

Apunts examen final d'IDI

1. Previ a il·luminació

Sobre els sòlids:

- Limitats per superfície
- Els moviments no modifiquen la forma

Models geomètrics

- Diferents maneres, a IDI els implementem amb triangles

Sobre implementació interna de triangles:

- Cada triangle a OpenGL és de la forma $(*v1, *v2, *v3, *n1, *n2, *n3, *m)$; és a dir, un triangle s'identifica amb tres referències a punts (emmagatzemades en un VBO de la forma $\langle x1, y1, z1, x2, y2, z2, \dots \rangle$), tres referències a normals i una referència a un material, definit per $\langle nom, ka, kd, ks, s \rangle$.

Com pintar un model a OpenGL?

- Crear en OpenGL un VAO amb les dades del model. Crear VBO amb els continguts de les coordenades dels vèrtexs (normals, colors, etc.).
- Guardar llista de vèrtex amb repetició en el VBO corresponent (+ guardar colors, normals...).
- Indicar quin VAO volem pintar i cridar a `glDrawArrays()` / `pinta_model()`.

Sobre processament de vèrtexs:

- Important diferenciar SCA (sistema coordenades aplicació / món / escena) de SCO (sistema de coordenades de l'observador).¹
- La posició de l'OBS no depèn del tipus d'òptica (obvi).
- Visualització: posició + orientació, òptica, fer la foto, emmarcar.
- El viewport és tota la finestra d'OpenGL.

Sobre pipelining d'OpenGL:

- Entra un vèrtex d'aplicació VA.
- Calculem vèrtex d'observador, VO, multiplicant per la View Matrix (definida per OBS + VRP + up).

¹ Un Patricio centrat a l'escena està a la posició (0, 0, 0) en SCA (perquè *està* centrat en l'escena); però, si mirem el Patricio des de davant, segurament el centre del Patricio serà a la coordenada (0, 0, -2R), perquè, des de la càmera (observador), ens hem de desplaçar 2 vegades el radi de l'escena per arribar al punt mitjà del Patricio.

- Passem a vèrtex de clipping amb la Project Matrix.

Vertex shader:

- Objectiu: $gl_Position = PM * VM * vec4(vèrtex, 1.0);$
- Recordem que si només fem $vec4(vèrtex, 1.0)$, estem veient amb una càmera amb l'OBS a (0, 0, 0) i volum de visió de cub de (-1, -1, -1) a (1, 1, 1).

Procés de visualització:

- Generar VAO.
- Generar matrius VM i PM, així com control del viewport.
- Pinta_model() / activar VAO + $glDrawArrays()$

Sobre models geomètrics i escenes:

- Una escena és un conjunt de models.
- Per visualitzar:
 - Fem un cop: per cada model, crear i omplir el seu VAO.
 - Cada cop que refresquem finestra: per cada model m, calculem la seva TG (transformacions geomètriques), indiquem a OpenGL que la volem usar ($modelMatrix(TG)$) i $pinta_model()$.
 - Cura, perquè apliquem la TG *abans* de tot!! Per tant, donat un vèrtex de model V, el passem a coordenades de clipping així²: $VC = PM VM TG V$.

Sobre els tres tipus de transformacions:

- Translació: $Translate(tx, ty, tz)$.
- Escalat: $Scale(sx, sy, sz)$. Normalment $sx = sy = sz$ per no deformar.
- Rotació: a teoria: $Rotatex(\varphi_x)$, $Rotatey(\varphi_y)$, $Rotatz(\varphi_z)$, tot i que també es fa servir $Rotate(\varphi, (x, y, z))$, on $(x, y, z) = (0, 0, 0)$ excepte un dels tres que és 1.
- Si volem fer translació + rotació, la matriu a aplicar és $R \cdot T$. Importa l'ordre!

Sobre rotacions i escalats:

- Un cop estem en l'origen (hem fet Translació(-t), on t és el centre/base de la capsa mínima contenidora), podem aplicar rotació + escalat o escalat + rotació, no importa!

Sobre càmera perspectiva:

- El volum de visió sempre és respecte l'OBS!!!
- Recordem que, en perspectiva, tenim l'angle d'obertura α (la meitat del FOV), zN , zF i relació d'aspecte del window (raw).
- El càlcul de l'altura del window és $aw = 2 zN \tan(\alpha)$, ja que $\tan(\alpha) = (hw/2) / zN$.
- L'amplada del window és $raw \cdot aw$.

² Recordem que el producte de matrius va sempre al revés: primer apliquem TG sobre V, després VM, etcètera.

- Per no tenir deformació cal $raw = rav$ (window és la meua imatge, viewport és el lloc de la pantalla que tinc per aquesta imatge).
- $Perspective(FOV, raw, zN, zF); projectMatrix(PM);$
- Quan multipliquem per la project matrix, tenim vèrtexs en coordenades de clipping. Per què això simplifica els càlculs? Perquè només hem de mirar, donat un $VC = (xc, yc, zc, wc)$ que xc estigui entre $-wc$ i $+wc$, yc entre $-wc$ i $+wc$, etc. Notem que wc és l'oposat a la coordenada z del vèrtex en coordenades d'observador (just abans de multiplicar per la Project Matrix!).
- Després d'aquest pas, es divideix el vector per wc , de manera que, un cop descartat els vèrtex fora del rang de visió, tenim $Vn = (xc/wc, yc/wc, zc/wc, 1)$. Això fa més grans els vèrtexs a prop de l'OBS i més petits si estan més allunyats: efecte de les vies del tren.
- El següent pas és el Device transform: simplement és la regla de tres de transformació del meu window al meu viewport: la cantonada de dalt a l'esquerra es correspon amb la cantonada de dalt a l'esquerra, etc. (sol ser en el rang $[-1, 1]$ a $[0, 1]$).

Sobre càmera en 3a persona + Euler:

- Tenim un OBS fora de la capsula mínima, un VRP definit al centre de la capsula i un up adient (normalment $(0, 1, 0)$).
- Per rotar l'escena, pensem els moviments a fer amb la capsula per tenir la visió "per defecte" (OBS a l'origen mirant cap a Z negatives).
- Assumim angles positius cap a la dreta i cap a dalt.
- Els passos són: Translació(centre capsula), RotacióY($-\psi$), RotacióX(θ), Translació($0, 0, -d$). Normalment $d = 2 R$.³

Càmera perspectiva sense retallar:

- Si $rav > 1$, no es retalla, no cal modificar el FOV. Només $raw^* = rav$.
- Si $rav < 1$, fem $raw = rav$ i incrementem l'angle d'obertura amb $FOV = 2 \alpha_V^*$, on $\alpha_V^* = \arctg(tg(\alpha_V) / rav)$.

Sobre òptica ortogonal:

- Paràmetres: $(l, r, b, t), zN, zF$.

³ Ull amb signes:

- Si s'ha calculat ψ positiu quan càmera gira cap a la dreta, serà un gir antihorari respecte eix Y de la càmera, per tant, matemàticament positiu; com girem els objectes en sentit contrari, cal posar $-\psi$ en el codi.
- Si s'ha calculat θ positiu quan pugem la càmera, serà un gir horari; per tant, matemàticament un gir negatiu; com objecte girarà en sentit contrari (antihorari), ja és correcte deixar signe positiu.

- Única diferència respecte perspectiva: el wc és sempre 1 (si el punt en coordenades de clipping està a Z més pròximes o més llunyanes a l'observador no importa).

Càmera ortogonal sense retallar:

- Si $rav > 1$ (massa ample), passem de $(-R, R, -R, R)$ a $(-rav \cdot R, rav \cdot R, -R, R)$.
- Si $rav < 1$ (massa alt), passem de $(-R, R, -R, R)$ a $(-R, R, -rav \cdot R, rav \cdot R)$.

Important sobre viewport:

- Relació d'aspecte = amplada / altura!!

Sobre *back-face culling*:

- Es fa amb coordenades de dispositiu (un cop ja les tenim al viewport).
- És un algorisme conservatiu: elimina *només* cares que no es veuran; però no totes!
- Funciona només amb un objecte opac i convex.
- Introdueix el paràmetre del color cv (color de vèrtex) al pipelining.

Sobre rasterització:

- Tres mètodes de rasterització: punts / contorn / triangle.
- El procés de rasterització converteix un vèrtex + color en Device = (xD, yD, zD, cv) en (xf, yf, zf, cf) .
- Amb un colorat constant, tenim $Cf = C1$ (el color d'un vèrtex, per exemple).
- Amb colorat de Gouraud / Gouraud shading / smooth shading, s'interpol·la el color.

Sobre el Z-buffer:

- L'últim pas és el Z-buffer.
- Tenim dos buffers de la mateixa mida que la pantalla.
- Anem apuntant colors i allunyament (la z de Device). Si tenim una z més petita, apuntem el color i allunyament en aquest píxel. Si és una z més gran, no fem res.
- Amb tot això ens cal al Fragment Shader: $\text{FragColor} = \text{vec4}(cf, 1)$;

Sobre inicialitzacions:

- Ens cal `glEnable(GL_CULL_FACE)` i `glEnable(GL_DEPTH_TEST)` a l'inicialitzar `GL()` i `glClear(GL_COLOR | GL_DEPTH)` al refresc de pantalla.

2. Il·luminació realista

Color d'un punt:

- El color va del material a la càmera ($P \rightarrow OBS$).
- Depèn de: fonts de llum, materials, altres objectes (no estudiat a IDI), posició de l'observador, medi pel qual es propaga (no estudiat).
- Només considerem llums R , G i B .
- Tenim models locals/empírics o models globals (traçat de raig, radiositat).
- Models locals és el que fem servir a IDI (no depenem dels altres objectes).

Sobre models empírics:

- Empíric ambient: és el Patricio pla. No es consideren focus de llum; la llum ambient és deguda a reflexions difuses. Tots els punts reben la mateixa aportació de llum. S'observa el mateix color en tots els punts d'un mateix objecte. En definitiva: $I_\lambda(P) = I_{a\lambda} \cdot k_{a\lambda}$. És a dir: llum ambient per reflexió ambient.
- Empíric difús (Lambert): el mateix però afegim el cosinus de l'angle entre la normal i la direcció punt P i focus de llum. $I_\lambda(P) = I_{f\lambda} \cdot k_{d\lambda} \cdot \cos(\Phi)$, si l'angle està entre -90° i 90° .
- Empíric especular (Phong). Ídem, però tenim un exponent de reflexió especular i l'angle de reflexió especular és el simètric de L ($P-OBS$) respecte la normal. Això fa un angle α respecte l'OBS i l'angle de reflexió. $I_\lambda(P) = I_{f\lambda} \cdot k_{s\lambda} \cdot \cos^n(\alpha)$.
- Fórmula general: $I_\lambda(P) = I_{a\lambda} + k_{d\lambda} + \sum_i (I_{f_i\lambda} k_{d\lambda} \cos(\Phi_i)) + \sum_i (I_{f_i\lambda} k_{s\lambda} \cos^n(\alpha^i))$.

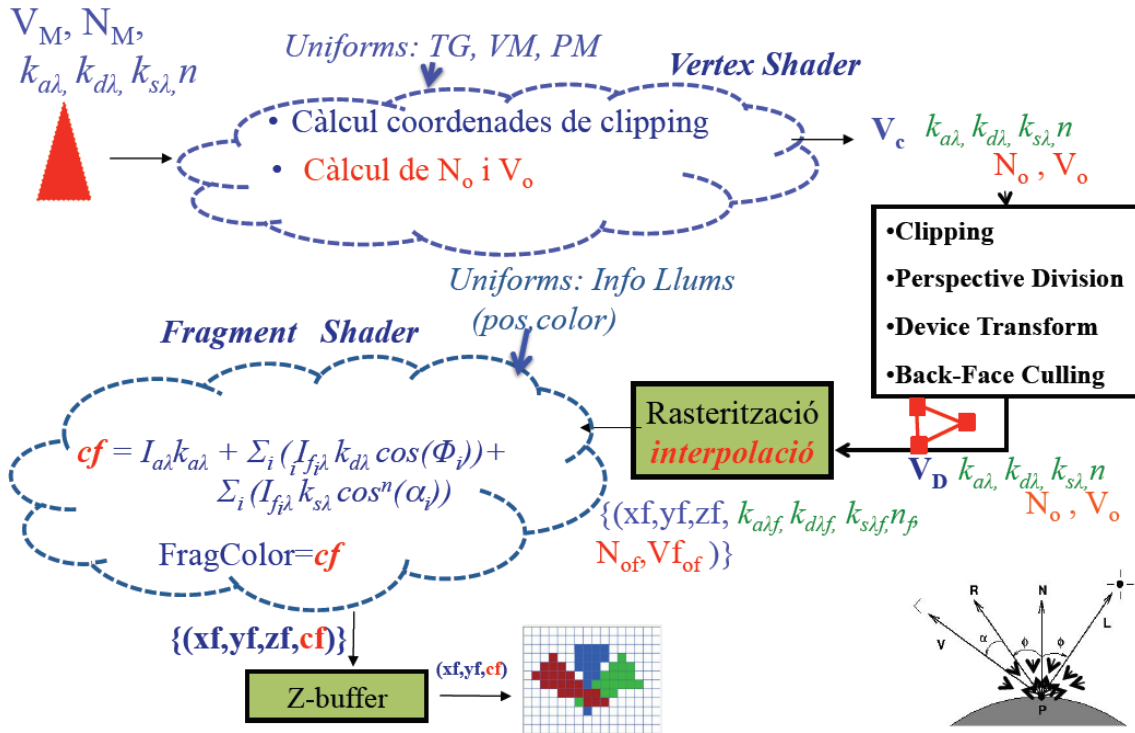
Sobre models globals:

- Traçat de raig: consideren focus de llum puntuals i altres objectes existents en l'escena però només transmissions especulars. Poden simular ombres, transparències i miralls. Són costosos.
- Models de radiositat: el focus de llum és un objecte qualsevol de l'escena. Els objectes només produeixen reflexions difuses pures. La radiositat no depèn de l'observador (perquè tot és difús). Poden modelar ombres i penombres però no miralls ni transparències. Són els més costosos.

Sobre suavitzat d'arestes:

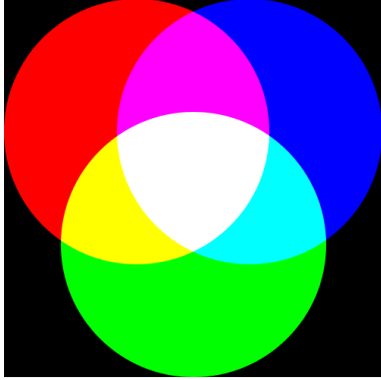

- Millor normals per vèrtexs que per cares.

Sobre uniforms i shaders:



3. Color

Diferenciem:

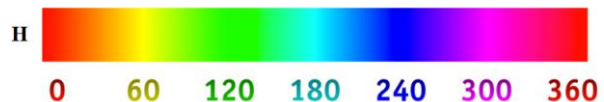
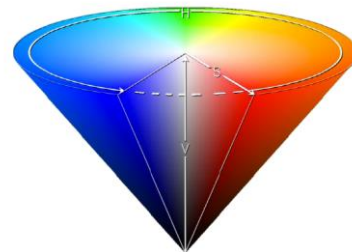
	Síntesi additiva	Síntesi subtractiva
Mitjà	Llum	Paper
Color base	Negre	Blanc
Esquema de colors	RGB	CMY(K)
Esquema visual		
	(R, G, B)	(C, M, Y)
Blanc	(1, 1, 1)	(0, 0, 0)
Negre	(0, 0, 0)	(1, 1, 1)
Red	(1, 0, 0)	(0, 1, 1)
Green	(0, 1, 0)	(1, 0, 1)
Blue	(0, 0, 1)	(1, 1, 0)
Cyan	(0, 1, 1)	(1, 0, 0)
Magenta	(1, 0, 1)	(0, 1, 0)
Yellow	(1, 1, 0)	(0, 0, 1)
Conversió	$(A, B, C) \leftrightarrow (1 - A, 1 - B, 1 - C)$	

Sobre CMY(K):

- La component K (black) cal perquè les tintes C, M i Y solen tenir impureses i, juntes, no fan un negre pur a la vida real.

Model HSV:

- Hue (*matiz*) + Saturació + Valor.
- Hue és el color (roig / blau / ...).
- La saturació indica la quantitat de gris del color.
- El valor (o intensitat).



Model CIE:

- Basat en estímuls R, G i B als cons i bastons.
- És una funció $f(R, G, B)$; s'acota en el pla $X + Y + Z = 1$

4. Usabilitat i més coses

Introducció a la HCI

HCl:

- Human-Computer Interface: com interaccionen humans i computadors.
- Va d'entendre tecnologies interactives i pràctiques dels humans.
- La HCI és principalment (però no només!!) usabilitat.

User Interfaces:

- User Interfaces: eines i mètodes que es fan servir per comunicar humans i sistemes.

Usabilitat:

- La usabilitat és (def. ISO): “habilitat en què un producte es pot fer servir per un grup específic d’usuaris (!!!!) per dur a terme tasques específiques (!!!!) de manera efectiva, eficient, i amb satisfacció en un entorn d’ús específic (!!!!)”.
- La usabilitat sempre es refereix a un grup concret d’usuaris i a un usuari concret d’una aplicació!!!!
- Per tant: usabilitat = eficàcia (es fa allò que volem) + eficiència (bon ús de recursos) + satisfacció.

User experience:

- La UX és quant a dispositiu: volem crear una bona sensació al fer servir el nostre dispositiu (!!!!).

Interaction design:

- Com interactuem amb els dispositius (per exemple: mida de pantalles dels mòbils, teclat físic / virtual en pantalla, etc.).
- Regla del polze (imatge).

Més d'HCl:

- En escriptoris: pantalles grans, espai per tot, ratolí, teclat, resolució gran.
- En mòbils: pantalla menuda, pensar com ajustar contingut a l'espai reduït. Problemes amb les notifikacions. Interacció amb dit/stylus. Sense teclat. Poca resolució i limitacions de software.
- En tablets: mida més gran, però hem de pensar encara com fer cabre les coses. Semblant als mòbils.



Sobre aplicaciones:

- Diferenciem web apps i apps natives.

- Web apps: develop once & deploy everywhere, fàcil d'actualitzar, llenguatges fàcils; però, no és tan rica en UI, protocols ineficients i insegurs. Solen estar dissenyades per a grans pantalles amb ratolins.
- Aplicacions natives: UI més rica, molts controls, accessos a recursos locals fàcils i segurs; menys varietat de llenguatges i eines de programació, dissenyades per pantalles petites i amb control de *touch*. Però, no hi ha accés universal (Android vs iOS), difícil d'actualitzar, menys general que el development d'escriptori.

Principis de disseny

Basat en Bruce Tognazzini:

- Interfícies efectives: donar als usuaris sensació de control / ocultar treball intern del sistema.
- Aplicacions efectives: màxim aprofitament de l'esforç dels usuaris.

Estètica:

- *Fashion should never TRUMP usability.*
- En qualsevol cas, tests d'usability.
- Important molt de contrast text / fons.

Anticipació:

- Anticipar-se a les accions de l'usuari.

Autonomia:

- Donem una mica de control (customization) a l'usuari.
- Els usuaris poden fer les seves decisions.
- Però informem l'usuari de la informació necessària per mantenir el control: estat actual, tasques...
- Ús de progress indicators, spinners...

Color:

- Tindre cura dels discapacitats quant al color.

Consistència:

- Molts nivells de consistència: plataforma / gamma de productes / in-app / estructures visibles / estructures invisibles / comportament d'usuari.
- Consistència de plataforma: guies de consistència (UX Android, iOS) i dins de la mateixa companyia.
- Tenim un *look & feel* comú entre productes i serveis.
- Consistència entre gammes de productes (Microsoft Office, Google apps).
- Estructures visibles: icones, símbols, posicionament similar...
- Estructures invisibles (per exemple, scroll bars invisibles en Macintosh).
- Comportament d'usuari: mai canviïs el significat d'una acció habitual!!! Canviar un gest après és molt frustrant!

Consistència:

- Inconsistència induïda: fem els objectes diferents i actuen de manera diferent.
- Forcem canvis visuals si les accions que triggerejen aquests objectes són diferents.
- Continuïtat induïda: cal esforçar-se per la continuïtat, no per la consistència.
- Les noves versions han de canviar grans àrees (noves característiques). Han de ser bastant diferents de la versió prèvia.
- Consistència amb user expectations: no forçis l'usuari a aprendre una nova manera de fer alguna cosa.

Valors per defecte:

- Si hi ha avantatge, fer-lo servir. Si no, no.

Discoverability:

- Generar la il·lusió de simplicitat no simplifica les coses.
- Si l'usuari no ho troba, no existeix!
- Fer servir active-discovery: oferir característiques no testades a usuaris experts.
- Tots els controls han de ser visibles (excepció: teclats en smartphones...).

Eficiència:

- Mirar la productivitat de l'usuari, no de l'ordinador!
- Els errors han d'ajudar. Explica què està malament, digues a l'usuari què fer.

Interfícies explorables:

- L'usuari ha de sentir-se lliure explorant les interfícies.
 - Fer accions reversibles
 - Sempre habilitar un *undo*.

Informar l'usuari:

- Cursor de ratolí animat si triguem $\frac{1}{2}$ – 2 segons. Més de 2 segons, digues a l'usuari quant trigarà. Si > 5 segons, fer servir un progress. Si > 10 segons, informar l'usuari i entretenir-lo. Si > 15 segons, afegir indicador visual per saber quan hem acabat.

Més principis:

- Assegurar-se que l'usuari mai perd el seu treball.

Principis universals i lleis de percepció en disseny

Regla 80/20:

- El 80% dels efectes generats en un sistema gran estan causats pel 20% de variables en aquest sistema (*80% of application usage on only 20% of its features*).

Test d'usabilitat-estètica:

- És important l'estètica. Estètica bona = més fàcil d'usar.

Alineació correcta:

- Coherència amb l'alineació.

Chunking:

- Un chunk és una unitat d'informació en la memòria a curt termini.
- Chunking vol dir que ubiquem la informació de manera que s'acomoda als límits dels humans a processar bits d'informació.
- Millor chunks petits que grans.
- Nombre màgic: 7 ± 2 .

Color:

- Reforça el layout de disseny.
- Com a molt, 5 colors.
- Fer servir un color principal i alternatiu.
- Simbolisme: diferent significat de colors en cultures.

Principi LATCH: la informació s'organitza en funció de:

- Location: Information comes from different places.
- Alphabet: Usually for large amounts of data (words in dictionary...).
- Time: Events with fixed durations. (meeting schedules).
- Category: To classify goods/elements of similar importance. Suitable for shops...
- Hierarchy: By magnitude, order of importance.

Garbage-in, garbage-out (GIGO):

- Necessari tenir cura dels inputs. Han de estar ben formatats! Com més "neta" sigui l'entrada d'un usuari, millor.

Representació icònica:

- Similitud (\Rightarrow), exemple (relacionat amb imatge), simbòlic (alt nivell d'abstracció, *candado*), arbirtari (símbol nuclear).

Lleis de Gestalt:


1. Llei de Prägnanz: llei de la bona figura, simplicitat. Formes simples.



2. Llei de tancament: 

3. Llei de similitud: agrupar elements en entitats col·lectives o totalitàries.



4. Llei de proximitat: diferència espai per veure grups 

5. Llei de simetria: [] ().

6. Llei de continuïtat: els elements en una ruta cinètica es perceben relacionats.

7. Llei de common fate: els elements amb la mateixa direcció o moviment es perceben com a una unitat col·lectiva.

Orientation sensitivity:

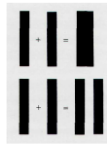
- Les orientacions vertical/horitzontal estan bé. Però les obliqües costen.
- Dos efectes:
 - Efecte d'obliquïtat: ens costa, a les neurones, interpretar dissenys amb orientació obliqua.
 - Pop-out effect: destaquen els elements amb bona separació de 30°.

Superioritat de les imatges en front dels textos.

Rule of thirds (imatge en 9 parts).

Ratio senyal a soroll:


- La idea és maximitzar senyal i minimitzar soroll.
- Fem el disseny simple.



Principi 1 + 1 = 3:

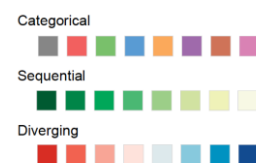
Percepció i color

Problemes:

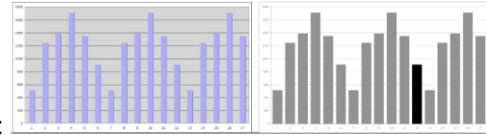
- Deuteranopia (poc verd).
- Protanopia (poc vermell).
- Trianopia (poc blau).
- Achromatopsia (no veu cap color).
- Es veuen amb un test d'Ishihara: .

Consells:

- Forcem contrast fons / color de text.
- Evitem colors adjacents o de lluminositat similar.
- El fons ha de ser consistent.
- Contrast colors light amb colors dark.
- Fer servir colors per a alguna cosa, no malbaratar-los!
- Usem colors softs per a la majoria d'informació i colors brillants/dark per a informació que requereix més atenció.



- Sense ordre / ordre sense valor neutral / amb valor neutral:



- Evitar soroll als gràfics (de-emphasizing):
- Evitem combinació verd + roig al mateix esquema (solució roig + gris).
- Evitar 3D.
- Fer servir colors oposats, anàlegs (taronja / verd / groc...), tríades (porpra / taronja / blau), quatriades (porpra / taronja / verd / blau, és el més estrany).

Usabilitat i problemes habituals

Problemes de disseny habituals:

- Es repeteixen molt, són difícils d'avaluar pel dissenyador.
- Non-standard GUI controls.
- Elements que semblen GUI control però que no ho són (coses que semblen botons però no ho són, o links!!).
- Inconsistència (Off / on amb vermell / verd, per exemple).
- No perceived affordance: menús invisibles, no saber què fer... Gestos: l'absència de senyals en la pantalla fa els gests impossibles de descobrir. Els usuaris han de memoritzar els gests. Mala idea!
- Targets de click massa petits.
- Falta de feedback: els usuaris han de saber l'estat de l'aplicació.
- Falta de progress indicator.
- Missatges d'error dolents.
- Demanar la mateixa informació dos cops.
- Falta de valors per defecte.
- Falta informació per saber com funciona l'aplicació: un *help*.
- Organitzar la informació d'acord amb el disseny de l'aplicació.
- Botons de reset en formularis webs.
- Crear dissenys passa entusiastes en disseny (millor no ocultar la navegació).
- Deixar la UX per a després, deixar el feedback per a més tard.

Interacció de disseny i avaluació

- Entropia de Shannon: la quantitat d'informació per trametre un missatge.
- Font + transmitter + canal + receiver.
- Amb {A, B, C} tenim $\log_2(3)$.
- Amb A1, A2, B1, B2, C1, C2 tenim suma d'incerteses = producte de continguts dels logaritmes = $\log_2(6)$.
- La probabilitat és P. Per tant $\log_2(M) = \log_2(1/P) = -\log_2(P)$ (es diu la "sorpresa").
- El sumatori de les p_i és 1.

- La entropia és, en definitiva $H = -\sum_{i=1}^N p_i \log_2 p_i$, on N és el nombre de caràcters.
- Interferència: la quantitat mitjana d'informació rebuda és $R = H(x) - H_y(x)$, on $H_y(x)$ és l'equivocació o entropia condicional de x quan y és coneguda.

Llei de Hick-Hyman:

- Descriu el temps de decisió en funció de la informació.
- Costa més respondre a un estímul quan aquest pertany a un gran conjunt, a diferència de quan hi ha menys estímuls.
- Per fer una decisió, el reaction time és $RT = a + b H_t$, on a i b són constants i H_t és la informació tramesa.
- Per tant, $RT = a + b \log_2 (n + 1)$. Aquest +1 és per la incertesa de si respondre o no.

Llei de Fitts:

- Relació linear entre la dificultat d'una tasca i el temps de moviment.
- Task difficulty = ID = $\log_2 (2A / W)$, on A és l'amplitud del moviment i W és l'amplada del target.
- Movement time $MT = a + b ID$. Paràmetres a i b .
- No funciona molt bé per a targets petits.
- És vàlida per a ratolins, joysticks, dit, stylus... però falla en altres.

Llei de Fitts. Variant de MacKenzie:

- $MT = a + b \cdot ID = a + b \cdot \log_2 \left(\frac{D}{W} + 1 \right)$

Implicacions de la llei de Fitts:

- Les cantonades tenen *width* infinit, per tant, són fàcils d'aplegar.
- Mantenir les coses relacionades a prop.
- Els filtres han d'estar pròxims a la barra de cerca.
- Els elements oposats separats.
- Menús de pop-up: redueix distància de travel, només usats per experts!
- Incrementem percepció si agrupem les coses amb buit pel mig.
- Com més gran és un botó, més usable és.
- Fer les coses destructives / delicades més difícils: fem servir slides en comptes de buttons per apagar el nostre mòbil.

Accelerar targets:



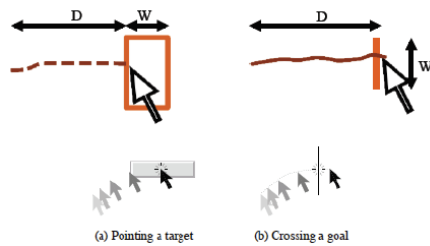
- Bubble targets: pot distraure ().
- Augmentar mida amb coses properes al cursor
- Bubble cursor (fa servir diagrames de Voronoi).
- Attract pointer: en 3D.



Control-display ratio:

- Hi ha un mapeig moviment mà a la vida real i moviment cursor (parlem de 3D). Com fem aquest mapeig? Constant, depèn velocitat mouse / cursor.

Llei de crossing:



- Pot ser continu o discret (soltem el cursor), ortogonal o colinear.
- Crossing i pointing són bastant semblants. Menys error en crossing, però.

Llei de Steering:

- Navegació per menús / navegació 3D / drag and drop / dibuix.

- $T_s = a + b \int_c \frac{ds}{W(s)}$, on ds és la mida del path, C longitud del path i $W(s)$ és el path al punt s .

Dispositius de control directe:

- Problemes: estratègia land-on: seleccionar a un clicking point, feedback més ràpid, propens a errors.
- Estratègia lift-off: el click inicial crea el cursor; en arrossegar tenim precisió.
- Són durables.

Dispositius de control indirecte (ratolí, trackball, joystick...).

Selecció 3D:

- Amb extensió de mà o basat en rajos.
- Amb extensió de mà: més intuïtiu si el món virtual representa el real.
- Amb rajos: posició de la mà + orientació del canell + posició del cap + direcció de la mà.

Teclats

QWERTY:

- Les tecles més usades estan lluny unes de les altres.
- Està dissenyat per l'anglès.

AZERTY:

- Dissenyat pel francès.

Disseny de teclats:

- Balancejar mà dreta i esquerra.
- Maximize the load on the home row.

DVORAK:

- Vocals en un costat

Teclats partits:

- No proporcionen cap millora de velocitat.

- Maximitzar la freqüència d'alternança entre seqüències de mans.
- Minimitzar la freqüència de fer servir el mateix dit.
- El temps per moure d'una tecla a una altra depèn de la distància de la tecla i l'amplada de les tecles. $MT = a + b \log_2 (D / W + 1)$.
- Per anar més ràpids, podem canviar la mida del target, canviar la distància o canviar la velocitat del cursor.

Problemes comuns:

- Autocorrecció. Millor no fer-la servir.
- Majúscula automàtica (per exemple, en e-mails).
- Suport de diferents idiomes.

5. Tests d'usabilitat

Tests d'usabilitat

Conceptes:

- Test d'usabilitat orientat a tasques, entorns i usuaris particulars.
- Com testejar? La facilitat d'ús és inversament proporcional al nombre i la severitat de dificultats que té la gent en usar un software. Així que anem a veure dificultats.

Tests d'usabilitat:

- Dos tipus: determinar problemes d'usabilitat (descobrir + prioritzar + resolució dels problemes; pot ser informal – test iteratiu) o mesurar la performance d'una tasca (per exemple, selecció 3D. Inclou dues tasques: desenvolupament d'objectius d'usabilitat i test iteratiu per determinar si el producte testejat ha complert els objectius).
- Varietat de tests d'usabilitat: molt informals o molt formals, mirar directament, a través d'un cristall, mirar un o dos participants alhora, think aloud...

Tècniques:

- Think aloud: es pot aplicar a quasi qualsevol test. Funciona millor per a parelles.
- Remote testing: introdueix entorns familiars + participants de països diferents... Però pot ser difícil feedback del participant

Laboratoris d'usabilitat:

- Dins / fora d'una habitació, dispositius usats, condicions d'il·luminació, qualitat de la connexió.
- A prova de sorolls externs.
- Diferents àrees i equipament: àrea del participant, àrea de l'observador, àrea de l'executiu, videocàmeres, micròfons, telèfons...

Rols de test:

- Administrador del test: dissenya l'estudi d'usabilitat. Especifica les condicions inicials per al test i els codis per fer login de les dades. Conduïx les reviews dels tests, fa l'anàlisi de les dades, fa la presentació final.
- Briefer: interactua amb els participants, fa que en un estudi think-aloud el participant parli, és familiar amb el producte per decidir què respondre als participants.
- Operador de càmera
- Data recorder: escriu notes durant un test (les dades principals de l'estudi), molt difícil, usa data-logging software.
- Help desk operator: reemplaça un operador d'escriptori real.
- Product expert: manté el producte i ofereix suport tècnic durant el test.

- Estadístic

Planificació de tests:

- Abans de començar, l'administrador ha de saber el propòsit del producte, decidir quines parts estan preparades per fer testing, determinar els tipus de persones que usaran el producte, determinar l'ús donat al producte, les condicions d'ús del producte
- Propòsit del producte: mesurar vs. identificar problemes d'usabilitat. Podem fer servir productes de la competència (o no).
- Per descobrir problemes: prioritzem problemes amb molta freqüència, que importin molt en l'ús normal, per magnitud d'impacte, facilitat de correcció. Nombre d'iteracions fixat *a priori*. Pocs participants però moltes iteracions.
- Per a tests de mesurar: fem categories: *ratios* d'èxit / fracàs i precisió, velocitat, eficiència, indicadors d'operabilitat, indicadors d'adquisició de coneixement (learnability i learning rate).
- En aquests tests tenim mesures globals fonamentals: ratio d'acabament exitós de tasques, temps mitjà d'acabament de tasques, puntuacions de satisfacció mitjana...

Participants / testers:

- Els participants estan determinats per l'administrador. Deuen encaixar amb els requeriments de l'administrador (nivell d'educació, sexe, edat...), es poden obtenir d'agències de treball. El número de participants depèn de diners/temps i tipus d'estudi single-shot (llarg) vs. iteratiu (més curt).
- Amb 5 testers tenim el 80% dels problemes; amb 3 la majoria.
- El millor són ≤ 5 testers amb tasques com més petites millor.
- El tercer participant ens dona menys informació que el primer o el segon. A partir del cinquè estàs perdent el temps.

Implementació:

- Els escenaris han de ser representatius (core tasks = característiques que tothom fa servir [escriure un text] vs. pheriferal tasks = característiques més estranyes [inserció d'una taula]). Definim els escenaris amb unes condicions inicials i especifiquem *què* fer i *per què* fer-ho. No tots els participants han de tenir els mateixos escenaris (pot dependre en el perfil de l'usuari).
- Procediment:
 - Introducció: proposta del test, confidencialitat.
 - Task performance: completar qüestions prèvies i formularis, completar entrenament (si cal) i dur a terme les tasques. Normalment, no donem ajut. Referim els usuaris a la documentació. Si cal, donem l'ajut, però anotem la tasca com a fallida. Evitem respondre directament a les preguntes. Evitar *sesgar* les respostes dels participants.
 - Passar un qüestionari de satisfacció al final de cada escenari.

- Després dels escenaris, un qüestionari final.
- Test pilot: el test d'usabilitat s'ha de provar. Normalment, un membre de l'equip d'usabilitat el prova.

Reporting:

- Fer un report dels resultats: descriure i prioritzar els problemes d'usabilitat. Presentar mesures quantitatives. (Hauria d'haver una recomanació).
- Evaluació dels problemes: freqüència, severitat dels problemes.
- Problemes d'usabilitat: haurien d'indicar la severitat. Es poden classificar: fallades (errors deguts a una intenció incorrecta), slips (errors deguts a una intenció correcta però una acció incorrecta). El nivell d'experiència (expertise) no afecta al nombre d'errors – però afecta a com ràpid se solucionen.
- Primer fem els canvis globals.
- S'ha de comprovar l'existència d'un *help*.
- Ha de donar mitjanes, desviacions estàndards i intervals de confiança.
- Conclusions: no donar un gran nombre de categories, tractar separatament la freqüència de la gravetat. No oblidar-se de destacar els descobriments positius.

Problem evaluation:

- Segons Jeff Rubin, nivells de problemes: 4 – unusable (no fa servir una part per com està dissenyada o implementada o no pot trobar-la), 3 – severe (intentarà fer-la servir, però tindrà moltes dificultats), 2 – moderate (l'usuari podrà fer-lo servir en la majoria de casos, però haurà d'invertir certs esforços a solucionar-lo), 1 – irritant (problemes estètics, només intermitentment...).
- Segons Dumas: level 1 – impedeix l'acabament de la tasca, level 2 – crea un retard important i frustració, level 3 – poca importància d'usabilitat, level 4 – suggeriments subtils i possibles millores.

Mètodes quantitatius i qualitatius per a experiments humà-participant

Introducció:

- Motivació dels tests: avaluar com els humans perceben, manipulen i raonen amb aplicacions o webs.
- És important fer-ho abans de llençar qualsevol producte.

Validesa dels experiments:

- Realment el nostre experiment mesura el que volem mesurar?
- Són els nostres resultats fiables? Si fem l'experiment un altre cop, tindrem els mateixos resultats? Tindran altres els mateixos resultats?
- Variables independents: el que estudia el nostre experiment. Poden ser *between-subjects* (cada participant veu un nivell diferent de la variable; per exemple, la meitat de persones en *stereo* i l'altra en *mono*) o *within-subjects* (cada participant veu tots els nivells d'una variable; per exemple, tots veuen ambdós *stereo* i *mono*).

- Variables dependents: el que mesura el nostre experiment. S'assumeix que se-
ran afectades per les variables independents. Cal que siguin mesurables (temps,
nombre de tasques completes, nombre d'errors, respostes a enquestes, punтуа-
cions...).
- Variables *de confusió*: no s'estudien, però afecten l'experiment (qualitat del *mono*
vs. el de l'*stereo*).

Disseny d'experiments:

- Per evitar desviacions, tenim cura del *learning* i de la fatiga:
 - Després de N repeticions, un usuari soluciona més ràpid un problema.
 - Després de N repeticions, l'usuari pot sofrir fatiga.
- Disseny counterbalancing (*de contrapès*): evitem *learning* i fatiga per fer les tas-
ques en ordre aleatori; no totalment *aleatori*, sinó ordenades de manera adient
(variacions sistemàtiques).
- Si tenim dues pàgines web a estudiar en dos dispositius diferents i amb quatre
repeticions, això són $3 \times 2 \times 4 = 24$ tasques. Creix factorialment! Per tant, cada
usuari fa un test diferent.
- Fem servir *latin squares*: evitem *learning* i fatiga. En el cas anterior fem un qua-
drat llatí 2×2 , i dos quadrats 3×3 . Per tant, tenim un rectangle gran 6×3 i un
altre 2×2 . Si fem producte cartesià, tenim permutacions per 12 persones.

Anàlisi de dades:

- Estadística descriptiva: descriure i explorar dades. Gràfics, histogrames... En-
tendre la distribució de les dades i pensar en tests de significació.
- Estadística inferencial: detectar relacions en les dades. Inferir característiques de
la població a partir de característiques dels exemples (com a PE).
- Fem servir *charts* (diagrames).
- Evitem pastissos (*pie charts*)!!!, projeccions 3D, mantenim bona ràtio data/chart,
fem servir el diagrama adient per a l'ús adequat.
- Tipus de diagrames: *trend*, *relative size* i *composition graphs*.
- *Chartjunk*: merda visual innecessària o confusa als charts i gràfics.
- Problemes típics dels graphs:
 - Tipus de gràfic equivocat.
 - Falta informació (títol, escala, etiquetes...).
 - Escala inconsistent.
 - Zero mal col·locat.
 - Efectes de gràfics pobres (*ducks* = element innecari, ombres).
 - No hi ha ajustament per inflació.
 - Massa precisió.
 - Poc balanç *data ink ratio* = *data ink* / *total ink to print graphic*.
- En comptes de *pie charts*, fem servir barres segmentades (de 0 % a 100 %)

6. Realitat virtual i augmentada⁴

Realitat virtual

Definició:

- La realitat virtual és una simulació interactiva per computador des del punt de vista del participant, en la qual se substitueix o s'augmenta la informació sensorial que rep

Elements:

- Simulació interactiva: Reproduir un món que només existeix a l'ordinador.
- Immersió sensorial: Desconnectar els sentits del món real per tal de portar-los cap al món virtual. "Visual immersion": Els objectes existeixen independentment del dispositiu de visualització. Utilitzem visió estereoscòpica per reproduir el món.
- Interacció implícita: El sistema decideix què vol l'usuari a partir dels seus moviments.

Elements de la RV:

- Perifèrics d'entrada: Capturen les accions del participant i envien aquesta informació al computador.
- Computador: Realitza la simulació a partir del model geomètric en 3D i les dades que ha rebut de la simulació física i sensorial.
- Model Geomètric 3D: Permet fer el càlcul d'imatges, so, col·lisions, etc.
- Perifèrics de sortida: Tradueixen els senyals d'àudio, vídeo, etc. generats pel computador en estímuls pels òrgans dels sentits (visuals, força/tacte, àudio, equilibri...).
- Programari de tractament de dades d'entrada: Llegeixen i processen la informació que proporcionen els sensors.
- Software de simulació física: S'encarreguen de les modificacions pertinents en la representació digital de l'escena, a partir de les accions de l'usuari i de l'evolució del sistema.
- Software de simulació sensorial: Representen digitalment les imatges, sons, etc. que el maquinari s'encarregarà de traduir a senyals i finalment a estímuls pels sentits. El més important és el de simulació visual i auditiva, i després la tàctil.
- 3D no implica RV, en canvi, RV sí implica 3D però no presencia (sensació d'estar allà).

Realitat augmentada

Introducció:

- Realitat augmentada: és la combinació de l'escena real i la virtual, tenim accés al món virtual i real al mateix temps.

⁴ Per Víctor Murciano Durán ©.

- Objectiu: millorar el rendiment i la percepció del món però mantenint la diferència entre el real i el virtual.
- A diferència de la RV, la RA no té una immersió total.
- És molt important registrar els objectes de la realitat per tal d'evitar problemes en aplicar la RA.

Conceptes:

- *Video see-through*: combina vídeo en temps real amb càmeres frontals que posen imatges virtuals.
- *Optical see-through*: l'usuari veu el món real directament, un exemple són les *Google Glasses* (requereix un calibratge constant, no es pot posar efectes oclusius).
- *AR projection*: les imatges són projectades directament sobre els objectes físics. No és necessari cap aparell visual per veure'ls, però cal el calibratge dels projectors per les diferents distàncies a més que només es poden utilitzar en interiors per la seva brillantor tan baixa.