Interacción y Diseño de Interfaces (IDI)

Cuatrimestre 4

https://github.com/AdriCri22/Interaccion-Interfaces-graficas-IDI-FIB

1. Procesos de visualización

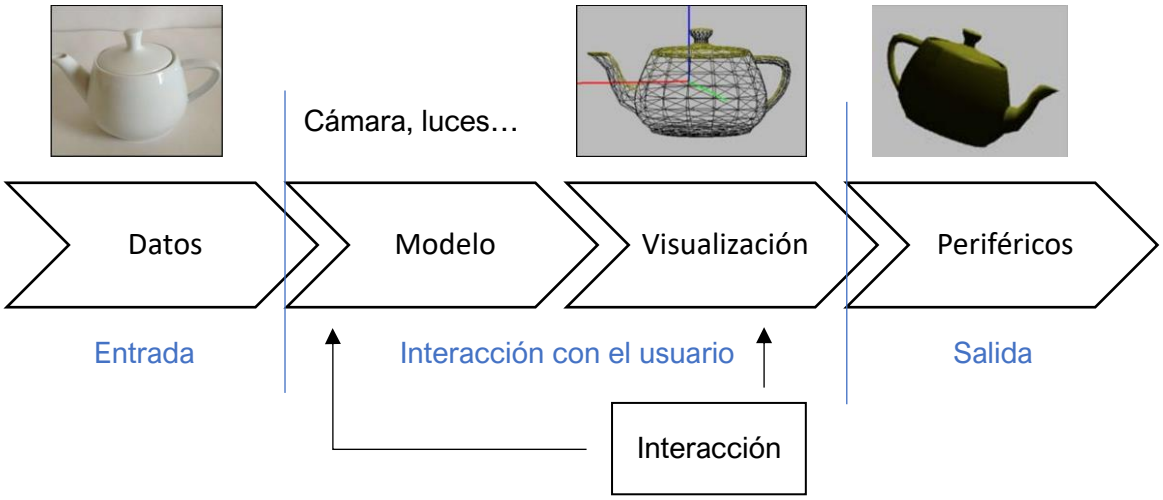
1.1. OpenGL y Qt

Esta asignatura trata sobre la representación de entornos reales a entornos gráficos con interacción del usuario, mediante el uso de OpenGL y Qt.

- **OpenGl**: Aplicación para la creación de gráficos en 3D (versión 3.3 o superior).
- **Qt**: Aplicación que permite al usuario interactuar con el entorno grafico (versión recomendada 5.x).

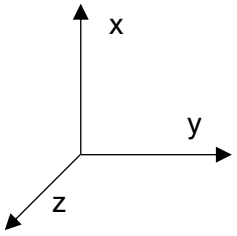
1.2. Modelos y visualizaciones

Primeramente, hay que tener en cuenta el orden, para entender el funcionamiento.



- Los **datos** son toda esa información que poseemos para crear el modelo.
- El **modelo** es creado a partir de los datos, en el centro de las coordenadas. El usuario puede hacer modificaciones sobre el modelo (escalar, rotar, trasponer…).
- La **visualización** son las diferentes modificaciones que le hace el usuario para ver el objeto (luces, cámara…).
- Los **periféricos** es el resultado final que podemos ver por pantalla.

En líneas generales, tendremos **modelos 3D** los cuales están formados por un conjunto de triángulos, cada triángulo formado por **3 vértices con coordenadas (x, y, z)**. Estos modelos se almacenan en ficheros (*.obj).



Los vértices se indican en **sentido antihorario**, para usar el sentido dado para saber la orientación de la cara.

Una vez creado un modelo, este se guarda en un buffer llamado **VAO**, utilizado para **encapsular datos** del modelo. La idea del VAO es aprovechar el modelo creado para evitar crear otro modelo en el caso de querer representar gráficamente el mismo modelo más de una vez. Por lo que:

- Sólo hay un **único VAO por modelo**
- Si queremos hacer **cambios** de medida, posición y orientación se harán aplicando **transformaciones geométricas**.

Aparte del VAO, tenemos el **VBO**, este es un buffer dónde se guardan las coordenadas del modelo (vértices), los colores del modelo (por vértice), las normales del modelo (por vértice)… estas son opcionales a excepción de las coordenadas. Por lo que:

- Para **cada VAO** existe como **mínimo un VBO** (el de coordenadas)
- Por cada copia del modelo habrá mínimo otro VBO.
- Por cada modelo que queramos **pintar** hay un **VBO**, y habrá **tantos VBOs cómo copias** de estos modelos. De la misma forma que para las normales del modelo.

Una vez que creamos un modelo y le asignamos un VAO y a este VAO, le asignamos distintos VBOs (contenedores de datos), OpenGL envía estos datos a la tarjeta gráfica para pintar el modelo cada vez.

Para la **validación del modelo** hay que comprobar que:

1. Todas las caras han de estar orientadas correctamente (mirando a la normal).
2. Todos los vértices han de estar orientados coherentemente.
3. Las aristas, de un modelo cerrado, separan dos triángulos/caras.

Llamamos **escena** a la imagen resultante que vemos por la ventana gráfica, con todos los objetos insertados.

1.3. Uso de los VAOs y VBOs

```
void pinta_modelo() {
    glBindVertexArray(VAO);      // Envía el VAO
    modelTransform(TG);          // Calcula la TG a aplicar al modelo
    modelMatrix(TG);             // envía uniform
    pintaModelo();               // Pinta la escena final
}
```

1.4. Proceso de visualización completa

Los siguientes puntos detallan en orden el proceso de visualización completo que se hace por cada vértice en la tarjeta gráfica.

1. **Procesado de vértices (Programable)**

Recibe las coordenadas del vértice SCM (TG) y las transforma a SCC (Clipping), pasando primero por SCA (VM) y SCO (PM).

2. **Clipping + Perspective división (Fijo)**

Pasamos las coordenadas a SCN: Se eliminan todos los triángulos que se encuentren enteros fuera del rango de visión.

3. **Device transform (Fijo)**

Pasamos las coordenadas a SCD: Se tiene en cuenta el viewport y la relación de aspecto (ra).

4. **Back-face culling (Fijo, se puede activar y desactivar)**

Al ser activado, se eliminan las partes escondidas de los triángulos, es decir, tiene en cuenta las normales. Sólo lo elimina si no se ve ninguna parte del triángulo. (Si se ve ni siquiera un 1% del triángulo no se elimina).

5. **Rasterización (Fijo)**

Recibe los triángulos y los pinta con puntos, segmentos o polígonos. Se envía después al Fragment Shader.

6. **Procesado de fragmentos (Programable)**

Se realiza en el Fragment Shader, dónde se aplican los colores, texturas…

7. **Z-Buffer/Depth-buffer (Fijo, se puede activar y desactivar)**

Al estar activado, elimina partes escondidas de los triángulos para otros triángulos, es decir, si tenemos un polígono que tapa un triángulo, este se deja de pintar, ya que no se ve. No tiene en cuenta el back-face culling.

1.5. Sistemas de coordenadas (SC)

SC Modelo (SCM)	Coordenadas iniciales para los modelos	I
SC Aplicación (SCA)	Obtenidas una vez las SCM son posicionadas en la escena	TG * I
SC Observador (SCO)	Obtenidas una vez las SCA son transformadas según las coordenadas del observador	VM * TG *I
SC Clipping (SCC)	Obtenidas una vez las SCO son transformadas según la óptica de la cámara	V = PM * VM * TG * I
SC Normalizadas (SCN)	Obtenidas del resultado del clipping	V = V / w
SC Dispositivo (SCD)	Obtenidos del resultado del Device Transform para copiar la ventana gráfica correctamente	X*V, siendo X la transformación para ocupar toda la pantalla
SC Fragment (SCF)	Obtenidas después de la rasterización, siendo las coordenadas de cada fragmento de la ventana	

2. Transformaciones geométricas

Las transformaciones geométricas son la manera con la que podemos interactuar con los modelos y crear diferentes versiones del mismo modelo, a partir del uso de VAOs. Las **TG**s (Transformaciones Geométricas) son matrices de 4x4.

2.1. Translaciones

Sirve para mover el objeto entero a una nueva posición en la escena.

- **OpenGL** mat4 TG = glm::translate(mat4(TG), vec3(tx, ty, tz))
- **Pseudocódigo** TG = Translate(tx, ty, tz)
- **Matriz**:

$$T(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.2. Rotaciones

- **OpenGL** mat4 TG = glm::rotate(mat4(TG), flot(angle_rot),vec3(eix_rotació))
- **Pseudocódigo** Rotate(angleRot,(eix_rot))

- **Matriz**:

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Nota: antes de rotar llevar el objeto al centro de coordenadas

2.3. Escalado

- **OpenGL** mat4 TG = glm::scale(mat4(TG), vec3(sx, sy, sz))
- **Pseudocódigo** TG = Scale(sx, sy, sz)
- **Matriz**:

$$S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Para escalar al tamaño deseado (altura exacta): $s_y = tamaño\_deseado/(y_{max} - y_{min})$. Para escalar sin deformar: $s_x = s_y = s_z$

Nota: antes de escalado llevar el objeto al centro de coordenadas

2.4. Cómo aplicar el código

1. **El código se lee de abajo a arriba o de derecha a izquierda**.
2. **Siempre** llevar el **objeto** al **origen de coordenadas**.

2.3. Caja mínima contenedora

Creamos una caja con todo el objeto dentro, englobamos el modelo, los límites de la caja contenedora están formados por los extremos del modelo.

- Límites de la caja: $P_{min}(x_{min}, y_{min}, z_{min})$ y $P_{max}(x_{max}, y_{max}, z_{max})$
- $amplitud = x_{max} - x_{min}$ $altura = y_{max} - y_{min}$ $profundidad = z_{max} - z_{min}$
- $centro\_caja = \left(\frac{x_{max}+x_{min}}{2}, \frac{y_{max}+y_{min}}{2}, \frac{z_{max}+z_{min}}{2}\right)$
- $base\_caja = \left(\frac{x_{max}+x_{min}}{2}, y_{min}, \frac{z_{max}+z_{min}}{2}\right)$

3. Cámara

Hasta ahora sólo hemos realizado cambios sobre la escena, pero ahora introduciremos el movimiento de la cámara. Primero hay que definir la posición de la cámara y el ángulo de visión (lookAt) y después escoger la óptica de la cámara (perspectiva u ortogonal).

3.1. Posicionamiento de cámara

El posicionamiento de cámara **no** depende del **tipo de óptica**. Su objetivo es generar una matriz **VM** (View Matrix) que dada una escena arbitraria posicione un observador en un punto externo de la escena para poderla ver. Hay dos maneras de establecer la posición de la cámara:

3.1.1. Mediante la función lookAt

- **OpenGL**: mat4 VM = glm::lookat(vec3(OBS), vec3(VRP), vec3(up))
- **Pseudocódigo**: VM = lookat(OBS, VRP, up); viewMatrix(VM)

Parámetros de posicionamiento:

- **OBS**: punto origen de la cámara (dónde se encuentra el observador o la cámara)

- **VRP**: punto objetivo de la cámara (hacia dónde apunta la cámara)
- **up**: vector vertical que orienta la cámara

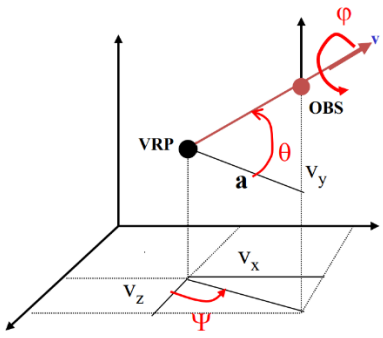
3.1.2. Mediante ángulos de Euler (Forma parte de la cámara en 3ª persona)

La idea es que la escena está centrada en el eje de coordenadas y el que se desplaza es el observador (o cámara) por una ventana exterior, por lo que para calcular la VM aplicamos transformaciones geométricas.

El orden para crear las transformaciones geométricas es:

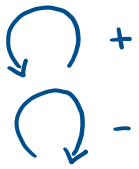
1. Hacer la translación des de VRP al centro.
2. Hacer las rotaciones deseadas
3. Trasladar la escena a una distancia d (VRP-OBS)

Ver apartado 3.6 para saber que es d



3 — VM = Translate (0, 0, -d)

2 — VM = VM * Rotate(φ, 0, 0, 1)
VM = VM * Rotate(θ, 1, 0, 0)
VM = VM * Rotate(ψ, 0, 1, 0)



1 — VM = VM * Translate(-VRP)

viewMatrix(VM)

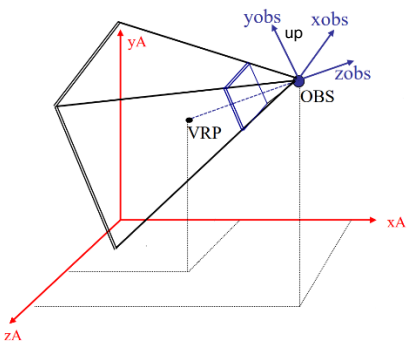
3.2. Cámara perspectiva

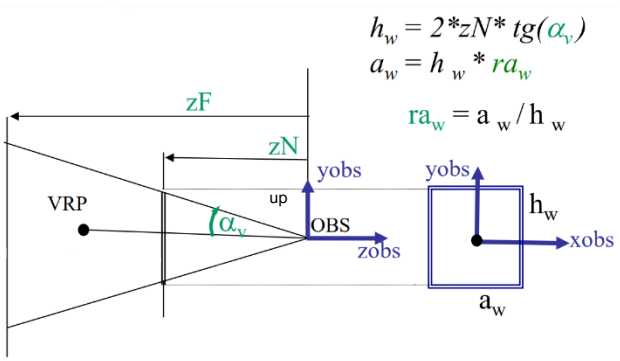
Este tipo de óptica de cámara permite dar profundidad a la escena, como si realmente fuera 3D.

- **OpenGL**: mat4 PM = glm::perspective(float(FOV), float(raw), float(zN), float(zF))
- **Pseudocódigo**: PM = perspective(FOV, raw , zN, zF); projectMatrix(PM)

Los parámetros son:

- **FOV**: el grado de apertura de la cámara $FOV = 2 * \alpha = 2 * \arcsin(a/h)$
- ra_w: la relación de aspecto
- **zNear**: distancia entre el OBS y el inicio de la escena (distancia entre el punto más cercano del objeto al observador)
- **zFar**: distancia entre OBS y el final de la escena (distancia entre el punto más lejano del objeto al observador**)**





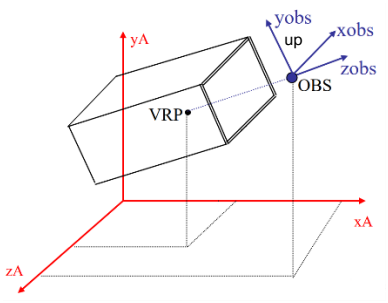
3.3. Cámara ortogonal o axonométrica

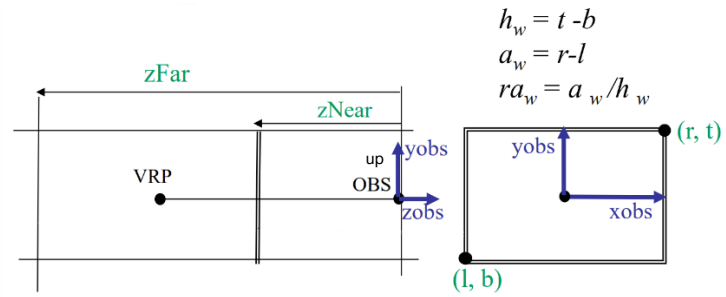
En este caso la escena se verá plana, en 2D, sin tener en cuenta la profundidad de los objetos.

- **OpenGL**: mat4 PM = glm::ortho(xl, xr, yb, yt, zN, zF); glUniformMatrix4fv(…)
- **Pseudocódigo**: PM = ortho(l, r, b, t, zN, zF); projectMatrix(PM)

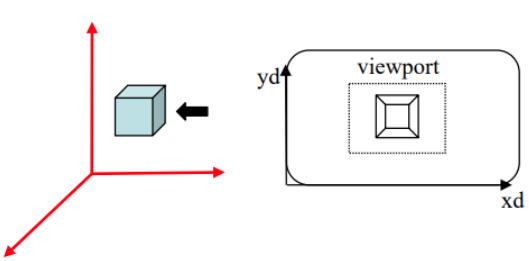
Los parámetros son:

- **l** (left)
- **r** (right
- **b** (bottom)
- **t** (top)
- **zNear**
- **zFar**

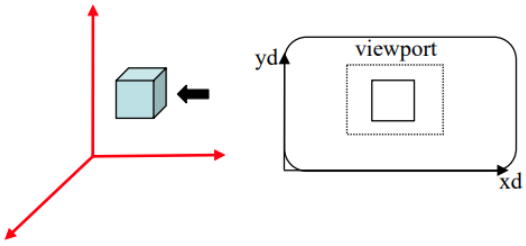




Comparación entre los dos tipos de óptica:



Perspectiva



Ortogonal

3.4. Redimensionado de ventana

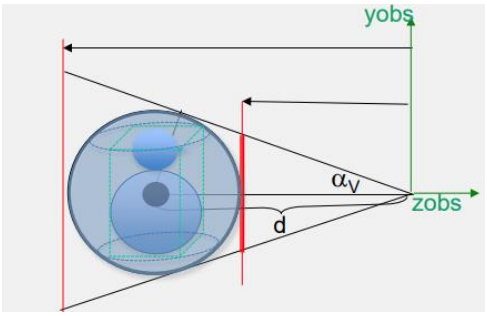
Para **evitar deformaciones de los modelos**, al redimensionar las ventanas debemos seguir **manteniendo** las relaciones de **aspecto del viewport**.

- Óptica perspectiva:
  - Si $ra_v > 1$: simplemente igualar $ra_w = ra_v$, para eso basta aumentando a_w
  - Si $ra_v < 1$: hay que igualar $ra_w = ra_v$, pero para ello hay que modificar la altura h_w, y esta está sujeta al FOV, por lo que hay que modificar los dos parámetros.

$$ra_w = a_w / h_w \qquad h_w = 2 * Znear * \tan(\alpha)$$

- Óptica ortogonal:
  - Si $ra_v > 1$:
    - l = -ra_v * R
    - r = ra_v * R
    - b = -R
    - t = R
  - Si $ra_v < 1$:
    - l = -R
    - r = R
    - b = -R / ra_v
    - t = R / ra_v

R es el radio de la esfera y d = 2*R



3.5. Zoom

Para poder hacer zoom en las escenas hay distintas opciones que podemos aplicar a la óptica de la cámara.

- Óptica perspectiva:
  - Modificar el **FOV**
  - Modificar el **zNear** y **zFar**
  - Modificar **OBS** y **VRP**
- Óptica ortogonal:
  - Modificar el **window**

Siempre manteniendo el ra_v

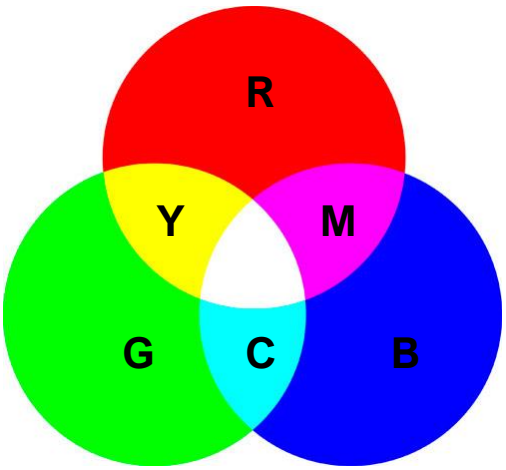
3.6. Cámara en 3ª persona

Indicaciones paso a paso de la inicialización de la cámara en 3ª persona, para no recortar la escena:

1. Encontrar **puntos máximos y mínimos de la caja contenedora** $P_{min}(x_{min}, y_{min}, z_{min})$ y $P_{max}(x_{max}, y_{max}, z_{max})$
2. Definir el **VRP** (centro de la caja)
3. Calcular el **radio** de la escena (R =(P_{max} - P_{min}) / 2)
4. Definir la **distancia** entre el OBS y VRP (d = 2*R)
5. Definir **OBS** con OBS = VRP + d * v (v es la dirección en la que se aleja)
6. Definir el parámetro **up**
7. Definir el **tipo de óptica**
8. Definir **zNear** y **zFar** (0 < zNear <= d-R) y (zFar >= d + R)
9. Definir **ra_w = 1**
10. Definir **parámetros de la óptica**:
    a. Si **perspectiva** FOV = 2 * α y $\alpha = \arcsin(R/d)$
    b. Si **ortogonal** l == b == -R y r == t == R

4. Colores

4.1. RGB



R = G = B = 0 → Negro

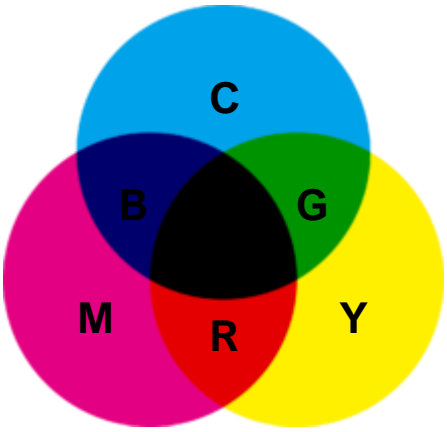
R = G = B = 1 → Blanco

R = 0; G = B = 1 → C

G = 0; R = B = 1 → M

B = 0; R = G = 1 → Y

4.2. CMY



C = M = Y = 0 → Blanco

C = M = Y = 1 → Negro

C = 0; M = Y = 1 → R

M = 0; C = Y = 1 → G

Y = 0; C = M = 1 → B

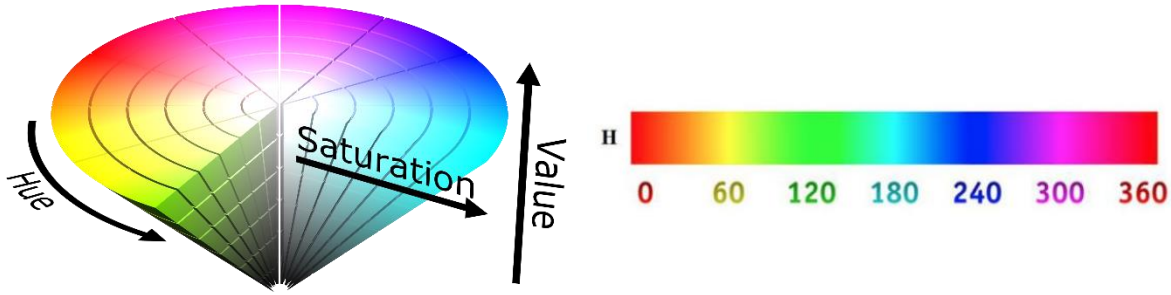
4.2.1. Pasar de RGB a CMY

C = 1 – R; M = 1 – G; Y = 1 – B;

4.2.2. Pasar de CMY a RGB

R = 1 – C; G = 1 – M; B = 1 – Y;

4.3. HSB (o HSV)



4.3.1. Pasar de HSV a RGB

Siempre y cuando $0 \leq H \leq 360$, $0 \leq S \leq 1$ y $0 \leq V \leq 1$:

$$C = V * S$$

$$X = C * (1 - |(H/60º)\ mod\ 2 - 1|)$$

$$m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & 0 \leq H < 60 \\ (X, C, 0) & 60 \leq H < 120 \\ (0, C, X) & 120 \leq H < 180 \\ (0, X, C) & 180 \leq H < 240 \\ (X, 0, C) & 240 \leq H < 300 \\ (C, 0, X) & 300 \leq H < 360 \end{cases}$$

$$(R, G, B) = \big((R' + m) * 255,\ (G' + m) * 255,\ (B' + m) * 255\big)$$

4.3.2. Pasar de RGB a HSV

Siempre y cuando R, G y B se encuentren en valores entre 0 y 255

$R' = R/255; \quad G' = G/255; \quad B' = B/255;$

$C_{max} = \max(R', G', B')$

$C_{min} = \min(R', G', B')$

$\Delta = C_{max} - C_{min}$

$$H = \begin{cases} 0 & \Delta = 0 \\ 60 * \left(\frac{G'-B'}{\Delta} \ mod \ 6\right) & C_{max} = R' \\ 60 * \left(\frac{B'-R'}{\Delta} + 2\right) & C_{max} = G' \\ 60 * \left(\frac{R'-G'}{\Delta} + 4\right) & C_{max} = B' \end{cases}$$

Si hay valores máximos iguales seguir este orden de preferencia. Ejemplo si R' = G' = B' = C_{max} → $H = 60 * \left(\frac{G'-B'}{\Delta} \ mod \ 6\right)$

$$S = \begin{cases} 0 & C_{max} = 0 \\ \frac{\Delta}{C_{max}} & C_{max} \neq 0 \end{cases}$$

$V = C_{max}$

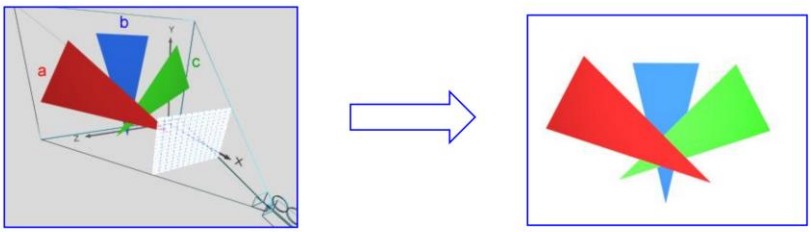
5. Realismo e iluminación

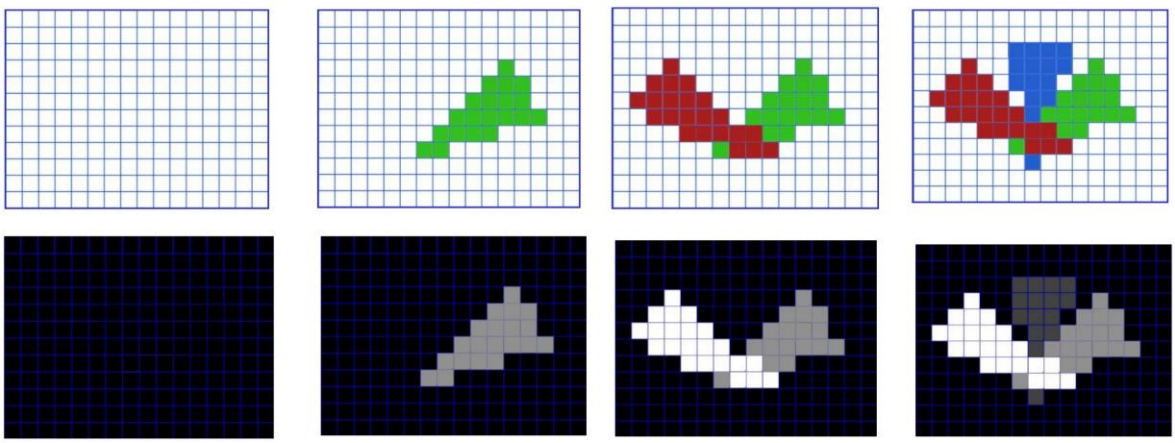
El realismo se basa en dos cosas:

- Eliminación de partes ocultas
- Modelos de iluminación

5.1. Depth Buffer (Z-Buffer)

Se encarga de eliminar las caras ocultas. Ha de conocer el valor de profundidad de cada uno de los píxeles y va eliminando aquellas que no se ven.

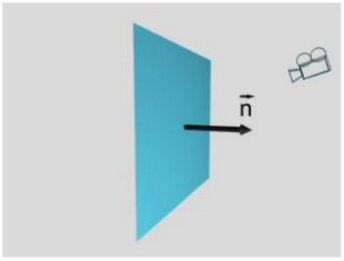


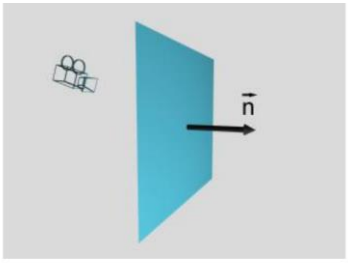


No importa el orden en el que se envían los triángulos y no hay que tener el back-face culling activado.

5.2. Back-face culling

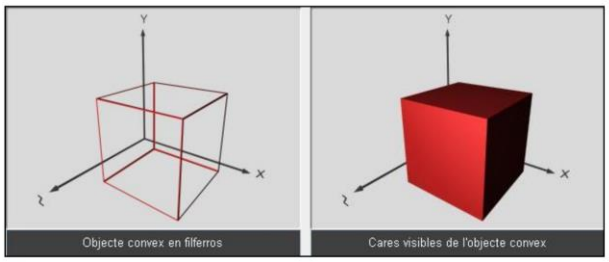
Elimina las caras ocultas que la normal no se puede ver desde la cámara. El cálculo se hace en la SCD.

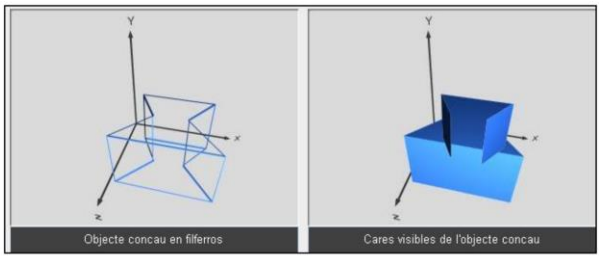


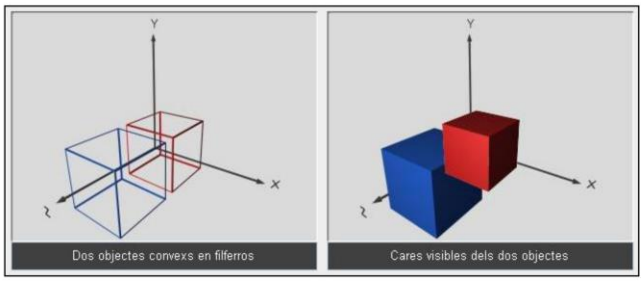


visible **no visible**

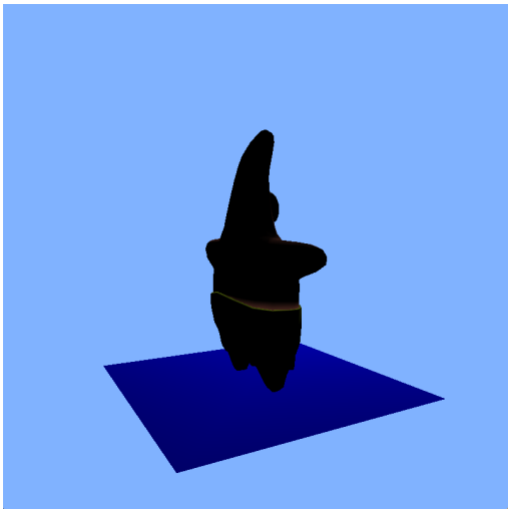
Aquí podemos ver la importancia de ordenar los vértices de manera antihoraria, para que la normal apunte donde queremos (por convención).







Considera solo las caras que están orientadas hace el observador



Sin back-face culling Con back-face culling

5.3. Iluminación

Con la inclusión de la iluminación en nuestras escenas conseguimos que estas sean más realistas, con una apariencia de volumen que con colores uniformes no podríamos conseguir.

5.3.1. Color de un punto

El color que un observador ve en un punto P de la escena es el color de la luz que llega al observador procedente de P: $I_\lambda(P \rightarrow Obs) \quad \lambda \in \{r, g, b\}$

Este depende de factores como el foco de luz, materiales y posición del observador.

5.3.2. Modelo de iluminación global

- Modelo de trazado de rayo: permiten simular sombras, transparencias y espejos, tienen un coste mayor en cálculo y tienen en cuenta:
  - Foco de luz
  - Otros objetos existentes en la escena, pero sólo transmisiones especulares.
- Modelo de radiosidad: Permite modelar sombras y penumbras (efectos como la luna que no está totalmente iluminada), pero no puede modelar espejos o transparencias. Son las más costosas.

5.3.3. Modelos locales o empíricos

Sólo tiene en consideración:

- El punto P que se calcula
- El foco de luz (siempre puntuales)
- La posición del observador
- NO considera objetos de la escena (sombras, espejos o transparencias)

5.3.3.1. Modelo ambiente

NO considerar el foco de luz de la escena → Todos los objetos reciben la misma luz → Se observa el mismo color en todo el objeto → Sensación de objeto plano (visto hasta ahora).

Ecuación:

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

$I_{a\lambda}$: color de luz ambiente

$k_{a\lambda}$: coeficiente de reflexión ambiente

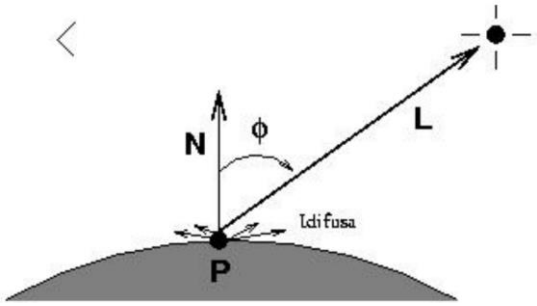
Ejemplo de una esfera:

$I_a = (1, 1, 1)$

k también blanca con intensidad variante entre 0.2, 0.4, 0.6, 0.8 y 1

5.3.3.2. Modelo difuso (Lambert)

Considera focos de luz puntuales, estos varían la posición. Imaginamos que este punto irradia la misma luz en todas direcciones y por lo tanto, su color depende de la dirección de visión.



Ecuación:

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

$$I_{f\lambda} * k_{d\lambda} * \cos(\phi) = I_{f\lambda} * k_{d\lambda} * dot(N, L) \qquad si\ |\phi| < 90º$$

$I_{f\lambda}$: color (r,g,b) de la luz del foco puntual f.

$k_{d\lambda}$: coeficiente de reflexión difusa del material.

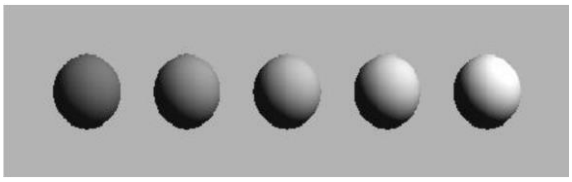
$\cos(\phi)$: coseno del ángulo entre la luz incidente y la normal en la superficie en el punto P.

$dot(N, L)$: producto escalar entre N y L. Se puede calcular así si están normalizados.

Ejemplo de una esfera:

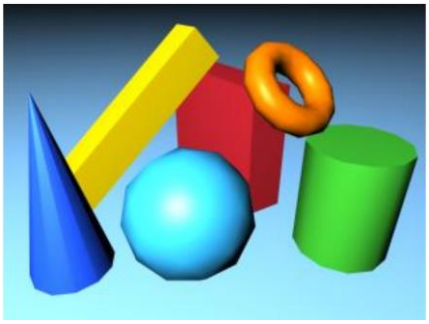
$I_f = (1,1,1)$

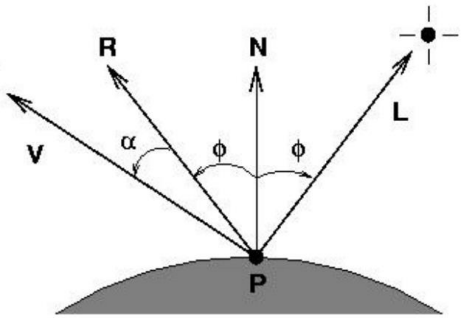
k_d también blanca con intensidad variante entre 0.4, 0.55, 0.7, 0.85 y 1



5.3.3.3. Modelo especular (Phong)

Tiene en cuenta focos de luz puntuales y objetos sólo reflexión especular. El observador sólo podrá observar reflexión especular en un punto si se encuentra en la dirección de reflexión especular, esta dirección es la simétrica de L respecto N y se puede calcular como: R=2N()





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

$$I_{f\lambda} * k_{s\lambda} * \cos^n(\alpha) = I_{f\lambda} * k_{s\lambda} * dot(R, v)^n \quad si\ |\phi| < 90º$$

$I_{f\lambda}$: color (r,g,b) de la luz del foco puntual f.

$k_{s\lambda}$: coeficiente de reflexión especular (x, x, x).

n: exponente de reflexión especular.

v: vector normalizado que une punto con observador

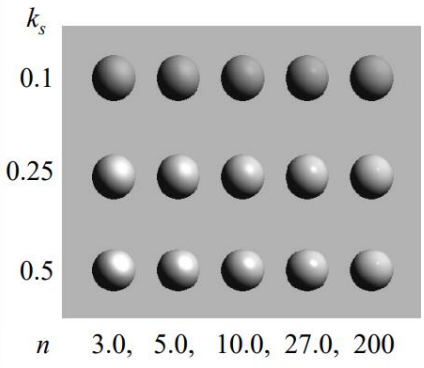
Ejemplo de una esfera:

$I_f = (1,1,1)$

k_d blanca de una intensidad de 0.5

k_s blanca con 0.1, 0.25 y 0.5

n de 3, 5, 10, 27 y 200

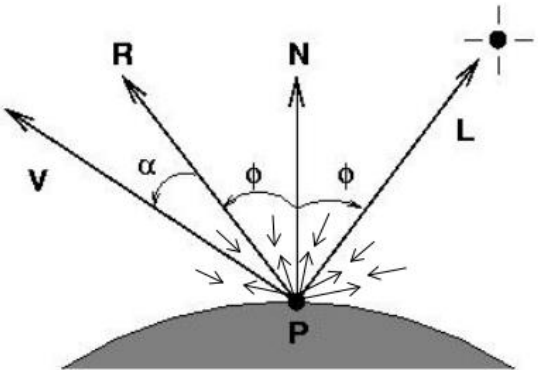


Resumen

Modelo en un punto P	**Depende de la normal?**	**Depende del observador?**
Modelo ambiente	No	No
Modelo difuso	Si	No
Modelo especular	Si	Si

La fórmula general aplicando todos los modelos:

$$I_\lambda(P) = I_{a\lambda} * k_{a\lambda} + \sum_i \left(I_{f_i\lambda} * k_{d\lambda} * \cos(\phi_i)\right) + \sum_i \left(I_{f_i\lambda} * k_{s\lambda} * \cos^n(\alpha_i)\right)$$

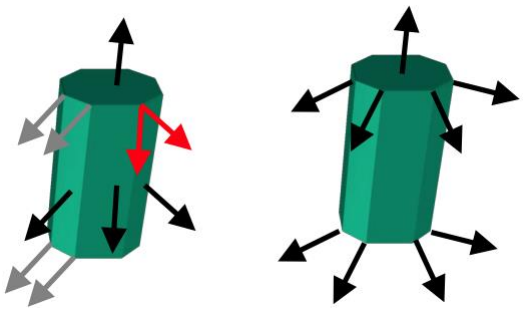


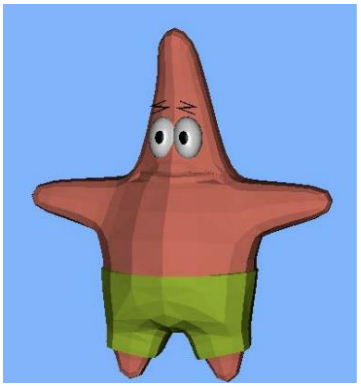
5.4. Cálculo de color (en el Vertex Shader)

```
in vec3 vertex, normal;
in vec3 matamb, matdiff, matspec;
in float matshin;
uniform mat4 proj, view, TG;

// Valores para los componentes que necesitamos de los focos de luz
vec3 colFocus = vec3(0.8, 0.8, 0.8);
vec3 llumAmbient = vec3(0.2, 0.2, 0.2);
vec3 posFocus = vec3(1, 1, 1);  // en SCA

out vec3 fcolor;
…
void main() {
    mat3 NormalMatrix = inverse(transpose(mat3(view * TG)));
    vec3 normSCO = normalize(NormalMatrix * normal);
    vec3 vertexSCO = (view * TG * vec4(vertex, 1.0)).xyz;
    vec3 posFocusSCO = (view * vec4(posFocus, 1.0)).xyz;
    vec3 L = normalize(posFocusSCO - vertexSCO);
    fcolor = Phong(normSCO, L, vec4(vertexSCO, 1.0));
    gl_Position = proj * view * TG * vec4 (vertex, 1.0);
}
```

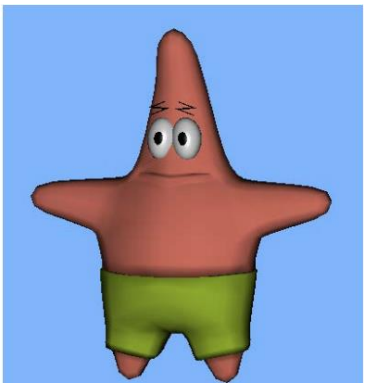
5.5. Suavizado de aristas

Se basa en cambiar de sitio las normales y no tenerlas declaradas únicamente en el centro de las caras, si no añadirlas en el vértice de cada cara, dando más definición al objeto.





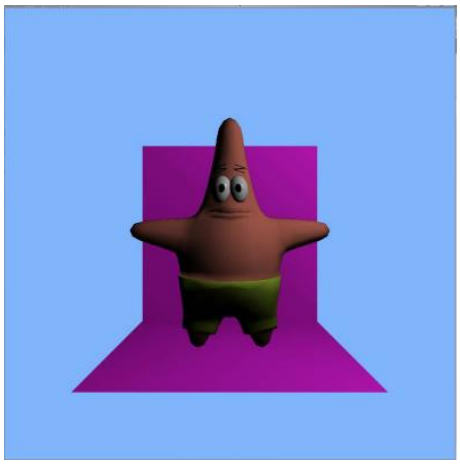
Normal por cara



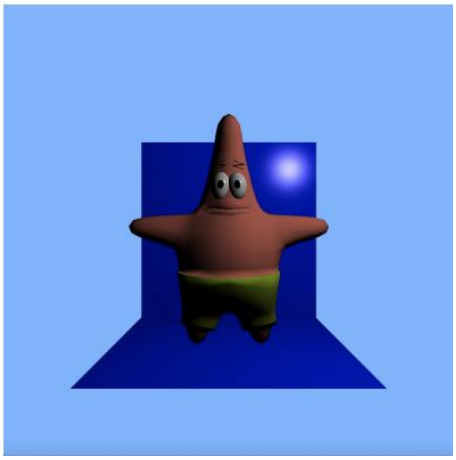
Normal por vértice

5.6. Cálculo de color (en el Fragment Shader)

Hacer el cálculo en el FS en vez del VS es **más costoso**, pero soluciona los problemas que teníamos con el VS (no se ven las manchas especulares en el centro de caras, se propagan las manchas de los vértices, perdemos iluminación al acercarnos a los polígonos grandes, las arestas varían entre ellas…). El FS calcula la iluminación en cada fragmento (cada píxel), recibimos los mismos inputs que en el VS, pasando antes por el VS, y los uniforms se mantienen. Para ser más eficientes vale la pena pasar a SCO los puntos en VS, pero los vectores se normalizan en FS.



VS



FS

Diseño de interfaces y usabilidad

1.1. Introducción

Una interfaz de usuario se define cómo las herramientas y métodos usados para la comunicación entre el usuario y el sistema.

HCI (Human Computer Interaction): Tienen como objetivo comprender y la evaluar críticamente las tecnologías interactivas que las personas utilizan y lo que experimentan. Su foco original fue la usabilidad, definida como la capacidad en la que un producto puede ser usado por usuarios específicos para llevar a cabo tareas específicas con:

- Eficacia Completar total y correctamente la tarea
- Eficiencia Relación entre los recursos usados y los resultados obtenidos
- Satisfacción El confort y aceptación del sistema por parte de los usuarios

UX (User Experience): creación de una experiencia significativa a través del dispositivo, recuerdos y sensaciones del uso del sistema.

Interaction Design: consiste en dar forma a las cosas digitales para el uso de las personas.

Características de los diferentes sistemas:

- Escritorio: Pantalla grande con mucho espacio e interacción con teclado, ratón.
- Smartphone: Pantalla pequeña con poco espacio (Hay que pensar dónde poner toda la información) e interacción con los dedos/lápiz táctil y teclado en pantalla.
- Tablet: Entre los dos casos anteriores, más espacio pero con la limitación de los smartphones.

GUI (Graphic User Interface)

- Web apps:
  - Pros:
    - Multiplataforma (sólo hay que programarlo 1 vez)
    - Fácil de actualizar
    - Herramientas técnicas conocidas
  - Contras:
    - Interfaces de usuario limitadas
    - No tenemos acceso a recursos locales, comunicación y UI peores que en apps nativas.
    - Protocolo de comunicación inseguro e ineficiente
    - Mayormente diseñadas para uso en escritorio
- Native Apps:
  - Pros:
    - UI enriquecida
    - Muchos más controles
    - Acceso rápido y seguro a recursos locales (GPS, cámara, archivos…) y comunicación más eficiente
    - Menor variedad de lenguajes y herramientas (SDK)
    - Diseñadas para pantallas más pequeñas
  - Contras:
    - No tienen acceso universal (1 App por cada plataforma)
    - Difícil administrar las actualizaciones
    - Menos general que la programación web

1.2. Principios de diseño

Conceptos base:

- **Interfaces efectivas**: dar al usuario la sensación de control y ocultar el trabajo interno del sistema.
- **Aplicaciones efectivas**: realiza el máximo de trabajo requiriendo un mínimo de información por parte de los usuarios.

Principios:

- **Aesthetics**
  - La moda no debe estar por encima de la usabilidad.
  - El usuario sabe los controles standard. Los No-standard requieren esfuerzo por parte del usuario.
  - El texto debe ser legible, tener un alto contraste con el fondo.
- **Anticipation**
  - Poner a disposición del usuario toda la información y las herramientas necesarias para cada paso del proceso.
  - Anticipar las necesidades del usuario (Requiere dominio de la tarea)
  - Información visible.
- **Autonomy**
  - Dejar que el usuario tenga cierto grado de personalización y toma de decisiones.
  - Mantener al usuario informado de estados, progresos…
  - Dar feedback de cada acción (ejemplo: al presionar un botón).
  - Tareas > 1 segundo → cursor ocupado.
  - Tareas > 10 segundos → barra de progreso.
- **Color**
  - Tener en cuenta a las personas que no distinguen ciertos colores (Usar colores con alto contraste).
  - Los colores tienen diferentes significados para las distintas culturas.
- **Consistency**
  - **Niveles de consistencia**:
    - **Consistencia de plataforma**: directrices establecidas según la plataforma (Android, iOS…), reglas no escritas que la comunidad asume y hay que mantener cierto aspecto a través de todos los servicios ofrecidos por una misma marca.
    - **Conjunto de productos**: mantener aspecto para toda la gama de productos.
    - **In-app**: tienen un look y sensaciones específicas.
    - **Estructuras visibles**: controlar la apariencia y mantener un posicionamiento similar.
    - **Estructuras invisibles**: se han de evitar o informar de su existencia (ejemplo: Hey google).
    - Nunca cambiar el significado de una acción habitual.
  - **Inconsistencia inducida**: hacer los objetos diferentes si actúan diferentes y notificar cambios funcionales.
  - **Continuidad inducida**: si el usuario sabe que algo actúa de cierta manera NO cambiarlo.
  - **Expectativas del usuario**: implementar funcionalidades cómo el usuario espera, no forzarlo a aprenderlas.
- **Default values**

  - Tener campos por defecto que sirvan de guía para el usuario.
  - Permitir actualizar campos, borrar o deshacerlos.
  - No todos los campos requieren un valor por defecto.
- **Discoverability**
  - Controles visibles.
  - Comunicar gráficamente las funciones por gestos.
  - Guiar al usuario en forma de tutorial.
- **Efficiency**
  - Priorizar la productividad del usuario.
  - Mantenerlo ocupado.
  - Informar, de manera útil, de los mensajes de error.
- **Explorable interfaces**
  - Libertad y seguridad para explorar.
  - Permitir volver atrás y deshacer cambios.
  - Pedir confirmación de cambios irreversibles.
- **Fitts's Law**
  - Objetos grandes son más importantes.
  - Objetos pequeños son menos importantes.
- **Informing users**
  - Informar a los usuarios de los tiempos de espera.
  - Reconocer clics de botones con un proceso visual de 50 ms.
- **More principles for usability**
  - Asegurarse de que el usuario no pierda su trabajo (hacer copias de seguridad).
  - Escoger metáforas extrapolables del mundo real al virtual.

1.3. Principios de diseño universal

- **Aesthetic-Usability Effect**: dedicar esfuerzos a mejorar nuestros diseños. Un diseño estético da sensación de más facilidad de uso y fomenta el uso de nuestros diseños.
- **Fix a visual hierarchy**: usar la escala modular garantiza unas proporciones armoniosas.
- **Correct alignment**: los elementos deben estar alineados, esto crea una sensación de unión y cohesión.
- Define a grid
- **Nesting**: es una cola visual de la jerarquía de la información mostrada.
- **Chunking**: agrupa un elemento en elementos más pequeños.
- Color: usar una paleta de colores (máx. 5 colores), colores saturados llaman la atención, pero los no saturados son más profesionales.
- **LATCH principle**: la información se organiza en función de: localización, alfabéticamente, tiempo, categoría, jerárquicamente.
- **Garbage-in garbage-out** (GIGO): comprobar los inputs hechos por los usuarios para asegurar que los datos introducidos están en un formato correcto y restringir los tipos de entrada de datos.
- **Iconic representation**: imágenes que tratan de representar objetos o acciones:
  - **Similarity**: el icono es visualmente similar a la acción/objeto que representa.
  - **Example**: los elementos pueden ser relacionados con la imagen.
  - **Symbolic**: alto nivel de abstracción.
  - **Arbitrary**: no tienen relación con el elemento o acción.

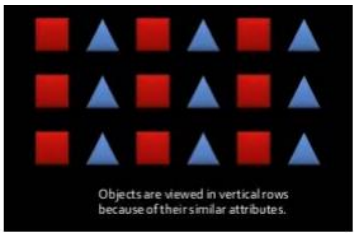



1.4. Leyes de percepción (Gestalt Laws)

- **Prägnanz Law**: tendencia a percibir formas simples


- **The law of closure**: tendencia a completar figuras regulares

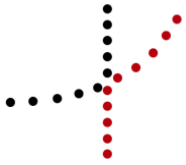

- **The law of similarity**: tendencia a agrupar elementos

- **The law of proximity**: tendencia a agrupar por proximidad


- **The law of symmetry**: tendencia a agrupar elementos simétricos

- **The law of continuity**: tendencia a seguir patrones



- **The law of common fate**: tendencias a agrupar elementos moviéndose en la misma dirección.

Más leyes de percepción:

- **Orientation Sensitivity**: líneas de dirección vertical y horizontal se distinguen bien, las direcciones oblicuas no tanto.
- **Pictorial superiority effec**t: a largo plazo, se recuerda más la información percibida en imágenes
- **Rule of thirds**: divide una imagen en nueve partes diferentes, utilizando dos líneas imaginarias paralelas y equiespaciadas de forma horizontal y dos más de las mismas características de forma vertical, y recomienda utilizar los puntos de intersección de estas líneas para distribuir los objetos de la escena.
- **Signal to noise ratio**: maximizar la señal (lo que queremos percibir) y minimizar el ruido (lo que acompaña).

1.5. Diseños con color

Color blindness: incapacidad de distinguir los colores de la misma manera que las personas sin discapacidades cromáticas, 5-10% de los hombres y 1-2% de las mujeres. Los tipos más comunes de color blindness:

- Deuteranopia (M cones): sensibilidad reducida a la luz verde (común).
- Protanopia (L cones): sensibilidad reducida a la luz roja (raro).
- Tritanopia (L cones): sensibilidad reducida a la luz azul (muy raro).
- Achromatopsia: no puede ver ningún color (muy raro).

Se identifica mediante el test de ishihara

Normas de color:

- Evitar colores muy saturados.
- Si queremos objetos del mismo color parezcan del mismo color, usar un color de fondo consistente.

- Utilizar colores con contraste para los textos.
- Usar colores diferentes sólo cuando correspondan a diferentes significados de los datos.
- Usar colores suaves y naturales para mostrar la mayor parte de la información y colores brillantes y/u oscuros para resaltar la información que requiere mayor atención.
- Usar paletas de colores (categóricas, secuenciales, divergentes…)

- Los componentes de las tablas y los gráficos que no son datos deben ser lo suficientemente visibles como para cumplir su función, pero no más, ya que una prominencia excesiva podría hacer que distraer la atención de los datos.
- Evitar usar la combinación de verde y rojo al mismo tiempo.
- Usar colores opuestos (color wheels).

2. Interacción

Teoría de la información (**entropía de Shannon**) mide la cantidad de información que debe transmitir un mensaje.

La **incertidumbre** en función de diversas posibilidades mide cual se producirá. La fórmula es:

$\log_2(M) = x\ bits = -\log_2(P)$, $M =$ número de posibilidades (equiprobables) y $P = 1/M\ =$ (probabilidad).

Dados n elementos:

$$incertidumbre = \sum_i^n(\log_2 m_i) = \log_2(\prod_i^n(M_i)).$$

La entropía de un mensaje (H) se calcula:

$H = -\sum_{i=0}^{n}(p_i * \log_2(p_i))$, donde $p_i =$ probabilidad de la información

Interferencia: cantidad mediana de información recibida es

$R = H(x) - Hy(x)$, donde $Hy(x)$ es la equivocación o entropía condicional de x cuando y es conocida.

Nota: $\log_a x + \log_a y = \log_a(x * y), \quad \log_a x - \log_a y = \log_a(x/y), \quad \log_a 1 = 0$

Hick-Hyman Law: tiempo para tomar una decisión (tiempo de reacción)

$RT = a + bH_T$, donde a, b son constantes y H_T es la información transmitida

$H_T = \log_2(n+1)$, donde n es el número de alternativas

Fitts' Law: establece una relación lineal entre el movimiento (MT) y la dificultad de la tarea.

$MT = a + b * ID$, donde a es el tiempo de movimiento, b la velocidad inherente y ID es el índice de dificultad

$ID = \log_2\left(\frac{2A}{W}\right)$, donde A la amplitud del movimiento y W amplitud del target

Variantes:

- Welford: $MT = a + b * \log_2\left(\frac{D+0.5W}{W}\right)$, donde D es la distancia del movimiento
- MacKenzie's: $MT = a + b * \log_2\left(\frac{D}{W} + 1\right)$

Movimientos en 2D:

- Crossman: $MT = a + b * \log_2\left(\frac{2D}{W}\right) + c * \log_2\left(\frac{2D}{H}\right)$
- Accot: $MT = a + b * \log_2\left(\sqrt{\left(\frac{D}{W}\right)^2 + \eta\left(\frac{D}{H}\right)^2} + 1\right)$
- Finger Fitts: $FFits = a + b\left[\log_2\left(\frac{xD}{W}\right)\right] + d\left[\log_2\left(\frac{e}{W}\right)\right]$

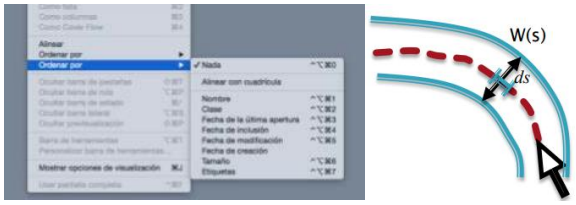
Crossing Law: movimiento descrito al cruzar con el cursor/dedo/lápiz por la pantalla. Pude ser:

- Continuo: se va de un lugar a otro de la pantalla sin levantar el dedo.

- Discreto: se puedes dejar de clicar durante el movimiento.

Sigue la misma caracterización que la ley de Fitts $MT = a + b * \log_2\left(\frac{D}{W} + 1\right)$, pero ahora W es la amplitud entre los dos objetivos.

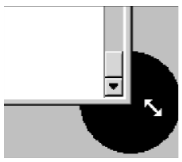
Steering Law:



Fitts' Law in UI Design Applications:

- Mantener los objetos relacionados cercanos y los objetos opuestos lejanos.
- Menús Pop-up: reducir la distancia a recorrer.
- Los pie menús: no debería tener oclusiones (son difíciles de diseñar)
- Perception: agrupar cosas puede mejorar con la distancia.

Accelerating Target Acquisition



- Expanding Targets:
  - Incrementar los tamaños de los objetos cercanos al cursor (barra de tareas MacOS)
  - Bubble Targets: incrementar la región seleccionable alrededor del target cuando el cursor esté cerca
  - Dynamic Bubble cursor: se basa en el Bubble cursor, pero el área de selección aumenta según la velocidad de este.
- Target Moving
  - Mover el target hacia el usuario.
  - Generar pop-ups al lado del usuario.
  - Sticky targets: atraer el puntero al target cuando está cerca.
- Control Display Ratio: relación entre la amplitud de los movimientos de la mano real del usuario y la amplitud de los movimientos del cursor virtual. Estrategias usadas: constante, dependiendo de la velocidad del cursor y dependiendo de la posición del cursor.

3. Dispositivos con los que interactuar

Pointing Devices

- Direct-control devices: actúan directamente sobre la pantalla. Algunas características: puede ser impreciso (dedos), más duraderos, permite gestos (multitáctil), no necesita extensión de pantalla…
  - Estrategia land-on: se hace el click donde apretamos sobre un punto de la pantalla, es más rápido pero es propenso a errores.
  - Estrategia lift-off: el click inicial sobre la pantalla crea un cursor que podemos arrastrar para tener más precisión, es un método más lento (teclado smartphones).
- Indirect-control devices: mouse, joystick, touchpad… Evitan el cansancio, son más precisos, no tapan la pantalla…

Teclados:

- QWERTY: las teclas que comúnmente se teclean juntas se colocan a gran distancia física (hecho para escribir en inglés).
- AZERTY: optimizado para escribir en francés.
- Dvorak: Las vocales están juntas, de la misma manera que todas las teclas más usadas se colocan en las posiciones más cómodas. Obtenemos una mejora del 30% y cometemos menos errores, pero sigue siendo una distribución optimizada para el inglés y tiene un nivel bajo de aceptación.

Problemas de los teclados táctiles: ocupan gran parte de la pantalla, tamaño limitado por la pantalla, en ocasiones con falta de feedback…

Layouts teclados móviles:

- Minuum: (1 o 2 dedos) agrupa las 3 filas de teclas en 1 requiere predictor/corrector y ocupa poca pantalla.
- Diagram-based layout: (1 dedo) optimiza las distancias.
- Single finger gesture: (1 dedo) se arrastra el dedo a través de las teclas (hace un recorrido), necesita un predictor que aprenda.
- KALQ: (2 dedos) divide el teclado en dos partes para cuando usas el teclado en horizontal.

Teclados virtuales: cosas a tener en cuenta:

- Corrección automática (evitarla en algunos campos).
- Auto-capitalización (cuidado en ciertos casos como al escribir la dirección de email).
- Layout apropiado según el tipo de dato de entrada.
- Teclados personalizados.
- Multiidioma.

Mobile Interaction Design

- Mantener una navegación simple, comunicando la actual sección de la App.
- Finger-friendly tap targets (manteniendo unos iconos de tamaño correcto y separados entre ellos).
- Progressive disclosure and cognitive load reduction.
- Hacer que el texto sea legible.
- Proveer de feedback en las interacciones.
- Reducir el desorden y reducir el contenido de elementos al mínimo.
- Reducir los inputs del usuario.
- Añadir funciones de confirmación para cambios definitivos (enviar un mensaje por correo, reiniciar el smartphone).
- No hacer que el usuario espere para ver el contenido.
- Usar gestos de manera prudente (informar al usuario de estos y no hacerlos muy complicados)

4. Test de usabilidad

Usabilidad: La medida en la que un producto puede ser utilizado por usuarios concretos para lograr objetivos específicos con eficacia, eficiencia y satisfacción en un contexto de uso específico. Para ser útil. La usabilidad tiene que ser específica, referirse a tareas concretas, entornos concretos y usuarios concretos.

Test de usabilidad tiene dos objetivos globales:

- Determinar problemas de usabilidad: descubrir, priorizar y resolver problemas de usabilidad y hacer pruebas iterativas.

- Medir el rendimiento de tareas: el desarrollo de los objetivos de la usabilidad y hacer pruebas iterativas para determinar si el producto ha alcanzado los objetivos.

Técnicas para pruebas:

- Pruebas **formales** de usabilidad, requieren un entorno controlado: dentro o fuera de una habitación para controlar la luminosidad (útil para estudiar la percepción), dispositivos usados (smartphone, ordenador…) y otras condiciones (como la conexión estable…).
  Se usan diferentes áreas y equipamiento:
  - Área de participantes dónde el experimento se lleva a cabo.
  - Área de observadores con cristal que sólo se ve desde un lado.
  - Área ejecutiva detrás del área de observación
  - Equipamiento: cámaras, micrófonos, teléfonos…

  Roles de la prueba de usabilidad:
  - **A**: administrador de la prueba
  - **B:** briefer
  - **CO:** operador de cámara
  - **DR:** el que registra los datos
  - **HD:** operador de servicios de asistencia técnica
  - **PE:** experto en productos
  - **S:** estadístico

  Flujo de trabajo de las pruebas de usabilidad

  - Preparación (A)
    - Comprensión del producto: finalidad del producto, partes a probar, tipos de usuarios… (A, PE).
    - Finalidad de la prueba: comparación del producto entre los sujetos… (A, S).
    - Medidas/Objetivos: número de iteraciones, recuento de errores, tiempos… (A, S).
  - Implementación
    - Selección de participantes: debe ser representativa (A)
    - Escenario de tareas: condiciones iniciales, pasos (A)
      - Core tasks: Features that everybody uses (write a text)
      - Peripheral tasks: Features used less often (table insertion)
    - Prueba piloto: miembros del equipo (A, B, CO, DR, HD)
    - Prueba (A, B, CO, DT, HD)
  - Presentación del informe (todo el equipo)
    - Análisis y evaluación de datos: frecuencia (número de usuarios con errores / número total de usuarios haciendo la prueba) y gravedad del problema (A, S)
    - Problemas/Medidas y recomendaciones (A, S, equipo)
    - Informe (A, S, equipo)
      - Describir priorizar los problemas de usabilidad.
      - Presentar mediciones cuantitativas.
- Pruebas de usabilidad simplificadas: que haga las pruebas una sola persona al principio, es mejor que 50 al final, el objetivo de las pruebas es informar sobre tu juicio.
  - Prueba de usabilidad Guerrilla
    - Llevar a alguien a una cafetería o espacio público y hacer que use una página web durante un par de minutos

    - Observar al usuario y hacerle preguntas abiertas, como por ejemplo: ¿Qué harías aquí?
    - Conocerlos un poco, invitarlos a una café.
    - Analizar los datos, considerando el tipo de audiencia.
- Prueba de usabilidad por 10 céntimos al día
    - Preparar algunas tareas a evaluar.
    - Escoger a una persona de la compañía como usuario.
    - Reunir a las partes interesadas en una sala para observar.
    - Dejar al usuario hacer el conjunto de tareas.
    - Capturar gestos, ratón, grabaciones…
    - Discutir durante la comida (comprar pizzas para todos)
    - Hacer un informe
- **Remote testing**
    - Es como una prueba tradicional, pero el participante y el facilitador se encuentran en países distintos. El participante hace las pruebas des de casa y el facilitador observa remotamente.
        - Ventajas: es barato y fácil, suele ser más rápido y podemos disponer de usuarios de todo el mundo.
        - Desventajas: no podemos observar el lenguaje corporal, difícil saber cuándo interactuar/hablar, la motivación de los participantes varia y no muestra mejores resultados que los estudios en persona.

      Dos tipos:
        - Sin moderador: los usuarios realizas las tareas completamente solos.
        - Con moderador: los usuarios tienen acceso al facilitador.
- **Evaluación heurística**: 3-5 expertos en usabilidad que evalúan la App o el UI, que usan principios predefinidos (heurísticos) y realzan los problemas de usabilidad antes que las pruebas con usuarios.
    - Ventajas: puede ser rápido y rentable (si disponemos de recursos), puede ser usado en las primeras fases del proceso de diseño, nos puede dar un estado de usabilidad completo de un producto y además es compatible con otros métodos de pruebas de usabilidad.
    - Proceso:
        - Recoger la interfaz de usuario.
        - Entender el negocio y las necesidades de los usuarios.
        - Entender las motivaciones de los usuarios y las tareas a lograr
        - Definir la heurística.
        - Usar un mínimo de 3 expertos.
        - Establecer un sistema de evaluación coherente.
        - Destacar el o los problemas y su clasificación.
        - Comparar y analizar los resultados de múltiples expertos.

Medidas de la planificación de las pruebas de usabilidad.

- Para descubrir un problema:
    - Centrarse en la priorización de los problemas.
        - Incluir la frecuencia con la que ocurren.
        - Probabilidad de ocurrencia en el uso normal.
        - Magnitud de impacto en los participantes.
    - Número de iteraciones previamente planificado.
    - Número de participantes pequeño, pero múltiples iteraciones.

- Para las pruebas de medición:
  - Categorías:
    - Indicadores de logro de metas (tasa de éxito y precisión)
    - Indicadores de ritmo de trabajo (velocidad y eficiencia)
    - Indicadores de operatividad (tasa de error y funcionalidad)
  - Medidas globales fundamentales:
    - Tasas de finalización de tareas con éxito.
    - Tiempos medios de realización de tareas.
    - Índices de satisfacción media de los participantes (tarea por tarea): existen cuestionarios estandarizados para ello.
    - Otras medidas podrían ser:
      - Número de tareas completadas dentro del límite de tiempo determinado, número de elecciones de menú erróneas, número de errores del usuario, número de errores repetidos (mismo usuario).
- Tras la elección de las medidas, el objetivo de la usabilidad se puede determinar.
  - Suele ser mejor establecer objetivos que hagan referencia a una medida (promedio) que a un percentil.
    - Las medidas muestrales extraídas de una distribución continuas son menos variables que las medianas muestrales.
    - A menos que falten datos debido a que los participantes fallen al completar las tareas.
  - Los objetivos percentiles requieren muestras de gran tamaño
    - No se puede medir con precisión el percentil 95 a menos que haya un mínimo de 20 mediciones.

5. Introducción VR y AR

5.1. Realidad virtual

Es la simulación interactiva por ordenador desde el punto de vista del participante, en el que la información sensorial se percibe sustituida y aumentada.

- Visualización interactiva: reproduce un mundo virtual que solo existe como modelo digital dentro del ordenador.
- Interacción implícita: El sistema decide que es lo que usuario quiere a partir de sus movimientos naturales.
- Inmersión: desconectar los sentidos del mundo real y conectarlos al entorno virtual.
  - Fusión: es la capacidad del cerebro humano de combinar dos imágenes con disparidad en una sola imagen con profundidad.

Sistemas de realidad virtual:

- Inmersivos
- Semi-inmersivos

Sistemas de configuración:

- Pantalla dinámica: las pantallas siguen los movimientos de la cabeza por lo que están fijas respecto a los ojos.
- Pantalla estática: es la configuración que siguen los sistemas basados en la proyección.

Interacción:

- Interacción 3D

    - Interacción HC en la que las tareas del usuario se llevan a cabo en un contexto espacial.
    - Utilización de dispositivos de entrada 3D o 2D con mapeos directos a 3D.
- Experiencia de usuario en 3D: interfaz de usuario que implica interacción en 3D.
- Técnicas de interacción en 3D: técnica diseñada para solucionar tareas, implica el uso de hardware y software.
- Selección 3D: tarea de selección en un entorno inmersivo.

Interacción VR y Selección 3D

- Hand extensión techniques: mapea la posición de la mano del usuario.
- Ray-based techniques: Usa la posición de la mano y otro elemento para indicar la orientación.

Presencia: es la paradoja que se forma al saber que lo que estás pensando, sintiendo y comportándote como si una situación fuera real sabiendo que no lo es.

5.2. Realidad aumentada

Es la combinación de una escena real y una escena virtual que aumenta la información de la escena. Se diferencia porque tenemos acceso a las dos escenas de manera simultánea.

El objetivo es mejorar el rendimiento del usuario y su percepción del mundo. Pero el reto es hacer que los usuarios sean capaces de diferenciar entre ambas escenas.

Realidad aumentada

- Sistema que aumenta la escena del mundo real.
- El usuario mantiene la sensación de presencia en el mundo real.
- Necesita de un mecanismo para combinar el mundo virtual y real.

Realidad virtual

- Entorno totalmente inmersivo.
- El sentido de la vista esta bajo control del sistema (a veces los auditivos y propioceptivos también)

La importancia del registro de los objetos:

- Los objetos virtuales generados por ordenador deben ser registrados con precisión con el mundo real en todas las dimensiones.
- Los errores en este registro impedirán que el usuario vea las imágenes reales y virtuales como fusionadas.
- El correcto registro debe mantenerse mientras el usuario se mueva por el entorno real.
- Las discrepancias o cambios en el registro provocan la distracción y pueden ser físicamente molestos.

Existen 3 maneras de presentar visualmente la realidad aumentada:

- Video see-trough: se sobrepone la imagen virtual y la real en un video.
- Optical see-trough: el usuario ve el mundo real directamente y se sobreponen ciertas imágenes (Google glasses)
- Proyección RA: Las imágenes son proyectadas directamente sobre los objetos físicos.

6. Principios de buenas representaciones gráficas

- La relación entre los datos y el gráfico debe ser alta, es decir que los datos sean claramente visibles y no se vean frustrados por los ejes, títulos…

- Usar un gráfico apropiadamente para nuestro objetivo. La mayoría de los gráficos presentados en Excel son malas elecciones, en particular nunca hay que utilizar el gráfico circular. Hay 3 tipos básicos de gráficos:
  - Gráficos de tendencia: sirven para destacar la tendencia de una serie temporal.
  - Gráficos de tamaño relativo: usar gráficos de barras paralelas, todas las barras deben estar ancladas al 0, deben ser de la misma anchura para que no se confundan por la diferencia de anchura por la longitud de las barras
  - Gráficos de composición: Aquí en vez de usar un gráfico circular (en forma de queso), debido a que no sabemos estimar ángulos, por ello, es mejor usar un gráfico de barras segmentado, en el que la barra (que va del 0 a 100%) está segmentado a trozos. Hay que poner los segmentos importantes en la parte superior o inferior de la barra (para que queden anclados en el 0 o el 100%) esto permite que el lector pueda estimar con precisión el porcentaje.

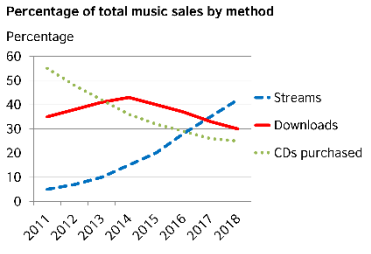
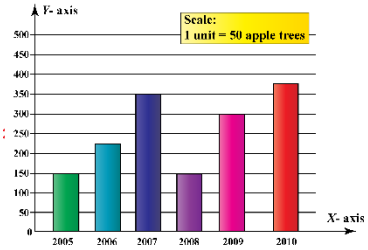
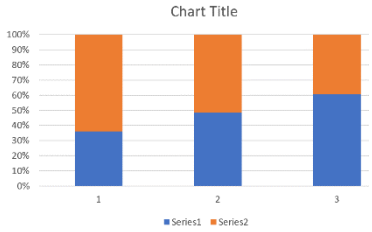


Gráfico de tendencia



Gráficos de tamaño relativo



Gráficos de composición

Errores comunes:

- Tipo de gráfico incorrecto
- Falta de texto: hay que etiquetar los ejes y las líneas además de que el gráfico debe tener un título.
- Escala inconsistente: la escala debe ser uniforme en todo el gráfico
- Punto cero mal colocado: la mayoría de la gente asume que el punto cero está en la parte inferior del gráfico.
- Efectos de gráfico deficientes: A menudo se añaden sombreados efectos 3D o efectos para animar el gráfico. En la mayoría de los casos son inútiles ya que distorsionan el gráfico y no aportan información.
- Confusión de área y longitud: si haces una imagen el doble de grande, parece que tiene 4 veces más área.
- No se ajusta a la inflación: Los importes en dólares deben ajustarse a la inflación. De lo contrario cualquier comparación es engañosa.
- Demasiada precisión: todos hemos visto gráficos que informan de que la cantidad de dinero recaudada es de 13.456.234,32 dólares. La mayoría de la gente no puede distinguir objetos con una resolución mejor que una parte entre cien. En consecuencia, dar 10 dígitos significativos es una tontería. Sería mucho mejor presentar esta cifra simplemente como 13 millones de dólares, es decir, eliminar todos los ceros adicionales y utilizar una escala adecuada.