Práctica NPI - NUI

Memoria técnica gestual

Asignatura: Nuevos paradigmas de interacción
Componentes del grupo 09: Carmen Azorín Martí, María Cribillés Pérez, Ana Graciani Donaire, Jaime Martínez Bravo, Jonas Riemann
Fecha: 22 de diciembre de 2024

Índice

1. Introducción

En este proyecto, hemos desarrollado un Natural User Interface (NUI) para la ETSIIT, facultad de la Universidad de Granada, con el objetivo de facilitar la interacción y la navegación por sus instalaciones mediante tecnología avanzada. La interfaz utiliza el dispositivo Leap Motion para la interacción gestual, permitiendo a los usuarios realizar gestos intuitivos que les ayudan a acceder a diversas funcionalidades. Este sistema está diseñado para beneficiar tanto a las personas que visitan la facultad por primera vez, como a estudiantes internacionales, personas con necesidades especiales, o cualquier usuario interesado en explorar y conocer más sobre la ETSIIT.

Para el desarrollo, hemos empleado Unity junto con el SDK de Ultraleap, implementando toda la programación en el lenguaje CSharp. El resultado es una solución que combina innovación tecnológica y accesibilidad en un entorno de aprendizaje avanzado.

El sistema cuenta con un menú principal que ofrece acceso a diversas secciones, incluyendo noticias y varios submenús interactivos. Los submenús más desarrollados son los del comedor y el de localización de aulas.

El submenú de comedor permite consultar los menús de todos los días de la semana, incluyendo información detallada sobre los alérgenos presentes en cada plato. Esto resulta especialmente útil para personas con restricciones alimentarias, ya que les ayuda a evitar posibles contaminaciones. Además, el sistema está diseñado para ser inclusivo con los estudiantes Erasmus y visitantes internacionales, permitiendo traducir los textos a cuatro idiomas diferentes para mejorar su comprensión y accesibilidad.

El submenú de localización de aulas guía a los usuarios en la ruta desde un punto inicial hasta un aula específica dentro de la facultad. Como ejemplo, hemos implementado una ruta que muestra cómo ir desde el tótem interactivo ubicado en el vestíbulo hasta el aula 3.3, donde se imparten clases prácticas de asignaturas. Este sistema es especialmente útil para nuevos estudiantes y visitantes que necesitan orientación dentro del edificio.

En definitiva, este proyecto no solo mejora la interacción con el entorno de la facultad, sino que también promueve la accesibilidad y la inclusión, haciendo uso de tecnología innovadora para atender las necesidades de todos los usuarios.

2. Interfaz Gestual

2.1. Diseño

En esta subsección vamos a explicar el diseño que hemos realizado en el tótem y el por qué lo hemos realizado así.

2.1.1. Diseño pantalla de inicio

Esta es la página inicial que se muestra si el dispositivo Leap Motion no detecta ninguna mano. Consta de un vídeo publicitario de la UGR que se repite en bucle para llamar la atención. Además, tiene la posibilidad de cambiar de idioma por si los extranjeros lo necesitan. En cuanto el dispositivo detecta una mano, pasa al menú principal ya que suponemos que un usuario va a hacer uso del menú.

Figura 1: Menú inicial con vídeo

Figura 2: Menú inicial con vídeo

Este menú está pensado también para las personas con diversidad funcional ya que el vídeo de fondo puede proporcionar un contexto visual que ayuda a las personas con discapacidades cognitivas o con dificultades para leer texto a entender mejor el propósito del menú. Además, el texto es grande y con contraste pero a la vez integrado en el menú para facilitar la lectura a personas con baja visión.

Para las personas que no entienden el español, es bastante rápido el cambio de idioma y está bastante claro a qué idiomas se puede cambiar.

En general, hemos pensado hacer un diseño simple y con un mensaje claro ("Ven si necesitas ayuda") para que facilite la compresión para todo tipo de capacidades.

En cuanto a los colores, hemos elegido el naranja por ser el color de la facultad para que sea fácil integrable con el resto de recursos que nos ofrece la universidad. Además, investigando un poco más sobre las personas con daltonismo, hemos encontrado que tanto el color morado (o violeta) y el naranja (o amarillo) son los colores que mejor se perciben como distintos sin importar el tipo de daltonismo que tenga una persona. Así podemos amplificar el público al que va dedicado nuestro menú.

2.1.2. Diseño del menú principal

Figura 3: Menú principal

Figura 4: Menú principal con botón seleccionado

En cuanto al diseño, seguimos el patrón de colores de la escena inicial. Vemos como tenemos la cabecera con el logo de la UGR y de la facultad. Además, tenemos un carrusel de noticias que se va cambiando solo con respecto al tiempo para que si alguien está interesado en leer la cabecera puede hacerlo. Si quiere acabar de leer la noticia tendrá que hacerlo en el móvil ya que queremos priorizar otras acciones en nuestro menú como es por ejemplo navegar entre los botones para encontrar la información necesitada.

En el centro tenemos un pequeño reloj que se actualiza solo, y abajo, al igual que en la escena inicial, para poder cambiar de idioma para personas extrajeras.

En cuanto a los botones tenemos 8 que además del texto con lo que está relacionado cada uno, tenemos un pequeño icono explicativo que se integra

en todo el diseño. Cuando está seleccionado es el propio icono el que se pone en gris para que sea de ayuda al usuario. Al incluir este pequeño dibujo, ayudamos a personas con diversidad funcional a entender que información nos aporta cada apartado. Además, los botones están bien espaciados para facilitar la navegación y con los gestos realizados, con poder mover la muñeca es posible deslizarse entre ellos.

En general, hemos buscado la simpleza para que si una persona ve el menú de primeras sepa toda la información que hay y como poder acceder a ella. Hemos pensado también en la gente con diversas funcionalidades para que le sea lo más fácil posible y se sientan incluidos.

2.1.3. Diseño del submenú Comedor

Figura 5: Submenú del comedor

Figura 6: Submenú del comedor con gestos

El submenú del comedor en la aplicación interactiva está diseñado para ofrecer información sobre las comidas disponibles en los comedores de la Universidad de Granada, además de la ubicación y método de pago.

En la parte superior del submenú, se encuentra un panel que muestra las comidas disponibles para el día seleccionado. Cada día incluye dos menús diferentes, entre los cuáles los usuarios pueden alternar utilizando las flechas laterales. Estas flechas permiten navegar entre el Menú 1 y el Menú 2 de manera sencilla.

Cada menú se presenta con imágenes de tres comidas distintas. Estas imágenes son interactivas: al hacer clic en una de ellas, esta se voltea para mostrar información sobre los alérgenos presentes en el plato correspondiente.

Debajo del panel principal, se encuentra un conjunto de tres botones dispuestos horizontalmente mediante un diseño de horizontal layout, cada uno con funcionalidades específicas:

- Cambio de día de la semana: este botón permite cambiar al siguiente día de la semana en el que hay menús disponibles. La navegación es secuencial, por lo que para regresar al día anterior es necesario pulsar el botón seis veces, ya que sólo se guarda los menús de los siete días de la semana.

- Información sobre pago: al pulsar este botón, se voltea para mostrar que el pago debe realizarse obligatoriamente con tarjeta.

- Ubicación de comedores: este botón ofrece información sobre la ubicación de los comedores. AL voltearlo, muestra que el comedor de la ETSIIT se encuentra en la planta 0. Además, incluye un código QR que permite acceder al listado completo de los comedores disponibles de la universidad.

Finalmente, en la parte más baja se encuentra un panel con cuatro banderas que representan diferentes idiomas:

- Reino Unido: inglés

- España: español

- Alemania: alemán

- Francia: francés

Al hacer clic en cualquiera de las banderas, el idioma de la aplicación cambia automáticamente al idioma correspondiente. Este cambio afecta tanto al texto mostrado en el panel superior como al de los botones inferiores, asegurando una experiencia completamente localizada para usuarios internacionales. A continuación, se explican las tres clases que interactúan entre sí.

La clase *LocalizedText* se incluye en cada componente de texto que necesita ser traducido y recibe las traducciones desde el *LocalizationManager*. Su función es actualizar el texto según el idioma actual. Esta clase define una clave asociada al texto que se desea traducir y se obtiene el componente *TextMeshProUGUI* del objeto y actualiza texto llamado a *UpdateText()*. En dicha funciñon se consulta el texto traducido correspondiente al idioma actual utilizando el método *GetLocalizedText()* de *LocalizationManager*.

La clase *LocalizationManager* es el núcleo del sistema de localización. Se encarga de cargar las traducciones desde un archivo JSON, almacenar los textos traducidos y gestionar el cambio de idioma. La clase es un singleton, por lo que solo hay una instancia global accesible desde cualquier parte del código. Durante su inicialización, carga las traducciones desde un archivo JSON ubicado en la carpeta *StreamingAssets*, es un diccionario con el siguiente formato:

```
{
    "es": {"botonQR": "D nde estamos", "botonQRAtras": "Planta 0"},
    "en": {"botonQR": "Where We Are", "botonQRAtras": "Floor 0"}
}
```

Al cambiar de idioma, actualiza la variable *currentLanguage* y llama a *UpdateAllTexts()* para que todos los textos visibles se actualicen.

Finalmente, está la clase *ChangeLanguage()* que se asocia a cada bandera y permite cambiar el idioma cuando el usuario hace clic en ella. Para hacerlo, define un código de idioma (*langaugeCode*) que representa el idioma que cambiará al pulsar la bandera. En el evento *OnButtonClick()*, se llama a *ChangeLanguage(languageCode)* del *LocalizationManager* para cambiar el idioma global.

2.1.4. Diseño del submenú Aulas

Figura 7: Submenú de aulas

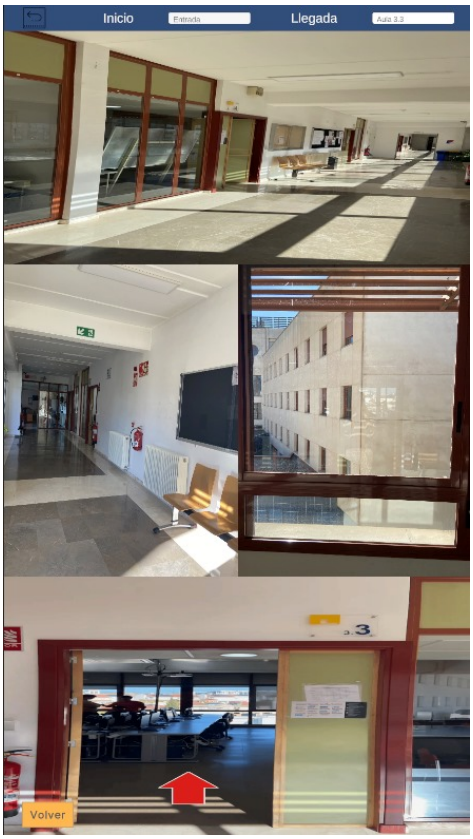

Figura 8: Submenú de aulas

El submenú de las aulas ha sido diseñado pensando en la accesibilidad y la facilidad de navegación, permitiendo a los usuarios explorar de manera intuitiva las instalaciones de la escuela. Este menú se compone de una cabecera funcional y cuatro imágenes grandes que ocupan toda la pantalla, proporcionando una experiencia visual accesible para cualquier persona.

La cabecera contiene los siguientes elementos:

- Botón de retroceso: ubicado en la parte izquierda de la cabecera, este botón permite retroceder al nodo anterior, facilitando la navegación bidireccional por la ruta.
- Barras de inicio y destino: por defecto, el sistema define el inicio como el tótem y el destino como el aula 3.3, destinada a las clases de prácticas.

Estos puntos pueden ser personalizados por el usuario para establecer una ruta específica dentro de la facultad.

El espacio principal de la pantalla está compuesto por cuatro imágenes grandes, cada una representando las vistas desde el nodo actual en las siguientes direcciones: frente, atrás, izquierda y derecha. Cabe destacar, que las referencias se hacen suponiendo que el usuario está mirando hacia el este.

Estas imágenes están diseñadas para maximizar la visibilidad y ser fácilmente identificables, atendiendo a las necesidades de personas con dificultades visuales.

Al seleccional un botón de navegación (una imagen), el usuario es conducido al siguiente nodo de la ruta, que también incluye cuatro imágenes correspondientes a las direcciones mencionadas. La transición entre nodos es fluida y está guiada por una flecha superpuesta en la imagen relevante, indicando claramente la dirección que se debe tomar para avanzar hacia el destino.

Para implementar el submenú que permite la navegación, he utilizado dos clases principales: *GrafoNavegacion* y *GrafoSetup*.

La clase *GrafoNavegacion* gestiona la lógica de navegación entre nodos. Cada nodo representa una posición dentro de la escuela, con conexiones a otros nodos que simulan posibles direcciones de movimiento (izquierda, derecha, frente y atrás). Su objetivo principal es actualizar la interfaz gráfica según el nodo actual y las posibles rutas disponibles.

Tiene imágenes como atributos que representan las vistas hacia las distintas direcciones desde el nodo actual. También hay un historial de nodos que permite volver hacia atrás ne la navegación. La función *SetNodoActual* cambia el nodo actual, actualizando las imágenes y registrando el nodo anterior en el historial.

La clase *GrafoSetup* se encarga de configurar la estructura del grafo. En el método *Start()* se crean los nodos, se les asignan imágenes y se establecen conexiones entre ellos. Un ejemplo de inicilización del nodo entrada es:

```
Nodo entrada = new Nodo { Id = "Entrada" };

entrada.ImagenIzquierda = Resources.Load<Sprite>("entradaIzq");
entrada.ImagenFrente = Resources.Load<Sprite>("entradaFrente");
entrada.ImagenDerecha = Resources.Load<Sprite>("entradaDrc");
entrada.ImagenAtras = Resources.Load<Sprite>("entradaAtras");

entrada.NodoFrente = esc1;
```

2.1.5. Diseño del resto de submenús

El resto de submenús han seguido el mismo patrón que el menú principal. Como solo hemos desarrollado más a fondo un submenú de gestión (Comedor) y otro submenú de localización (Aulas), el resto está implementado de la forma más básica posible:

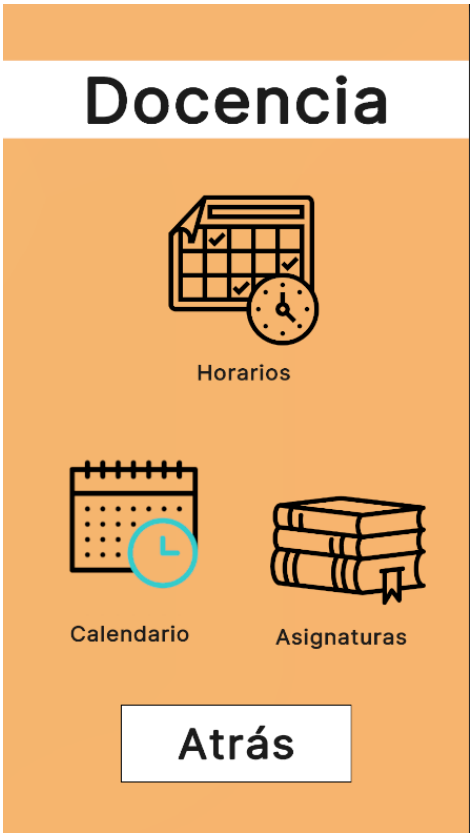

Figura 9: Submenú de docencia

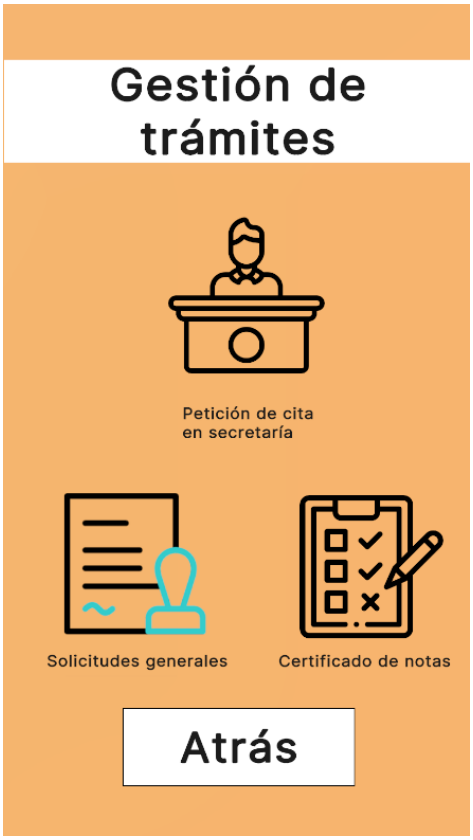

Figura 10: Submenú de trámites

Figura 11: Submenú de profesorado

Figura 12: Submenú de espacios comunes

Figura 13: Submenú de servicios externos

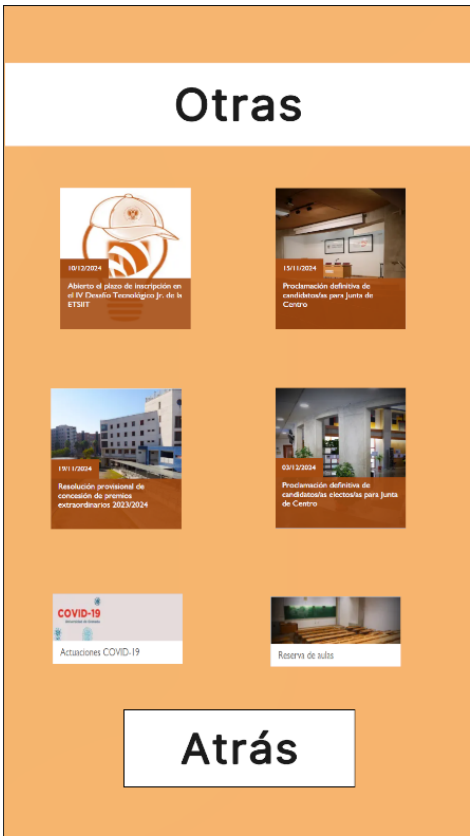

Figura 14: Submenú de Otros

2.2. Gestos con Leap Motion

En este apartado vamos a especificar que gestos se han implementado, cuáles son de la librería de Unity y cuáles hemos implementado nosotros mismos desde cero. Además, se pondrá pequeños trozos de código (los más significantes ya que el proyecto entero vendrá adjuntado en la entrega) para poder complementar la información y el procedimiento de creación de los gestos.

2.2.1. Gestos de la escena inicial

Vamos a explicar más en detalle el código relacionado con esta escena.

La escena Init tiene el script asociado initSceneBehaviour donde se implementan todas sus funcionalidades. Primero de todo, tenemos que importar las librerías de Unity y Leap Motion necesarias. Después, declaramos la clase InitSceneBehaviour, que es un clase extendida de MonoBehaviour. La clase de esta escena tiene como atributos el leap provider y un atributo Hand donde almacena la información sobre la mano que sea detectada por el dispositivo. A continuación, en el método Start(), se busca e inicializa el elemento leapProvider para conectarlo con nuestro dispositivo. Si no lo encuentra, salta un error. Se obtiene el frame actual del leapProvider y verifica si hay manos detectadas. Si no las hay manda un Debug.Log para indicarnos que no hay manos. Además, con Screen.SetResolution(900, 1600, true); establecemos la resolución de nuestra pantalla vertical donde haremos la defensa y con el parámetro true se pondrá en pantalla completa.

En el método Update(), obtenemos el frame actual con CurrentFrame y volvemos a verificar si hay manos. En cuanto detecta una mano cargamos la escena del menú principal.

```
private void Update()
{
    Frame frame = leapProvider.CurrentFrame;

    if (frame != null)
    {
        if (frame.Hands.Count > 0)
        {
            SceneManager.LoadScene("Main");
        }
    }
}
```

En conclusión, la escena de Init se estará ejecutando hasta que encuentre una mano el dispositivo y pasará a la siguiente escena para poder manejar el menú.

2.2.2. Gestos del menú principal

Esta escena es la principal del menú en el tótem. Se ejecuta una vez que ha detectado una mano en la escena inicial. Se implementa en el script MainSceneBehaviour donde volvemos a importar las librerías, dependencias y a asociar el leapProvider y Hand. Estéticamente, en la escena hay 8 botones:

1 2 3 4
5 6 7 8

Por tanto, tenemos dos filas y cuatro columnas de botones:

- *NUM_Y=2*
- *NUM_X=4*
- *NUM_BUTTONS=NUM_X*NUM_Y*

En el método Start(), volvemos a activar el SetResolution para ponerlo a las dimensiones de la pantalla vertical (900x1600) y en pantalla completa. Asignamos en un array los botones que se asocian con los eventos declarados (cambiar al submenú de cada botón).

En el método Update(), si detectamos un puño, se ejecuta la acción del botón actualmente seleccionado. Si no te has movido de botón, se empieza por defecto seleccionado el primer botón. Por tanto, si detecta el puño entrará al submenú del botón que esté seleccionado. Si no se detecta un puño, se comprobará si se están haciendo los gestos para cambiar de botón a las cuatro posiciones posibles: izquierda, derecha, arriba y abajo. Como nuestra disposición de botones está diseñada así:

1 2 3 4
5 6 7 8

para ir del botón 1 al botón 4 se puede ir directamente si hacemos el gesto de scroll izquierda y para ir del botón 1 al 5 se puede ir o haciendo scroll hacia arriba o hacia abajo.

Para poder implementar los gestos que no están en la librería de Unity (el puño sí es de la librería), hemos calculado el ángulo que forma el antebrazo con la mano. Si este ángulo es mayor de 15 grados, es decir, no tenemos alineados la mano con el antebrazo, empezaremos a comprobar si es cumple las siguientes condiciones:

- **Scroll down:** la dirección z de la palma es menor de -0.15. La posición de cada componente está entre -1 y 1, por tanto, cuando estamos posicionando la mano para abajo sin mover el antebrazo, ponemos los demás gestos a false y este gesto a true. Además, empieza a contar el tiempo para asegurarse de que el gesto sea sostenido durante al menos 1 segundo antes de ejecutarlo para asegurarse que es lo que quiere hacer el usuario. Si se mantiene el gesto, aumentamos en Y el botón ya que estamos cambiado a la siguiente fila (NUM_Y es las filas que hay). Además, tenemos que hacer el módulo ya que si en el último pasamos al siguiente queremos que vuelva al primero:

  ```
  selectedButtonY = (selectedButtonY + 1) % NUM_Y;
  ```

- **Scroll up:** el gesto hacia arriba el procedimiento es igual que el anterior, solo que es cuando detecta la palma de la mano cuando en el eje z es mayor de 0.3. En el anterior, el umbral es más bajo ya que al Leap le costaba diferenciar más cuando la mano estaba boca abajo. Sin embargo, cuando está hacia arriba se detecta bastante bien, entonces hemos subido el umbral para que si hay un gesto natural no cambie, que solo cambie cuando de verdad queremos. En este caso, lo que queremos es restar uno en la fila por lo que utilizaremos esta fórmula:

  ```
  selectedButtonY=(selectedButtonY-1 + NUM_Y) % NUM_Y;
  ```

  Para que funcione bien el operador módulo con número negativos tenemos que volver a sumarle NUM_Y. De ahí el significado de la fórmula.

- **Scroll left:** en este caso detecta cuando la palma de la mano en el eje x es mayor que 0.10. Al ser un gesto para la izquierda, tenemos que restar uno en las columnas (NUM_X):

  ```
  selectedButtonX = (selectedButtonX - 1 + NUM_X) % NUM_X;
  ```

  La explicación de la fórmla es igual solo que trabajamos con las columnas en vez de con las filas.

- **Scroll right:** se detecta cuando la palma de la mano en el eje x es menor que -0.10. Es el gesto para cambiar de botón al de la derecha, por lo que la fórmula es la siguiente:

  ```
  selectedButtonX = (selectedButtonX + 1) % NUM_X;
  ```

El cálculo para el índice de cada botón seleccionado es el siguiente:

```
selectedButton = selectedButtonX + (selectedButtonY * NUM_X);
```

Si mantenemos el mismo gesto, se hará la acción correspondiente hasta que se deje de hacerlo. Es decir, para que el usuario no tenga que deslizar dos veces, hemos implementado el código para que si deja su mano en la dirección de deslizar, cada segundo avanza la posición seleccionada un botón.

Además, para saber que botón está seleccionado, ponemos tanto el icono como las letras un poco más grises en vez de negro, así le será más fácil al usuario saber por donde va y cuantos gestos le queda para llegar a su objetivo.

Para finalizar, tenemos funciones declaradas para comprobar con la librería de Unity si está el puño cerrado (IsFist(Hand hand) donde ponemos un umbral mínimo de 0.8) y las funciones correspondientes para cargar todos los submenús con LoadScene. Estas funciones se llaman: OnDocenciaButton() donde el nombre de Docencia se cambia por los demás submenús declarados.

En conclusión, en el menú principal tenemos dos tipos de gestos: scroll implemetado por nosotros mismos (no es de la librería de Unity) en cuatro direcciones para poder cambiar de botón y el puño para poder seleccionar el botón.

2.2.3. Gestos del menú Comedor

Cada funcionalidad del submenú del comedor tiene gestos específicos que permiten a los usuarios navegar y obtener informacion de manera eficiente. La getsión de los gestos se realiza mediante la clase *SubMenuSceneBehaviour-Comedor*. Esta clase se encarga de interpretar los movimientos de las manos para ejecutar las acciones correspondientes en la interfaz.

En cada frame, el método *Update()* se encarga de:

1. Detectar las manos visibles
2. Identificar si la mano detectada es la izquierda o la derecha
3. Determinar el número de dedos extendidos de la mano activa
4. Llamar a la función correspondiente según la mano activa y los dedos extendidos

Para identificar qué mano está interactuando, se utiliza e siguiente código:

```
    foreach (Hand detectedHand in frame.Hands)
{
    if (detectedHand.IsRight)
    {
        hayManoDerecha = true;
```

```
            hayManoIzquierda = false;
            hand = detectedHand;
        }
        else
        {
            hayManoIzquierda = true;
            hayManoDerecha = false;
            hand = detectedHand;
        }
}
```

Una vez identificada la mano activa, se evalúan los dedos extendidos con este código:

```
    int count = 0;
foreach (Finger finger in hand.Fingers)
{
    if (finger.IsExtended) count++;
}
```

Para interactuar con las imágenes de los platos y visualizar los alérgenos, se usa el número de dedos extendidos por la mano izquierda:

- Un dedo extendido: gira la imagen del primer plato principal llamando a la función *SimulateClick(image1)*
- Dos dedos extendidos: gira la imagen del segundo plato principal llamando a la función *SimulateClick(image2)*
- Tres dedos extendidos: gira la imagen del acompañamiento llamando a la función *SimulateClick(image3)*

Los tres botones del horizontal layout también responden a gestos basados en la cantidad de dedos extendidos, pero esta vez con la mano derecha:

- Un dedo extendido: se cambia al siguiente día de la semana llamando a la función *CambiarDia()*
- Dos dedos extendidos: se accede a la información sobre el método de pago llamando a la función *Girar()*
- Tres dedos extendidos: se muestra la ubicación del comedor de la ETSIIT y el código QR con información de todos los comedores de la universidad llamando ala función *GirarQR()*

Para alternar entre los menús disponibles del día seleccionado, se utiliza un gesto con la mano extendida en paralelo al suelo. Girando la mano izquierda hacia la izquierda hasta formar un ángulo de aproximadamente 30 grados entre el antebrazo y la mano, se cambia de un menú al otro. Este gesto simula el movimiento de pasar una página.

Para detectar este gesto, se realiza el cálculo del ángulo entre la dirección del antebrazo y la dirección de la palma de la mano:

```
Vector3 forearmDirection = hand.Arm.Direction;
Vector3 palmDirection = hand.Direction;

float angle = Vector3.Angle(forearmDirection, palmDirection);

if (angle > 30f)
{
    SwitchPanel(1);
    lastGestureTime = Time.time;
}
```

La selección del idioma se realiza mediante un gesto colaborativo que involucra ambas manos: juntándolas frente al Leap Motion, se cambia al siguiente idioma disponible en el orden establecido. Por defecto, la aplicación se inicia en español.

El reconocimiento del gesto se basa en comprobar la presencia de al menos dos menos y verificar si están suficientemente cerca. Esto se gestiona al principio del método *Update*, utilizando el siguiente código:

```
if (hand != null && frame.Hands.Count >= 2 && AreHandsTogether(frame))
{
    ChangeToNextLanguage();
    lastGestureTime = Time.time;
}
```

La función *AreHandsTogether(Frame frame)* se encarga de calcular si las palmas de las manos están lo suficientemente cerca como para considerar que están juntas. para ello se accede a las manos detectadas mediante *frame.Hands[0]* y *frame.Hands[1]* y se obtiene la posición de cada palma con *hand1.PalmPosition*. Posteriormente, se calcula la distancia euclidiana entre las posiciones de las palmas utilizando la fórmula:

$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Si la distancia calculada es menor que un umbral definido 0.1f, las manos se consideran juntas.

```
private bool AreHandsTogether(Frame frame)
{
    Hand hand1 = frame.Hands[0];
    Hand hand2 = frame.Hands[1];

    Vector3 palmPosition1 = hand1.PalmPosition;
    Vector3 palmPosition2 = hand2.PalmPosition;

    float distance = Mathf.Sqrt(
        Mathf.Pow(palmPosition1.x - palmPosition2.x, 2) +
        Mathf.Pow(palmPosition1.y - palmPosition2.y, 2) +
        Mathf.Pow(palmPosition1.z - palmPosition2.z, 2)
    );

    bool areHandsTogether = distance < 0.1f;

    return areHandsTogether;
}
```

Para manejar la claridad de interacción y evitar pulsaciones accidentales, se ha establecido un tiempo de espera de dos segundos entre la activación de un gesto y la posibilidad de realizar otro. Este retraso asegura que cada acción sea intencionada y facilita que los usuarios comprendan los resultados de su interacción antes de continuar. Para lograrlo, se implementa un sistema de control de tiempo entre gestos, utilizando las siguientes variables como atributos de instancia de la clase:

```
private float gestureCooldown = 3f;
private float lastGestureTime = -3f;
```

Este mecanismo se gestiona dentro del método *Update()*, que es llamado en cada frame, mediante el siguiente bloque de código:

```
if (Time.time - lastGestureTime < gestureCooldown)
return;
```

Donde la expresión dentro del condicional calcula el tiempo que ha pasado desde el último gesto detectado y si dicho tiempo es menor que el valor de gestureCooldown, el sistema no procesa más gestos y retorna inmediatamente. Por tanto, cada vez que se procesa un gesto, se debe incluir la línea:

```
lastGestureTime = Time.time;
```

Por último, también existe la opción de volver al menú principal, que se explica en el apartado de Gestos del submenú Aulas. Esto se consigue cerrando el puño con cualquiera de las dos manos.

2.2.4. Gestos del submenú Aulas

En la implementación de la interacción con Leap Motion, la clase *SubmenuSceneBehaviourAula* se encarga de interpretar los gestos realizados por el usuario.

Para volver al menú principal el sistema detecta si la mano realiza un puño. Si esto ocurre, inmediatamente se ejecuta la acción para volver al menú principal. Esto se logra comprobando que hay al menos una mano visible y luego verificando si cumple la condición de puño con el método *IsFist(hand)*:

```
if (IsFist(hand))
{
    SceneManager.LoadScene("Main");
}
```

El método es común a todos los submenús y permite al usuario regresar al menú principal. Si la mano no forma un puño, se procede a contar el número de dedos extendidos. Si solo hay un dedo extendido, el sistema interpreta esto como el comando para retroceder al nodo anterior.

```
if (count == 1)
{
    Debug.Log("Retroceder");
    grafoNavegacion.RetrocederNodo();
    lastGestureTime = Time.time;
}
```

El sistema utiliza el ángulo entre la palma de la mano y el antebrazo para determinar hacia dónde mover el cursor. Se calculan las direcciones con base en los vectores *forearmDirection* (dirección del antebrazo) y *palmDirection* (dirección de la palma):

```
Vector3 forearmDirection = hand.Arm.Direction;
Vector3 palmDirection = hand.Direction;

float angle = Vector3.Angle(forearmDirection, palmDirection);
```

Según el ángulo y la dirección de la palma, se ejecuta el movimiento correspondiente dentro del grafo navegación:

- Mover atrás: si el eje Z de la palma es menor a -0.15
- Mover adelante: si el eje Z de la palma es mayor a 0.30
- Mover izquierda: si el eje X de la palma es mayor a 0.10

- Mover derecha: si el eje X de la palma es menor a 0.10

```
if (angle > 15f)
{
    if (palmDirection.z < -0.15)
    {
        grafoNavegacion.MoverAtras();
        lastGestureTime = Time.time;
    }
    else if (palmDirection.z > 0.30)
    {
        grafoNavegacion.MoverFrente();
        lastGestureTime = Time.time;
    }
    else if (palmDirection.x > 0.10)
    {
        grafoNavegacion.MoverIzquierda();
        lastGestureTime = Time.time;
    }
    else if (palmDirection.x < -0.10)
    {
        grafoNavegacion.MoverDerecha();
        lastGestureTime = Time.time;
    }
}
```

Para evitar que los gestos se ejecuten de forma accidental o se repitan demasiado rápido, se implementa un tiempo de espera entre gestos de 3 segundos. Este tiempo ya se ha explicado en el apartado anterior.

2.2.5. Gestos del resto de submenús

En cuanto al resto de submenús, tenemos un mismo script asociado ya que solo se implementa la acción de volver al menú principal. Este script es SubMenuBehaviour. En él hemos importado las librerías necesarias, hemos buscado el leapProvider y HandPoseDetector correspondientes y hemos vuelto a poner la resolución a la de la pantalla vertical indicando que es en pantalla completa. Además, tenemos la función de IsFist():

```
private bool IsFist(Hand hand)
{
    float fistThreshold = 0.8f;
    return hand.GrabStrength >= fistThreshold;
}
```

Esta función comprueba si el puño está cerrado con un umbral. Si devuelve un true, se llamará a la función de volver hacia el menú principal:

```
private void OnBackToMenuButton()
{
    SceneManager.LoadScene("Main");
    Debug.Log("Boton 'Volver' presionado.");
}
```

2.3. Errores del Leap y evaluación de la detección de los gestos

En este párrafo evaluamos como detecciones falsas del Leap afectan la satisfacción de la experiencia del usuario con la aplicación.
En general, el sistema de detección de gestos implementado con Leap Motion funciona correctamente para la mayoría de las acciones y menús. Sin embargo, hemos identificado algunos problemas específicos y evaluado su impacto:

1. Problema con el gesto de cambiar entre menús del submenú comedor. El gesto de mover la mano hacia la izquierda para cambiar del menú 1 al menú 2 no funciona de manera óptima. A pesar de estos inconvenientes, los demás botones de este submenú funcionan correctamente.

2. Confusión al contar el número de dedos extendidos. En algunos casos, el Leap Motion puede confundirse al interpretar el número de dedos visibles, especialmente si la mano no está completamente extendida o si hay oclusión de algunos dedos. Sin embargo, la implementación de un tiempo de espera de 3 segundos entre gestos ha mejorado significativamente este aspecto. Al obligar al sistema a procesar un gesto único en un intervalo de tiempo, se reducen las ambigüedades y errores de detección.

3. Problema al detectar un gesto cuando tenemos la mano desalineada con el antebrazo. Al basarse en eso los gestos del menú principal, hemos puesto un umbral de 15 grados para el ángulo entre la mano y el antebrazo para que haya una cierta naturalidad, es decir, nosotros habitualmente no tenemos perfectamente alineadas estas dos partes, así que introduciendo este umbral se le añade naturalidad al gesto.

A excepción del problema puntual en el menú del comedor, el resto de gestos funcionan correctamente tanto en este submenú como en el resto de los menús. La detección es precisa y la interacción fluida, lo que mejora la experiencia del usuario.

3. Reparto de trabajo

Toda la parte relacionada con el Leap ha sido realizada por Carmen Azorín Martí, María Cribillés Pérez y Jonas Riemann. Aquí queda detallado más concretamente:

- Gestos de la escena inicial, menú principal y submenús: María y Jonas.
- Gestos del menú del comedor: Carmen y Jonas.
- Gestos del menú Aulas: son los mismos que los dos anteriores pero adaptados para este submenú.
- Diseño submenús Comedor y Aulas: Carmen
- Diseño pantalla inicial, menú principal y resto de submenús: María

4. Bibliografía

En el desarrollo del proyecto se utilizó la documentación oficial de Ultraleap y Unity [3], así como la guía para la descarga del SDK de Unity [2]. También se hizo uso de un repositorio en GitHub relacionado con la implementación de la interfaz natural de usuario (NUI) compartido por todos los miembros del grupo [1].

Para implementar el carrusel de imágenes en Unity, se siguieron dos tutoriales de YouTube [5, 6]. Finalmente, se consultó la guía oficial de Ultraleap para desarrollar el primer proyecto con su tecnología [4].

Referencias

[1] GitHub. Repositorio compartido por nuestro grupo, 2024.

[2] Ultraleap. Descargar el sdk de unity, 2024.

[3] Ultraleap. Getting started with xr and unity, 2024.

[4] Ultraleap. Your first project in unity with ultraleap, 2024.

[5] YouTube. Crear un carrusel en unity, 2024.

[6] YouTube. Tutorial adicional para carrusel en unity, 2024.