

Resumen IDI

Contacto:

Borja Fernández Ruizdelgado

Email: borjafruízdelgado@gmail.com

Telef: +34 635127575

Índice de contenidos

| | |
|---|-----------|
| Modelo geométrico 1 | 3 |
| Modelo geométrico 2: Escenas | 4 |
| Transformaciones geométricas | 5 |
| Proceso de visualización 1 | 6 |
| Ópticas de cámara: perspectiva | 7 |
| Definiendo cámara en 3a persona | 9 |
| Ángulos de Euler | 10 |
| Óptica axonométrica | 11 |
| Vista en 3a persona axonométrica | 12 |
| Zoom | 13 |
| Proceso de pintado OpenGL | 14 |
| Realismo | 15 |
| Back-face Culling | 15 |
| Depth Buffer | 15 |
| Iluminación | 16 |
| Modelos de iluminación | 16 |
| Locales o empíricos | 16 |
| Modelos globales | 17 |
| Proceso de visualización: VS vs FS | 18 |
| Introducción a la HCI | 20 |
| Algunas definiciones: | 20 |
| HCI y UX para dispositivos móviles: | 20 |
| Aplicaciones móviles: | 21 |
| Color | 22 |
| Principios de diseño | 25 |
| Qué tener en cuenta a la hora de diseñar una interfaz. | 28 |
| Las 7 leyes de Gestalt | 31 |
| Interacción diseño y evaluación | 32 |
| Percepción y color | 36 |
| Teclados: | 37 |

| | |
|--|-----------|
| Realidad virtual | 38 |
| Realidad aumentada | 38 |
| Tests de usabilidad | 40 |
| Métodos cuantitativos para experimentos humanos | 43 |

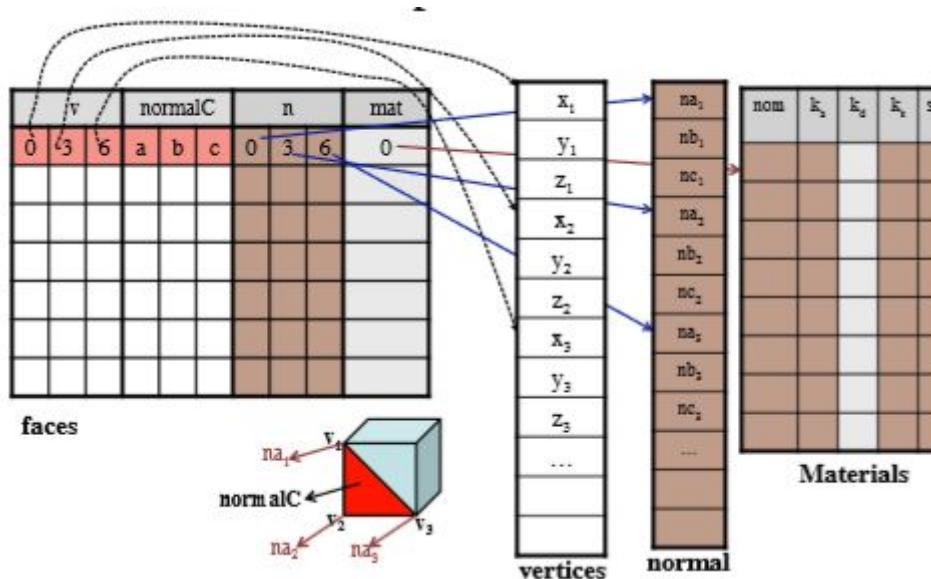
Modelo geométrico 1

Varios tipos de modelos:

1. Modelo de fronteras → aproximación por caras planas (triángulos) es el que utilizamos en IDI.
2. Procedural, GSG, etc.

Los modelos 3D están formados por triángulos. **¿Cómo guardamos los triángulos?**

1. Guardaremos los modelos por caras y estas se relacionan con sus tres vértices.



2. Validación de un modelo de fronteras:
 - a. Todas las caras han de estar orientadas, es decir, todas las caras han de tener una normal que indica hacia donde mira.
 - b. La orientación de los vértices ha de ser coherente con la ordenación de las caras.
 - c. Toda arista separa dos caras. (Modelos cerrados)

En IDI utilizaremos OpenGL para visualizar modelos 3D. Para su visualización necesitaremos:

1. Crear un **VAO** → Elemento que contiene los datos del modelo. Se aloja en la memoria de la GPU.
2. Crear un **VBO** → guarda las coordenadas de los vértices. Puede que necesitemos varios para las normales, colores, iluminación, etc.
3. Cada vez que queramos pintar un modelo hay que indicar que VAO se quiere pintar, en teoría se simplifica utilizando `pinta_model()`;

Modelo geométrico 2: Escenas

1. Una escena es un conjunto de modelos que hay que pintar. Tendremos varios VAO, uno para cada modelo diferente y VBOs que corresponderán a estos VAO. Hay que tener en cuenta que no es lo mismo un objeto de la escena que un modelo. Un VAO se puede pintar varias veces para replicar el modelo a lo largo de la escena, cada réplica del modelo es lo que llamaremos un objeto.
2. Por cada modelo solo hemos de rellenar el VAO **una** vez.
3. Cada objeto de la escena ha de ser pintado cuando queramos mostrar cambios en la misma.

Transformaciones geométricas

Para poder cambiar la posición de los modelos 3D les aplicaremos transformaciones geométricas a sus vértices.

Tipos de TGs:

1. Traslación → **T(tx,ty,tz)** desplaza los vértices acorde al vector especificado.
2. Rotación → **Gx,y,x(ángulo)** rota el modelo el ángulo especificado en relación al **origen**; x,y,z especifican en qué eje se ha de rotar. Para saber si los grados han de ser positivos o negativos se utiliza la regla de la mano derecha, situando el pulgar en el eje de giro. Rotar la mano en sentido antihorario nos indica los grados positivos y viceversa los negativos.
3. Escalado → **S(sx,sy,sz)** escala el modelo en relación al **origen**. Se indica el factor de escalado en cada eje que deseamos, no el tamaño final que tendrá el modelo. Si queremos escalar el modelo sin perder su posición es recomendable primero llevarlo al origen de coordenadas, escalar y luego llevarlo a la posición deseada o aplicarle las TG que queramos.



Es importante recordar que las TG se pueden acumular multiplicándose entre sí y que se han de multiplicar en **orden inverso** al que las queremos aplicar. Ej: si queremos primero escalar y luego transformar multiplicaremos la de transformar por la de escalar.

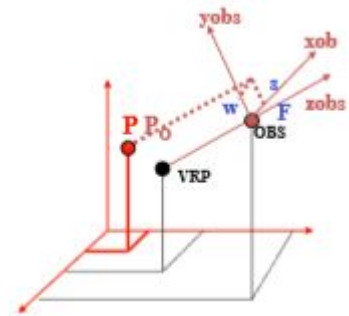
¿Cómo aplicamos las TGs? → se coge la TG y se multiplica por cada vértice que tiene el modelo: **TG*verticeEnQuestion**. Este proceso se aplica en el Vertex Shader.

Proceso de visualización 1

Para poder visualizar escenas necesitamos pasar del mundo de la aplicación a una pantalla; para ello declararemos una cámara.

Esencial para definir una cámara:

1. **OBS** (observador) → punto donde se sitúa la cámara.
2. **Vector UP** → vector que define la vertical de la cámara.
**Importante no poner ese vector paralelo a la dirección de visualización de la cámara.
3. **VRP** → punto donde mira la cámara.
4. $\mathbf{F} = \text{OBS} - \text{VRP} \Rightarrow \mathbf{F} = \mathbf{F}/|\mathbf{F}|$
5. $\mathbf{s} = \text{up} * \mathbf{F} \Rightarrow \mathbf{s} = \mathbf{s}/|\mathbf{s}|$
6. $\mathbf{w} = \mathbf{F} * \mathbf{s}$



Con estos parámetros se define la view matrix (VM) de la siguiente manera:

$$VM = \begin{bmatrix} s.x & s.y & s.z & 0 \\ w.x & w.y & w.z & 0 \\ F.x & F.y & F.z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \text{Trans}(-\text{OBS})$$

Otra opción es llamar a la función **lookAt(OBS,VRP,UP)** que nos hará el cálculo de la VM.

¿Cómo aplicamos la VM? → Al igual que la TG en el Vertex Shader de la siguiente manera: $VM * TG * \text{verticeEnQuestion}$.

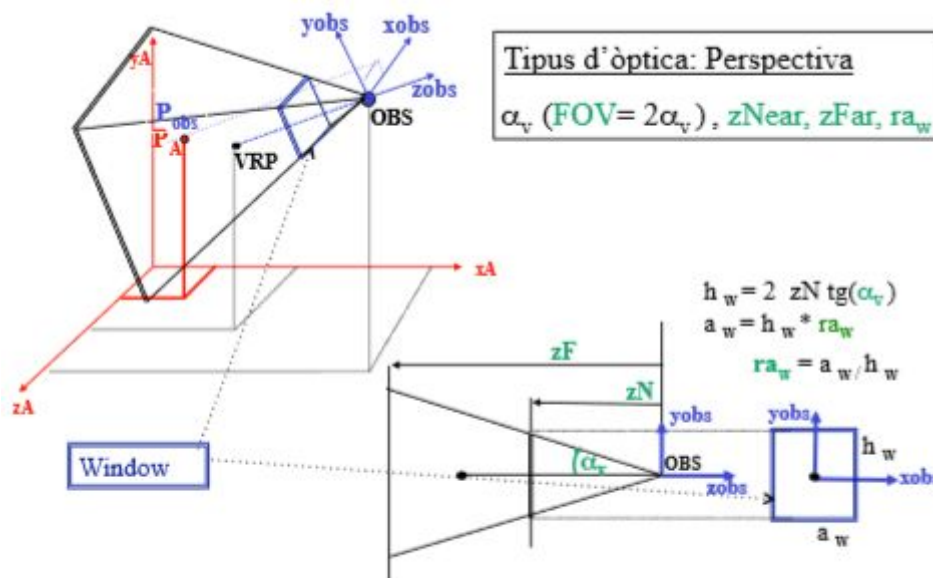
Ópticas de cámara: perspectiva

La óptica de cámara determina el volumen de visión que tendremos con la cámara.

El volumen de visión es definido en relación al observador.

Parámetros necesarios para la óptica perspectiva:

1. FOV (Field Of View) $\rightarrow 2\alpha_v$
2. zNear \rightarrow distancia entre OBS y window.
3. zFar \rightarrow límite de renderizado.
4. raw (relación de aspecto del window) $\rightarrow 4:3, 16:9, \text{ etc.}$



Con estos parámetros se define lo que se llama como Projection Matrix, calculada de la siguiente manera:

$$PM = \begin{pmatrix} 1/ra \cdot a & 0 & 0 & 0 \\ 0 & 1/a & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & -1 & 0 \end{pmatrix} \quad \begin{aligned} a &= \tan(FOV/2) \\ c &= (f+n)/(n-f) \\ d &= 2nf/(n-f) \end{aligned}$$

¿Cómo aplicamos la PM? \rightarrow se aplica en el Vertex shader de la siguiente manera:
 $PM \cdot VM \cdot \text{vertexEnQuestion}$.

En el momento en que empezamos a jugar con el tamaño del viewport se producen deformaciones esta es la manera de **evitar las deformaciones**:

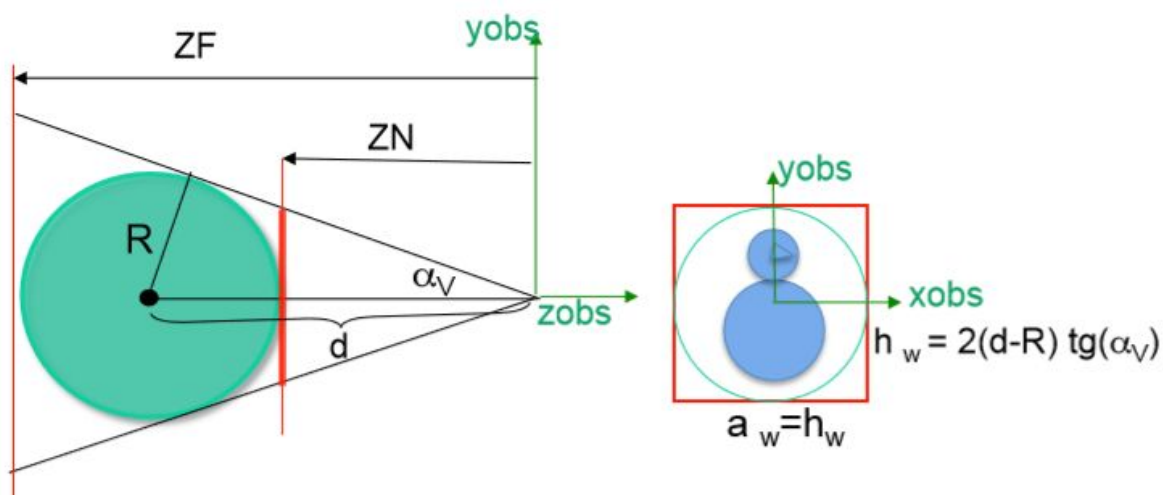
1. Hay que conseguir que la relación del viewport y el window coincidan. Como el viewport es definido por el pc sólo podemos tocar el widow en cuanto al dominio de la aplicación se refiere.
2. Evitar deformaciones 2 casos:
 - a. Si $rav < 1$ --> $FOV = 2a*v$ dónde $a*v = \arctan(\tan(av)/ra)$ $ra*w = rav$
 - b. Si $rav > 1$ --> $ra*w = rav$

Definiendo cámara en 3a persona

Que una cámara esté en 3a persona significa que toda la escena se ve sin recorte alguno.

Pasos para conseguir una cámara en 3a persona:

1. Hay que encontrar el **centro de la escena** y situar el **VRP** allí.
2. OBS ha de situarse fuera del volumen de la escena.
3. CajaContenedora → definida por los puntos máximo y mínimo de la escena $P_{min}(x_{min}, y_{min}, z_{min})$ y $P_{max}(x_{max}, y_{max}, z_{max})$.
4. CentroEscena → Centro(CajaCont) → $((x_{min} + x_{max})/2, (y_{min} + y_{max})/2, (z_{min} + z_{max})/2)$
5. Radio → $R = \text{dist}(P_{min}, P_{max})/2$
6. Distancia entre VRP y OBS → d por ejemplo puede ser $2R$
7. $OBS = VRP + x_{veces} * d * v$ donde v = vector unitario de dirección.



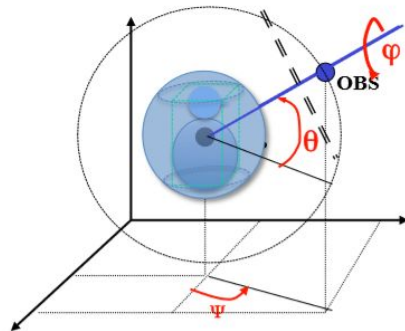
8. $\alpha_v = \arcsin(R/d)$
9. $R = d * \sin(\alpha_v)$
10. $FOV = 2 * \alpha_v$
11. Si deseamos que el window no sea cuadrado $a_w = r_a * h_w$

Ángulos de Euler

Los ángulos de Euler nos permiten calcular la VM de manera directa y se usa generalmente para cámaras en 3a persona. Es como mover toda la escena al origen y a partir de ahí aplicar transformaciones geométricas para visualizarlo como deseamos. Esto es posible ya que OpenGL por defecto la cámara se sitúa en el origen de coordenadas con el vector up siguiendo las Y positivas y mirando hacia las Z negativas.

¿Como calcular ángulos de Euler?:

1. $T(-VRP)$ --> Mover la escena al origen
2. $Gy(-\psi)$ --> Rotación en el eje de las Y
3. $Gx(-\theta)$ --> Rotación en el eje de las X
4. $T(0,0,-d)$ --> Movemos la escena a distancia d

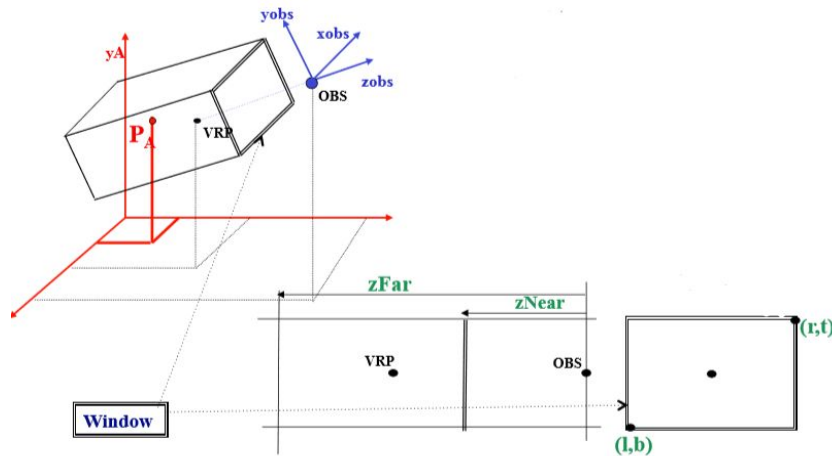


$$VM = T(0,0,-d) * Gx(-\theta) * Gy(-\psi) * T(-VRP)$$

**También podemos incluir cabeceo entre $T(0,0,d)$ y $Gx(-\theta)$ se verá similar a como si inclinamos la cabeza lateralmente.

Óptica axonométrica

1. Óptica axonométrica **no es realista** → distancia de los objetos **no** influye a la hora de representar su tamaño en pantalla.
2. Elementos necesarios para representar óptica axonométrica:
 - a. Window → l,r,b,t
 - b. zNear
 - c. zFar
 - d. $heightW = t-b$
 - e. $widthW = r-l$
 - f. $raW = widthW/heightW$



Vista en 3a persona axonométrica

Parámetros necesarios para definir la vista en 3a persona axonométrica:

1. $zNear$ y $zFar$ → Mismo razonamiento que en perspectiva. Véase apartado:
Definiendo cámara en 3a persona.
2. Window min $\rightarrow (-R, -R, R, R) \rightarrow raw = 1$; de esta manera hacemos coincidir exactamente la ventana con el total de la esfera (recordamos que R es el radio de la esfera).
3. Resize sin deformación:
 - a. Si $raw > 1 \rightarrow$ modificar el window $a*w = raw*hw = raw*2R \Rightarrow inc_a = a*w - aw$
 $\Rightarrow window = (-R*raw, R*raw, -R, R)$
 - b. Si $raw < 1$ se aplica un razonamiento similar al anterior y entonces window = $(-R, R, -R/raw, R/raw)$

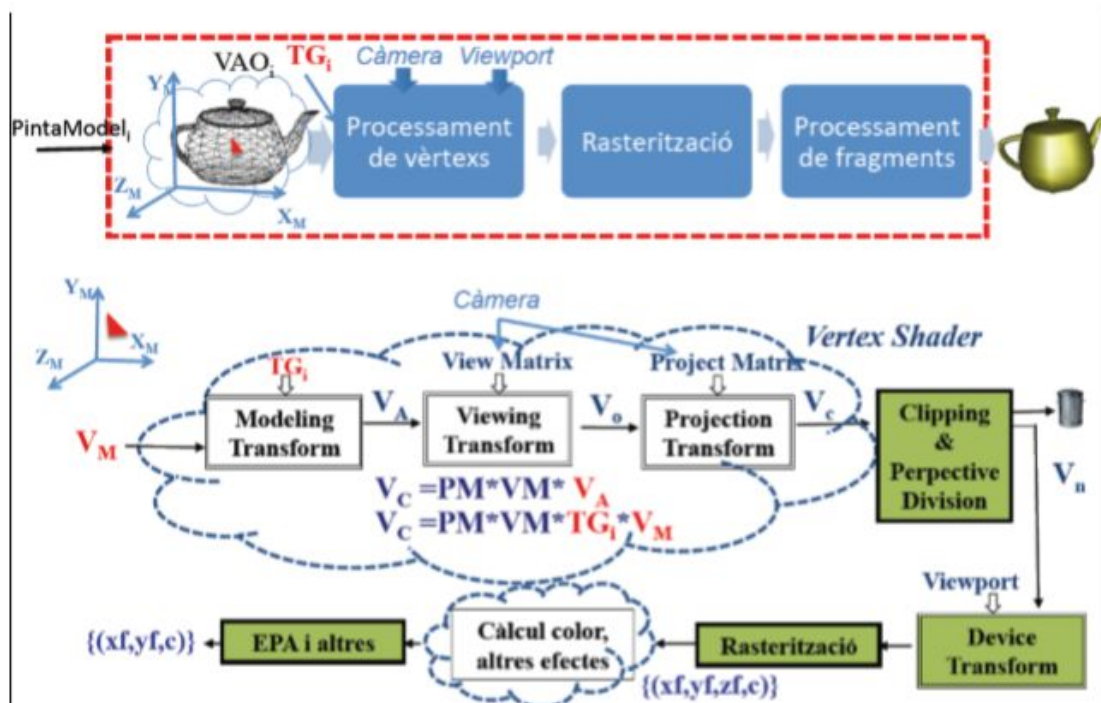
Zoom

Hay dos maneras de conseguir un efecto de zoom a la hora de visualizar una escena dependiendo del tipo de óptica:

1. Zoom con perspectiva: reducir el ángulo del FOV (similar que lo que hacen los objetivos de las cámaras de fotografía actual).
2. Zoom con axonométrica: Tan simple como escalar el window de manera regular.

Proceso de pintado OpenGL

Breve repaso gráfico de cómo funciona el proceso de visualización de OpenGL:



Es importante saberse el esquema para el examen puesto que suele salir a menudo.

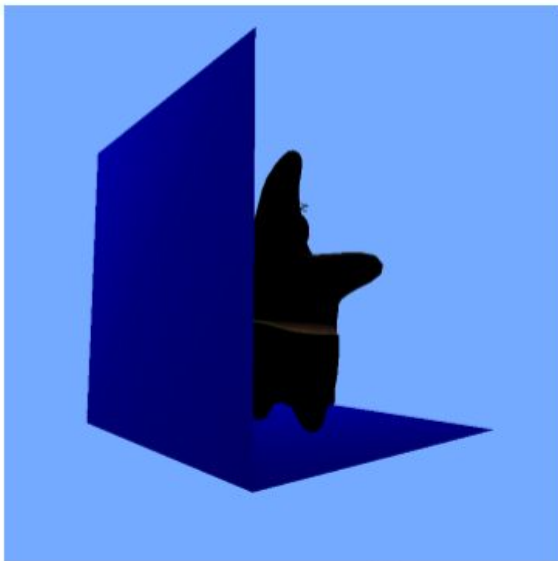
Notad que lo único que hemos de programar nosotros son los shaders, primero el vertex shader (primera “nube”) y luego el fragment shader (segunda “nube”).

Realismo

1. Back-face Culling

- a. Elimina partes ocultas a nivel de **triángulo**.
- b. Necesita tener las caras orientadas (**normal por cara**).
- c. Considera la escena formada por la cara y el observador.
- d. Elimina las caras que seguro no son visibles.
- e. OpenGL hace el cálculo en coordenadas de dispositivo, calculando las normales de cara a partir de los 3 vértices.

Sense culling



Amb culling



En resumen, las caras que dan la espalda al observador no se pintan.

2. Depth Buffer

- a. EPA en espacio de imagen (nivel de píxel/fragmento).
- b. Se aplica después del raster.
- c. No requiere de back-face para funcionar.
- d. Necesita la profundidad del píxel respecto al observador para funcionar.
- e. No importa el orden de pintado de los triángulos.

Iluminación

- Color que llega a un observador desde un punto, este depende de: Fuentes de luz, Materiales, Otros objetos, Posición del observador, Medio de propagación.

Modelos de iluminación

Locales o empíricos

1. Solo tienen en cuenta un punto P, los focos de luz y la posición del observador.
2. No tienen en cuenta otros objetos y sus propiedades.
3. **Tipos de modelos empíricos:**

a. Ambiente:

- i. Focos de luz no considerados.
- ii. Todos los puntos reciben la misma luz.

Equació: $I_{\lambda}(P) = I_{a\lambda} k_{a\lambda}$

- $I_{a\lambda}$: color de la llum ambient
- $k_{a\lambda}$: coef. de reflexió ambient

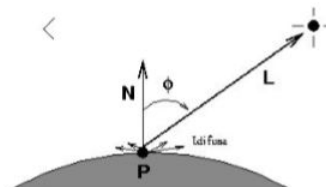


b. Empírico difuso (Lambert):

- i. Focos de luz puntuales considerados.
- ii. Color no depende del ángulo de visión.

$$I_{\lambda}(P) = I_{f\lambda} k_{d\lambda} \cos(\Phi)$$

$si |\Phi| < 90^\circ$



- $I_{f\lambda}$: color (r,g,b) de la llum del focus puntual f
- $k_{d\lambda}$: coef. de reflexió difusa del material
- $\cos(\Phi)$: cosinus de l'angle entre la llum incident i la normal a la superfície en el punt P



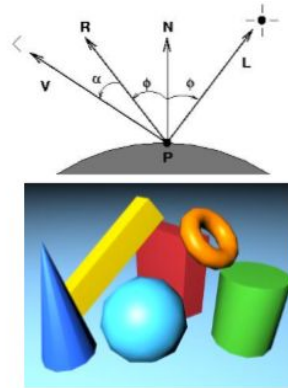
c. Especular (Phong):

- i. Tiene en cuenta focos de luz puntuales y reflexión especular de objetos.
- ii. Dirección de la especularidad es simétrica de L respecto a N. $R = 2N(N \cdot L) - L$ si todos los vértices están normalizados.

$$I_{\lambda}(P) = I_{f\lambda} k_{s\lambda} \cos^n(\alpha)$$

si $|\Phi| < 90^\circ$

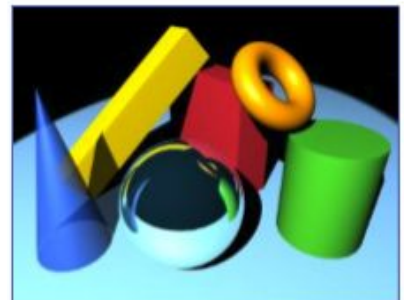
- $I_{f\lambda}$: color (r,g,b) del focus puntual f
- $k_{s\lambda}$: coef. de reflexió especular (x,x,x)
- n : exponent de reflexió especular



****Ecuación resultante:** $I_{\lambda}(P) = I_{a\lambda} k_{a\lambda} + \sum_i (I_{f_i\lambda} k_{d\lambda} \cos(\Phi_i)) + \sum_i (I_{f_i\lambda} k_{s\lambda} \cos^n(\alpha_i))$

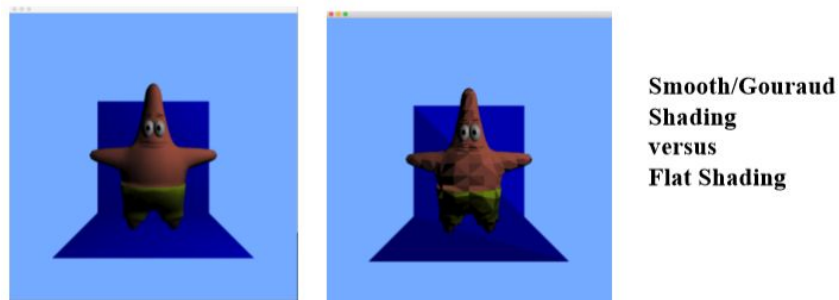
Modelos globales

1. **Trazado de rayo:** consideran focos de luz puntuales y otros objetos existentes en la escena pero solo las transmisiones especulares. Simulan sombras, transparencias y espejos. Más costosos.
2. **Modelos de radiosidad:** El foco de luz es un objeto cualquiera de la escena. Los objetos producen reflexiones difusas puras. La radiosidad no depende del observador. Pueden modelar sombras y penumbras pero no espejos ni transparencias. Son los más costosos.



Proceso de visualización: VS vs FS

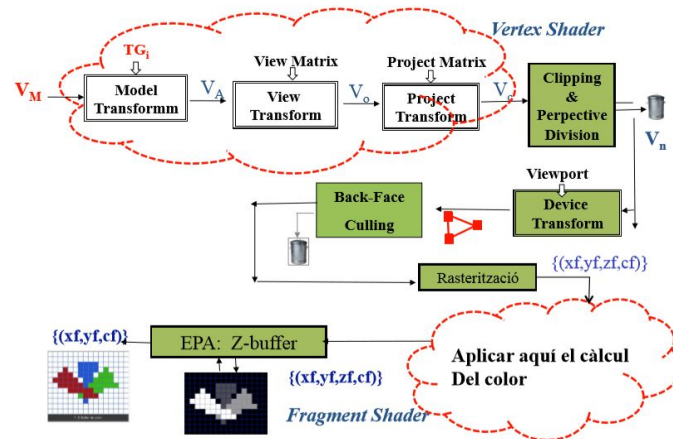
1. En cuanto a iluminación se refiere, utilizar el VS es mucho más eficiente pero si nuestro modelo tiene pocos triángulos veremos una iluminación muy pobre, sobre todo en cuanto al modelo Phong se refiere puesto que la mancha especular desaparece y se aplicará sólo sobre un vértice si pertoca.. En el FS por contraparte tendremos una iluminación más rica pero que requiere de más potencia de cálculo.
2. El cálculo del shading en el VS provoca un suavizado del modelo “perdiendo definición” por así decirlo.



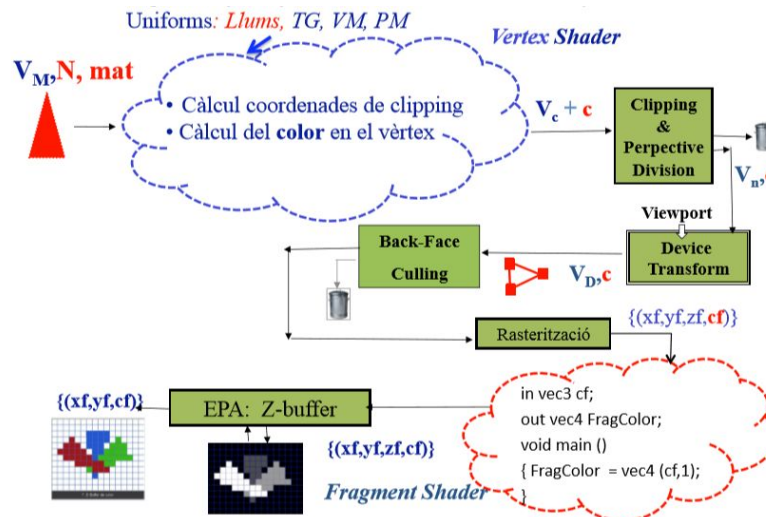
3. Suavizado de aristas: mejor tener normales por vértices que por caras puesto que nos permite tener una mayor definición.

4. Cómo se computa la iluminación en FS vs VS:

Càlcul il·luminació en FS



Càlculo en VS



Introducción a la HCI

Algunas definiciones:

1. **HCI:** Human Computer Interaction es un campo que trata con el estudio de cómo los humanos interactuamos con las máquinas. Una app tiene que cumplir sus requisitos y proveer acceso fácil a sus características.
2. **Eficacia:** capacidad de conseguir un objetivo de manera correcta y completa.
3. **Eficiencia:** relación entre recursos y la exactitud y lo completo de los objetivos conseguidos.
4. **Satisfacción:** comodidad y aceptación de un sistema por los usuarios y otras personas afectadas por su uso.
5. **Experiencia de usuario (UX):** Es inherente a cómo el usuario se siente y qué recuerda del uso de un dispositivo.

HCI y UX para dispositivos móviles:

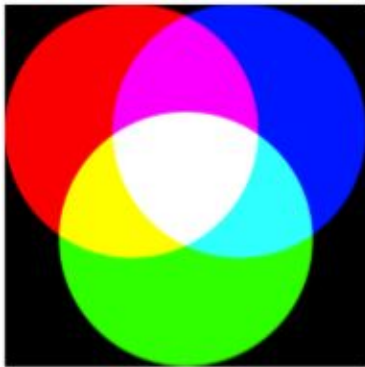

1. En cuanto al alcance de nuestro pulgar en relación a la pantalla se utiliza lo que denominamos como regla del pulgar. En la imagen vemos que zonas son alcanzables y cuáles no para una mano de tamaño medio.
2. Hay que tener en cuenta a la hora de diseñar una interfaz que: tamaños de pantalla reducidos, por lo tanto necesitaremos planificar bien donde situamos cada cosa, teclado pequeño y limitaciones SW.



Aplicaciones móviles:

1. Dos maneras de desarrollar:
 - a. Web apps: se desarrolla una vez y es accesible por cualquier sistema con buscador. Fácil de actualizar. Por contraparte no tienen una interfaz tan rica, el protocolo de comunicación es inseguro y suelen estar enfocadas a dispositivos con pantallas grandes.
 - b. Aplicaciones nativas para el SO: UI rica, más seguro y de rápido acceso, diseñada para tamaños de pantalla pequeños. No tienen acceso universal, es más difícil sacar actualizaciones.

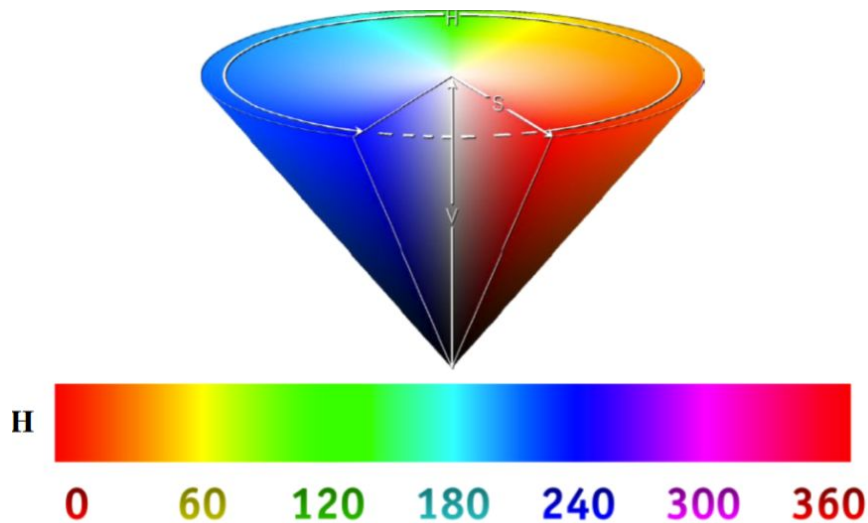
Color

| | Aditivo | Sustractivo |
|------------------------------|--|--|
| Aplicado en | Luz | Papel |
| Color base | Negro | Blanco |
| Esquema de colores | RGB | CMY(K) |
| Esquema visual |  |  |
| | R-G-B | C-M-Y |
| Blanco | 1-1-1 | 0-0-0 |
| Negro | 0-0-0 | 1-1-1 |
| Rojo | 1-0-0 | 0-1-1 |
| Verde | 0-1-0 | 1-0-1 |
| Blue | 0-0-1 | 1-1-0 |
| Cian | 0-1-1 | 1-0-0 |
| Magenta | 1-0-1 | 0-1-0 |
| Amarillo | 1-1-0 | 0-0-1 |
| Conversión RGB a CMY | $(A,B,C) \rightarrow (A-1,B-1,C-1)$ | |
| Conversión RGB a CMYK | $(A,B,C) \rightarrow (1-R-K, 1-G-K, 1-B-K, \min(1-R, 1-G, 1-B)*s/100)$ | |

** la K en CMY(K) viene a raíz de que los colores de las impresoras no son perfectas y se añade un tanto de negro para compensar.

HSV (Hue, Saturation, Intensity(Value))

1. **Hue:** color es un ángulo de 0° a 360°.
2. **Saturación:** Mide el % de gris que se añade, de 0% a 100%(color puro).
3. **Intensidad:** el brillo del color.



Conversiones:

RGB to HSV (VIG):

max = maximum of RGB

min = minimum of RGB

V = max

S = (max - min) / max

if S = 0, H is undefined, else

delta = max-min

if R = max, H = (G-b)/delta

if G = max, H = 2 + (B-R)/delta

if B = max, H = 4 + (R-G)/delta

H = H*60

if H < 0, H = H + 360

HSV to RGB

if S = 0 and H = undefined, R = G = B = V

if H = 360, H = 0

H = H / 60

i = floor(H)

f = H - i

p = V*(1-S)

q = V*(1-(S*f))

t = V*(1 - (S * (1-f)))

if i = 0, R = v, G = t, B = p

if i = 1, R = q, G = v, B = p

if i = 2, R = p, G = v, B = t

if i = 3, R = p, G = q, B = v

if i = 4, R = t, G = p, B = v

if i = 5, R = v, G = p, B = q

Principios de diseño

1. **Estética:** Tener una estética agradable no ha de interponerse a la usabilidad.
2. **Anticipación:** Dar al usuario la información y herramientas necesarias para cada paso del proceso. Si el usuario no puede encontrar una característica de la app nunca la va a utilizar. Si la experiencia es frustrante podemos llegar a perder al usuario.
3. **Autonomía:**
 - a. Hay que dejar que el usuario tome sus propias decisiones. No significa que no haya reglas en la app.
 - b. Hay que proveer al usuario con información sobre el estado y procesos actuales.
 - c. Hay que mostrar de manera precisa el tiempo que tardarán los procesos (Ej: load bar).
4. **Color:** hay que tener en cuenta las posibles discapacidades visuales de nuestros usuarios, intentando no coger colores que pueden producir confusión.
5. **Consistencia:**
 - a. **Plataforma:** Hay que mantener un look general a través de los productos de una compañía entre plataformas con el fin de dar imagen de marca y facilidad de adoptar un producto nuevo por parte de un usuario.
 - b. **Suite de productos:** Si una misma compañía tiene varios productos es buena praxis tener un look y comportamiento similar (Ej: Google apps, Microsoft Office).
 - c. **In-app:**
 - d. **Estructuras visibles:** iconos, símbolos, posicionamiento similar.
 - e. **Estructuras no visibles:** Si se utilizan estructuras invisibles, lease scrollbars que aparecen al hacer una acción o similar, utilizarlas siempre de la misma manera. Su uso puede ser difícil para el usuario de primeras.
 - f. **Comportamiento del usuario:** Nunca cambiar el significado de una acción habitual para un usuario.

g. Tipos de consistencia:

- i. **Inconsistencia inducida:** Hacer objetos diferentes si actúan diferente. Si una app cambia sustancialmente se puede cambiar completamente el diseño para reforzar ese hecho.
- ii. **Continuidad inducida:** Con el paso del tiempo hay que conseguir una buena continuidad no consistencia puesto que nuevas releases pueden introducir cambios grandes que hagan difícil seguir con la consistencia.
- iii. **Consistencia con expectativas del usuario:** Si los usuarios esperan que algo se haga de una determinada manera no hay que forzarlos a aprender una manera nueva a no ser que ofrezca una clara ventaja.

6. Valores por defecto: Han de ser fáciles de escribir, no todos los campos los requieren, si no hay una ventaja al usarlos, mejor no hacerlo.

7. Descubribilidad:

- a. Intentar bajar complejidad puede incrementarla.
- b. Si el usuario no lo puede encontrar no existe.
- c. Descubribilidad activa: enseñar al usuario las características de la app (Ej: tutorial).
- d. Todos los controles deben estar visibles.

8. Eficiencia:

- a. Hay que evaluar la productividad del usuario no la de la máquina.
- b. Mantener al usuario ocupado lo máximo posible, evitar que pierda la concentración.
- c. Muchas pérdidas de eficiencia vienen dadas por cambios arquitecturales.
- d. Los mensajes de error han de ayudar, explicando que ha sucedido, no solo mostrando un código de error.

9. Interfaces explorables: Hay que dejar a los usuarios explorar la interfaz, para ello las acciones han de ser reversibles, permitir “deshacer”.

10. Ley de Fitts: El tiempo que tardamos en llegar a un objetivo es la función de la distancia al objetivo y el tamaño del objetivo. Objetos grandes para funciones importantes y reducir el número de objetivos en pantalla, no solo la distancia.

11. Objetos de interfaz humana: Si se usan metáforas de interfaz humana, deberían verse y actuar como sus contrapartes reales (Sliders se deslizan, botones se aprietan, etc.).

12. Informar usuarios: hay que informar a los usuarios si hay algún tipo de espera.

13. Usabilidad:

- a. Elige metáforas que permitan a los usuarios captar al instante los detalles más finos del modelo conceptual.
- b. Asegurar que el usuario no pierde su trabajo.
- c. El texto legible debe tener alto contraste con el fondo para que se lea de manera fácil.

Qué tener en cuenta a la hora de diseñar una interfaz.

1. **Ley del 80/20:** Aproximadamente el 80% de los efectos de un sistema grande son generados por el 20% de las variables.
2. **Efecto Estético-Usable:** Los diseños estéticos son importantes y parecen más fáciles de usar a la par que animan a ser utilizados. No hay que permitir que la estética de un diseño afecte a la usabilidad.
3. **Correcto alineamiento:** Los elementos de una lista han de estar bien alineados ya que provoca una cohesión entre ellos y no induce a errores de selección.
4. **Chunking:** Un “Chunk” es un pedazo de información en la memoria a corto plazo. Este método busca la manera de colocar la información a memorizar.
 - Los trozos pequeños son más fáciles de memorizar.
 - El número de trozos ha de ser de 5+-2.
5. **Colores:**
 - a. Usar hasta 5 colores.
 - b. Combinaciones de colores: análogos, complementarios, combinaciones de colores encontrados en la naturaleza.
 - c. Saturación: a más saturación sugiere más dinamismo y excitación, a menos saturación se ve más profesional.
 - d. Hay que tener en cuenta que en diversas culturas los colores pueden representar una u otra cosa.
6. **Principio de LATCH:**
 - a. La organización de la información ha de depender de:
 - i. Localización.
 - ii. Alfabeto.
 - iii. Categoría.
 - iv. Jerarquía.
7. **Garbage-in, garbage-out:**
 - a. Si el resultado de una entrada es erróneo este producirá malos resultados.
 - b. Errores:
 - i. Escritura: si una mala entrada no es detectada puede generar mucha basura. Hay que detectar y comprobar el formato de entrada.

- ii. **Error cualitativo:** Escrito correctamente pero con algunos defectos. Con el fin de evitarlos en gran medida hay que pedir al usuario confirmaciones y darle una preview.
- 8. Iconografía:** Hay muchos tipos de iconos entre ellos: los que son similares a la acción que representan, relatados en una imagen, simbólicos y arbitrarios que describen algo por consenso o similar (peligro nuclear).
- 9. Anticipación:** Hay que dar al usuario las herramientas necesarias en cada paso. Anticiparse a sus necesidades. Información en el sitio y visible, ya que si no la ve no la va a utilizar. Un fallo en la anticipación puede provocar que perdamos el usuario.
- 10. Autonomía:** Hay que dejar que el usuario personalice y tome sus propias decisiones. Si permitimos al usuario tomar decisiones hay que mantenerlo informado en todo momento para el paso en el que se encuentra. Hay que mantener el timing de la app -> mostrar notificaciones a destiempo o mentir al usuario es contraproducente.
- 11. Colores:**
 - a. Color friendly:** evitar utilizar colores adyacentes, contraste oscuro contra colores de fondo claros. Hay que utilizar colores que prevengan o faciliten el uso a daltónicos y personas con déficit de visión.
 - b. Reglas de diseño con color:**
 - i. Si deseamos que dos objetos de un mismo color parezcan del mismo hay que ponerlos sobre un fondo de color consistente.
 - ii. Par que los objetos se distingan hay que utilizar colores que contrasten.
 - iii. Solo se utilizan colores cuando sea estrictamente necesario.
 - iv. Los colores a la hora de representar datos han de ser utilizados cuando correspondan a diferencias en el significado de los datos.
 - v. Colores suaves para información usual y colores fuertes para resaltar información.
 - vi. Paletas de colores: categóricas: sin orden, secuenciales: con orden, pero sin valor neutro, derivadas: con orden y color neutro.
 - vii. Los componentes sin datos de tablas y gráficos han de ser mostrados solo visibles, sin colores, para que hagan su función y nada más, de otra forma provocaría distracciones en los que las interpretan.
 - viii. Evitar colores verde y rojo en el mismo sitio.
 - ix. Evitar efectos visuales como 3d en los gráficos.

x. Utilizar colores opuestos para representar gráficos.

12. Orientation sensitivity: Orientaciones verticales y horizontales son bien percibidas mientras que las oblicuas no tanto.

13. Efecto de superioridad pictórica: las personas tienden a retener mejor las imágenes que el texto después de 30 segundos de observación.

Las 7 leyes de Gestalt

1. **Ley de prägnanz:** las personas tendemos a ver figuras simples.



2. **Ley del cierre:** Si un elemento tiene una forma conocida y es parcial o está incompleto la mente humana tiende a cerrarlo y completar la forma.



3. **Ley de similitud:** Se tiende a agrupar elementos con el mismo color, forma, brillo, etc.



4. **Ley de proximidad:** Elementos cercanos tienden a verse como una unidad.



5. **Ley de simetría:** Las imágenes simétricas son percibidas como un conjunto, aunque estén separadas ({}).
6. **Ley de continuidad:** Elementos que siguen un patrón suelen verse del mismo conjunto aun teniendo propiedades distintas.
7. **Ley del destino común:** Elementos con la misma dirección son percibidos como un colectivo.

Interacción diseño y evaluación

1. **Entropía de Shannon:** existe una interferencia y no toda la información llegará al receptor.

- a. Formulación de información Recibida:

$$R = H(x) - H_y(x)$$

- $H_y(x)$ is the *equivocation* or conditional entropy of x when y is known

$$H = \sum_{i=1}^N p_i \log_2 \left(\frac{1}{p_i} \right) = - \sum_{i=1}^N p_i \log_2 p_i$$

- N is the number of alternatives
- p_i is the probability of the i th alternative.
- H is the entropy of the message that is to be transmitted, the amount of information expected to be received (no noise).

2. **Ley de Hick-Hyman:**

- a. Considera el tiempo que le toma hacer una decisión a un usuario.
- b. A un usuario le cuesta responder a un estímulo más tiempo cuando este pertenece a un gran conjunto.
- c. Formulación de tiempo de respuesta: **$RT = a + b \cdot \log_2(n+1)$** donde a y b son constantes y n el número de estímulos. El $+1$ corresponde a la incertidumbre de contestar o no.

3. **Ley de Fitts formulación original:**

- a. Establece una relación entre la dificultad de una tarea y su tiempo de movimiento (MT).
- b. Formulación: **$ID = \log_2 (2A / W)$** donde ID es la dificultad de la tarea A es la amplitud del movimiento y W es el ancho del objetivo. **$MT = a + b \cdot ID$** donde a es el tiempo start/stop en segundos y b es inherente a la velocidad del dispositivo.

4. **Welford variante para la ley de Fitts:** **$MT = a + b \cdot \log_2(D + 0.5W)/W$** donde D es la distancia de movimiento y W el ancho del objetivo.

5. **MacKenzie variante para la ley de Fitts:** $MT = a + b \cdot \log_2(D/W + 1)$ donde D es la distancia de movimiento y W el ancho del objetivo.

- a. En este caso los movimientos verticales y horizontales se tratan igual.
- b. D es solo hasta llegar al objetivo, no hasta su centro.

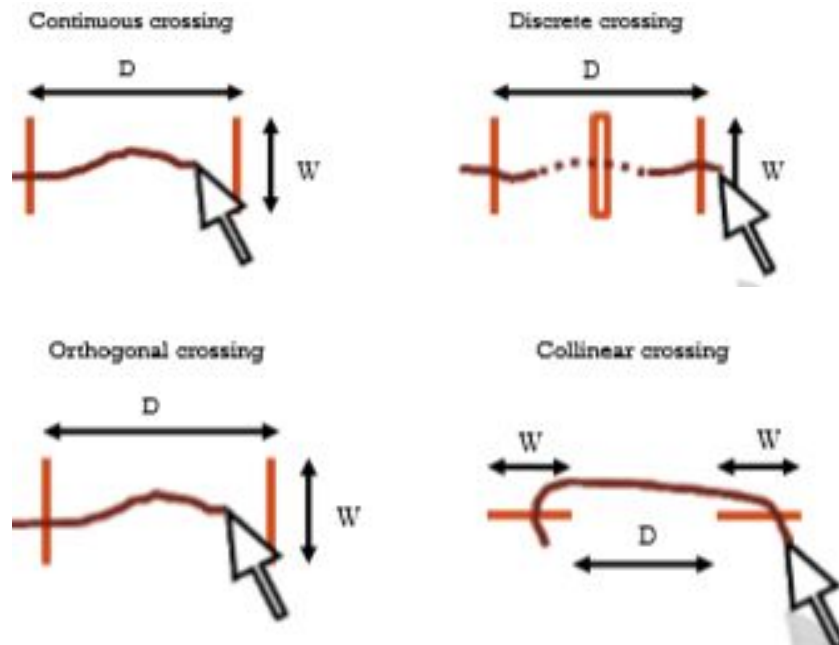
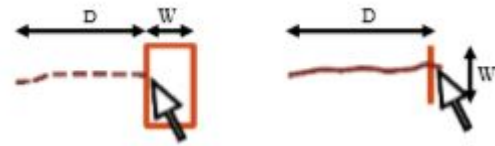


6. **Ley de Fitts en diseño de UI:**

- a. Los laterales de la pantalla tiene anchura infinita.
 - b. Los lugares más fáciles de llegar son las esquinas.
 - c. Aplicaciones:
 - i. Mantener las cosas relacionadas cerca y los elementos opuestos lejos.
 - ii. Los menús tipo pop-up reducen el recorrido. El problema es que solo lo utilizan usuarios expertos.
 - iii. Pie menús: no han de tener oclusiones y necesitan espacio para ser desplegados. En los teléfonos móviles es mejor utilizar medio círculo y no todo el círculo.
 - iv. Han de ser creados en todo caso bajo demanda.
 - v. Para dispositivos móviles hay que tener en cuenta la regla del pulgar.
 - vi. Para la toma de decisiones se puede aplicar la regla del pulgar, poniendo los elementos más destructivos en una posición de difícil alcance.
7. **Objetivos expandibles:** el hecho de utilizar objetivos que se expanden cuando el mouse se acerca mejora el tiempo de selección, por contraparte necesita más espacio o mover a los demás.
8. **Cursores expandibles (Bubble cursor):** el cursor aumenta su rango cuando está cerca de los objetivos, se ha de determinar el alcance de cada objetivo previamente de manera que si este tiene un radio X se pueda determinar sobre qué objetivo tiene más influencia.
9. **Control display ratio:** la relación entre el movimiento de la mano del usuario y el movimiento del cursor virtual se puede mapear de diferentes maneras: Constante, dependiendo de la velocidad de movimiento o dependiendo de la posición del cursor.

10. Ley de Crossing:

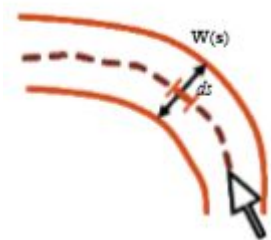
- Formulación: $T = a + b \cdot \log_2(D/W + 1)$
- D es la distancia entre objetivos y W el ancho de los objetivos, al igual que en Fitts a y b son constantes a determinar.
- Podemos determinar los siguientes tipos de cruce:



- Ley de steering:** ley que estudia lo que tarda un usuario en seguir un camino, puede aplicarse a varias cosas como navegación por los menús, dibujar, etc.

$$T_s = a + bID_s$$

$$ID_s = \int_c \frac{ds}{W(s)}$$



Donde **C** es la longitud del camino y **W** es el ancho del camino hasta el punto s.

12. Pointing devices:

- Land-on strategy: se selecciona al clicar en el objeto pero es más propenso a errores.
- Lift-off strategy: se selecciona al levantar el cursor, esto permite una mayor precisión.

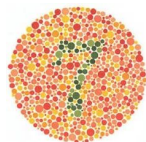
13. Selección 3d: se utilizan 2 técnicas

- a. Hand extension: elementos a veces difíciles de alcanzar, se requieren amplios movimientos debido a el mapeado directo con el mundo 3d, es intuitivo.
- b. Ray-based technique: es más preciso pero algunos elementos pueden estar ocluidos al rayo, es mas difícil buscar.

Percepción y color

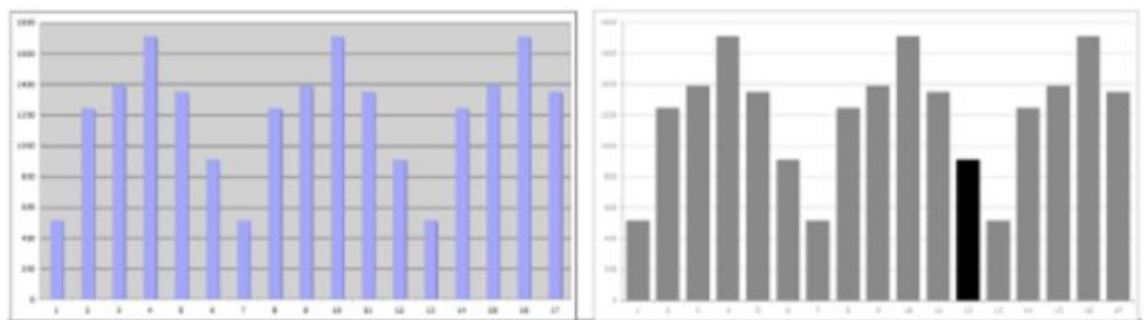
1. Problemas:

- a. Deuteranopia → poco verde.
- b. Protanopia → poco rojo.
- c. Tritanopia → poco azul.
- d. Achromatopsia → ningún color.
- e. Estos problemas pueden ser detectado con un test de Ishihara



2. Consejos:

- a. Forzar el contraste entre el fondo y el color del texto.
- b. Evitar colores adyacentes o de misma luminosidad.
- c. Fondo consistente.
- d. Colores brillantes contrastan bien con los oscuros.
- e. Uso de colores solo si sirven para clarificar no hay que malgastarlos.
- f. Uso de colores suaves para la mayoría de información pero brillantes u oscuros para la información que requiere más atención.
- g. Evitar ruido en los gráficos: en la primera imagen las rayas y el fondo distraen al receptor.



- h. Evitar efectos 3D al utilizar gráficos.
- i. Utilizar colores análogos (naranja, verde, amarillo...), tríadas (púrpura/naranja/azul), cuatríadas(púrpura/naranja/verde/azul).

Teclados:

1. **QWERTY:** teclas más usadas están más lejos las unas de las otras.
2. **DVORAK:** vocales situadas en un lado.
3. **AZERTY:** diseño para el francés.
4. **Teclados partidos:** no mejora la velocidad de escritura.
5. **Sobre diseño de teclados:** Los teclados han de ser diseñados en base al reparto de trabajo entre manos, maximizar el estar en la fila central y alternar el uso de dedos, minimizando en la medida de lo posible el uso de el mismo dedo constantemente.
6. **Problemática común:**
 - a. Autocorrección, es mejor no utilizarla.
 - b. Mayúsculas automáticas fallan a menudo.

Realidad virtual

1. **Definición:** simulación interactiva por computador desde el punto de vista del participante, en la cual se substituye o aumenta la información sensorial que recibe.
2. **Elementos:**
 - a. Simulación interactiva: reproducir un mundo que solo existe en el ordenador.
 - b. Inmersión sensorial: desconectar los sentidos del mundo real para llevarlos al mundo virtual. Los objetos existen independientemente del dispositivo de visualización utilizado. Se utiliza la visión estereoscópica para reproducir.
 - c. Interacción implícita: el sistema decide que quiere el usuario a partir de sus movimientos naturales tales como gestos o posibles movimientos del mouse.
3. **Elementos para utilizar realidad virtual:**
 - a. Periféricos de entrada: capturan las acciones del participante y envían la información a la máquina.
 - b. Computador: Realiza la simulación a partir del modelo geométrico y los datos de los periféricos de entrada.
 - c. Modelo geométrico 3D.
 - d. Periféricos de salida: Traducen las señales generadas por el ordenador tales como video, audio, etc en estímulos perceptibles por los sentidos.
 - e. Programa de tratamiento de datos de entrada: Leen y procesan la información que proporcionan los sensores.
 - f. Software de simulación física: Se encargan de las modificaciones pertinentes en la representación digital de la escena a partir de las acciones del usuario y la evolución del sistema.
 - g. Software de simulación sensorial: Representan digitalmente las imágenes, sonidos, etc.

Realidad aumentada

1. **Definición:** es la combinación de la escena real y virtual, tenemos acceso simultáneamente al mundo real y virtual.
2. **Objetivo:** mejorar el rendimiento y la percepción del mundo pero manteniendo una diferencia notable entre el mundo real y el virtual.
3. La realidad aumentada no tiene una inmersión total a diferencia de la realidad virtual.

4. Es importante registrar los objetos de la realidad para evitar una mala superposición de imágenes.
5. Se puede presentar de 3 formas: video see-through (Pokémon Go), optical see-through (Google glasses) y AR projections.
6. En cuanto a las proyecciones son ideales puesto que el usuario no ha de utilizar elementos de visión adicionales pero requieren de calibración cada vez que cambia la escena por lo que se limitan a estar indoor.

Tests de usabilidad

Hay dos tipos de tests unos para **determinar problemas de usabilidad** (descubrir + priorizar + resolución de problemas) o **medir el rendimiento de una tarea**.

1. Técnicas:

- a. Think aloud: es una técnica donde los participantes hablan sobre lo que hacen en ese momento ya que es más importante lo que piensan en el momento que lo que recuerdan después. Se puede aplicar en casi cualquier test. Funciona mejor para parejas.
- b. Remote testing: introduce entornos familiares más diversos participantes. Es más complejo recibir feedback del participante.

2. Laboratorios de usabilidad:

- a. Dentro o fuera de una habitación, ciertas condiciones de iluminación, calidad de conexión, a prueba de sonidos externos.
- b. Áreas y equipamiento: área del participante, área del observador, área del ejecutivo, videocámaras, micrófonos, teléfonos.

3. Roles:

- a. Administrador del test: diseña el estudio de usabilidad. Especifica las condiciones para hacer el test. Conduce las reviews de los tests, hace el análisis de los datos y hace la presentación final.
- b. Briefer: interactúa con los participantes, hace que en un estudio think-aloud el participante hable, está familiarizado con el producto para poder contestar a los participantes.
- c. Operador de cámara.
- d. Data recorder: escribe notas durante un test y utiliza data-logging software.
- e. Help desk operator: reemplaza un operador de help desk real.
- f. Product expert: mantiene el producto y ofrece soporte técnico durante el test.
- g. Estadístico.

4. Planificación de los tests:

- a. Antes de comenzar el administrador ha de saber el propósito del producto y decir qué partes están preparadas para hacer testing, determinar los tipos de personas que utilizarán el producto, determinar el uso del producto y las condiciones de uso.

- b. Propósito del producto: medir vs. identificar problemas de usabilidad. Podemos utilizar productos de la competencia (o no).
- c. Para descubrir problemas: se priorizan problemas con mucha frecuencia, que importan mucho en el uso normal, por magnitud de impacto y facilidad de corrección.
- d. Para tests de métricas: hacemos categorías de ratios de éxito/fracaso y precisión, velocidad, eficiencia, identificadores de operabilidad y learning rate.

5. Participantes:

- a. Los participantes son seleccionados a propósito según las necesidades del test.
- b. Con 5 participantes descubriremos el 80% de los problemas, con 3 la mayoría.
- c. Recomendable ≤ 5 testers con asignaciones de tareas pequeñas llamado **ley de diminishing returns**.
- d. Hay que evitar efectos de aprendizaje y cansancio.

6. Implementación:

- a. Los escenarios han de ser representativos (tareas core = las básicas y tareas periféricas = más complejas). Definimos un escenario con unas condiciones iniciales y se especifica que hacer y el porqué. No todos los participantes tienen porqué tener el mismo escenario.

7. Reporte:

- a. Hay que hacer un reporte con la descripción y prioridad de los problemas de usabilidad.
- b. Evaluar los problemas: frecuencia y severidad.
- c. Intentar aportar una solución como mínimo a cada problema.

8. Evaluación de los problemas:

- a. Según Jeff Rubin, niveles de problemas: 4 - unusable (no usa una parte por cómo está diseñada o implementada o no puede encontrarla), 3 - severe (intentará usarla, pero tendrá muchas dificultades), 2 - moderate (l el usuario podrá usarlo en la mayoría de casos, pero tendrá que invertir ciertos esfuerzos a solucionarlo), 1 - irritante (problemas estéticos, sólo intermitentemente ...).
- b. Según Dumas: level 1 - impide la finalización de la tarea, level 2 - crea un retraso importante y frustración, level 3 - poca importancia de usar, level 4 - sugerencias sutiles y posibles mejoras.

Métodos cuantitativos para experimentos humanos

1. Diseño de experimentos:

- a. Para evitar desviaciones, tenemos cuidado del learning y de la fatiga:
 - i. Tras N repeticiones, un usuario soluciona más rápido un problema.
 - ii. Tras N repeticiones, el usuario puede sufrir fatiga.
- b. Diseño **counterbalancing** (de contrapeso): evitamos learning y fatiga para hacer las tareas en orden aleatorio; no totalmente aleatorio, sino ordenadas de manera adecuada (variaciones sistemáticas).
- c. Si tenemos dos páginas web a estudiar en dos dispositivos diferentes y con cuatro repeticiones, eso son $3 \times 2 \times 4 = 24$ tareas. Crece factorialmente! Por lo tanto, cada usuario hace un test diferente.
- d. Usamos **latin squares**: evitan learning y fatiga. En el caso anterior hacemos un cuadrado latino 2×2 , y dos cuadrados 3×3 . Por lo tanto, tenemos un rectángulo grande 6×3 y otro 2×2 . Si hacemos producto cartesiano, tenemos permutaciones por 12 personas.

2. Representación de los datos:

- a. **Estadística descriptiva:** describir y explorar datos. Gráficos, histogramas ...
Entender la distribución de los datos y pensar en tests de significación.
- b. **Estadística inferencial:** detectar relaciones en los datos. Inferir características de la población a partir de características de los ejemplos.
- c. Usamos charts (diagramas).
- d. Evitamos pie charts, proyecciones 3D, mantenemos buen ratio fecha / chart, usamos el diagrama adecuado para el uso adecuado.
- e. Tipo de diagramas: trend, relative size y composition graphs.
- f. Chartjunk: evitar cosas que no aportan nada a la hora de representar información.
- g. Problemas típicos de los graphs:
 - i. Tipo de gráfico equivocado.
 - ii. Falta información (título, escala, etiquetas ...).
 - iii. Escala inconsistente.
 - iv. Zero mal colocado.
 - v. Efectos de gráficos pobres (ducks = elemento innecesario, sombras).
 - vi. No hay ajuste por inflación.
 - vii. Demasiada precisión.
 - viii. Poco balance *data ink ratio = data ink / total ink to print graphic*.
- h. En vez de pie charts, usamos barras segmentadas (de 0% a 100%)