Programación de interfaces naturales mediante la combinación de sensores de teléfonos móviles

Javier Hernández-Aceituno
Dpto. Ingeniería Informática y de Sistemas
Universidad de La Laguna
San Cristóbal de La Laguna, España
jhernaac@ull.edu.es

Isabel Sánchez-Berriel
Dpto. Ingeniería Informática y de Sistemas
Universidad de La Laguna
San Cristóbal de La Laguna, España
isanchez@ull.edu.es

***Abstract*—Uno de los aspectos más llamativos del aprendizaje de la programación, orientada a la creación de aplicaciones para teléfonos móviles, es la posibilidad de producir interfaces basadas en movimientos y habilidades básicas del cuerpo humano, en lugar de verse limitadas al uso de teclados y ratones. En este trabajo se describe una serie de actividades prácticas, planteadas para dotar a los alumnos de las capacidades necesarias para el acceso y procesamiento de la información, proveniente de los sensores de los teléfonos móviles, y su combinación para la creación de interfaces naturales.**

***Index Terms*—Interfaces inteligentes, interfaces naturales, teléfonos móviles, programación, sensores.**

I. Introducción

Uno de los objetivos prioritarios en el diseño de aplicaciones informáticas, especialmente aquellas orientadas a funcionar principalmente en plataforma móviles tales como tablets o smartphones, es el uso de interfaces que no se encuentren limitadas a los métodos convencionales de interacción con el usuario, como pueden ser el teclado o el ratón de un ordenador.

Se denomina *interfaz natural* a toda aquella que permite interactuar con un sistema o aplicación mediante el uso de movimientos gestuales y capacidades básicas humanas, tales como la mirada o la voz, de una forma que resulte sencilla y cómoda al usuario, sin que requiera entrenamiento alguno. El uso de este tipo de interfaces ha sido ampliamente estudiado y aplicado a diversos ámbitos de estudio [1]–[4].

El interés en el desarrollo de este tipo de interfaces encuentra motivación en la gran cantidad de sensores de que disponen los dispositivos móviles, tales como giróscopos, acelerómetros, posicionamiento global, cámaras y micrófonos, y en las facilidades que los lenguajes de programación modernos dan para su uso integrado en cualquier tipo de aplicación.

Es por tanto de gran interés formar a los estudiantes universitarios de carreras basadas en la informática, para que sean capaces de utilizar las herramientas de desarrollo que se encuentran a su disposición y les permiten crear aplicaciones informáticas con interfaces naturales.

En la actualidad existe una tendencia al uso de interfaces naturales como soporte para la docencia, especialmente en casos en los que los métodos tradicionales no están disponibles o no son efectivos. La Facultad de Telemática de la Universidad de Colima (México) ha desarrollado interfaces gestuales y tangibles para educación especial y la activación física de niños de primaria [5]. También en la Universidad de Zaragoza ha elaborado un sistema de detección visual para educación infantil, basado en el reconocimiento de formas y colores mediante una cámara [6]. La Universidad de La Coruña ha realizado un estudio analizando posibles diseños para utilizar realidad aumentada como interfaz de aprendizaje [7].

El presente trabajo se enmarca dentro de la asignatura cuatrimestral "Interfaces Inteligentes", perteneciente al itinerario de Computación del Grado en Ingeniería Informática de la Universidad de La Laguna. En esta asignatura se imparten los fundamentos y tecnologías para el análisis y diseño de experiencias interactivas, realidad virtual, realidad aumentada e interfaces naturales, organizados en los siguientes epígrafes:

1) Fundamentos y tecnologías en el análisis y diseño de experiencias interactivas
2) Realidad virtual y realidad aumentada
3) Interfaces naturales
4) Interacción afectiva y emocional

Las competencias que sus alumnos adquieren tras su estudio son las siguientes:

- Competencias generales

CG4 Capacidad para definir, evaluar y seleccionar plataformas hardware y software para el desarrollo y la ejecución de sistemas, servicios y aplicaciones informáticas.

CG6 Capacidad para concebir y desarrollar sistemas o arquitecturas informáticas centralizadas o distribuidas integrando hardware, software y redes.

CG9 Capacidad para resolver problemas con iniciativa, toma de decisiones, autonomía y creatividad. Capacidad para saber comunicar y transmitir los conocimientos, habilidades y destrezas de la profesión de Ingeniero Técnico en Informática.

- Competencias específica del itinerario

C44 Capacidad para desarrollar y evaluar sistemas interactivos y de presentación de información compleja y su aplicación a la resolución de problemas de diseño de interacción persona computadora.

- Competencias transversales

T7 Capacidad de comunicación efectiva (en expresión y comprensión) oral y escrita, con especial énfasis en la redacción de documentación técnica.
T10 Capacidad de integrarse rápidamente y trabajar eficientemente en equipos unidisciplinares y de colaborar en un entorno multidisciplinar.
T21 Capacidad para el razonamiento crítico, lógico y matemático.
T23 Capacidad de abstracción: capacidad de crear y utilizar modelos que reflejen situaciones reales.

Al ser eminentemente práctica, la evaluación de la asignatura de "Interfaces Inteligentes" se basa en trabajos individuales realizados semanalmente en el laboratorio y en el desarrollo grupal de dos prototipos, uno de realidad virtual y otro de alguna de las restantes técnicas explicadas durante el curso.

Se ha de resaltar que la asignatura de Interfaces Inteligentes no versa sobre el funcionamiento físico de los sensores de los dispositivos móviles, sino solamente sobre su uso en la programación de aplicaciones con interfaces naturales. Sin embargo, la Universidad de La Laguna ofrece en su Grado de Ingeniería Informática otras asignaturas, tales como Robótica Computacional [8], a través de las cuales los alumnos pueden complementar su formación con una explicación detallada a nivel de hardware sobre el funcionamiento interno de los dispositivos sensoriales.

A. Evaluación

La importancia de las prácticas en la asignatura queda reflejada en las estrategias seguidas para la evaluación. En cada sesión práctica de laboratorio se programa una actividad que conlleva el desarrollo de una aplicación con funcionalidades mínimas, en la que se ponga en práctica el aspecto específico que se trabaje en ella. De esta forma, las prácticas se pueden agrupar en alguna de las categorías siguientes:

- Aprendizaje de programación de gráficos 3D utilizando Unity
- Desarrollo de aplicaciones utilizando los sensores:
  - Desarrollo de aplicaciones de RV
  - Desarrollo de aplicaciones de RA
  - Desarrollo de aplicaciones con interacción basada en sensores

Cada semana se evalúan las tareas que debe resolver el alumno, suponiendo un 50% de la nota de la asignatura. Además todas las técnicas y tecnologías aprendidas se deben poner en práctica en el desarrollo de un prototipo de una aplicación de RV. El tema del prototipo es libre y, si bien los alumnos por lo general optan por el desarrollo de algún videojuego, en ocasiones también han optado por simulaciones inmersivas, como es el caso de una "simulación espacial de minería con agentes reactivos" (curso 2017–2018). También se les pide que la aplicación incorpore alguno de los sensores adicionales para la interacción, como puede ser el acelerómetro o el micrófono, por ejemplo.

Todos estos trabajos son realizados por equipos de a lo sumo 3 personas, por lo que se le hace especial hincapié en el uso de herramientas colaborativas para el desarrollo de proyectos. Este aspecto es especialmente importante en el desarrollo de la competencia T10, que consideramos fundamental en el trabajo cotidiano de cualquier Graduado en Ingeniería Informática. Por otra parte, desde el punto de vista del profesor facilita la evaluación ya que permite hacer un seguimiento de las aportaciones de cada miembro del grupo.

Si bien no se restringe la herramienta a utilizar, se recomienda el uso de *Git* y la plataforma *Github* además de *Collaborate*, la herramienta que provee Unity para el desarrollo colaborativo. Independientemente de qué herramienta decida utilizar el equipo para el trabajo colaborativo, todos los prototipos deben entregarse mediante un enlace a Github que contenga todo el código del proyecto y un documento en el que se debe incluir:

- Los hitos logrados
- Los problemas y dificultades encontrados en el proyecto, tanto los que se resolvieron como aquellos que se debieron simplificar por escapar de lo factible
- Acta de acuerdos del grupo y seguimiento del proyecto

La puntuación obtenida en el desarrollo de este prototipo supone un 20% de la nota de la asignatura, y la exposición del proyecto otro 20%.

II. Métodos

Las clases prácticas de esta asignatura se basan en el desarrollo de aplicaciones en Unity3D [9], utilizando el lenguaje C#, generalmente para su ejecución en teléfonos móviles con sistema operativo Android. Tras una iniciación a la implementación de código en Unity3D, los alumnos son introducidos semanalmente a diferentes herramientas de acceso y procesamiento de la información proveniente de los sensores de un teléfono móvil, y su combinación para la generación de interfaces naturales.

En general no se impone a los alumnos una descripción estricta de qué aplicación deben programar en cada caso, sino que se les da libertad para utilizar la información obtenida de forma creativa, creando aplicaciones personalizadas sujetas sólo a un conjunto reducido de requisitos, relativos al uso de los diferentes sensores.

A. Realidad virtual

Como primera práctica, los alumnos se familiarizan con el paquete de realidad virtual de Google Cardboard [10]. Haciendo uso de los acelerómetros de un teléfono móvil, montado en una estructura de cartón, es posible crear unas gafas de realidad virtual de coste muy reducido y sencillas de programar (Fig. 1).

La integración de las funcionalidades de Google Cardboard en una aplicación de Unity es relativamente sencilla y se explica detalladamente en las páginas web oficiales de ambos proyectos. En general, basta incluir el paquete *GoogleVRForUnity_*.unitypackage* en la lista de recursos de la aplicación, lo que produce una vista estereoscópica alineada con la orientación relativa del dispositivo móvil en el que se ejecute (Fig. 2).

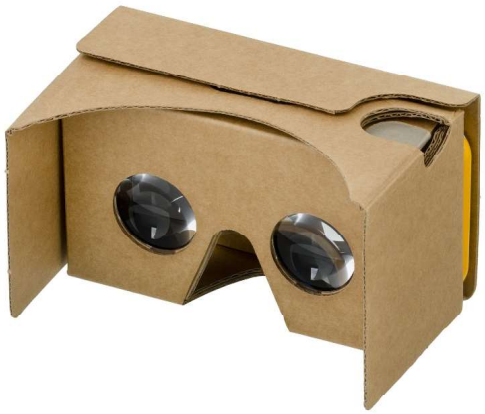

Fig. 1. Estructura Google Cardboard

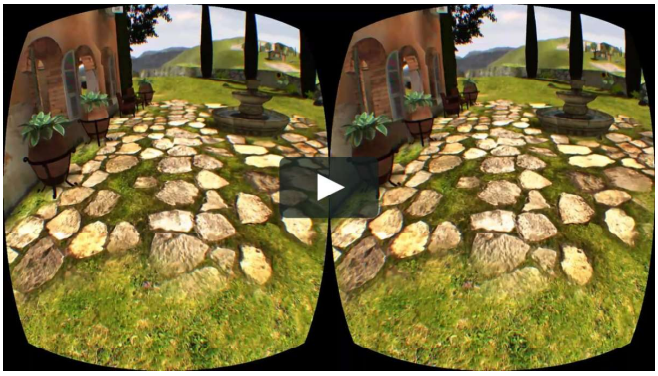

Fig. 2. Vista estereoscópica

La principal dificultad a la hora de utilizar este recurso proviene de los cambios constantes del framework para el desarrollo de aplicaciones de realidad virtual de Google. Desde que se comenzó a uitilizar esta tecnología en las tareas de enseñanza–aprendizaje práctico en la asignatura de "Interfaces Inteligentes", en el curso 2015–2016, ha habido una variación en la API para el desarrollo. En el curso 2015–2016 se disponía del paquete de Google Cardboard para Unity; en el curso 2016–2017, Google había lanzado DayDream, su primera apuesta por el lanzamiento de dispositivo móvil de prestaciones de gama media que mejorase la experiencia de RV.

En el curso 2017-2018, Google actualizó el framework de RV, unificando en el mismo paquete de Unity ambos dispositivos, Cardboard y Daydream. En ambos cursos, el cambio se produjo apenas un mes antes del periodo en el que se imparte la asignatura, lo que dejó obsoleto el material docente del curso previo. Existe además la dificultad añadida de contar exclusivamente con la documentación de Google, ya que la comunidad no tiene tiempo de generar otros recursos. La gran ventaja, sin embargo, es la alta productividad que se logra al utilizar el editor de escenas Unity, que permite al alumno obtener un prototipo jugable de una aplicación de RV altamente inmersiva y a bajo coste.

En este tipo de aplicaciones, por lo general altamente atractivas para los usuarios, se corre el riesgo de mermar la experiencia del usuario por la incomodidad generada por el efecto conocido como *mareo del simulador*. Por esta razón se programa una práctica para el estudio y puesta en práctica de las recomendaciones de diseño que el equipo de Google ha recopilado para aplicaciones de RV.

Por una parte tenemos consideraciones a tener en cuenta para evitar el mareo, todas aquellas destinadas a reducir la discrepancia entre la percepción real del usuario y lo que sucede en el mundo virtual. Por otra parte tenemos las pautas destinadas a orientar al usuario en esta nueva forma de interacción con la que aún no se encuentra familiarizado, como sucede con aplicaciones convencionales para PC o móviles. En esta línea es de especial importancia la práctica programada para el uso de la retícula y la interacción con objetos en la escena mediante *raycast*. La retícula es un objeto visual que advierte al usuario cuando su línea de visión ha enfocado (seleccionado) un elemento de la escena con el que puede interactuar.

B. Sensores de posición

Posteriormente, los alumnos aprenden a acceder manualmente a los valores devueltos por dichos acelerómetros, así como por la brújula interna y el sistema de posicionamiento global (GPS) del dispositivo. El código de desarrollo para todos estos elementos queda recogido en la clase *Input* del paquete *UnityEngine*, que debe importarse a la aplicación en la que se desee usar.

Los datos de salida de la brújula interna vienen dados por el atributo *Input.compass.trueHeading*, de tipo numérico en coma flotante (*float*), que indica la orientación del dispositivo respecto al Polo Norte geográfico. Aunque este dato es suficiente para la mayoría de aplicaciones útiles, el paquete *Input* también permite acceder a la orientación respecto al Polo Norte magnético, dada por el atributo *Input.compass.magneticHeading*, y a la información geomagnética en crudo, medida en microteslas, dada por el objeto *Input.compass.rawVector*. Para poder acceder a todos estos valores es necesario inicializar en la aplicación el sistema de localización del dispositivo, mediante una llamada a la función *Input.location.Start()* (Fig. 3).

La información de los acelerómetros del dispositivo móvil es fácilmente accesible en todo momento, en el objeto de tipo vector (*Vector3*) *Input.acceleration*. Los datos de tipo *Vector3* contienen tres atributos de tipo *float*, etiquetados como *x*, *y*, *z*, que en el caso de los acelerómetros indican la aceleración del dispositivo en cada eje cartesiano (Figs. 4 y 5).

Por último, el sistema de posicionamiento global del dispositivo permite acceder a sus coordenadas, en forma de valores de latitud (*latitude*), longitud (*longitude*) y altitud (*altitude*), como atributos del objeto *Input.location.lastData*. Estos atributos sólo se actualizan una vez realizada la llamada a la función *Start()* del objeto *Input.location*, también usada para obtener datos de la brújula.

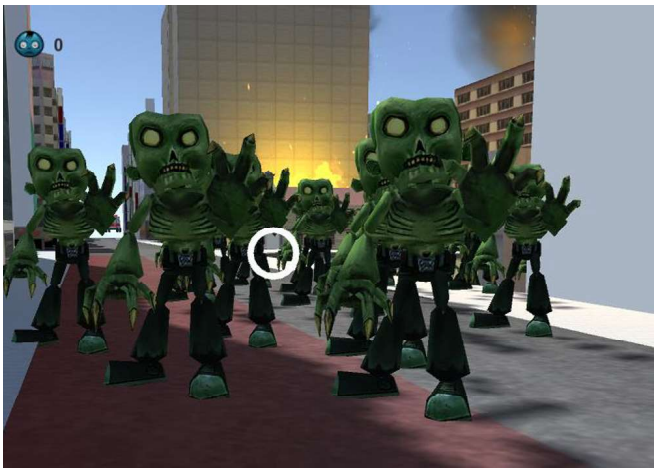

Fig. 3. Captura de la aplicación "Zombie Timer α", que integra el uso de la brújula del dispositivo móvil

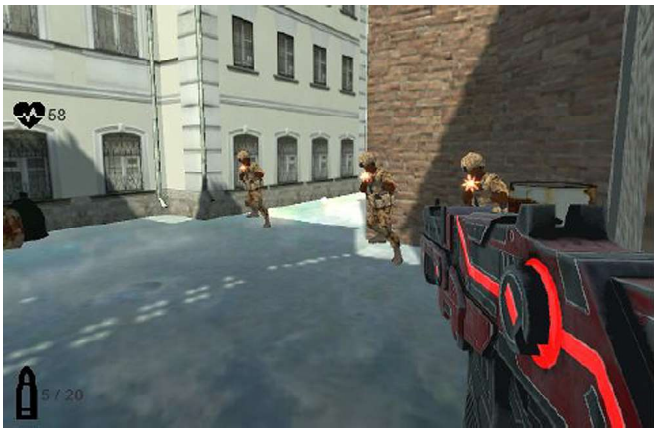

Fig. 4. Captura de la aplicación "VR Combat Arcade", que utiliza los acelerómetros del dispositivo móvil como método de control

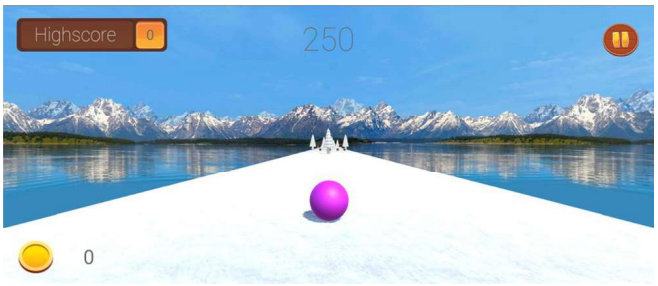

Fig. 5. Captura de la aplicación "GRESBALL", que utiliza los acelerómetros del dispositivo móvil como método de control

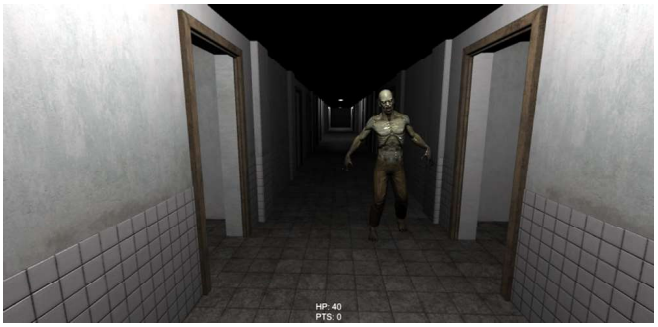

Fig. 6. Captura de la aplicación "Program for the Investigation of Pursuing Objectives", que integra el uso del micrófono para interactuar con los personajes

La función *Start()* tiene dos argumentos opcionales: la precisión deseada y la distancia mínima de actualización, ambas en metros y con un valor por defecto de $10\,m$. Conviene además comprobar el valor del atributo *Input.location.status*, que indica si el servicio está activo (*LocationServiceStatus.Running*), antes de leer *Input.location.lastData*. Una vez termine de usarse el servicio de localización, debe detenerse con una llamada a la función *Input.location.Stop()*.

C. Sensores de entrada audiovisual

En la siguiente práctica, se indica a los alumnos cómo acceder mediante código al micrófono y a la cámara de sus teléfonos móviles, cuyos datos son procesados en las siguientes prácticas (Fig 6).

Unity permite obtener sonido desde el micrófono del ordenador o el teléfono móvil usando la función *Start (entrada, bucle, duración, frecuencia)* de la clase *Microphone*, que devuelve un objeto de tipo *AudioClip* y cuyos parámetros son:

- *entrada*: Nombre del dispositivo de entrada a utilizar, dado como una cadena de caracteres (*string*) o una cadena vacía si se desea usar el que esté disponible por defecto.
- *duración*: Duración máxima en segundos del objeto *AudioClip* generado, dada como un número entero (*int*).
- *bucle*: Valor booleano (*bool*) que indica si, una vez alcanzada la duración máxima, se debe volver al principio del *AudioClip* o parar de grabar.
- *frecuencia*: Frecuencia de muestreo, dada como un valor *float*, que define la calidad del audio generado.

Para detener la grabación, basta llamar a la función *Microphone.End (entrada)*.

La forma más general de obtener imágenes de la cámara del teléfono móvil o la webcam del ordenador es mediante la clase de Unity *WebCamTexture*, que traduce la información obtenida de dichos dispositivos a texturas 2D. Para ver las imágenes grabadas, debe asociarse el objeto *WebCamTexture* al atributo de textura visual *mainTexture* de un material y seguidamente llamar a la función *Play()* de dicho objeto. Para pausar la reproducción basta llamar a la función *Pause()* y, para detenerla, a la función *Stop()*. En el ejercicio práctico, se pidió a los alumnos hacer uso de todas estas funciones, de forma

que fuese posible capturar fotogramas aislados y conservarlos en memoria como imágenes fijas.

D. Reconocimiento de voz

Unity ofrece herramientas de reconocimiento de voz, aunque por el momento se encuentran limitadas a su uso en dispositivos equipados con el sistema operativo Windows 10 [11]. Para la utilización de estas herramientas se debe importar el paquete *UnityEngine.Windows.Speech*. Aunque dicho paquete ofrece diversos métodos para reconocer palabras y frases a partir del micrófono del dispositivo, en la asignatura de "Interfaces Inteligentes" sólo se propuso el uso de las clases *KeywordRecognizer* y *DictationRecognizer*, de menor complejidad.

La clase *KeywordRecognizer* intenta casar el audio recibido por el micrófono del ordenador o teléfono móvil con una lista de palabras clave proporcionadas. Es posible mantener varios objetos de esta clase activos al mismo tiempo, siempre que sus conjuntos de palabras no tengan ningún elemento común.

Para iniciar un objeto de este tipo, se utiliza el constructor *KeywordRecognizer (lista)*, donde el array de *string lista* contiene todas las palabras a detectar. Un objeto de esta clase puede activarse y desactivarse con sus funciones *Start()* y *Stop()*, respectivamente. Se advirtió a los alumnos la importancia de llamar a la función *Dispose()* al terminar de usar este tipo de objetos, para liberar sus recursos.

Para la detección de voz debe asociarse al objeto *KeywordRecognizer* una función de escucha del tipo *void función (datos)*, donde el parámetro *datos* de tipo *PhraseRecognizedEventArgs* contiene:

- la frase recibida *datos.text*, de tipo *string*;
- el grado de certeza *datos.confidence*, de tipo enumerado *Windows.Speech.ConfidenceLevel*, cuyos valores pueden ser *High*, *Medium*, *Low* y *Rejected*;
- la duración de la frase *datos.phraseDuration*, de tipo *TimeSpan*;
- el momento de comienzo *datos.phraseStartTime*, de tipo *DateTime*;
- el significado semántico *datos.semanticMeanings*, de tipo *SemanticMeaning[]*. Este atributo no tiene utilidad para *KeywordRecognizer*, pero sí para otras clases de reconocimiento de voz más complejas, como *GrammarRecognizer*

. Una vez creada la función de escucha, se asociará al objeto *KeywordRecognizer* mediante la instrucción *objeto.OnPhraseRecognized += función*.

Por otro lado, la clase *DictationRecognizer* intenta transcribir el audio recibido por el micrófono del ordenador o teléfono móvil, sin necesidad de un listado de palabras clave. Un objeto de esta clase también puede activarse y desactivarse con sus funciones *Start()* y *Stop()*, respectivamente. De nuevo, es importante llamar a la función *Dispose()* al terminar de usarlo para liberar sus recursos. Un objeto *DictationRecognizer* espera que se le asocien cuatro funciones de escucha:

- *objeto.DictationResult*, llamada cada vez que se reconozca una frase y que será de tipo *void función (string texto, Windows.Speech.ConfidenceLevel confianza)*
- *objeto.DictationHypothesis*, llamada cada vez que se actualice la frase capturada, de tipo *void función (string texto)*
- *objeto.DictationComplete*, llamada cada vez que se interrumpa el dictado, de tipo *void función (Windows.Speech.DictationCompletionCause causa)*, donde el parámetro *causa* es de tipo enumerado y sus valores pueden ser los siguientes:
  - *Complete*, si el dictado terminó correctamente;
  - *AudioQualityFailure*, si el dictado se interrumpió por mala calidad del audio;
  - *Canceled*, si el dictado fue cancelado o la aplicación fue pausada;
  - *TimeoutExceeded*, si se excede el tiempo estipulado para la captura;
  - *NetworkFailure*, si el dictado se interrumpió por un fallo de red;
  - *MicrophoneUnavailable*, si el dictado se interrumpió por no detectarse el micrófono;
  - *UnknownError*, en cualquier otro caso.
- *objeto.DictationError*, llamada cuando se detecte un error durante el dictado, de tipo *void función (string error, int resultado)*.

En todos los casos, las funciones de escucha se asociarán al atributo correspondiente del objeto *DictationRecognizer* mediante la instrucción *objeto.atributo += función*.

E. Realidad aumentada

Finalmente, se realiza un entrenamiento sobre la salida de la cámara para reconocer elementos visuales concretos y generar a partir de ellos una aplicación de realidad aumentada [12]. Para ello, basta agregar a un proyecto de Unity el paquete de desarrollo *vuforia-unity-xx-yy-zz.unitypackage* y el objeto prefabricado (prefab) *ARCamera*, que sustituye a la cámara principal de la escena (*MainCamera*). A continuación, basta añadir una base de datos que relacione imágenes objetivo, a capturar por la cámara del dispositivo, con objetos en 3D a mostrar en la escena.

F. Rúbrica de la evaluación

La rúbrica de la evaluación se define para contemplar cada uno de los aspectos cuya importancia se ha resaltado en este trabajo:

- **Modelos usados en la escena –** Escala: Simples, Geométricos, Complejos sin animación, Complejos con animación, Además de los modelos se cuidan los escenarios: terreno, skybox, ...
- **Complejidad de las acciones de los personajes –** Escala: Los personajes sólo avanzan por el juego, Los personajes además de avanzar interaccionan con elementos en la escena, Algún personaje interacciona de varias formas diferentes con algunos objetos de la escena,

Además de lo anterior, alguna interacción de algún personaje genera efectos especiales como como por ejemplo sistemas de partículas.

- **Variedad de personajes –** Escala: Además del jugador hay objetos estáticos, Además del jugador hay personajes dinámicos como enemigos, Además de lo anterior existe varios tipos de personajes no estáticos y que no son el jugador, Además de lo anterior el comportamiento de cada tipo de personaje ante el jugador es diferente.
- **Originalidad –** Escala: Baja, Media, Alta
- **Complejidad del desarrollo –** Escala: El código se basa exclusivamente en prefabs de la asset store y del paquete standard assets, Además de los prefabas utilizados se implementan scripts sobre al menos dos objetos de la escena invocados en update, Además de lo anterior se implementan eventos de los components de Unity, Se implementa un GameController basado en eventos que incluye algún evento definido por los alumnos.
- **Calidad del código –** Escala: Código de baja calidad, sin documentar, sin jerarquía lógica de objetos, etc., Código bien documentado., Código bien documentado en el se define una colección coherente de objetos, Además de lo anterior, se contempla de forma adecuada qué propiedades o métodos deben ser públicos, cuales privados, y se tiene en cuenta la eficiencia de cara al rendimiento en dispositivos móviles.
- **Grado de inmersión –** Escala: Grado bajo, Grado medio,Grado alto
- **Interacción con las cardboard –** Escala: No se contempla más interacción que los movimientos de cabeza, Se utiliza el botón magnético, Además de lo anterior también se utiliza un GamePad o algún otro método de interacción
- **Calidad de la interacción de RV –** Escala: En todo momento se hace seguimiento de la cabeza, Además de lo anterior se siguen las recomendaciones de la retícula, Se contemplan otras recomendaciones para las aplicaciones de RV como evitar discrepancias entre la percepción física y el mundo virtual
- **Trabajo en equipo –** Escala: El acta sólo refleja división de tareas sin ninguna conexión, El acta refleja división de tareas y al menos una puesta en común del trabajo, El acta refleja división de tareas y más de una puesta en común del trabajo.
- **Trabajo individual –** Escala: El alumno en las respuestas individuales muestra haber adquirido un dominio bajo del desarrollo de aplicaciones de RV: confunde conceptos, no sabe responder con precisión, no identifica a qué corresponden fragmentos de código., El alumno en las respuestas individuales muestra haber adquirido un dominio medio del desarrollo de aplicaciones de RV: confunde conceptos avanzados de RV, no responde con precisión a cuestiones avanzadas de RV, El alumno muestra un dominio alto en el desarrollo de aplicaciones de RV: responde con exactitud y precisión a cuestiones avanzadas de RV, así como a qué corresponde cada script implementado.
- **Otros sensores –** Escala: No se usa ningún sensor, Uso simple del micrófono o el acelerómetro, Uso complejo del micrófono o el acelerómetro, pero no los dos. O bien: uso simple de ambos sensores, Uso complejo de ambos sensores

III. Resultados y conclusiones

Las clases prácticas de la asignatura de "Interfaces Inteligentes" permiten a los alumnos desarrollar de forma creativa sus propias aplicaciones con interfaces naturales, haciendo un uso combinado de los sensores disponibles en sus teléfonos móviles. Estos ejercicios introducen al código necesario para utilizar la brújula, los acelerómetros, el sistema de posicionamiento global, la cámara y el micrófono de los dispositivos, en aplicaciones de realidad virtual o aumentada. Estos elementos son la base para las distintas técnicas de diseño de interfaces naturales, incluyendo el reconocimiento de voz e imágenes, que pueden utilizarse en el desarrollo de aplicaciones complejas y adaptativas.

El problema a destacar está en las dificultades propias del uso de tecnologías que aún están en evolución, cuyos frameworks de desarrollo cambian de un curso a otro, o aún no están disponibles para una amplia variedad de dispositivos. Por contra, la principal ventaja que aporta la metodología utilizada corresponde a la motivación que genera en los alumnos el uso de estas tecnologías utilizando herramientas productivas que en un periodo corto de tiempo les permite generar prototipos funcionales.

Al término de la asignatura, durante el curso 2017/2018, se solicitó a los 37 alumnos matriculados que compartiesen su opinión respecto a la misma en una encuesta on-line. A continuación se muestran las respuestas obtenidas:

- De todos los trabajos y prácticas encargados en la asignatura, ¿cuáles consideras que te han ayudado más en tu formación?
  - Seminarios/debates (0%)
  - Presentación oral de trabajos (7.14%)
  - Prácticas Laboratorio (92.86%)
- ¿Qué dificultades has encontrado para la realización de los trabajos/prácticas?
  - Carencia de información (23.08%)
  - Desconocimiento de las fuentes a utilizar (23.08%)
  - Falta de indicaciones por parte del responsable de la asignatura (7.69%)
  - Demasiado volumen de trabajo (0%)
  - Periodo de tiempo concedido para su elaboración y entrega (7.69%)
  - Dificultad para entender el significado de las cuestiones planteadas (23.08%)
  - Dificultades del trabajo en equipo (15.38%)
- Valora los siguientes aspectos utilizados en prácticas, desde el punto de vista del aprendizaje de la programación de interfaces naturales y la programación de aplicaciones basadas en sensores. (1 No ha sido adecuada.

2 Ha dificultado el aprendizaje. 3 No ha influido. 4 Ha facilitado el aprendizaje. 5 Ha sido fundamental.)
  - Lenguaje de programación C# (4.14)
  - Unity 3D (4.43)
  - Realidad Virtual en dispositivos móviles (4.00)
  - Framework Google Cardboard (3.43)
  - Vuforia (3.29)
  - Acelerómetro (3.79)
  - Brújula (3.69)
  - Micrófono (3.93)
  - Cámara (4.00)
- Valora los siguientes aspectos utilizados en prácticas, desde el punto de vista de tu motivación respecto a la programación de interfaces naturales y la programación de aplicaciones basadas en sensores. (1 Me ha desmotivado totalmente. 2 Ha influido negativamente en mi motivación. 3 No ha influido 4 Ha contribuido en mi motivación. 5 Me ha mantenido totalmente motivado.)
  - Lenguaje de programación C# (3.71)
  - Unity 3D (3.86)
  - Realidad Virtual en dispositivos móviles (3.86)
  - Framework Google Cardboard (2.86)
  - Vuforia (2.86)
  - Reconocimiento de voz en Unity (3.57)
- Valora los siguientes aspectos utilizados en prácticas, desde el punto de vista de los conocimientos nuevos útiles desde el punto de vista profesional que has adquirido. (1 No se ha aportado ningún conocimiento nuevo. 2 Se han aportado conocimientos nuevos insuficientes. 3 Se han aportado conocimientos nuevos aceptables. 4 Se han incrementado aspectos importantes el conocimiento 5 Totalmente nuevo)
  - Lenguaje de programación C# (3.71)
  - Unity 3D (4.14)
  - Realidad Virtual en dispositivos móviles (4.00)
  - Framework Google Cardboard (3.71)
  - Vuforia (3.64)
  - Acelerómetro (3.93)
  - Brújula (3.93)
  - Micrófono (4.00)
  - Cámara (4.00)
- Valora los siguientes aspectos utilizados en prácticas, desde el punto de vista de la dificultad que han supuesto respecto a la programación de interfaces naturales y la programación de aplicaciones basadas en sensores. (1 Ha supuesto una dificultad excesiva en las prácticas. 2 Ha supuesto una dificultad en las prácticas. 3 No ha añadido dificultad a las prácticas 4 Ha facilitado el desarrollo de las prácticas. 5 Ha facilitado totalmente el desarrollo de las prácticas.)
  - Lenguaje de programación C# (3.46)
  - Unity 3D (3.36)
  - Realidad Virtual en dispositivos móviles (3.14)
  - Framework Google Cardboard (2.79)
  - Brújula (3.07)
  - Micrófono (3.07)
  - Cámara (3.29)
  - Reconocimiento de voz en Unity (2.86)
- Indica cualquier sugerencia que tengas respecto a el aprendizaje de interfaces naturales y programación de sensores que creas que ayudarían a mejorar la enseñanza de los contenidos. (Entre las sugerencias proporcionadas por los alumnos en este apartado, destacan:)
  - Una reducción de la carga teórica de la asignatura en favor de más ejercicios prácticos.
  - El uso de Blender [13] como herramienta de modelado.
  - Equipos informáticos más preparados para la carga gráfica requerida en las prácticas de la asignatura.
  - Mayor peso a la explicación de los fundamentos de realidad virtual en dispositivos móviles.
- Indica tu valoración personal respecto a la enseñanza de la programación de interfaces naturales y sensores desarrollando una aplicación de Realidad Virtual en móviles.
  - Salvo excepciones muy puntuales, la opinión general respecto a la asignatura es muy positiva. Los alumnos han valorado su utilidad, al ser la asignatura del grado más centrada en herramientas gráficas, y mencionan haber disfrutado desarrollando juegos, pues les ha permitido observar satisfactoriamente los resultados del trabajo realizado.

Se observa por tanto una acogida mayoritariamente positiva al enfoque eminentemente práctico de la asignatura de "Interfaces Inteligentes". Los alumnos han agradecido el aprendizaje del lenguaje C# y la plataforma Unity 3D y se han visto motivados por la posibilidad de crear sus propias aplicaciones de realidad virtual, aunque no se aprecia un cierto desencanto respecto al Framework Google Cardboard y al uso de Vuforia. Se puede por tanto plantear reducir la carga práctica de dichos elementos, en favor de una introducción a la herramienta de modelado Blender.

REFERENCIAS

[1] G. Fischer and B. Reeves, "Beyond intelligent interfaces: Exploring, analyzing, and creating success models of cooperative problem solving," Applied Intelligence, vol. 1, issue 4, pp. 311–332, May 1992, Kluwer Academic Publishers.

[2] A. Malizia and A. Bellucci, "The Artificiality of Natural User Interfaces," Magazine Comm. ACM, vol. 55, issue 3, pp. 36–38, March 2012, ACM, New York, NY, USA

[3] T.M. Alisi, A. Del Bimbo and A. Valli, "Natural interfaces to enhance visitors' experiences", IEEE MultiMedia, Vol. 12, issue 3, pp. 80–85 Sept. 2005.

[4] R. Francese, I. Passero and G. Tortora, "Wiimote and Kinect: gestural user interfaces add a natural third dimension to HCI," AVI '12 Proc. Int. Working Conf. Advanced Visual Interfaces, pp. 116–123, May 2012, ACM New York, NY, USA.

[5] P. Santana, "Interfaces Naturales de Usuario - La Experiencia de la Universidad de Colima," Software Guru, vol. 43.

[6] J. Marco, E. Cerezo and S. Baldassarri, "Desarrollo de interfaces naturales para aplicaciones educativas dirigidas a niños," VIII Congreso Internacional de Interacción Persona Ordenador, pp. 79–82, 2007.

[7] J. Videla, A. Sanjuán, S. Martínez and A. Seoane, "Diseño y usabilidad de interfaces para entornos educativos de realidad aumentada," Digital Education Review, vol. 31, pp. 61–79, June 2017.

[8] R. Arnay, J. Hernández-Aceituno and E. González, "Teaching kinematics with interactive schematics and 3D models," Computer Applications in Engineering Education, vol. 25, pp. 420–429, 2017, Wiley Periodicals.
[9] Motor de videojuegos multiplataforma Unity3D (unity3d.com)
[10] Plataforma de realidad virtual Google Cardboard (vr.google.com/cardboard)
[11] Paquete de reconocimiento de voz para Windows10 en Unity3D (UnityEngine.Windows.Speech)
[12] Paquete de realidad aumentada Vuforia para Unity3D (developer.vuforia.com)
[13] Programa de modelado 3D Blender (blender.org)