TRABAJO FIN DE GRADO

INGENIERÍA EN ...

Titulo del Proyecto

Subtitulo del Proyecto

Autor

Nombre Apellido1 Apellido2 (alumno)

Directores

Nombre Apellido1 Apellido2 (tutor1)

Nombre Apellido1 Apellido2 (tutor2)

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, mes de 201

Título del proyecto

Subtítulo del proyecto.

Autor
Nombre Apellido1 Apellido2 (alumno)

Directores
Nombre Apellido1 Apellido2 (tutor1)
Nombre Apellido1 Apellido2 (tutor2)

Título del Proyecto: Subtítulo del proyecto

Nombre Apellido1 Apellido2 (alumno)

Palabras clave: palabra_clave1, palabra_clave2, palabra_clave3, ...

Resumen

Poner aquí el resumen.

Project Title: Project Subtitle

First name, Family name (student)

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Write here the abstract in English.

Yo, **Nombre Apellido1 Apellido2**, alumno de la titulación TITULACIÓN de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI XXXXXXXXX, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Nombre Apellido1 Apellido2

Granada a X de mes de 201 .

D. **Nombre Apellido1 Apellido2 (tutor1)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

D. **Nombre Apellido1 Apellido2 (tutor2)**, Profesor del Área de XXXX del Departamento YYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **Nombre Apellido1 Apellido2 (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Nombre Apellido1 Apellido2 (tutor1) **Nombre Apellido1 Apellido2 (tutor2)**

Agradecimientos

Poner aquí agradecimientos...

Índice general

Capítulo 1

Introduccion

Capítulo 2

Objetivos

Capítulo 3

Planificacion y coste

En este capítulo vamos a tratar la planificación que se hizo del proyecto, así como los costes derivados de este. También estudiaremos si estas estimaciones iniciales fueron correctas, y en caso contrario, cuál sería el impacto en el coste final.

3.1. Fases del proyecto

El proyecto ha sido desarrollado en cuatro fases: planificación del experimento, desarrollo de la aplicación, obtención de resultados mediante la utilización de la aplicación en distintos usuarios y análisis de los resultados obtenidos.

1. Planificación del experimento: en esta etapa se definió el experimento que íbamos a realizar, teniendo en cuenta contenido, fases de las que consistiría y grupos de población en los que íbamos a realizarlo.

2. Desarrollo de la aplicación: esta etapa se dedicó a desarrollar la aplicación que se utilizaría después para la realización de los experimentos. A su vez esta etapa se dividió en tres subetapas:

   a) Instalación de los paquetes necesarios para utilizar el dispositivo System 3D Touch.

   b) Desarrollo del código necesario para implementar la aplicación.

   c) Revisión del código para utilizar los parámetros idóneos. Esta etapa se realizó una vez acabada la primera prueba de realización del experimento.

3. Obtencion de resultados: en esta etapa distintos usuarios llevaron a cabo el experimento planeado, haciendo uso de la aplicación implementada en la fase anterior. A su vez esta etapa se dividió en dos:

a) Fase de prueba, en la que se eligió a un primer grupo de sujetos para la realización del experimento, cada uno con un set de parámetros diferentes. Esta fase nos ayudó a decidir los parámetros que queríamos utilizar finalmente.

b) Fase final, en la que se eligió a un segundo grupo de sujetos (ninguno de los cuales había participado en la fase de prueba) y se realizó el experimento con los parámetros que finalmente habíamos elegido. Estos resultados fueron los que se utilizaron para fase Análisis de los resultados.

4. Analisis de los resultados: en esta etapa analizamos las graficas obtenidas con los datos de la etapa anterior, para así obtener conclusiones del experimento.

Por tanto podemos decir que el proyecto ha seguido un modelo de ciclo de vida en cascada, en el que en algunas etapas ha habido realimentación,dado que la información conseguida en algunas etapas nos ha servido para modificar etapas anteriores. Este proceso puede verse reflejado en la Figura:

Por otro lado la toma de decisiones en las distintas fases se ha llevado a cabo mediante reuniones regulares (telemáticas y presenciales) y la comunicación mediante correo electrónico.

3.2. Planificación del proyecto en tiempo

El proyecto fue pensado para ser desarrollado entre los meses de septiembre y junio. La planificacion inicial fue:

- Septiembre: planificación del experimento.
- De octubre a diciembre: desarrollo de la aplicación.
- De enero a marzo: obtención de resultados
- De marzo hasta mayo: análisis de resultados y redacción de la memoria.

La etapa mas larga fue de la del desarrollo de la aplicación pues es la que sustentaba todo el trabajo que podíamos realizar después. Por eso fue la que más carga de trabajo acumuló, y a la que más horas se le dedicó.

El reparto de horas puede verse en la siguiente grafica:

- Planificación del experimento: 10 horas.
- Desarrollo de la aplicación:
  - Instalación del software y familarización con el: 35 horas
  - Desarrollo del programa: 60 horas

- Obtención de resultados: 40 horas
- Análisis de resultados:
- Total de horas:

Por tanto, teniendo en cuenta que el salario medio de un profesional junior es de /Hora, el coste en personal seria de ....

Por otro lado tenemos que añadir el coste del dispositivo System3D Touch: ´

Por último, dado la necesidad de realizar el experimento en individuos externos al proyecto y teniendo en cuenta la duración media de este (45 minutos) decidimos ofrecer una compensación material a los individuos que se presentaran voluntarios, en concepto de coste de transporte. La compensación la hemos fijado en 5 euros por persona, y dado que necesitamos – voluntarios (– para la fase de prueba y – para la fase final), el coste adicional será de — euros.

Por tanto el presupuesto final fue de – euros, repartido como puede verse en la siguiente figura:

Capítulo 4

Material

4.1. Touch

Touch ha sido el dispositivo sobre el que se ha hecho la aplicación, y que se ha utilizado posteriormente para la realización del experimento.

Es un dispositivo háptico desarrollado por la empresa 3D Systems. Los dispositivos hápticos simulan respuestas táctiles, mediante las cuales podemos percibir objetos tridimensionales en un ambiente de realidad virtual. Según la propia documentación de 3D Systems: "Touch es un dispositivo motorizado que aplica retroalimentación de fuerza a la mano del usuario, lo que le permite sentir objetos virtuales y producir sensaciones táctiles reales a medida que el usuario manipula los objetos 3D en la pantalla". Touch es utilizado en aplicaciones de control robótico, rehabilitación, medicina y cirugía, etc.

Foto Touch

4.1.1. OpenHaptics

OpenHaptics es un software para desarrolladores hápticos que permite crear aplicaciones que utilicen Touch. Está basado en la API de OpenGL y está disponible para Windows de 64 bits 10 y 11 y para Linux.

OpenHaptics incluye varias APIs desarrolladas en C++, y viene con varios ejemplos de aplicaciones, una guía de usuario y dos guías para los desarrolladores: OpenHaptics Programmers Guide y OpenHaptics API Reference Guide.

Capítulo 5

Fundamentos matemáticos

Un proceso clave a la hora de desarrollar la aplicación ha sido trasladar las coordenadas(x,y,z) del sistema de referencia del dispositivo Touch al sistema de referencia de la pantalla del ordenador. De esta forma hemos podido obtener las trayectorias que cada sujeto ha realizado tal y como ellos las han visto reflejadas en la pantalla.

Para conseguirlo hemos utilizado dos conceptos muy importantes en matemáticas como son la multiplicación de matrices y los sistemas de coordenadas.

Álgebra lineal

5.1. Cambio de sistema de referencia

// hablar un poco de la historia de las matrices https://dl.acm.org/doi/abs/10.1145/800248.807398 https://www.ugr.es/ rcamino/docencia/geo1-03/temario.htm

5.1.1. Definiciones y teoremas

Matrices:

Definición 1. *Sea G un conjunto. Decimos que $(G, *)$ es un grupo si existe una operación o ley de composición interna:*

$$* : GxG \to G(x, y) \mapsto x * y \tag{5.1}$$

que cumpla las siguientes propiedades:

1. *Asociativa: $x * (y * z) = (x * y) * z$, $\forall x, y, z \in G$.*
2. *Elemento neutro: $\exists e \in G$) (elemento neutro) tal que $e * x = x * e = x$, $\forall x \in G$.*
3. *Elemento simétrico. $\forall x \in G$, $\exists \bar{x} \in G$ (elemento simétrico de x) tal que $x * \bar{x} = \bar{x} * x = e$.*

Si además cumple:

- *Conmutativa:* $x * y = y * x$, $\forall x, y \in G$.

diremos que el grupo es abeliano.

Definición 1. *Diremos que* $(A, +, *)$ *es un anillo si verifica:*

1. $(A, +)$ *es un grupo conmutativo.*
2. $(A, *)$ *verifica la propiedad asociativa.*
3. $(A, +, *)$ *verifica la propiedad distributiva de la suma respecto al producto, esto es,*

$$x * (y + z) = x * y + x * z (y + z) * x = y * x + z * x \tag{5.2}$$

para todo $x, y, z \in A$.

Un anillo unitario no trivial es un anillo cuyo producto tiene elemento neutro distinto de cero.

Definición 1. *Un cuerpo* $(K, +, *)$ *es un anillo unitario no trivial en el que el producto verifica la propiedad elemento simétrico. El cuerpo es conmutativo si el producto* $*$ *verifica la propiedad conmutativa.*

Definición 1. *Sea* $(K, +, *)$ *un cuerpo conmutativo. Un espacio vectorial sobre* $(K, +, *)$ *es una terna* $(V, +, *K)$ *formada por un conjunto* V *dotado de una operación (ley de composición interna) en* V,

$$+ : VxV \to V \tag{5.3}$$

y una aplicación

$$* : KxV \to V \tag{5.4}$$

o ley de composición externa de K *sobre* V *tales que:*

1. $(V, +)$ *es un grupo conmutativo. Esto es, la operación* $+$ *en* V, *que asocia a cada par* $(u, v) \in VxV$ *un único* $u + v \in V$, *verifica las propiedades de grupo abeliano.*
2. *La ley de composición externa* $* : KxV \to V$, *que asocia a cada* $a \in K$ *y cada* $v \in V$ *un único vector que denotaremos* $a * v \in V$, *verifica las propiedades:*
   a) *Pseudoasociativa:* $(a * b) * v = a * (b * v)$ *para todo* $a, b \in K$ *y todo* $v \in V$.
   b) *Unimodular:* $1 * v = v$, *para todo* $v \in V$, *donde 1 es el elemento neutro de* $(K\{0\}, *)$.

c) *Distributiva respecto de la suma en* K : $(a+b)*v = a*v+b*v$, *para cualesquiera* $a, b \in K$ *y* $v \in V$.

d) *Distributiva respecto de la suma en* V : $a*(u+v) = a*u+a*v$, *para todo* $a \in K$ *y cualesquiera* $u, v \in V$.

Definición 1. *Sea* $V(K)$ *un espacio vectorial y* $S = \{v_1, ..., v_m\}$ *una familia finita no vacía de vectores de* V. *Una combinación lineal de* S *es cualquier vector de* V *obtenido al multiplicar cada* v_i *por un escalar* $a_i \in K$ *y después sumar los vectores resultantes:*

$$a_1 * v_1 + ... + a_m * v_m, dondea_i \in K \forall i = 1, ..., m. \tag{5.5}$$

Llamaremos $L(S)$ *al subconjunto de* V *formado por los vectores obtenidos como combinación lineal de* S*:*

$$L(S) = L(v_1, ..., v_m) = \{a_1 * v_1 + ... + a_m * v_m : a_i \in K, \forall i = 1, ...m\}. \tag{5.6}$$

Definición 1. *´Sea* $V(K)$ *y* $S = \{v_1, ..., v_m\}$ *una familia finita no vacía de vectores de* V. *Diremos que* S *es linealmente independiente si se cumple:*

1. *Caso* $m = 1$*:* $v_1 \neq 0$.
2. *Caso* $m \geq 2$*: ningún vector de* S *es combinación lineal de los restantes vectores de* S, *esto es,* $v_i \notin L(S - \{v_i\})$, *para cada* $i = 1, ..., m$.

Definición 1. *Sea* $V(K)$ *y sea* $S \subset V$ *un conjunto no vacío. Se dice que* S *es un sistema de generadores si* $V = L(S)$. *Esto equivale a que todo vector de* V *se expresa como combinación lineal finita de vectores de* S, *es decir, para cada* $v \in V$, *existen* $m \in N$, *vectores* $v_1, ..., v_m \in S$ *y escalares* $a_1, ..., a_m \in K$, *tales que* $v = a_1 * v_1 + ... + a_m * v_m$.

Definición 1. *Sea* V *un espacio vectorial sobre un cuerpo* K. *Una base de* V *es una familia no vacía* $B \subset V$ *tal que* B *es un sistema de generadores de* V *y* B *es linealmente independiente.*

Definición 1. *Supongamos* V *un espacio vectorial sobre un cuerpo* K *con* $dim_k(V) = n$. *Tomamos dos bases* $B = \{v_1, ..., v_n\}$ *y* $B' = \{v_1, ..., v_n\}$ *de* V *y* $v \in V$. *La matriz de cambio de base de* B *a* B' *es la matriz que denotaremos* $M(I, B' \leftarrow B)$ *cuya columna j-ésima contiene las coordenadas del vector* v_j *de* B *respecto de* B'. *Lo simbolizamos así:*

$$M(I, B' \leftarrow B) = ((v_1)_{B'}|...|(v_n)_{B'}) \tag{5.7}$$

Proposición 1. *En las condiciones anteriores se tiene que:*

$$v_{B'} = M(I, B' \leftarrow B) * v_B \tag{5.8}$$

Demostración. Supongamos que:

$$v_j = a_{1j} * v'_1 + ... + a_{nj} * v'_n, paracadaj = 1, ..., n \tag{5.9}$$

De esta forma la j-ésima columna de $M(I, B' \leftarrow B)$ contiene exactamente a los escalares a_{ij} con $i = 1, ..., n$. Supongamos también que:

$$v = x_1 * v_1 + ... + x_n * v_n \tag{5.10}$$

es decir, v_B contiene a los escalares x_i con $i = 1, ..., n$. Buscamos expresar v como combinación lineal de B'. Sustituyendo la primera ecuación en la segunda tenemos la igualdad:

$$v = x_1*(a_{11}*v'_1+...+a_{n1}*v'_n)+...+x_n*(a_{1n}*v'_1+...+a_{nn}*v'_n) = (a_{11}*x_1+...+a_{1n}*x_n)*v'_1+...+(a_{n1}*x_1+. \tag{5.11}$$

que expresa v como combinación lineal de B'. Esto significa que los coeficientes de la combinación lineal anterior son las coordenadas de v respecto de B'. Por definición de producto de matrices, la entrada i-ésima de $v_{B'}$ es el producto de la fila i-ésima de $M(I, B' \leftarrow B)$ con el vector columna v_B. □

Definición 1. *Un espacio afín es una tripleta* $(A, V, \rightarrow)$ *donde*

- A *es un conjunto no vacío*
- $V \equiv (V, +, *\Re)$ *es un espacio vectorial real*
- $\rightarrow : AxA \rightarrow V$, $(p, q) \mapsto \overrightarrow{pq}$, *es una aplicación satisfaciendo* $A_1 : \overrightarrow{pq} + \overrightarrow{qr} = \overrightarrow{pr}$. $A_2 : \forall p \in A, \forall v \in V\ , \exists! q \in A : \overrightarrow{pq} = v$.

Al espacio vectorial V *lo vamos a denotar como* $\overrightarrow{A}$. *La dimensión de* A *es la de* $\overrightarrow{A}$.

Definición 1. *Una colección de puntos* $\{p_0, ..., p_k\}, k \in K$, *en un espacio afín* A *se dice afínmente independiente si los vectores* $\{\overrightarrow{p_0p_1}, ..., \overrightarrow{p_0p_k}\}$ *son linealmente independientes si*

$$dim < p_0, p_1, ..., p_k >= k \tag{5.12}$$

Definición 1. *Dado un espacio afín* A *con* $dim\ A = n \in N$, *un sistema de referencia* R *en* A *es un sistema ordenado* $\{p_0, ..., p_n\}$ *de* $n + 1$ *puntos afínmente independientes o equivalentemente satisfaciendo:*

$$< \{p_0, p_1, ..., p_n\} >= A \tag{5.13}$$

Al punto p_0 *se le llama origen del sistema, y a la base ordenada* $B = \{\overrightarrow{p_0p_1}, ..., \overrightarrow{p_0p_n}\}$ *de* $\overrightarrow{A}$ *se le llama base asociada de las direcciones de* R. *Si* $B = \{v_1, ..., v_n\}$ *es una base ordenada de* $\overrightarrow{A}$ *y* $p_0 \in A$, *el sistema ordenado de puntos*

$$R = \{p_0, p_0 + v_1, ..., p_0 + v_n\} \tag{5.14}$$

es un sistema de referencia de A con origen p_0 y base asociada de direcciones B. Por tanto, y de forma alternativa, podríamos definir un sistema de referencia como un par

$$R = \{p_0, B\} \tag{5.15}$$

donde p_0 es un punto de A y B una base ordenada de $\overrightarrow{A}$.

Definición 1. *Sea A un espacio afín, y consideremos un sistema de referencia afín $R = \{p_0, ..., p_n\} \equiv \{p_0, B\}$, donde $B = \{\overrightarrow{p_0p_1}, ..., \overrightarrow{p_0p_n}\}$. La aplicación biyectiva*

$$\Phi_R : A \to \Re^n, \phi_R = \phi_B \circ F_{p_0} \tag{5.16}$$

es conocida como la aplicación asignación de coordenadas (con notación columna) en el sistema de referencia R. De forma simplificada escribiremos

$$p_R := \Phi_B(F_{p_0}(p)) \equiv (\overrightarrow{p_0p})_B \in \Re^n, \tag{5.17}$$

y diremos que p_R son las coordenadas de $p \in A$ en R.

Proposición 1. *Sea A un espacio afín y $R = \{p_0, ..., p_n\} \equiv \{p_0, B\}$ un sistema de referencia afín, donde $B = \{\overrightarrow{p_0p_1}, ..., \overrightarrow{p_0p_n}\}$. Entonces*

1. $(p + v)_R = p_R + v_B$
2. $(\overrightarrow{pq})_B = q_R - p_R$

Demostración. Usando la definición anterior y que Φ_B es lineal, tenemos que:

$$(p + v)_R = (\overrightarrow{p_0(p + v)})_B = (\overrightarrow{p_0p} + v)_B = (\overrightarrow{p_0p})_B + v_B = p_R + v_B \tag{5.18}$$

lo que prueba (i). Análogamente,

$$q_R - p_R = (\overrightarrow{p_0q})_B - (\overrightarrow{p_0p})_B = (\overrightarrow{p_0q} - \overrightarrow{p_0p})_B = (\overrightarrow{pp_0} + \overrightarrow{p_0q})_B = (\overrightarrow{pq})_B \tag{5.19}$$

lo que prueba (ii). □

Proposición 1. *La fórmula del cambio de sistema de referencia es:*

$$p_{R'} = (p_0)_{R'} + M(Id_{\overrightarrow{A}}, B, B')p_R \tag{5.20}$$

Demostración. Utilizando la proposición y la definición anterior:

$$p_{R'} = (p_0 + \overrightarrow{p_0p})_{R'} = (p_0)_{R'} + (\overrightarrow{p_0p})_{B'} = (p_0)_{R'} + M(Id_{\overrightarrow{A}}, B, B') * (\overrightarrow{p_0p})_B, \tag{5.21}$$

y por tanto,

$$p_{R'} = (p_0)_{R'} + M(Id_{\overrightarrow{A}}, B, B')p_R \tag{5.22}$$

□

Proposición 1. *La ecuación anterior puede escribirse de forma compacta utilizando una única matriz de orden $n+1$. Esta matriz es de la forma:*

$$M(Id_A, R, R') := \begin{pmatrix} M(Id, B, B') & (p_0)_{R'} \\ 0 & 1 \end{pmatrix} \tag{5.23}$$

Podemos escribir la fórmula del cambio de sistema de referencia como:

$$\begin{pmatrix} 1 \\ p_{R'} \end{pmatrix} = M(Id_A, R, R') * \begin{pmatrix} 1 \\ p_R \end{pmatrix} \tag{5.24}$$

Demostración. Para demostrarlo... □

Por tanto podemos de esta forma podemos conocer las coordenadas en un sistema de referencia R' de un punto $p \in A$ a partir de las coordenadas de p en otro sistema de referencia R' a partir de los datos:

- Coordenadas en R' del origen p_0 de R.
- Matriz del cambio de base en $\overrightarrow{A}$ de la base B de las direcciones de R a la base B' de las direcciones de R'.

5.1.2. Aplicaciones en informática gráfica

‘ Con OpenGL utilizamos diferentes sistemas de coordenadas:

- Coordenadas locales (local coordinates): coordenadas del objeto relativas a su propio origen.
- Coordenadas del mundo (world coordinates): coordenadas del objeto relativas a un origen global del sistema, donde están situados los demás objetos.
- Coordenadas de vista (view coordinates): coordenadas del objeto desde el punto de vista de la cámara.
- Coordenadas de recorte (clip coordinates): coordenadas respecto a un cubo virtual desde el que determinaremos qué vértices se verán en la pantalla. El cubo tiene dimensiones [-1,1]x[-1,1]x[-1,1].
- Coordenadas de pantalla (screen coordinates): coordenadas del objeto tal y como lo vemos representado en la pantalla. El origen de las coordenadas de pantalla está situado en la en la esquina inferior izquierda.

Además vamos a introducir un sistema de coordenadas relativo a los puntos objetivos de la aplicación. Es el sistema de coordenadas que utilizaremos para dibujar las trayectorias.

- Coordenadas finales: en este sistema de coordenadas el origen es el punto inicial de la aplicación, y los demás puntos están situados en la circunferencia de radio 2.

Nuestro objetivo es tomar las coordenadas que nos proporciona Touch, que están en el sistema de referencia local, y transformarlas para obtener las coordenadas finales. De esta forma podemos representar las gráficas y obtener la distancia a los puntos objetivos.

Para transformar las coordenadas de un punto de un sistema de coordenadas a otro utilizaremos las matrices de cambio de sistema de referencia que hemos definido antes. Estas matrices las vamos a llamar de la siguiente forma:

- Matriz modelo (model matrix): matriz de cambio del sistema de referencia local al sistema de referencia del mundo.
- Matriz de vista (view matrix): matriz de cambio del sistema de referencia del mundo al sistema de referencia de vista.
- Matriz de proyección (projection matrix): matriz de cambio del sistema de referencia de vista al sistema de referencia de recorte.
- Viewport: matriz de cambio de sistema de referencia de recorte al sistema de referencia de pantalla.

Estas matrices las podemos obtener a través de OpenGL. OpenGL combina la matriz modelo y la matriz vista en una matriz llamada modelview. Además tenemos que definir otra matriz para pasar de sistemas de coordenadas de pantalla a sistema de coordenadas finales, que llamaremos matriz final.

Por tanto el diagrama que vamos a seguir es:

Coordenadas locales -¿coordenadas de vista -¿coordenadas de recorte -¿coordenadas de pantalla -¿coordenadas finales

vector en coordenadas locales -¿modelview -¿proyection -¿viewport -¿matriz final

En la librería de Glu.h contamos con una función que mapea las coordenadas locales a coordenadas de pantalla. Vamos a explicar cómo funciona teniendo en cuenta los fundamentos matemáticos explicados antes.

Vamos a definir los sistemas de coordenadas que vamos a utilizar:

- R: sistema de coordenadas locales
- R': sistema de coordenadas de vista
- R'': sistema de coordenadas de recorte
- R''': sistema de coordenadas de pantalla

Y las correspondientes matrices de cambio de coordenadas son:

- (Id, R', R): modelview
- (Id, R'', R'): projection
- (Id, R''', R''): viewport

Vamos a tomar un punto v en el sistema de coordenadas R y vamos a estudiar las transformaciones que OpenGL le aplica para obtener el punto en el sistema de coordenadas R'''. Las matrices modelview y projection las podemos obtener con la función glGetDoublev de Gl.h.

La matriz modelview que utiliza la aplicación es:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -6.49 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Como hemos explicado antes, esto es una notación abreviada de:

$$p_{R'} = \begin{pmatrix} 0 \\ 0 \\ -6.49 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} * p_R \tag{5.25}$$

Por tanto lo que estamos haciendo es una transformación afín, en concreto una traslación del origen de sistema de coordenadas en el eje z.

Ahora vamos a hacer el cambio de sistemas de coordenadas de vista al sistema de coordenadas de recorte.

El objetivo de este cambio de sistema de coordenadas es transformar la región visible del espacio en un cubo [-1,1]x[-1,1]x[-1,1], utilizando una proyección en perspectiva, como podemos ver en la imágenes: (imagenes de la proyeccion en perspectiva y del frustum)

Para ello primero calcularemos la proyección en el eje z de las coordenadas x,y. Como hemos estudiado antes, para obtener la proyección sobre el plano euclídeo tenemos que dividir por la última coordenada. En este caso como además queremos que los puntos se proyecten sobre el valor $z = -n$, en lugar de $z = 1$, tendremos que multiplicar por $-n$. Si $(x, y, z) \in P$.

$$x' = x * (-n)/z \tag{5.26}$$

$$y' = y * (-n)/z \tag{5.27}$$

$$z' = z * (-n)/z = -n \tag{5.28}$$

Tenemos que (x', y') sería el punto que obtendríamos sobre el plano euclídeo (plano situado sobre $z = -n$).

Sabemos ahora que $x \in [l, v]$ y $y \in [b, t]$. Para conseguir que ambos estén en el intervalo $[-1, 1]$ tenemos que aplicar un escalado y una traslación.

$$x'' = 2 * (\frac{x' - l}{r - l}) - 1 \tag{5.29}$$

$$y'' = 2 * (\frac{x' - b}{t - b}) - 1 \tag{5.30}$$

Sustituyendo en las ecuaciones anteriores, y utilizando las constantes a_0, a_1, b_0, b_1.

$$x'' = \frac{a_0 * x}{-z_e} - a_1 = \frac{a_0 * x + a_1 * z}{-z} \tag{5.31}$$

$$y'' = \frac{b_0 * y}{-z_e} - b_1 = \frac{b_0 * y + b_1 * z}{-z} \tag{5.32}$$

donde

$$a_0 = \frac{2n}{r - l} \tag{5.33}$$

$$a_1 = \frac{r + l}{r - l} \tag{5.34}$$

$$b_0 = \frac{2n}{t - b} \tag{5.35}$$

$$b_1 = \frac{t + b}{t - b} \tag{5.36}$$

Estas transformaciones no se pueden implementar en una matriz, ya que no son lineales (aparece un denominador). Para poder escribirlo en forma de matriz vamos a hacer uso de una coordenada adicional, que nos va a servir para guardar el término no lineal, en este cao $-z$:

$$x'' = a_0 * x + a_1 * z \tag{5.37}$$

$$y'' = b_0 * y + b_1 * z \tag{5.38}$$

$$w'' = -z \tag{5.39}$$

La matriz que transforma las coordenadas (x, y, z) en las coordenadas (x'', y'', w'') es:

$$\begin{pmatrix} x'' \\ y'' \\ w \end{pmatrix} = \begin{pmatrix} a_0 & 0 & a_1 \\ 0 & b_0 & b_1 \\ 0 & 0 & -1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

De esta forma, una vez aplicadas las transformaciones afines, podemos obtener las coordenadas del plano de la siguiente forma:

$$(x'', y'', w'') -> (x''/w'', y''/w'') \tag{5.40}$$

Con este procedimiento tendríamos calculadas las coordenadas x, y de pantalla. Sin embargo OpenGL utiliza EPO (eliminación de partes ocultas) para dibujar los objetos en la pantalla, lo que significa que tenemos que guardar información sobre la profundidad, para saber qué puntos están más cerca de otros y así poder superponerlos adecuadamente.

Para ello vamos a utilizar una función lineal en la coordenada z, que lleve el rango $[-f, -n]$ a $[-1, 1]$:

$$z'' = \frac{c_0 * z + c_1}{-z} \tag{5.41}$$

donde

$$c_0 = \frac{n+f}{n-f} \tag{5.42}$$

$$c_1 = \frac{2fn}{n-f} \tag{5.43}$$

Incluyendo ahora esta nueva ecuación en la matriz anterior tenemos:

$$\begin{pmatrix} x'' \\ y'' \\ z'' \\ w \end{pmatrix} = \begin{pmatrix} a_0 & 0 & a_1 & 0 \\ 0 & b_0 & b_1 & 0 \\ 0 & 0 & c_0 & c_1 \\ 0 & 0 & -1 & 0 \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Este sería el cambio de sistema de coordenadas de estábamos buscando. Comprobamos ahora que la matriz de proyección que utilizamos en la aplicación tiene esta forma (la obtenemos con glGetDoublev):

$$\begin{pmatrix} 2.74748 & 0 & 0 & 0 \\ 0 & 2.74748 & 0 & 0 \\ 0 & 0 & -3.74748 & -22.5922 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

La matriz tiene la estructura que hemos definido, siendo:

$$a_0 = 2.74748; a_1 = 0; b_0 = 2.74748; b_1 = 0; c_0 = -3.74748; c_1 = -22.5922;$$

Por lo tanto ya tenemos las coordenadas de recorte y ahora vamos a calcular las coordenadas de pantalla. Las coordenadas de pantalla se expresan en píxeles y nos dicen cuál va a ser la representación en la pantalla del ordenador del objeto.

Los parámetros de la ventana que vamos a representar son:

- x_l, y_b: número de columna y fila que ocupa la esquina inferior izquierda de la ventana.
- w, h: anchura y altura de la ventana
- n_d, f_d: rango de valores de z, siendo n_d la profundidad más cercana al observador y f_d la más lejana.

Podemos verlo en la imagen:

Las condiciones que tenemos que realizar para pasar de coordenadas de recorte a coordenadas de pantalla son:

- en el eje de coordenadas x tenemos que llevar el intervalo $[-1, 1]$ al intervalo $[x_l, x_l + w]$.
- en el eje de coordenadas y tenemos que llevar el intervalo $[-1, 1]$ al intervalo $[y_b, y_b + h]$.
- en el eje de coordenadas z tenemos que llevar el intervalo $[-1, 1]$ al intervalo $[n, f]$.

Por tanto las transformaciones que tenemos que aplicar son:

$$x' = (x + 1) * \frac{w}{2} + a \tag{5.44}$$

$$y' = (y + 1) * \frac{h}{2} + b \tag{5.45}$$

$$z' = (z + 1) * \frac{f - n}{2} + n \tag{5.46}$$

Las cuales las podemos representar en la matriz:

$$\begin{pmatrix} x' \\ y'' \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{w}{2} & 0 & 0 & \frac{w}{2} + a \\ 0 & \frac{h}{2} & 0 & \frac{h}{2} + b \\ 0 & 0 & \frac{f-n}{2} & \frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{5.47}$$

En la que podemos ver que estamos aplicando un escalado $[\frac{w}{2}, \frac{h}{2}, \frac{f-n}{2}]$ y una traslación $[\frac{w}{2} + a, \frac{h}{2} + b, \frac{f+n}{2}]$. OpenGL toma $f = 1$ y $n = 0$ y guarda los valores de $a, b, w, h)$ en el vector viewport.

Por último tenemos que pasar de coordenadas de pantalla a las coordenadas finales que son las que utilizaremos para dibujar las trayectorias.

Para ello vamos a realizar dos transformaciones afines:

- El origen de nuestro sistema de coordenadas queremos que esté en el punto de referencia. Al haber hecho una proyección para pasar de sistemas de coordenadas locales a sistemas de coordenadas de pantalla sabemos que un punto en las coordenadas de pantalla corresponde a una recta en las coordenadas de dispositivo. En concreto el punto de referencia que estamos buscando es la recta que pasa por el punto $(0, 0)$ en las coordenadas de dispositivo. Por tanto, utilizando solo un punto de esa recta (el $(0, 0)$) podemos saber cómo se representa en la pantalla. Aplicamos las transformaciones que hemos estudiado antes (modelo-vista-proyección-viewport) al punto $(0, 0)$ y nos da $(250, 250)$. Por lo tanto la traslación que estamos buscando es la que lleva el punto $(250, 250)$ al punto $(0, 0)$, es decir:

$$x' = x - 250 \tag{5.48}$$

$$y' = y - 250 \tag{5.49}$$

- Por otro lado queremos que en nuestro sistema de referencia los puntos estén situados en la circunferencia de radio 2, por lo tanto tenemos que aplicar un escalado. Para saber el escalado que tenemos que hacer aplicamos la transformación modelo-vista-proyección-viewport al punto objetivo $(2 * cos(pi/4), 2 * cos(pi/4)) = (raizde2, raizde2)$. Después calculamos la distancia del punto que nos ha dado al punto origen, y dividimos 2 (distancia de nuestro punto origen a nuestro punto objetivo) entre esa distancia. El resultado es 0.00945962, que será el factor de escala.

Por último vamos a representar estas transformaciones afines en forma de matriz:

$$\begin{pmatrix} x' \\ y'' \\ 1 \end{pmatrix} = \begin{pmatrix} 0.0009 & 0 & -250 \\ 0 & 0.0009 & -250 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{5.50}$$

Y de esta forma ya tenemos todas las transformaciones que vamos a hacer para obtener las coordenadas con las que dibujaremos las trayectorias.

Capítulo 6

Desarrollo del proyecto

En este capítulo vamos a hablar de cómo se han desarrollado las diferentes fases del proyecto así como de los resultados obtenidos.

6.1. Planificación del experimento

En esta fase del proyecto decidimos cuál iba a ser la forma que iba a tener el experimento que posteriormente íbamos a realizar.

Desde el primer momento sabíamos que el objetivo era tener una aplicación en la que aparecieran una serie de puntos en la pantalla, y en la que el usuario tuviera que mover el cursor de un punto a otro haciendo movimientos balísticos. Posteriormente queríamos ser capaces de guardar la trayectoria de los movimientos realizados para poder analizarlas después.

6.1.1. Experimento de referencia

Para diseñar el experimento tomamos como referencia el artículo Cerebellar Contributions to Reach Adaptation and Learning Sensory Consequences of Action. En este artículos los investigadores llevan a cabo un experimento en el que se estudian dos tipos de aprendizaje de movimientos diferentes: de modelo directo y de modelo inverso.

En el modelo inverso se asocia un objetivo con el comando motor necesario para alcanzarlo, mientras que en el modelo directo se asocia un comando motor con sus consecuencias sensoriales. La motivación del artículo es dar soporte a la teoría de que el cerebelo no está implicado en el aprendizaje del modelo inverso, lo que explicaría por qué algunos pacientes con daños en el cerebelo son capaces de realizar con éxito determinados tipos de movimientos.

Para ello llevaron a cabo una serie de experimentos enfocados en disociar estos dos tipos de modelos.

Durante la prueba los individuos tuvieron que alcanzar con un brazo robot un objetivo que se proyectaba en una pantalla, mientras que se ocultaba su mano para que no tuvieran visión de ella, pero sí del brazo robot.

En la primera fase (fase de referencia), aparecieron una serie de dianas en un ángulo de 45º, que el sujeto tenía que golpear con el brazo robot. Primero se realizaron 50 repeticiones en la que se mostraba el cursor sin perturbación (la posición que se mostraba era la posición real), y luego 25 repeticiones sin mostrar el cursor.

Después (sin mostrar ningún objetivo ni el cursor) sino que se pidió a los sujetos que realizaran un movimiento dentro del primer cuadrante de un círculo. Al terminar el movimiento y traer el brazo robot al centro los sujetos tenían que señalar con la mano izquierda dónde creían que la mano derecha había cruzado el círculo. Esto se repetió 25 veces.

Por último se presentó un objetivo en ángulos de [15°, 25°, 35°, 45°, 55°, 65°, 75°]. El feedback visual (la representación del cursor) se proporció solo en los objetivos en ángulo de 45º, por lo que se probaba la capacidad de generalización de los individuos.

En la segunda fase (fase de adaptación) el target objetivo fue siempre 45º, y a lo largo de las repeticiones fue incrementando una perturbación entre el movimiento realizado y la posición del cursor en la pantalla.

En la última fase (fase de postadaptación) se probó la capacidad de generalización de los individuos aplicando una perturbación de 30º, mostrando solamente el target 45º.

El experimento fue llevado a cabo con 10 pacientes sanos y 10 pacientes con algún tipo de daño en el cerebelo.

6.1.2. Diseño de nuestro experimento

Dado que el objetivo del proyecto era probar el funcionamiento de System3D Touch en este tipo de experimentos, y no disponiendo de otros dispositivos complementarios, decidimos que el experimento consistiera en la aparición en pantalla de una serie de puntos sobre los que el sujeto debería situar el cursor.

Primero tendemos un punto de referencia en el centro, y cuando el sujeto se haya posado sobre él, desaparece este y aparece otro sobre una circunferencia alrededor del punto de referencia, en ángulos de 0, pi/4, pi/2, 3pi/4, pi, 5/4pi, 3pi/2, 7pi/4. Al haber situado el cursor sobre el punto objetivo, o al haber terminado el tiempo límite, el punto objetivo desaparece y aparece otra vez el punto de referencia, volviendo a empezar otra iteración. Las trayectorias que nos interesan son las del punto de referencia al punto objetivo.

Tenemos además cinco fases, en las que cambian las condiciones en como se comporta 3DSystem Touch:

- Fase de adaptación: en esta fase solo se muestra el cursor sin ningún punto objetivo. El sujeto es libre para hacer cualquier tipo de movimiento. Esta fase sirve para que el sujeto se adapte a la utilización del dispositivo, así como para que tenga en cuenta las condiciones en las que el dispositivo no funciona correctamente, como cuando se acerca o se aleja en profundidad.
- Fase inicial, sin ninguna perturbación y mostrando el cursor.
- Segunda fase, en la que se muestra el cursor y se aplica una fuerza constante en la dirección x, en sentido negativo.
- Tercera fase, en la que no se aplica ninguna fuerza y se oculta el cursor en los movimientos desde el punto de referencia al punto objetivo. Al terminar el movimiento se vuelve a mostrar el cursor para volver al punto de referencia.
- Cuarta fase, en la que se aplica la misma fuerza de la segunda fase y se oculta el cursor siguiendo lo explicado en la tercera fase.

Los parámetros que teníamos que elegir eran:

- Número de repeticiones por fase. Dado que la realización del experimento requiere de movimientos del brazo durante un periodo de tiempo moderado, teníamos que encontrar un acuerdo entre tener un número suficiente de repeticiones por punto objetivo y que el experimento no fuese demasiado exigente para los sujetos. Es por eso por lo que en la fase de prueba probamos diferentes configuraciones con sujetos de diferentes edades, desde 20 a 90 años. Finalmente elegimos tener 80 repeticiones por fase, para así tener 10 repeticiones por punto. Habría que tener en cuenta que en el caso de finalmente realizar el experimento con personas mayores de 80 años, sería recomendable bajar el número de repeticiones.
- Tiempo para la realización del movimiento. Este es un parámetro clave a la hora de diseñar el experimento, pues queríamos que los sujetos estuviesen obligados a hacer movimientos balísticos. Por lo tanto teníamos que encontrar un balance entre tener tiempo suficiente para que pudiesen alcanzar el punto y la obligación de que esos movimientos fuesen rápidos. Aquí también encontramos grandes diferencias en los resultados de personas jóvenes y más mayores, pero finalmente decidimos utilizar un tiempo de ..., con el que, una vez pasado un periodo de aprendizaje, todos los sujetos fueron capaces de alcanzar en ese tiempo el punto objetivo.
- Número y disposición de los puntos objetivos. Hicimos pruebas con diferentes sujetos con 8 puntos (como hemos mencionado arriba), 4

puntos (0, pi/2, pi, 4pi/3) y 2 puntos (0, pi). Pensamos que el experimento más interesante era con 8 puntos, pues así podíamos estudiar las diferencias en el aprendizaje de movimientos cuando hay que mover el cursor en un solo eje (los puntso 0, pi/2, pi, 4pi/3) a cuando hay que moverlo en dos ejes a la vez (pi/4, 3pi/4, 5/4pi, 7pi/4). En todo caso, y en relación con la dificultad en realizar un número grande de iteraciones en personas mayores, podría estudiarse el caso de reducir el número de puntos para así poder reducir a su vez el número de iteraciones.

- Disposición de los puntos objetivos siguiendo un orden o de forma aleatoria.

Otra decisión que tomamos fue la del criterio para considerar que un movimiento es correcto. En primer lugar pensamos que el criterio fuera que el sujeto pasase el cursor por encima del punto objetivo, de forma similar a como hacen en Çerebellar Contributions to Reach Adaptation and Learning Sensory Consequences of Action". Después pensamos que sería interesante estudiar también, no solo que los sujetos realizaran el movimiento en la dirección correcta, sino con la distancia correcta. Por eso, para que el movimiento se considere correcto, el sujeto debe situar durante ... segundos el cursor encima del punto. Hicimos lo mismo a la hora de situar el cursor dentro del punto de referencia, e iniciar así el movimiento. Esto fue debido a que, en una pequeña demo inicial vimos que, si lo hacíamos solo con pasar el cursor sobre el punto de referencia, todos los movimientos empezaban con un desplazamiento en sentido contrario, el correspondiente al movimiento de volver al punto de referencia. De esta forma el sujeto está obligado a pararse dentro del punto inicial e iniciar el movimiento desde ahí.

También introdujimos un sonido cuando el movimiento no se ha realizado de forma correcta. Esto sirve de feedback al usuario, para saber cuando el movimiento se ha realizado correctamente y cuando no.

Como hemos mencionado llevamos a cabo dos tandas de experimentos: una primera fase de prueba, para decidir los parámetros que íbamos a utilizar; y una segunda fase final para poder obtener resultados.

Uno de los requisitos era que en cada una de las fases tuviésemos representación de individuos de distintas edades, para no tener un sesgo a la hora de implementarla y obtener resultados. Teniendo en cuenta esto, y la disponibilidad de personas que teníamos a nuestro alrededor, los sujetos elegidos fueron:

Fase 1		Fase 2	
Sexo	Edad	Sexo	Edad
Mujer	23	Mujer	20
Mujer	24	Mujer	22
Hombre	53	Hombre	60
Mujer	52	Hombre	50
Hombre	62	Hombre	60
Mujer	60	Mujer	59
Hombre	90		
Mujer	90		

Como se puede comprobar, en la segunda fase la muestra de individuos fue similar a la primera, salvo porque no tuvimos individuos mayores de 80 años. Esto fue debido en parte a la poca disponibilidad que teníamos de individuos de edad más avanzada, y en parte a que decidimos realizar el experimento en unas condiciones que les eran menos favorables, con demasiadas repeticiones. Ninguno de los sujetos del experimento tenían un problema cerebral reconocido, por lo que podemos suponer que eran individuos sanos. No pudimos probar el experimento en individuos con problemas en el cerebelo, como hubiese sido lo ideal, por falta de disponibilidad.

6.2. Desarrollo de la aplicación

El objetivo de esta fase era desarrollar una aplicación sobre la que pudiésemos llevar a cabo el experimento diseñado en la fase anterior. Dicha aplicación está pensada para ejecutarse desde el ordenador, pues necesitamos utilizar a la vez el dispositivo Touch.

La fase del desarrollo de la aplicación tuvo tres partes: instalación del software necesario, implementación del código y revisión del código una vez elegidos los parámetros.

Por otro lado también fue necesario desarrollar un script de python que tomara los datos generados por la aplicación cuando los sujetos realizaban el experimento y los analizara para generar las gráficas.

6.2.1. Instalación del software de OpenHaptics

La primera decisión que tuvimos que tomar fue si instalar el software de OpenHaptics en una máquina Ubuntu, o en una máquina Windows. Aunque en un principio la opción inicial fue instalarlo en Ubuntu 20.04, después de una prueba nos dimos cuenta de que Windows soportaba mejor la interfaz gráfica necesaria para llevarlo a cabo. Por tanto procedimos a realizar la instalación en Windows.

Los paquetes utilizados pueden encontrarse aqui: https://support.3dsystems.com/s/article/OpenH
for-Windows-Developer-Edition-v35?language=en_US.

Además para realizar la implementación del código hemos utilizado Visual Studio Code.

6.2.2. Implementación del codigo

...

Guardado de datos

Para guardar los datos obtenidos de las trayectorias punto_inicial-punto_objetivo utilizamos excel. Guardamos los datos en cuatro ficheros diferentes, uno por cada fase del experimento. Cada fichero contiene cinco columnas:

- Coordenada x del cursor
- Coordenada y del cursor
- Número de iteración. Una iteración es un movimiento desde el punto inicial al objetivo.
- Punto objetivo. Numerados del 1 al 8.
- Tiempo transcurrido desde que se empieza el movimiento, en nanosegundos.

Las coordenadas guardadas vienen dadas en coordenadas de pantalla, con respecto a la ventana de visualización donde aparecen los puntos. Para obtener las coordenadas de pantalla a partir de las coordenadas locales, ver sección ...

6.2.3. Script en python

Para poder estudiar los resultados obtenidos implementamos un script en python.

Este script toma los datos de los ficheros excel (utilizando el módulo pandas), y, por cada iteración toma los últimos valores pertenecientes a la posición x, a la posiciçon y y al tiempo. Esto es debido porque queremos quedarnos con la posicion final del cursor. Despues calcula la distancia al punto objetivo de la siguiente manera:

Por ultimo dibuja varias graficas:

- Una con la trayectoria realizada entre el punto inicial y el objetivo
- Por cada punto objetivo una grafica con los errores cometidos (la distancia desde el ultimo punto al punto objetivo).
- Una grafica global con todos los errores cometidos.
- Una grafica con el tiempo transcurrido por cada iteracion

6.3. Obtencion de resultados

Explicar cómo se han llevado a cabo los experimentos.

6.4. Analisis de resultados

For analysis of hand trajectories, we used Savitzky–Golay smoothing filter for the measured hand position and velocity.

Capítulo 7

Discusion y conclusiones

Capítulo 8

Bibliografía

https://quasardynamics.com/dispositivos-hapticos-la-realidad-virtual/ https://es.3dsystems.com/devices/touch