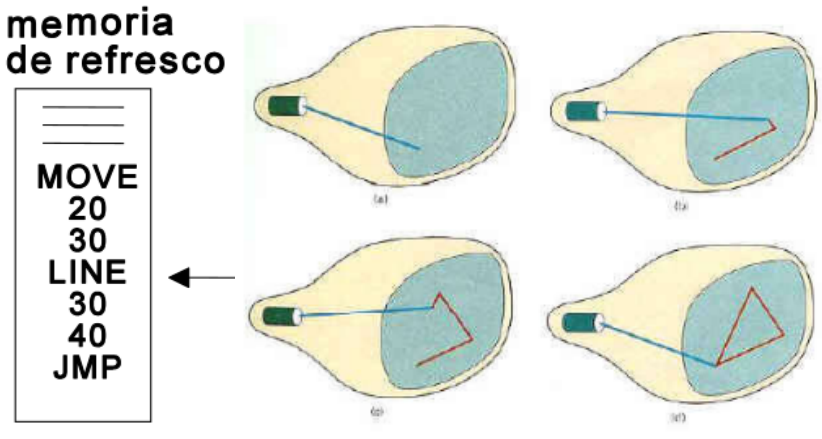
1. terminal raster y vectorial.

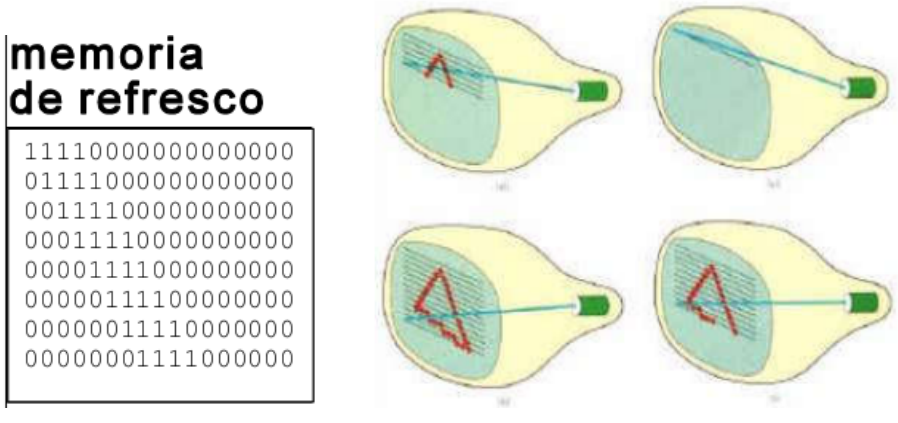
Ambas usan tubo de rayos catódicos (TRC).

Vectorial: TRC en el que el haz de electrones recorre la pantalla de acuerdo a unas órdenes de dibujo.



Raster: Años 70, usados en tv y posteriormente en computación.

Direccionamiento de líneas extrapolado a una matriz bidimensional de datos mostrados en pantalla; haz no recorre libremente sino ordenado desde superior izquierda hasta inferior derecha.



Otros tipos: Pantallas de plasma, LCD, OLED, LED. Sus características: Frecuencia de refresco, Resolución horizontal o vertical, Profundidad de color.

2. calcular tasa media de refresco

Tengo un juego en mi ordenador que tiene las siguientes tasas de refresco. ¿Cuál es la tasa media de refresco en frames por segundo?

- ➔ 25f/s durante 100 frames → 100/25 = 4 segundos
- ➔ 50f/s durante 100 frames. →100/50 = 2 segundos
- ➔ 10f/s durante 100 frames → 100/10 = 10 segundos
- ➔ 25f/s durante 100 frames → 100/25 = 4 segundos

Total de frames = 400 frames
Total de segundos = 4 + 2 + 10 + 4 = 20 segundos

Tasa de refresco media = 400/20 = 20 frames/segundo

3. definición pixel, frame buffer

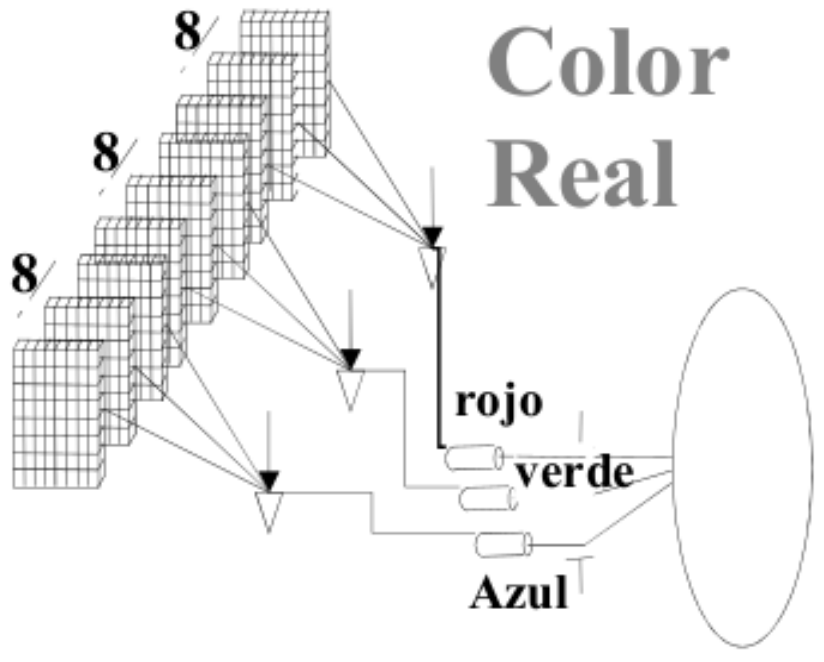
Píxel: Unidad básica de una imagen digitalizada en pantalla a base de puntos de color o en escala de grises.
Frame buffer: El conjunto del valor de los píxeles se almacena en una área de memoria que se denomina (frame-buffer). Las aplicaciones varían estos valores y son volcados (swap) en un momento determinado. (ver siguiente pregunta)

4. calcular tamaños frame buffer, double buffer, quad buffer, z-buffer, ciclos de reloj por píxel

Frame Buffer

1 bit por píxel= 2^1 = 2 colores, también llamado monocromo.
2 bits por píxel= 2^2 = 4 colores, o CGA.
3 bits por pixel = 2^3 = 8 colores Zx spectrum.
4 bits por píxel: 2^4 = 16 colores EGA. Mac original.
8 bits por píxel: 2^8 = 256 colores, también llamado VGA.

Color Real: Si son más de 8 bits se utiliza modelo RGB estándar (y transparencia).
8 bits por cada uno de los colores → 24 bits por píxel (+8 con transparencia)



Dependiendo del tipo de color, Tamaño del frame-buffer = bits por píxel X resolución

Si mi pantalla tiene una resolución VGA y color real cuanto ocupa el frame buffer en megabytes. ¿y si la resolución de la pantalla es 1280*1024 píxeles?

Datos: resolución VGA = 640 x 480
Color real = 24 bits por píxel
1 MByte = 10^6 Bytes = $8*10^6$ bits

Tamaño frame-buffer = 24 * 640 * 480 = 7372800
En MBytes = $7372800/8*10^6$ = 0,9216 MB

Doble buffer: De la WIKIPEDIA En gráficos de computadora, el doble buffer es una técnica para la presentación de gráficos en tiempo real que elimina o reduce considerablemente el número de parpadeos, ruido u otros artefactos.

Es difícil que un programa refresque la imagen en una pantalla sin que los píxeles cambien más de una vez. Por ejemplo, para actualizar una página de texto es mucho más fácil borrar toda la página y, a continuación, dibujar las letras de nuevo, que borrar todos los píxeles que están presentes en la nueva imagen. Sin embargo, esta imagen intermedia (cuando se borran todos los píxeles) es percibida por el usuario como un parpadeo. Los monitores de ordenador constantemente redibujan su imagen visible (en torno a 60 veces por segundo), por lo que incluso una actualización perfecta puede ser visible momentáneamente como un divisor horizontal entre la "nueva" imagen y la "vieja" (efecto conocido como lagrimeo).

quad buffer de Wikipedia El término "buffering cuádruple" se utiliza en las implementaciones de estereoscópico, y significa el uso de doble buffer para cada una de las imágenes del ojo izquierdo y derecho, por lo tanto hay cuatro topes totales. El comando para intercambiar o copiar el buffer normalmente se aplica a los dos pares a la vez.

Supongo que al ser literalmente dos o cuatro buffers al calcular tamaños haces x2 o x4 y listo. ¯_(ツ)_/¯

Depth buffer o z-buffer: se utiliza para determinar si un objeto o parte de un objeto es visible en una escena. Tiene el mismo ancho y alto que el búfer de color. Cada píxel tiene una coordenada XYZ de profundidad o distancia de la cámara (16, **24** o 32 bits). Compara el valor de cada fragmento con el anterior y en función del resultado se sustituye o no.

Se renderiza un objeto (escritura frame buffer y z buffer) → Se va a renderizar el siguiente, si el valor del z-buffer es inferior se dibuja y actualiza.

Las comparaciones se basan en la ecuación : $F_{depth} = \frac{1/z - 1/near}{1/far - 1/near}$

```
glEnable( GL_DEPTH_TEST);
glClear (GL_DEPTH_BUFFER_BIT);
glDepthFunc (GL_ALWAYS);

GL_NEVER, GL_LESS,GL_EQUAL, GL_LEQUAL, GL_GREATER, GL_NOTEQUAL, GL_GEQUAL, GL_ALWAYS
```

(ver siguientes 2 preguntas)

---

5. ¿El z-buffer se aplica antes de la rasterización con el fin de no hacer cálculos sobre los objetos ocultos?

Rasterización es el proceso por el cual una imagen descrita en un formato gráfico vectorial se convierte en un conjunto de píxeles o puntos para ser desplegados en un medio de salida.

La pregunta básicamente es ¿Cuando se hacen las comprobaciones del depth buffer, antes o después de hacer el frame buffer?

Generalmente antes → is supposed to let one do the depth test before the fragments are processed and through that test discard fragments which fail the test. This would give an increased running performance as a fragment shader with a heavy implementation wouldn't need to be executed as many times.

---

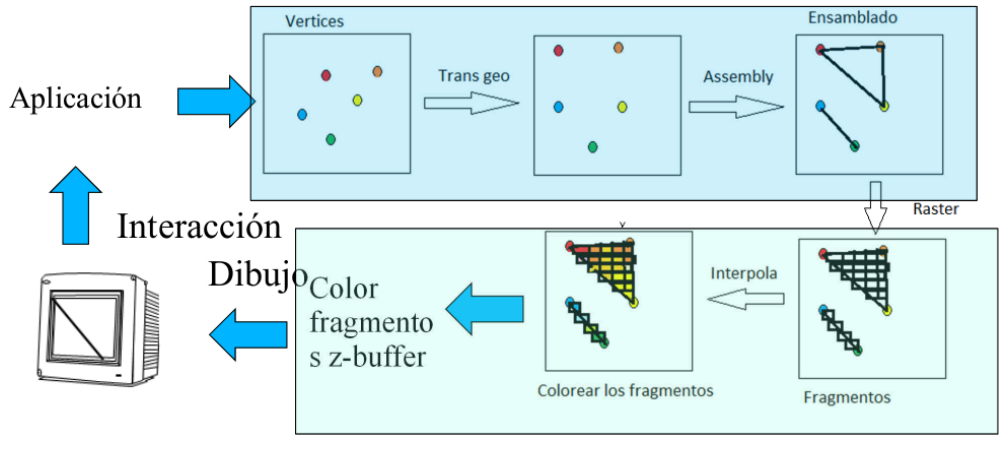
6. z-buffer, rendimiento front to back y back to front

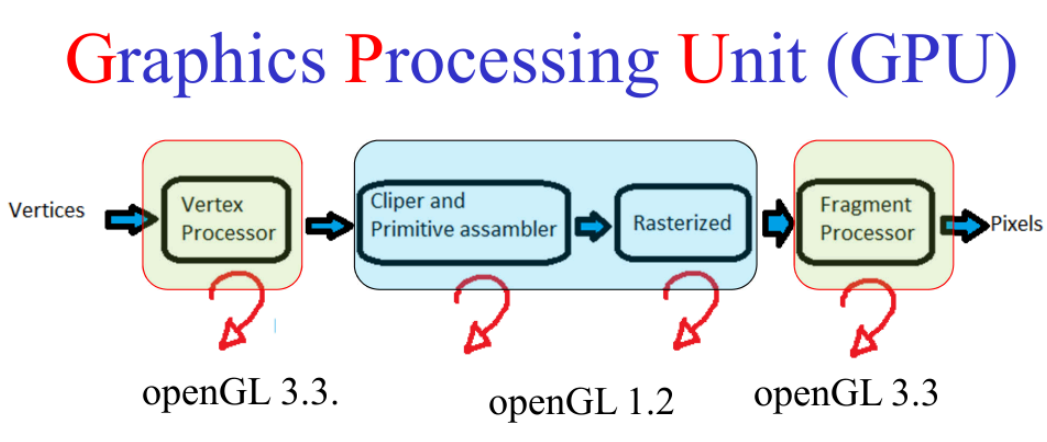
De frente hacia atrás porque vas descartando antes los píxeles, si lo haces de atrás hacia delante vas a sustituir el color en ese píxel muchas veces (ver imagen). Por defecto los sistemas lo hacen así pero se puede desactivar.

---

7. proceso del pipeline de openGL estático. proceso de visualización de escena

Al principio, usar OpenGL significaba desarrollar en modo inmediato (fija), que es un método fácil de usar. La mayor parte de la funcionalidad de OpenGL estaba oculta y no tenían mucha libertad para determinar la forma en que OpenGL realiza sus cálculos. El modo inmediato es fácil de usar, pero también es lento. Por esa razón, la especificación comenzó a despreciar la funcionalidad del modo inmediato de la versión 3.3 nos obliga a usar prácticas modernas. que es muy flexible y eficiente es más difícil de aprender.





Transformaciones de vértices
Ensamblado y rasterización
Fragment texturización y coloreado
Interpolación
Dibujo.

El vertex shader. Realiza la transformación de la posición de vértice en la pantalla. Sus entradas son el vértice del triángulo junto con cualesquiera datos, que se necesite. La salida del sombreado de vértices es un vértice en una posición transformada y posiblemente otra información, como la normal, color.
Fragment shader. Recibe la información del elemento a procesar. La salida es el color del pixel.
Rasterizado. zbuffer y Alpha test.

Vertex shaders.
Función que recibe como parámetro vértices.

Manipula vértices en 3D; Color, coordenadas de textura, orientación y tamaño de vértices.

Su propósito es transformar las coordenadas 3D de los vértices en otras coordenadas que ya veremos cuales son.
La salida debe ser la posición de los vértices.

Fragment shaders (OpenGL)
Recibe la información de cada fragmento.
Un fragmento es toda la información necesaria para dibujar un pixel.
Se aplica en la última etapa de rasterizado.
Su propósito es obtener el color de cada pixel.
La salida es el color de cada píxel.

Qué pasa después del Fragment Shader
Se realizan una serie de comprobaciones relativas al Alpha test, blending test y Depth test.

En primer lugar, se parte de modelos de diferentes objetos, a los que se les pueden aplicar diferentes transformaciones (de modelado), y que conforman una escena de un mundo en tres dimensiones. Aplicando a este mundo las transformaciones de vista o visualización pertinentes, obtenemos una vista de esa escena en tres dimensiones.
Esta vista en tres dimensiones pasa a continuación por un proceso denominado clipping cuya finalidad es ajustar la imagen que se quiere visualizar al espacio disponible en la ventana o pantalla en la que se proyectará. El siguiente paso es la proyección en el que se calcula cómo representarla en una pantalla en dos dimensiones.
Por último, a esta proyección en dos dimensiones de nuestra escena en tres dimensiones ya ajustada para ser encuadrada en el espacio disponible para su proyección se le aplica un último proceso conocido como rasterización, que convierte toda la información que se tiene de la imagen a un conjunto de píxeles que pueden ser introducidos en el framebuffer de un dispositivo para que este los proyecte.

---

8. MODELADO. modelado geométrico y texturizado

¿ Qué es modelar un objeto sólido? Dar forma a un objeto. Representación de sus superficies, el contorno del objeto, proporcionando información. Debe dar +cuenta de su geometría y propiedades visuales (color, textura, alguna propiedad física (elasticidad, cómo se comporta frente a la luz)).

---

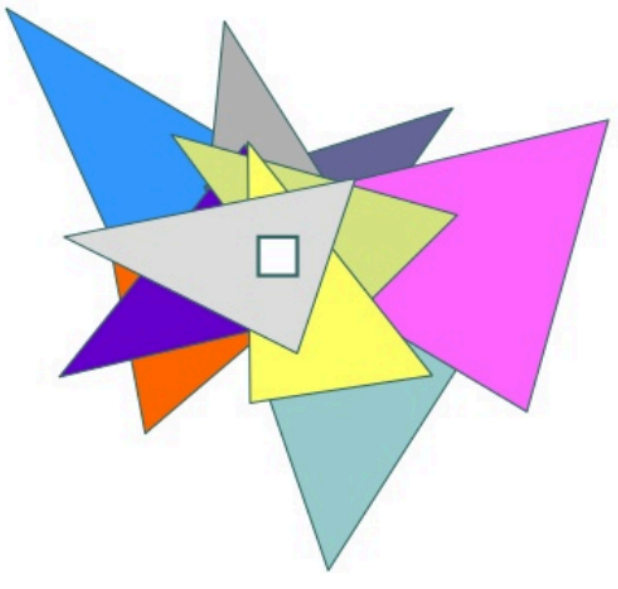
9. dibujar las secuencias de vértices (según el glBegin())

GL_POINTS - Treats each vertex as a single point.

GL_LINES - Treats each pair of vertices as an independent line segment.
GL_LINE_STRIP - Draws a connected group of line segments from the first vertex to the last.
GL_LINE_LOOP - Draws a connected group of line segments from the first vertex to the last, then back to the first.

GL_TRIANGLES - Treats each triplet of vertices as an independent triangle.
GL_TRIANGLE_STRIP - Draws a connected group of triangles. One triangle is defined for each vertex presented after the first two vertices. For odd n, vertices n, n+1, and n+2 define triangle n.For even n, vertices n+1, n, and n+2 define triangle n.
GL_TRIANGLE_FAN - Draws a connected group of triangles. One triangle is defined for each vertex presented after the first two vertices. Vertices 1, n+1, and n+2 define triangle n. N-2 triangles are drawn.

GL_QUADS - Treats each group of four vertices as an independent quadrilateral.
GL_QUAD_STRIP - Draws a connected group of quadrilaterals. One quadrilateral is defined for each pair of vertices presented after the first pair.
GL_POLYGON - Draws a single, convex polygon.



---

10. Identificación de caras

Es muy importante el orden en que se especifican los vértices ya que esto determina la normal de la cara.

glEnable (GL_CULL_FACE); → indica que se ocultaran caras siguiendo la regla

Normales/caras

Una normal por cara, normales no unitarias. La normal se calcula a partir de los vértices y se normaliza o se puede especificar un conjunto de ellos.

glEnable(GL_NORMALIZE) → If enabled, normal vectors are normalized to unit length after transformation and before lighting.

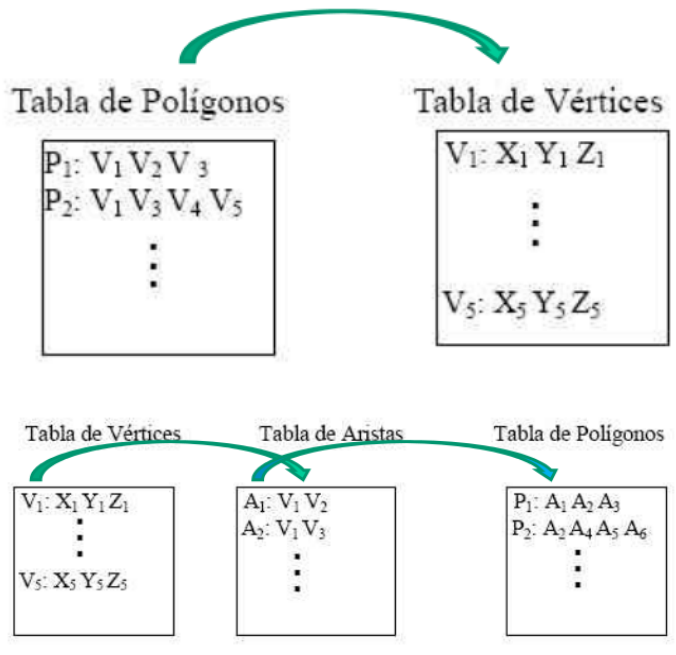
---

11. Representación de superficies tablas indexadas

Para la representación de un objeto en base a polígonos necesitamos conocer vértices y sus atributos y representarlos. Y almacenarse de alguna forma.
– Tablas geométricas incorporan la información sobre los vértices y los parámetros que proporcionan información sobre los polígonos o la superficie que determinan.
– Tablas de atributos informan sobre las propiedades físicas de los polígonos, color transparencia...texturas...

Puede ser:

- Sin indexar
- Indexada (simplificar eliminando las aristas utilizado únicamente una lista de vértices y una de polígonos)
- Doblemente indexada (para cada polígono guardaremos las aristas que lo forman; para cada arista, tendríamos los dos vértices que la forman y, si se desea, los dos polígonos que la comparten)



12. Modelo poliédrico

Un modelo poliédrico es aquel que utiliza poliedros (cuerpos geométricos tridimensionales formados por polígonos planos) para representar tridimensionalmente estructuras complejas. Los polígonos más frecuentemente utilizados para formar los poliedros son los triángulos.

El modelo exige que en un poliedro se cumpla que:

- Un vértice debe pertenecer al menos a una arista.
- Los polígonos no deben intersecarse entre sí, salvo en las aristas.
- En una arista no deben confluir más de dos polígonos.
- En un vértice pueden coincidir cualquier número de polígonos.

En una buena representación poliédrica, debemos:

- Evitar la degeneración de los polígonos (lados con longitud cero).
- Evitar vértices en T, ya que puede provocar problemas a la hora de rellenar el color por interpolación entre los vértices, y otras singularidades en ciertos algoritmos.
- Utilizar primitivas poligonales que permitan especificar el objeto sin repetir varias veces los vértices que pertenezcan a varios polígonos.
- Cuidarnos de la falta de coplanaridad (cualidad de estar en un mismo plano) en polígonos de más de tres vértices.

---

13. Listas de visualización

Son índices que nos permiten renderizar un conjunto de vértices, previamente almacenados, simplemente invocando dicho índice.

```
IndiceLista = glGenLists(1);

glNewList(IndiceLista,GL_COMPILE);
       ...
glEndList();

glCallList(IndiceLista);
```

---

14. clipping, viewport, mapping

- Clipping: es el estado del pipeline de gráficos que determina qué primitivas (vértices y demás) se descartan y cuáles pasan al siguiente estado.
- Viewport: permite dividir la ventana principal en más ventanas. Se utiliza para ello glViewport();
- Mapping: técnica de textura para aplicar efectos a una superficie. Explicado más abajo.

---

15. coordenadas homogéneas, permiten traslación, escalado y rotación?

Las coordenadas homogéneas son coordenadas de posición de 4 dimensiones, en la que el cuarto valor es un número que representa un factor de escala, que permiten ser representadas en una matriz para así realizar cálculos matriciales de traslación, escalado y rotación.

Las coordenadas homogéneas son un instrumento utilizado para describir un punto (x, y) representado con la terna (x/w, y/w, w), donde w es un número real. Su uso permite tratar cualquier transformación geométrica como una multiplicación de matrices, facilitando el cálculo de transformaciones. El uso de coordenadas homogéneas ofrece las siguientes ventajas:

- Facilitan las operaciones de concatenación de transformaciones mediante el producto de matrices.
- Las tarjetas aceleradoras de gráficos implementan en hardware las operaciones matriciales con coordenadas homogéneas, mejorando el rendimiento de estas operaciones. Si las operaciones se realizarán por software requerirían un gran poder de cómputo.

---

16. coordenadas de texturas (u, v), técnicas de texturizado

Son coordenadas (u, v) de textura que indican la posición de un punto en una textura 2D que se va a asociar a un punto en un cuerpo en 3D.

- Texture Mapping: Crea una textura a partir de un patrón (imagen) sobre la superficie.
- Bump Map: Perturba la superficie mediante una modificación de las normales u otros.
- Environmental Map: Mapea el entorno sobre la superficie.
- Displacement mapping: Modifica la geometría de un objeto mediante un mapa de texturas.
- Light mapping: Cálculo del brillo en superficies previo al cómputo, que permite mostrar sombras y luces estáticas en las propias texturas disminuyendo enormemente el costo de calcularlas.

(u, v): Son los vectores de superficie, los ejes u y v de dicha textura. Para calcular el valor de textura en un píxel determinado, es necesario utilizar la relación entre las coordenadas (s, t) de la textura en el plano y los vectores de superficie (u, v).

---

17. obtención de imágenes, recortado, proyección.

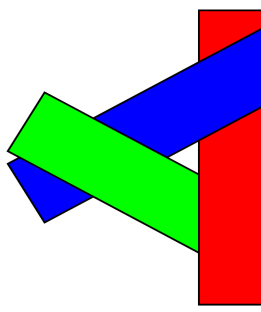
---

18. proyección de perspectiva, proyección ortogonal, realismo visual

Tipo de proyección que muestra los objetos cómo se verían en la vida real, con una sensación de profundidad, reduciendo el tamaño cuanto más lejos un cuerpo se encuentre. Esto permite aumentar el realismo visual provocando esa sensación de profundidad y distancia. Por otra parte, la perspectiva ortogonal, mantiene el tamaño de los cuerpos sin importar la distancia a la cámara, esta perspectiva es muy utilizada en arquitectura y diseño para mostrar los tamaños reales.

---

19. algoritmo del pintor, ordenación de los objetos



En el algoritmo del pintor, los objetos se pintan en el orden en el que llegan. En las implementaciones más básicas, el algoritmo del pintor puede ser poco eficiente, ya que fuerza al sistema a renderizar cada punto de todos los polígonos visibles, incluso si estos polígonos están ocultos en la escena final.

El algoritmo puede fallar en determinados casos. Por ejemplo, no se podría representar esta imagen puesto que están intersectadas en tres dimensiones.

---

20. modelos global y local de iluminación

Hay dos tipos principales de modelos de iluminación.

Locales: Luz específica de un objeto. La luz especular y difusa son luces locales. Se caracterizan por su velocidad.

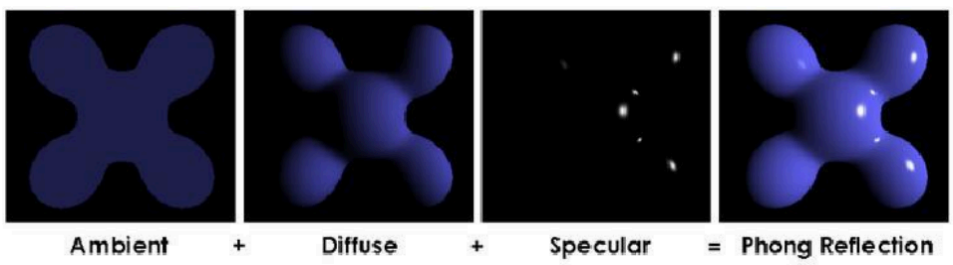
Globales: Consideran la escena en un conjunto, suelen ser lentos aunque sus resultados sean mucho mejores. El más conocido es el de radiosidad.

La luz ambiente es una luz global.

$$I = I_a \cdot K_a + I_L \cdot \left(K_d \cdot \left(\vec{N} \bullet \vec{L}\right) + K_S \cdot \left(\vec{R} \bullet \vec{V}\right)^n \right)$$ Ecuación de la luz

21. modelo iluminación de phong, eficiencia, tipos de sombras, color de un punto interior de una cara se calcula a partir de los vértices y aristas

El modelo de iluminación de Phong es una técnica de iluminación digital en tres dimensiones en el que la luz y sus efectos sobre los objetos se divide y simplifica en tres tipos: luz ambiente, luz difusa y luz especular.



- La luz ambiente no está asociada a ningún foco de luz ni a ninguna dirección. Provoca cambios uniformes en la iluminación de los objetos, sin generar sombras ni matices en el color.
- La luz difusa sí está asociada a un foco de luz y proviene de una dirección en particular. Se refleja en todas las direcciones. Su variación se asocia con cambios de intensidad o posición del foco. Genera sombras y matices de color en función del ángulo de incidencia.
- La luz especular también está asociada a un foco de luz proveniente de una dirección concreta, pero se refleja únicamente en un ángulo determinado. Su variación está también asociada a los cambios de intensidad y posición del foco, al igual que la luz difusa; pero, además, también es dependiente de la posición del observador. Con ella se obtienen efectos de brillo sobre las superficies.

---

22. Similitud y diferencias entre el sombreado de Gouraud y Phong

Ambas utilizan interpolación bilineal. Sin embargo, Gouraud calcula el color interpolando vértices mientras que Phong interpola para calcular la normal de cada punto, asociando así un color.

Phong da un mejor resultado que Gouraud, aunque necesita más recursos.

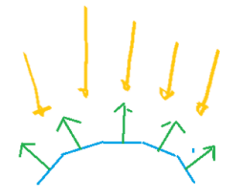
Normales /caras

- Una normal por cara, normales no unitarias.
- glEnable(GL_NORMALIZE)

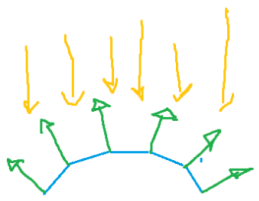
La normal se calcula a partir de los vértices y se normaliza o se puede especificar para un conjunto de ellos.

```
glNormal3fv(normals[1]);          // (1, 0, 0)
glVertex3fv(vertices[1]);
……..
glVertex3fv(vertices[5]);
glVertex3fv(vertices[6])
;
```

Iluminación plana (FLAT) glShadeModel (GL_FLAT);

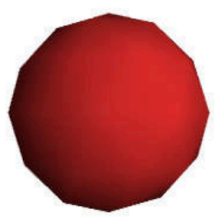


Sombreados de interpolación. Gouraud.



A cada vértice se la asigna una normal. y entonces promedia los valores de color en toda la superficie del objeto para así alcanzar un efecto de sombreado suave que parece más realista sobre superficies curvas.

```
glNormal3fv(Nx,Ny,Nz);
glVertex3fv(vertices[0]);
glNormal3fv(Nx,Ny,Nz);
glVertex3fv(vertices[2]);
glNormal3fv(Nx,Ny,Nz);
glVertex3fv(vertices[3]);
        //etc.
```



---

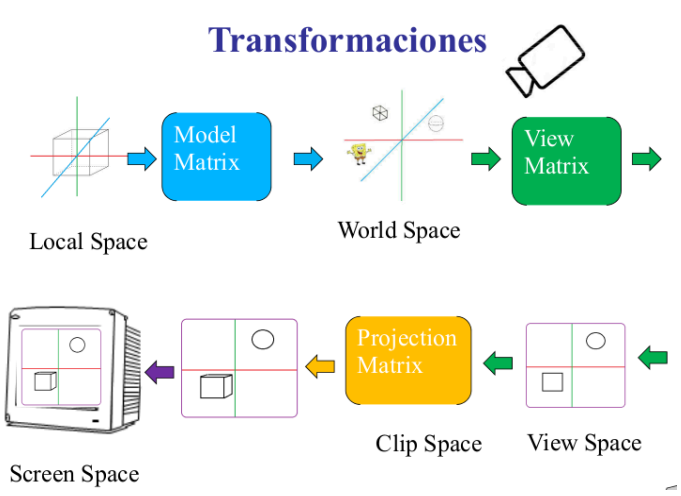
23. proceso de texturización a la vez que el de coloreado

---

24. transformaciones (traslaciones, rotaciones, escalados). ejercicios de explicar pasos

Local Space. Son Coordenadas locales de su objeto en relación con su origen local. Los vértices están referenciados a estas coordenadas

La matriz del modelo transforma las coordenadas locales en coordenadas de la escena (mundo).



En estas coordenadas es donde aparecen representados todos los objetos del World Space. Las coordenadas de los vértices tendrán unos nuevos valores.

Ahora, a través de la matriz de visión se obtiene el view space que es el espacio de visualización de la cámara que enfoca la escena. Esto se logra con una combinación de traslaciones y rotaciones. (colocar la cámara).

Finalmente, y mediante la matriz de proyección obtenemos el clip space que lo que hace es eliminar los elementos que estén fuera del frustum (espacio visible por la cámara). Por último, se convierte la imagen de 3D a 2D obteniendo así la imagen final que se observará por pantalla.

25. explicar el movimiento de un coche (teclas, transformaciones, escalado, rotaciones, estructura)

Tengo una bicicleta por piezas. Todos sus elementos están centrados en el origen y tienen un tamaño unitario. El cuadro tiene un tamaño 1,2 veces el unitario mientras que la rueda 0.6 veces. La rueda está situada 0.7 metros en el eje X respecto al origen del cuadro y gira a .3 radianes por segundo. Indica que instrucciones se utilizaría para dibujarla. El frame rate es de 1 f/s.

```
typedef struct {
	float x, y, z;
	float scaleX, scaleY, scaleZ;
	float rotacion, velocidad;
	int indice;
} Coche;

Coche coche;

void display() {
	glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);

	glPushMatrix();
		glTranslatef(coche.x, coche.y, coche.z);
		glRotatef(coche.rotacion, 0,1,0);
		glScalef(coche.scaleX, coche.scaleY, coche.scaleZ);
		glCallList(indice);
	glPopMatrix();
```

9

```
	glUseProgram(shader);

	glm::mat4 model;
	unsigned int modelLoc = glGetUniformLocation(shader, "model");

	model = glm::mat4();
	model = glm::translate(model,
		glm::vec3(coche.x * sin(coche.rotacion * M_PI/180), 0.0,
		coche.z * cos(coche.rotacion * M_PI/180)));
	model = glm::rotate(model, coche.rotacion, glm::vec3(0.0f, 1.0f,
	0.0f));
	model = glm::scale(model, glm::vec3(coche.scaleX, coche.scaleY,
	coche.scaleZ));

	glUniformMatrix4fv(modelLoc, 1, GL_FALSE, glm::value_ptr(model));
	glCallList(indice);

	glutSwapBuffers();
	glFlush();
}
```

```
void idle() {
	coche.x += coche.velocidad * sin(coche.rotacion * M_PI/180);
	coche.z += coche.velocidad * cos(coche.rotacion * M_PI/180);

	glutPostRedisplay();
}

void myTeclado(unsigned char tecla) {
	switch (tecla) {
		case 'w': coche.velocidad += 0.1;
		case 'a': coche.rotacion += 0.1;
		case 's': if (velocidad > 0) coche.velocidad -= 0.1;
		case 'd': coche.rotacion -= 0.1;
	}
}

void main() {
	glutInit(&argc, argv);
	glutInitWindowSize(500, 500);
	glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
	glutCreateWindow("Coche");

	glutDisplayFunc(display);
	glutIdleFunc(idle);
	glutKeyboardFunc(myTeclado);

	glutMainLoop();
}
```

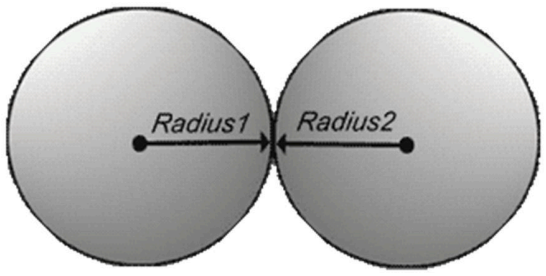
---

26. algoritmos de colisión

Para detectar colisiones se utilizan bounding boxes y esferas que engloban los objetos.

Detección de colisiones:

Esferas con esferas -> La distancia entre sus centros es menor que la suma de sus radios



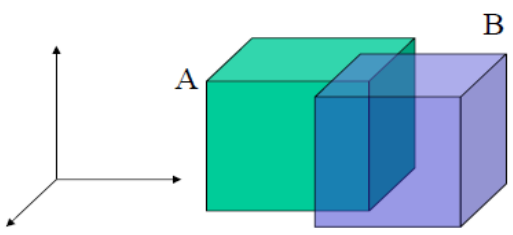
Caja con caja

Axis Aligned Bounding Boxes (cajas alineadas) : El más utilizado ya que es fácil, rápido y barato de calcular. Lo que hacemos es determinar si las esquinas de cada de sus ocho esquinas está dentro de la caja del otro.

Cada caja se caracteriza por dos puntos: el puntoMinimo y el puntoMaximo que marcarán los límites de dicha caja.

Por lo tanto, para comprobar si hay colisión se debe determinar si los puntos de una caja está dentro de la caja del otro:

AxMin < BxMax and AxMax > BxMin
AND
AyMin < ByMax and AyMax > ByMin
AND
AzMin < BzMax and AzMax > BzMin



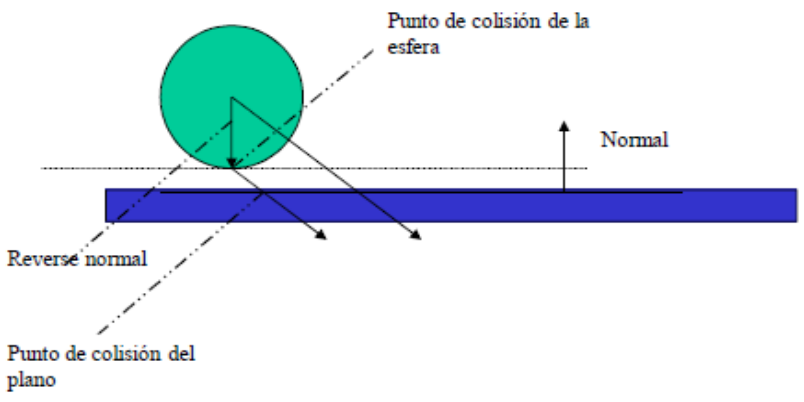
Test Ejes Separados (SAT) : para ejes no alineados.

Métodos generales: uso del trazado de rayos

Esfera-Plano: En este caso, la esfera se simplifica a un Rayo. Un rayo se representa con un vector que denota su punto de comienzo y su dirección y un vector velocidad que detalla cómo el rayo viaja.

Para calcular el punto de choque sobre la esfera se debe sumar al origen de la esfera la normal invertida del plano multiplicada por el radio de la esfera si esta no es de radio unidad.

El siguiente paso consiste en calcular el punto sobre el plano. Similar a cuando una recta (Rayo  corta un plano.



27. preguntas shaders

TEST 2021

En el examen de 2022 no hay tipo test pero lo pongo por si le cunde a alguien para repasar

1. La matriz de vista

    a. la determina la multiplicación de la matriz del modelo por la de proyección.

    b. por defecto es la matriz identidad.

    c. se calcula a partir del glLookat(.....).

    d. la determina el glOrtho(...) y el gluPrespective(...);

2. En un shader una variable uniforme

    a. Se utiliza para comunicar el programa principal con vertex shader pero no con el fragment shader.

    b. Su valor es constante en el shader.

    c. Se utiliza para comunicar el vertex shader con el fragment shader.

    d. Se utiliza para comunicar el program shaders con el programa principal.

3. Quiero hacer el google earth, ¿Que harías para aumentar la performance?

    a. Usar environment mapping ya que está diseñada específicamente para entornos naturales.

    b. Usar mipmapping ya que al generar niveles de detalle se verá mejor e irá más rápido.

    c. Usar bump mapping ya que al considerar las normales de cada fragmento y no por cara se verá mejor e irá más rápido.

    d. No usar luces y usar lightmapping para crear efectos de luz sin necesidad de generarlas.

4. El Vertex Array Object VAO.

    a. permite la utilización de vértices, complementando funciones tales como, glVertex3f(..),glNormal3f(…) o glTexture(..).

    b. Es un objeto de OpenGL que indica cómo la información de los vértices debe ser procesada.

    c. es un objeto de OpenGL que almacena la información para enlazar los vértices en el vertex shader.

    d. es un objeto OpenGL que almacena toda la información necesaria para el procesado de los vértices de un objeto.

5. El fragment shader se aplica.

    a. Después del ensamblado y antes del coloreado.

    b. Después del coloreado y antes del ensamblado.

    c. Después del coloreado y después del clipping.

    d. Antes del ensamblado y después del clipping.

6. glOrtho() y gluPerspective()

    a. Determina la posición de la cámara.

    b. Determinan la matriz de vista.

    c. Determinan la matriz de proyección.

    d. Determinan la matriz de vista y la de proyección

7. La última coordenada w en las coordenadas homogéneas.

    a. debe ser 0.

    b. puede tomar cualquier valor, pero se debe normalizar por el far/near.

    c. puede tomar cualquier valor, pero se debe homogeneizar.

    d. debe ser 1.

8. En el vertex shader puedo cambiar el color y las normales de los vértices de los objetos

    a. No

    b. Si

    c. Colores no, normales si.

    d. Colores si, normales no.

9. En general, ¿cuántos fragmentos son generados en una escena?

    a. Se puede calcular mediante la resolución de la ventana por la profundidad del color y por la profundidad del zbuffer.

    b. Uno por pixel.

    c. Más que píxeles.

    d. Uno o dos por pixel, dependiendo si tengo el Double buffer activado o no.

10. La matriz de proyección.

    a. Realiza la transformación de una escena para determinar el punto de vista del usuario.

    b. Determina la posición de la cámara y el frustum.

    c. Realiza la transformación del view space al clipping space.

    d. Realiza la transformación desde el word space al view space.

11. Si tenemos una figura definida en un Vertex Array Object cuyos vértices no están indexados, Los dibujo mediante

    a. glDrawArrays(.....);

    b. glDrawElements(.....);

    c. glVertex3f(.....);

    d. glCallList(.....);

12. Las tablas geométricas

    a. Incorporan toda la información necesaria para formar las caras de las superficies abiertas.

    b. Incorporan la información sobre los materiales de cada cara.

    c. Incorporan la información sobre los vértices, normales, coordenadas de textura, materiales.

    d. Incorporan la información sobre los vértices y los parámetros que proporcionan información sobre los polígonos o la superficie que determinan.

13. En el vertex shader para calcular la posición de un punto en Clip Space, la expresión es.

    a. gl_Position=model* view * projection * vec4(aPos, 1.0);

    b. gl_Position =view * model* projection * vec4(aPos, 1.0);

    c. gl_Position= view * projection * model* vec4(aPos, 1.0);

    d. gl_Position=projection * view * model* vec4(aPos, 1.0);

14. En el modelo de iluminación de phong es lento.

    a. Por el cálculo de las sombras entre los objetos.

    b. Por el cálculo de las interacciones entre los objetos.

    c. Por el cálculo complejo a realizar debido a la interpolación bilineal de las normales para el cálculo del color de los fragmentos.

    d. Por el cálculo de los rayos reflejados y transmitidos.

15. Mediante el mipmapping.

    a. Aumenta la velocidad/calidad de la visualización en función de la distancia al tener mayor número de fragmentos.

    b. Aumenta la calidad al utilizar texturas más grandes con un filtrado bilineal anisotrópico.

    c. Aumenta la velocidad/calidad de la visualización en función de la distancia al tener un menor número de fragmentos.

    d. Aumenta la velocidad/calidad de la visualización en función de la distancia.

16. En un terminal raster.

a. El haz de electrones recorre la pantalla siguiendo las instrucciones de la unidad de control realizando así el dibujo.

b. Representa las imágenes mediante una fórmula matemática.

c. El haz de electrones recorre ordenadamente la pantalla formada por filas y columnas realizando así el dibujo.

d. La imagen se escribe en el double buffer y luego se vuelca a la pantalla.

17. Quiero aplicar el modelo de sombreado de FLAT y varias caras comparten vértices indexados en tablas o VAO´s.

a. Al tener las mismas normales las caras se verían igual.

b. Las caras se verían diferentes debido a la luz ambiente.

c. Las caras se verían diferentes debido a la luz difusa y especular.

d. Sí, pero tengo que tener una tabla de materiales.

18. Si descartamos un fragmento.

a. El píxel que representaría aparece del color del fondo si no existe otro fragmento con valor de z-buffer mayor.

b. Aparece en negro.

c. El píxel que representaría el color del fondo.

d. Aparece del color con el que borramos el buffer de color.

19. Donde se aplica el modelo de iluminación.

a. Antes de la rasterización.

b. Cuando se aplica el z-buffer y Alpha test.

c. Después de la rasterización.

d. Al final de la pipe gráfica una vez se determina el fragmento que se visualizará.

20. Las transformaciones geométricas suponen un cambio en el sistema de coordenadas.

a. Verdadero transformamos el sistema de coordenadas Word Space al Local Space.

b. Falso, ya que solo se hacen translaciones, rotaciones y escalados.

c. Verdadero transformamos el sistema de coordenadas al de la cámara.

d. Verdadero transformamos el sistema de coordenadas Local Space al World Space.

21. Mediante el light mapping.

a. Utilizo una textura con las componentes ambiente, difusa y especular de cada punto y así aplicar el modelo de phong.

b. Se crea una textura con el brillo/luces de una superficie para posteriormente aplicar la en los objetos generando efectos lumínicos.

c. Creo una textura con los valores de las normales de una superficie y así puedo generar los efectos lumínicos.

d. Se crea una textura con el brillo/luces de una superficie para posteriormente aplicar a objetos en movimiento simplificando el proceso de iluminación dinámica.

22. Si tenemos una figura definida en un Vertex Array Object cuyos vértices están indexados. Los dibujo mediante

a. glVertex3f(.....);

b. glDrawElements(.....);

c. glDrawArrays(.....);

d. glPolygonMode(.....);

23. En una proyección ortográfica.

a. Los objetos no cambian su tamaño si no habilitas el z-buffer.

b. Los objetos no cambian su tamaño al ser los proyectores paralelos.

c. Los objetos no cambian su tamaño porque el frustum es cuadrado.

d. Los objetos si cambian su tamaño al existir un centro de proyección.

24. El Clip Space es

a. La región del espacio que se sitúa dentro del frustum.

b. La región del espacio donde se transforma el world space.

c. El espacio que se recorta para hacerlo coincidir con la ventana.

d. La región del espacio vista desde la cámara.

25. Puedo utilizar el mismo vertex shader en dos program shaders

a. No.

b. Si.

c. Si, pero debo cambiar el nombre de las variables uniformes.

d. Sí, pero debo cambiar el nombre de las variables in y out.

QUIZ: https://take.quiz-maker.com/QJRVDWDA9
QUIZ(Editable): https://www.quiz-maker.com/#QP-2996549x2f79AF1a-138