 3)p
- P'=(T 3 •T 1 •T 2)p

Siendo

- T 1 = R x [90]
- T 2 = T[-9,4,3]
- T 3 = R z [45]

Para estas secuencias, calcule tras cada transformación de la composición el valor temporal del punto y su posición final.

- P'=(T 3 •T 2 •T 1)p

Primero rotamos en x 90º:

se queda igual → [3,0,0]

Trasladamos con T2:

se queda → [-6,4,3]

Rotamos en z 45º:

se queda → [-6cos(45) – 4sin(45), -6sin(45) + 4cos(45), 3] = [-7'071, -1'4142,3]

- P'=(T 1 •T 2 •T 3)p

Rotamos en z 45º:

se queda → [3cos(45),3sin(45),0]

T2: [3cos(45) -9, 3sin(45) + 4, 3]

T1: [-6'878, -3, 2'1213]

- P'=(T 3 •T 1 •T 2)p

T2: [-6,4,3]
T1: [-6, -3, 4]
T3: [-6cos(45)+3sin(45), -6sin(45) – 3cos(45), 4] = [-2’121, -6’364, 4]

46. Escribe funciones en C para manipulación de transformaciones y para aplicar transformaciones a tuplas de 3 reales.

*Componer(Transformacion *res, Transformacion *m1, Transformacion *m2)*
Compone dos matrices de transformación m1 y m2, y guarda el resultado en res

*Aplicar(Real res[4], Transformacion *m, Real coo[4])*
Aplica la matriz de transformación m a la tupla de 4 reales coo, y almacena el resultado en res.

*Rotacion(Transformacion *res, Real a, Real ex, Real ey, Real ez)*
Crea una matriz de rotación de a grados entorno al eje definido por (ex, ey, ez). Escribe la matriz resultado en res.

*Traslacion(Transformacion *res, Real dx, Real dy, Real dz)*
Escribe en res la matriz de traslación asociada al vector de desplazamiento (dx, dy, dz).

*Escalado(Transformacion *res, Real sx, Real sy, Real sz)*
Escribe en res la matriz de escalado con factores de escala sx,sy y sz.
Siendo *typedef Real Transformacion[4][4];*

47. Escribe una función en OpenGL, llamada gancho para dibujar con OpenGL la figura:
Cada segmento recto tiene longitud unidad, y el extremo inferior está en el origen. Use para ello glBegin , glVertex2f y glEnd
48. Usando exclusivamente la función gancho del problema anterior, construye otra función gancho_x4 para dibujar el polígono que aparece en la figura:
Hay que tener en cuenta que la figura se puede obtener exclusivamente usando rotaciones de la original, pero en esas rotaciones el centro no es el origen.

49. Documente en sus apuntes: ¿Cómo se hace una rotación con respecto a cualquier punto?

Para rotar un objeto lo que hay que hacer es trasladarlo al origen, rotarlo ahí y despues trasladarlo donde estaba.

50. Escribe el pseudocódigo OpenGL otra función gancho_2p para dibujar la figura delejercicio 52, pero escalada y rotada de forma que sus extremos coincidan con dos puntos arbitrarios p0 = (x0, y0) y p1 = (x1, y1), que se pasan como parámetro a dicha función.