PROYECTO FIN DE MÁSTER

POSGRADO EN INFORMÁTICA

UNIVERSIDAD DE ALMERIA

ESCUELA POLITÉCNICA SUPERIOR

MÁSTER EN INGENIERÍA INFORMÁTICA

"Accesibilidad en Videojuegos Mediante Servicios Cognitivos"

Curso 2017/2018

Alumno:
Raúl García Muñoz

Director:
Antonio Leopoldo Corral Liria

AGRADECIMIENTOS

La realización de este proyecto ha sido posible gracias al apoyo y ayuda de muchas personas a las que me gustaría mostrar mi agradecimiento. De una forma u otra, su aportación ha sido fundamental para llegar al final del camino.

En primer lugar, me gustaría dar las gracias a José Antonio Álbarez Bermejo por prestarle su tiempo y sus conocimientos a un antiguo alumno. Sin él nunca me habría decantado por volver a la universidad y realizar este máster.

También me gustaría agradecer a José Andrés Moreno Ruiz por sus consejos e interés a la hora de hablar de temas que nos apasionan a ambos. Fueron esas charlas sobre .NET las que me llevaron a enfocar mi vida profesional por estos caminos.

A José Antonio Piedra Fernández, por sus consejos e interés. Ha sido muy positivo poder hablar de videojuegos a nivel de desarrollador e intercambiar ideas para este proyecto.

A mi familia y amigos, por su paciencia y comprensión. Han sido unos meses complicados y estresantes, pero siempre me ha reconfortado saber que estabais a mi lado.

Y por último a mi tutor Antonio Leopoldo Corral Liria, por su apoyo y entusiasmo con el proyecto. Hemos tenido mil y un problemas y contratiempos, pero al final lo hemos conseguido.

Gracias a todos.

CONTENIDO

INTRODUCCIÓN

En el Congreso Nacional de Discapacidad "Accesibilidad Universal en el Siglo XXI", celebrado en noviembre de 2005, tuvo como objetivo la reflexión sobre las posibilidades que han abierto las nuevas tecnologías en el campo de la accesibilidad, y su aprovechamiento por parte de las personas con discapacidad. [1]

Este proyecto pretende mostrar la posibilidad de integrar servicios cognitivos en el ámbito de los videojuegos para mejorar la accesibilidad de estos. En concreto, este sistema permitirá a un usuario con deficiencia visual interactuar con el juego mediante su voz y haciendo uso del lenguaje natural, sin necesidad de memorizar comandos o procedimientos preestablecidos.

El proyecto hace uso de los servicios cognitivos y de aprendizaje automático o machine learning de la plataforma Azure, concretamente sus servicios de voz y lenguaje [2]. Estos servicios se encargarán de realizar el trabajo de transcripción e interpretación, con lo que permiten la toma de decisiones en base a los resultados obtenidos. En la *Ilustración 1* se muestra de forma conceptual el procedimiento descrito.

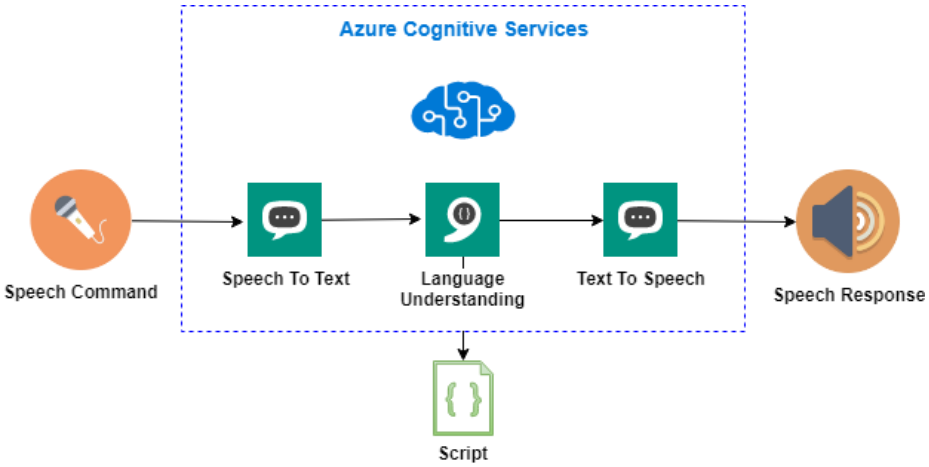


Ilustración 1. Proceso de transcripción, interpretación, acción y realimentación.

Para demostrar el potencial de estos servicios y su integración, se modificará un pequeño juego desarrollado en Unity3D [3] y programado en C# donde el usuario deberá realizar una serie de tareas e interacciones sin la necesidad de recibir información visual. El usuario se comunicará con el juego mediante su voz y recibirá información a través de su oído.

El proyecto estudiará los requerimientos de accesibilidad en base a guías desarrolladas para tal fin. Con estos requerimientos, se pretende implementar el mayor número de características de accesibilidad a nivel audiovisual que complementen al sistema de comandos de voz.

El objetivo último es llamar la atención sobre la necesidad de implementar sistemas que mejoren la accesibilidad a los videojuegos para personas con algún tipo de minusvalía. Además, se explorarán las

posibilidades que ofrece la computación cognitiva [4] a la hora de resolver problemas desde un punto de vista diferente al de la algoritmia clásica.

El proyecto ha sido dividido en 7 fases:

- Recopilación del documentación e información sobre los diferentes servicios y tecnologías empleados.
- Análisis del problema y planificación del desarrollo del proyecto mediante la redacción de un anteproyecto.
- Implementación de las librerías de interacción con los servicios cognitivos de Azure, en preparación para su integración dentro del videojuego.
- Adaptación del videojuego a las características del proyecto.
- Integración de los servicios cognitivos en el videojuego e implementación de una lógica de negocio para la toma de decisiones.
- Depuración y prueba del videojuego.
- Redacción de la memoria del proyecto.

En la gráfica de la *Ilustración 2* se representa la temporización del proyecto según las fases y el número de horas empleados. En total, el proyecto empleará 192 horas de trabajo para su elaboración.

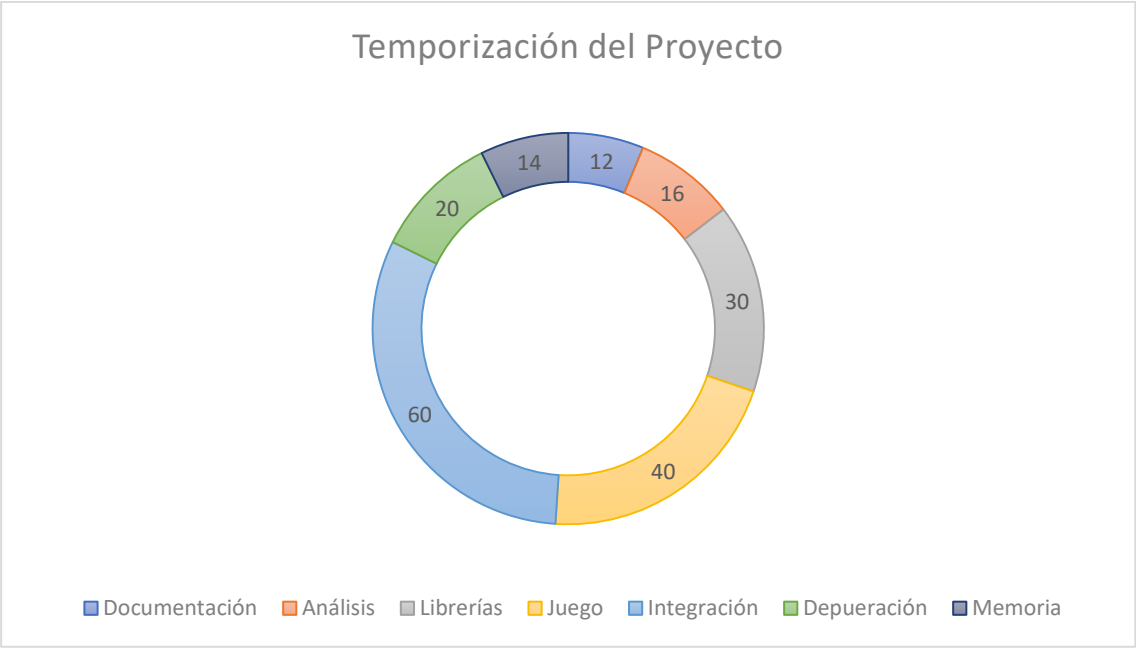


Ilustración 2. Temporización del Desarrollo del Proyecto

La memoria se estructura en seis capítulos. En el primero capítulo se exponen conceptos básicos sobre accesibilidad, clasificación y normativa vigente. En el segundo capítulo se introduce la computación cognitiva, sus caracterizas y los diferentes servicios que ofrece la plataforma Azure. En el tercer capítulo se habla del videojuego, de sus mecanismos y adaptaciones. En el cuarto capítulo se expone el desarrollo de las librerías para el uso de servicios cognitivos, su diseño y compatibilidad con Unity3D. En el quinto capítulo se expone la integración cognitiva en el videojuego, el estudio de su accesibilidad, el desarrollo del cliente, lógica de negocio y toma de decisiones. En el capítulo seis se exponen las conclusiones, dificultades encontradas y proyectos futuros.

CAPÍTULO 1: ACCESIBILIDAD

Se define la accesibilidad o accesibilidad universal como el grado en el que todas las personas pueden utilizar un objeto, visitar un lugar o acceder a un servicio, independientemente de sus capacidades técnicas, cognitivas o físicas. [5]

La accesibilidad es por tanto una condición previa necesaria para la vida autónoma e independiente de las personas, así como su plena participación social y económica.

Los sistemas interactivos son un campo en el que se centra cada vez más la mejora de la accesibilidad. Esto es debido a que dichos sistemas se fundamentan en la interacción Humano-Ordenador, lo que conlleva un ciclo de alimentación y realimentación en ambas direcciones, como por ejemplo un cajero automático o una aplicación de ordenador.

Se entiende por tanto que los videojuegos son sistemas interactivos ya se requiere una interactuación entre el jugador y el sistema. El tipo de interactuación puede ser auditiva, visual, motora y cognitiva.

Según la fundación *AbleGamers* [6], más de 33 millones de jugadores poseen algún tipo de discapacidad. Esto implica la dificultad o imposibilidad de acceder a una gran cantidad de videojuegos debido a la falta de opciones de accesibilidad.

1.1 NORMATIVA

La *Convención de las Naciones Unidas sobre los derechos de las Personas con Discapacidad (CRPD)* [7], celebrada en 2006, es el primer instrumento internacional jurídicamente vinculante que establece unos estándares mínimos para los derechos de las personas con discapacidad y la primera convención de derechos humanos a la que la UE ha pasado a ser parte.

El consejo adoptó la decisión relativa a la celebración de la Convención el 26 de noviembre de 2009. Por lo que respecta a la UE, la Convención entro en vigor el 22 de enero de 2011. Todos los países de la UE la han firmado y ratificado.

Esto significa que tanto la UE como los Estados miembros que toman parte en la Convención de las Naciones Unidas se comprometen a defender y proteger los derechos de las personas con discapacidad, tal como está consagrado en la Convención de la ONU. Los elementos centrales de la Convección de las Naciones Unidas se reflejan en la **Estrategia Europea para la Discapacidad 2010-2020**.

La *Ley Europea de Accesibilidad* [8] tiene como objetivo mejorar el funcionamiento del mercado interno de productos y servicios accesibles eliminando los obstáculos creados por una legislación

divergente. Esto facilitará el trabajo de las empresas y traerá beneficios para discapacitados y personas mayores en la UE.

La Ley de Accesibilidad cubre los productos y servicios que han sido identificados como los que tienen el mayor riesgo de verse afectados por los requisitos de accesibilidad divergentes en todos los países de la UE.

Entre los productos y servicios enumerados, se encuentran ordenadores y sistemas operativos, Smartphones y medios audiovisuales. Si bien los videojuegos no están explícitamente mencionados, cabe destacar el uso de estos productos y servicios mencionados a la hora de hacer uso de dichos videojuegos.

1.2 Clasificación y Niveles de Accesibilidad en Videojuegos

A la hora de mejorar la accesibilidad en un sistema interactivo, es importante clasificar y valorar el nivel de accesibilidad requerido o deseado para cada caso. En el caso de los videojuegos, existen guías diseñadas para orientar al desarrollador a la hora de mejorar la accesibilidad de su juego.

Tabla 1. Guía de accesibilidad

Movilidad	**Nivel 1**	Configuraciones Alternativas Controles de Cámara Teclas Reasignables
	Nivel 2	Acceso a Terceros IU Móvil / Redimensionable Uso de Macros Ajuste de Dificultad y Ayuda Puntos de Guardado Ajuste de Sensibilidad Click-to-Move / Mouse-to-Move Asistente de Movimiento con Teclado
	Nivel 3	Dispositivos de Entrada Ajustes de Velocidad
Audición	**Nivel 1**	Subtítulos Cerrados
	Nivel 2	Tamaño de Fuente Ajustable Color de Fuente Ajustable
	Nivel 3	Ruido Ambiente Reacción de Entrada Alternativa
Visión	**Nivel 1**	Color de Fuente Ajustable Tamaño de Fuente Ajustable Opciones para el Daltonismo Retícula de Alto Contraste Marcado de Enemigos
	Nivel 2	Fuentes Personalizables Interfaces Personalizables Opciones de Mapeado de Color / Vistas Alternativas
	Nivel 3	Opciones de Velocidad Texto-a-Voz
Cognitivo	**Nivel 1**	Tutorial Niveles de Dificultad

	Nivel 2	Niveles de Entrenamiento Menús Intuitivos
	Nivel 3	Opciones de Velocidad Marcado de Enemigos

La *Tabla 1* muestra una clasificación de los diferentes sistemas que pueden ser implementados en un videojuego para mejorar la accesibilidad de este. Ha sido promovida por la fundación *AbleGamers* [6] y escrita por desarrolladores y jugadores con discapacidad. Está guía, llamada "*Includification*", recoge casi una década de investigación, análisis, experimentación y experiencias de jugadores con discapacidad. La tabla está estructurada en base a su funcionalidad y su nivel de integración.

La segunda guía, "*Game Accessibility Guidelines*" [9], se fundamenta en el estudio colaborativo entre académicos, desarrolladores y especialistas. Su objetivo es ayudar a evitar la exclusión innecesaria de jugadores a la hora de diseñar videojuegos.

Su lista se divide en tres niveles de accesibilidad (básica, intermedia y avanzada) y dentro de cada nivel se subdivide en las diferentes funcionalidades a las que va dirigido (motora, cognitiva, visual, auditiva, de habla y general).

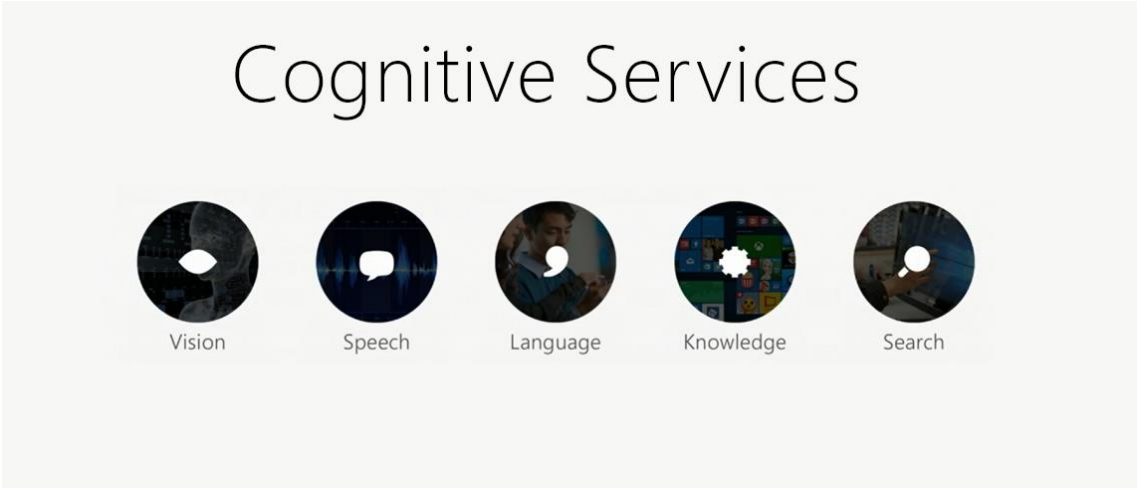
Para el desarrollo de este proyecto, se hará uso de algunas de las indicaciones mostradas en la *Tabla 1*. La inclusión de subtítulos o el ajuste de fuentes formarán parte de las ayudas a la accesibilidad, aunque el pilar central será el uso de comandos de voz. En el apartado *5.1 Tabla de Accesibilidad* del Capítulo 5 se mostrará en detalle los elementos de accesibilidad elegidos para el proyecto.

CAPÍTULO 2: COMPUTACIÓN COGNITIVA

La palabra 'cognitivo' hace referencia a actividades mentales conscientes, tales como pensar, comprender, aprender y recordar.

En relación con la computación, no hay una definición universalmente aceptada para la Computación Cognitiva, pero a alto nivel, podemos pensar en cosas como el reconocimiento de voz, reconocimiento de imagen, texto a voz y voz a texto, […]

A un nivel más profundo, comprende cosas como la Inteligencia Artificial, Aprendizaje Automático, Interacción Persona-Ordenador, Procesamiento del Lenguaje Natural, entre otras. [10]



La computación cognitiva posibilita la computación de una nueva clase de problemas. Aborda situaciones complejas que se caracterizan por la ambigüedad e incertidumbre, características de los problemas propios de los humanos. La información tiene un carácter dinámico, cambiante, volátil y en ocasiones conflictivo. Esto hace que los objetivos del usuario cambien, evolucionando a la par con la información. Para hacer frente a la naturaleza de estos problemas, la computación cognitiva no solo ofrece una síntesis de la información, sino también de influencias, contextos y perspectivas. Para hacer esto, los sistemas a menudo necesitan ponderar la evidencia conflictiva y sugerir una respuesta que es "mejor" en lugar de "correcta". [4]

Para alcanzar este nivel de computación, los sistemas cognitivos deben de ser:

- **Adaptativos**. Deben aprender a medida que la información cambia y los objetivos y requerimientos evolucionan. Deben resolver la ambigüedad y tolerar la imprevisibilidad. Deben diseñarse para consumir datos de forma dinámica en tiempo real, o casi tiempo real.

- **Interactivos**. Deben interactuar fácilmente con los usuarios para que esos usuarios puedan definir sus necesidades cómodamente. También pueden interactuar con otros procesadores, dispositivos y servicios en la nube, así como con otras personas.
- **Iterativo y con estado**. Deben ayudar a definir un problema haciendo preguntas o buscando información adicional de la fuente si la declaración del problema es ambigua o incompleta. Deben "recordar" interacciones previas en un proceso y devolver información que sea adecuada para la aplicación específica en ese punto en el tiempo.
- **Contextual**. Deben comprender, identificar y extraer elementos contextuales tales como el significado, la sintaxis, el tiempo, la ubicación, el dominio apropiado, las regulaciones, el perfil del usuario, el proceso, la tarea y el objetivo. Pueden recurrir a múltiples fuentes de información, incluida información digital estructurada y no estructurada, así como a las entradas sensoriales (visuales, gestuales, auditivas o provistas por sensores).

La computación cognitiva desempeña un papel fundamental en el proyecto gracias a estas características. Para que el videojuego sea capaz de entender los comandos de voz del usuario, será necesario un aprendizaje interactivo y contextualizado. Esto se debe a que las órdenes dadas por el usuario presentarán variaciones léxicas, sintácticas y no siempre con una clara intención o contextualización, por lo que deberán de ser interpretadas.

Existen diferentes ofertas en computación cognitiva, como los servicios cognitivos de *IBM Watson* [11] o los servicios de *Google Cloud* [12]. Para el desarrollo del proyecto, se hará uso de los servicios de *Microsoft Azure* por su facilidad de integración y familiaridad. No obstante, otras posibilidades son igualmente viables.

2.1 Servicios Cognitivos de Microsoft

Microsoft define sus servicios cognitivos, conocidos anteriormente como *Proyecto Oxford*, como un conjunto de APIs, SDKs y servicios disponibles para que los desarrolladores hagan que sus aplicaciones sean más inteligentes, atractivas y reconocibles. Estos servicios permiten una interactuación natural entre el usuario y la aplicación.

A través de la plataforma Azure, Microsoft ofrece servicios orientados a 5 campos diferentes:

- **Visión**. Permite que las aplicaciones y los servicios identifiquen y analicen correctamente el contenido de imágenes y videos.
- **Voz**. Permite la integración de capacidades de procesamiento de voz en cualquier aplicación o servicio. Cuenta con servicios para convertir el lenguaje hablado en texto o producir una voz natural a partir de texto mediante fuentes de voz estándares (o personalizables).
- **Lenguaje**. Permite que las aplicaciones y servicios puedan comprender el significado de un texto sin estructura o reconocer la intención tras la pronunciación de un hablante.
- **Conocimiento**. Permite crear recursos ricos en conocimientos que pueden integrarse a aplicaciones y servicios. Puede extraer bases de conocimiento de listas de preguntas y respuestas y texto sin estructura.
- **Búsqueda**. Permite que las aplicaciones y los servicios hagan uso de un motor de búsqueda a escala Web. Ofrece servicios de búsqueda personalizados y clasificados por entidades.

2.2 Servicios de Voz y Lenguaje

De los servicios nombrados anteriormente, camben destacar los de voz y lenguaje pues son sin duda los más relevantes para el proyecto.

Los servicios de voz hacen referencia a la capacidad de entender la voz humana. Esto se traduce en poder escuchar y hablar con los usuarios filtrando ruido, identificando a los oradores y entendiendo su intención. Dentro de estos servicios se encuentra los de texto a voz, voz a texto, reconocimiento del orador y traducción simultánea entre otros. Es posible entrenar modelos que permitan ajustar los servicios de voz para, por ejemplo, reconocer acentos, filtrar ruido de fondo o vocabulario especifico. [13]

Por otro lado, los servicios de lenguaje ofrecen la posibilidad de procesar texto para reconocer que quiere el usuario. El análisis de texto permite identificar el idioma, las frases clave y entidades que participan en el mismo. El corrector ortográfico permite comprobar y corregir errores producidos por el usuario, reconocer la diferencia entre nombres, argot y comprender homófonos. El traductor permite dar soporte a diferentes idiomas en tiempo real mediante el uso de redes neuronales para la traducción. Los moderadores de contenido facilitan la supervisión y filtrado de imágenes, texto y videos. Permiten la colaboración entre la máquina y el humano para la revisión de contenidos. La comprensión del lenguaje ayuda al ordenador a comprender la intención de la persona, así como las entidades que participan en la misma. Este es sin duda el servicio más importante en la construcción del proyecto. [14]

2.3 LUIS

Language Understanding Intelligent Service [15] es el servicio cognitivo de la plataforma Azure para el reconocimiento del lenguaje contextual. Este servicio aplica aprendizaje automático a una entrada de texto generada por el usuario en lenguaje natural para predecir su significado general y extraer información relevante y detallada.

A los modelos diseñados para un dominio especifico se les denomina aplicaciones. Estos modelos se construyen mediante una lista inicial de *intenciones* generales del usuario, como por ejemplo “abre la puerta” o “apaga la luz”. Después se aportan frases de ejemplo, llamadas *enunciados*, para cada intención. Adicionalmente, es posible se marcan las palabras que se consideran relevantes dentro del enunciado. Estas palabras, llamadas *entidades*, aportan información adicional al enunciado como por ejemplo el destino de un vuelo o la hora de una reserva.

Una vez que el modelo ha sido diseñado, entrenado y publicado, está preparado para empezar a recibir y procesar enunciados. La aplicación de LUIS recibe el enunciado como una petición HTTP y responde con la intención del usuario extraída. La aplicación cliente envía el enunciado y recibe la respuesta de LUIS como un objeto JSON. La *Ilustración 3* muestra el proceso descrito anteriormente.

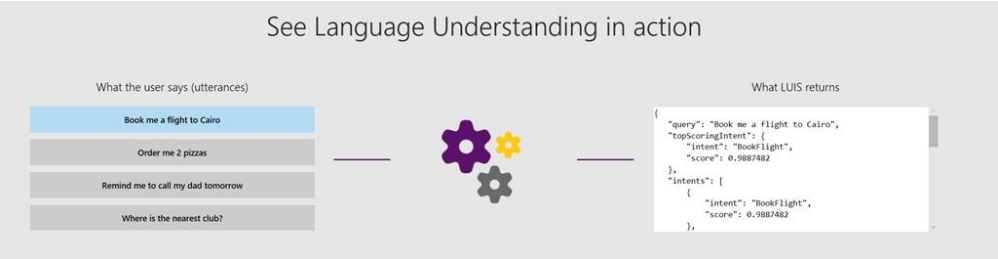


Ilustración 3. Proceso de Análisis de LUIS

2.3.1 INTENCIONES

Una intención representa una tarea o acción requerida por el usuario. Este propositivo u objetivo está expresado en la entrada del usuario, como reservar un vuelo, pagar un recibo o encender una luz.

Una aplicación está compuesta por un conjunto de definiciones de intenciones que corresponde con acciones que el usuario quiere realizar. Por ejemplo, una aplicación para controlar la domótica de una casa puede definir una función llamada '*EncenderLuz*', que se extrae del enunciado "*Enciende la luz del salón*".

Todas las aplicaciones vienen con una intención predefinida denominada '*None'*. Esta intención se utiliza generalmente para clasificar las declaraciones del usuario que son irrelevantes para la aplicación como, por ejemplo, el enunciado "*Resérvame un vuelo*" carece de contexto para la aplicación de domótica. Otro posible uso es el de negación o cancelación de una acción. Enunciados como "*para*", "*olvídalo*" o "*vuelve*" pueden ser interpretados como una cancelación de la acción.

Es recomendable usar solo las intenciones necesarias para el funcionamiento de la aplicación. Si se definen demasiadas intenciones, LUIS tendrá más dificultades para clasificar enunciados correctamente. Si por el contrario el número de intenciones es insuficiente, se corre el riesgo de solapar dichas intenciones debido a su generalidad.

2.3.2 ENTIDADES

Las entidades son palabras importantes dentro de los enunciados que contienen información relevante a la intención, siendo en ocasiones esenciales para la misma. Las entidades pertenecen a clases de objetos similares.

En el enunciado "*Enciende la luz del salón*", se puede entender la palabra '*salón'* como una entidad de tipo ubicación. LUIS reconoce las entidades mencionadas en la entrada del usuario y ayuda a elegir acciones específicas que cumplan con la intención. No es necesario crear una entidad por cada concepto presente en la aplicación. Basta con definir las que son requeridas por el modelo para realizar una acción. En la *Tabla 2* se enumeran los tipos de entidades disponibles en LUIS.

Tabla 2. Tipos de entidades

Tipo	Descripción
Predefinidas	Tipos preexistentes que representan conceptos comunes como fechas, tiempo y geografía.
Lista	Entidades que representan un conjunto fijo de palabras relacionadas en el sistema. Cada entidad lista puede tener una o más formas. No son aprendidas por aprendizaje automático, por lo que son adecuadas para representar un concepto con variaciones fijas (ej. sinónimos). Estas entidades no tienen que ser etiquetadas en los enunciados o entrenadas por el sistema.
Simple	Es una entidad genérica que describe un concepto simple.
Jerárquica	Una entidad heredada define una categoría y a sus miembros. Está formada por entidades hijo que forman los miembros de la categoría. Se pueden utilizar estas entidades para definir relaciones jerárquicas o heredadas entre entidades, en la que los hijos son subtipos de la entidad padre.
Compuestas	Están formadas por otras entidades que forman parte de un todo. Estas entidades se construyen a partir de entidades simples preexistentes, hijos de entidades jerárquicas o entidades predefinidas.

2.3.3 Enunciados

Los enunciados son entradas del usuario que necesitan ser interpretadas por la aplicación. El proceso de entrenamiento para extraer intenciones y entidades requiere capturar una variedad de diferentes entradas para cada acción. El aprendizaje activo, o el proceso de entrenamiento continuo para nuevos enunciados, es esencial para el aprendizaje automático inteligente que aporta LUIS.

A la hora de seleccionar enunciados que sean relevantes para el modelo, es importante prestar atención al lenguaje, la terminología o los posibles errores cometidos por el usuario.

Si los usuarios suelen cometer errores de entrada, es importante añadir un servicio de corrección ortográfica o entrenar al modelo con errores.

Cuando se elige un enunciado, hay que tener en cuenta que un término o frase común al desarrollarlo puede no ser conocida por parte del usuario de la aplicación. Puede que no conozcan o tengan poca experiencia con el dominio, por lo que es importante no utilizar frases o expresiones que el usuario solo utilizaría si fuera un experto.

Es importante utilizar términos variados y evitar las repeticiones. LUIS es capaz de inferir sinónimos por contexto, aunque es mejor introducir variedad para entrenar.

2.3.4 Portal

LUIS ofrece un portal web para la creación, gestión y entrenamiento de aplicaciones. Este portal permite la definición de intenciones, entidades y parámetros relacionados con el modelo, así como su entrenamiento y publicación.

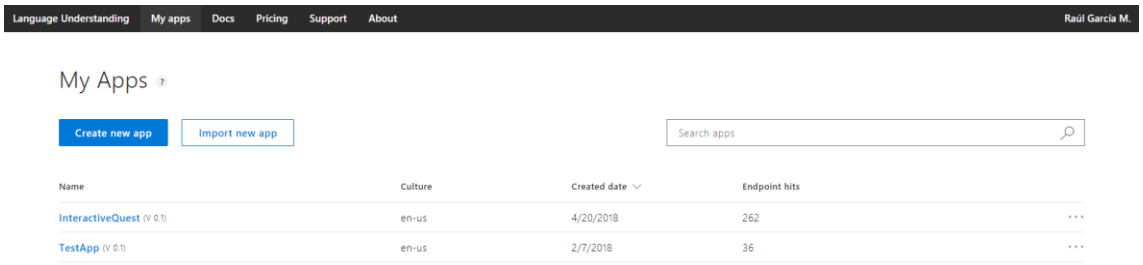


Ilustración 4. Portal de LUIS

Las aplicaciones creadas están sujetas un idioma y cultura específicos, los cuales no pueden ser cambiados una vez han sido instanciadas dichas aplicaciones. Esto quiere decir que por cada idioma al que se le de soporte, se tendrá que crear una aplicación en LUIS. En la *Ilustración 4* se muestra la página principal del portal de LUIS con algunas aplicaciones disponibles.

Una vez creada la aplicación, se presenta un panel de gestión mostrando por defecto la lista de entidades, tal y como muestra la *Ilustración 5*. A la hora de crear una nueva entidad, se especifica el nombre de dicha entidad y se proporcionan algunos enunciados de ejemplo para empezar a entrenar el modelo. Estos ejemplos han de ser representativos de lo que se espera que el usuario introduzca.

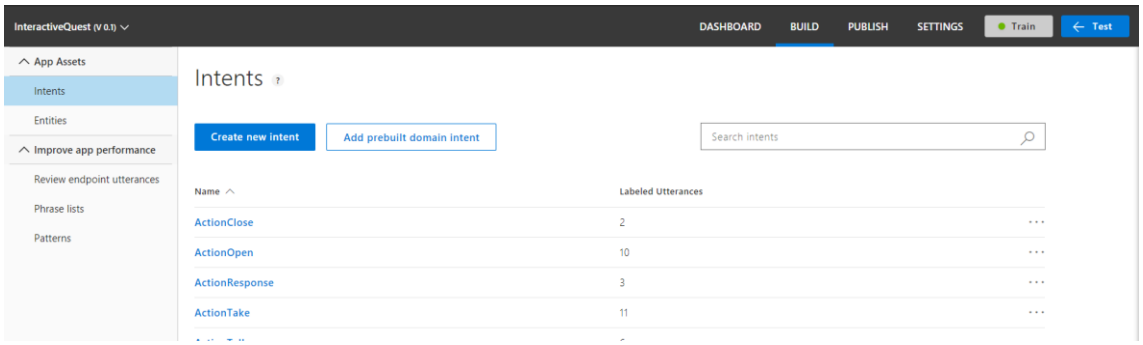


Ilustración 5. Listado de Intenciones

En el caso de existir entidades relevantes para la intención, estas son marcadas en los enunciados. En la *Ilustración 6* se muestran las entidades marcadas (izquierda) y el grupo al que pertenecen (derecha). Como ya se ha mencionado anteriormente, es importante entrenar al modelo con algunos errores que pueda cometer el usuario.

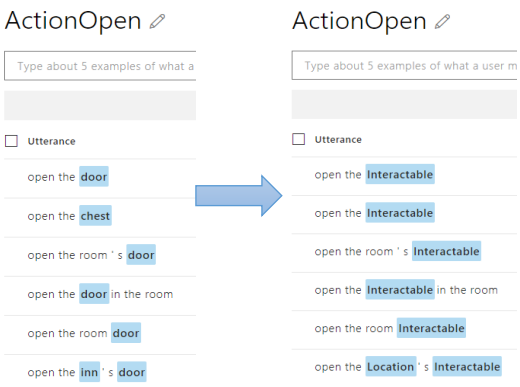


Ilustración 6. Marcado de Entidades en el Enunciado

En el apartado de entidades, se muestran una lista con el nombre de la entidad, su tipo y el número de enunciados en los que aparece. En la *Ilustración 7* se muestra la lista de entidades de la aplicación. Nótese el caso particular de las entidades de tipo lista, que no muestran los enunciados en los que se encuentran referenciadas. Esto se debe a que este tipo de entidades están compuestas por un conjunto de palabras clave con sus respectivos sinónimos.

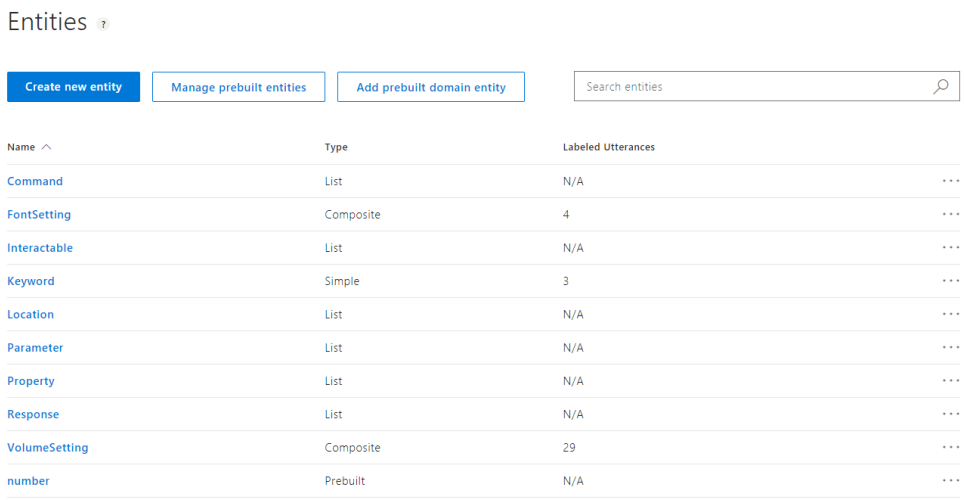


Ilustración 7. Lista de Entidades

Una vez que las intenciones y la entidades han sido asignadas, se procede al entrenemiento del modelo de la aplicación. El portal ofrece la posibilidad de realizar pruebas en tiempo real y comparar los resultados entre el modelo actual y el de producción. La *Ilustración 8* muestra una prueba del modelo con los resultados obtenidos del enunciado propuesto.

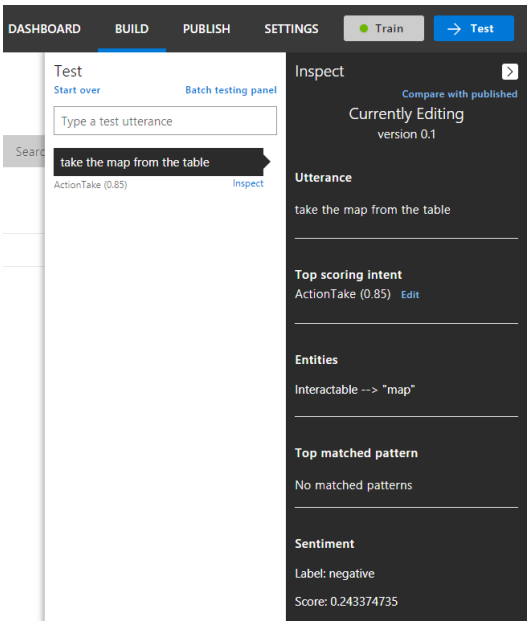


Ilustración 8. Entrenamiento y Test del Modelo

Una vez que el modelo es entrenado, es necesario publicarlo antes de poder acceder mediante un endpoint a este. La *Ilustración 9* muestra la pantalla de publicación, donde es posible especificar el modo de publicación (producción o pre-producción), habilitar servicios externos de análisis o habilidar la corrección ortográfica entre otros.

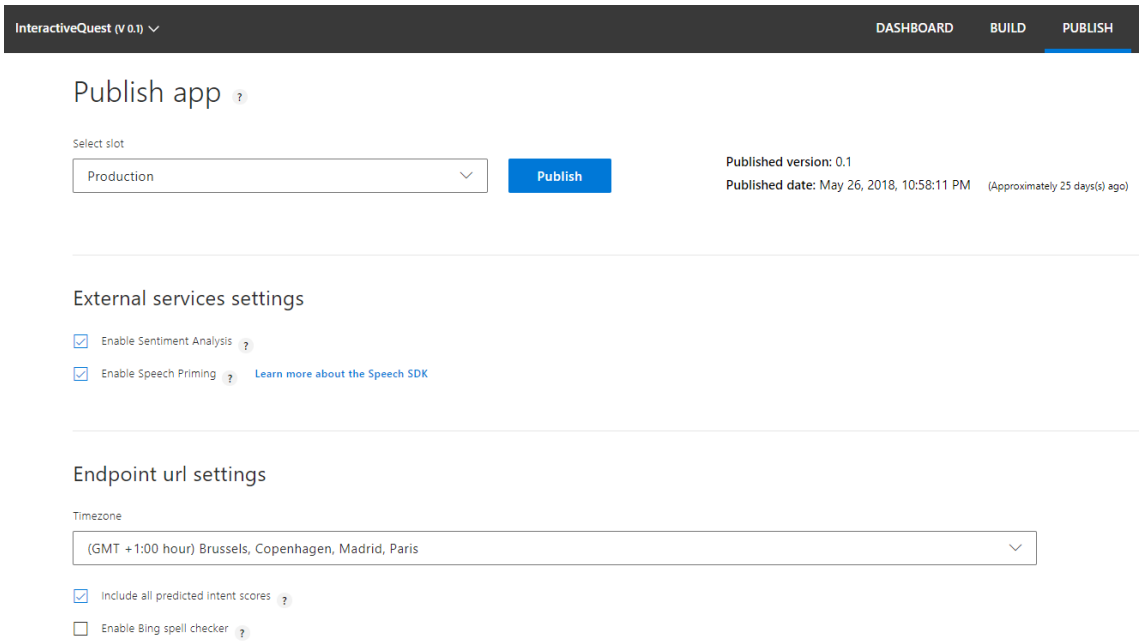


Ilustración 9. Publicación de la Aplicación

Otra de las características que ofrece el portal es la de revisión de enunciados del usuario. Cuando el usuario envía una orden cuya intención no es clara, el sistema marca dicho enunciado como dudoso para su posterior revisión. De este modo se puede comprobar si la intención recibida se ha categorizada correctamente o se trata de un error introducido por el usuario. Una vez que se realiza el arbitraje correspondiente, el modelo puede ser nuevamente entrenado y de esta forma ajustarlo al comportamiento del usuario. En la *Ilustración 10* se muestran algunos enunciados pendientes de revisión.

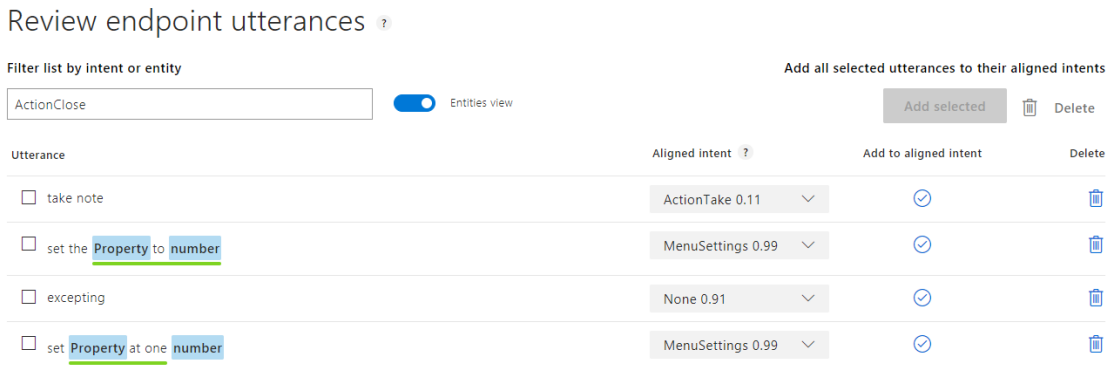


Ilustración 10. Revisión de los Enunciados del Endpoint

CAPÍTULO 3: EL VIDEOJUEGO

Un videojuego es un juego que jugamos gracias a un aparato audiovisual y que puede basarse en una historia. [16]

Tal y como propone el profesor *Nicolas Esposito* en su trabajo [16], un videojuego puede ser definido a través de otros términos estrechamente relacionados con el mismo, como son el juego, la interacción y la narrativa.

Esposito hace referencia al dispositivo audiovisual en referencia a un sistema de computación (ordenador, consola, etc.), dispositivos de entrada (teclado, mando, etc.) y dispositivos de salida (pantallas, altavoces, etc.). Independientemente del dispositivo en el que se juegue, se produce una interacción entre humano y computador por lo que los juegos se pueden ver como interfaces de usuario. Es por esto por lo que podemos hablar de interactividad.

Para poder la explorar las diferentes posibilidades de accesibilidad e integración, se propone utilizar como base para el juego el ejemplo propuesto en la sección de tutoriales de Unity3D [17]. Este ejemplo de juego de aventuras cuenta con un sistema de interacción fácilmente extensible y que permite mostrar las diferentes posibilidades que ofrece la computación cognitiva a la hora de mejorar la accesibilidad. Además, es posible añadir más sistemas, como menús o diálogos, y mostrar un corte vertical como ejemplo de videojuego en producción.

3.1 CONCEPTOS BÁSICOS

El juego se basa en el concepto de una aventura interactiva de tipo *point-and-click* en un escenario 3D. La historia tratará sobre un aventurero que se encuentra hospedado en una posada y se prepara para emprender su viaje. El jugador tendrá que recopilar una serie de objetos situados en su habitación y abrir la puerta para poder dirigirse al comedor, situado en la planta baja.

Después podrá interactuar con el posadero mediante un simple dialogo, con la opción de aceptar un regalo por parte del posadero o rechazarlo. Si posee todos los objetos en su inventario, entonces podrá pagar por la habitación y abrir la puerta de la posada para terminar la aventura. En caso contrario, el jugar deberá ir a buscar los objetos que haya dejado atrás.

El jugador tendrá la posibilidad de acceder a los menús del juego para ajustar una configuración básica, además de pausar el juego en cualquier momento.

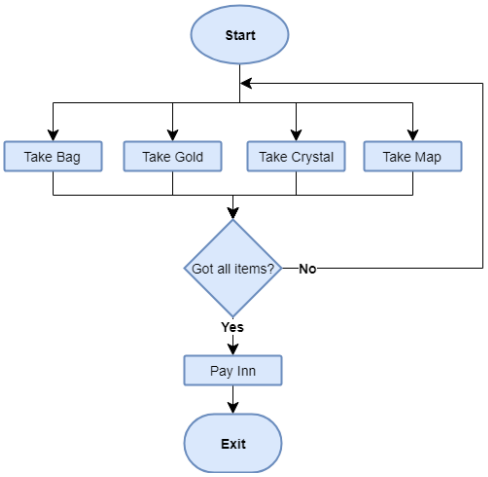


Ilustración 11. D.F. de Lógica del Juego

La *Ilustración 11* muestra la lógica simplificada del juego para llegar a la condición de victoria. El jugador debe recoger 4 objetos, sin orden especifico. Una vez que todos los objetos están en su posesión, puede hablar con el posadero, pagar la posada y salir por la puerta.

3.1.1 MOVIMIENTO

Al tratarse de un *point-and-click*, la interacción con el juego se realiza principalmente con el ratón. Para mover al personaje por el escenario, se realiza un clic de ratón en el punto de destino deseado. El sistema de eventos registra el clic y obtiene el punto de impacto mediante el uso de *raycasting* desde la cama al escenario, tal y como se muestra en la *Ilustración 12*.

Para que el punto calculado sea válido, ha de producirse en una zona especial definida por la entidad *NavMesh* [18] y con un atributo que designe dicha zona como navegable.

Una vez que se registra el clic se invoca a la función *OnGroundClick*, encargada de gestionar dicho evento. Esta función se encarga en primer lugar de comprobar que exista un camino entre el personaje y el destino. Es útil en el caso de encontrarse una puerta cerrada o un obstáculo insalvable.

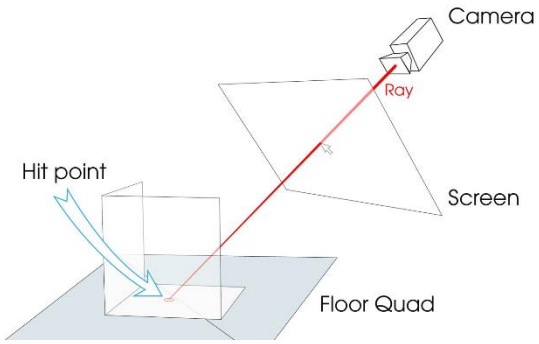


Ilustración 12. Proceso de Raycasting [30]

Adicionalmente, se comprueba que las coordenadas de destino se encuentren dentro de la zona de navegación. En caso positivo, se asigna el punto calculado como nuevo destino. En caso contrario, se calcula un punto cercano al original y que se encuentre dentro de la zona de navegación antes de ser asignado.

Después de asignar el nuevo destino, el movimiento del personaje se realiza mediante el agente de navegación [19]. Este agente se encarga de calcular el camino a seguir desde la posición actual del personaje hasta el punto de destino.

La *Ilustración 13* muestra el diagrama de flujo de la función de movimiento. Cuando el sistema de eventos registra un clic en el escenario, se invoca la función *OnGroundClick*. Lo primero que se hace es reiniciar la variable que contiene el interaccionable actual, pues se trata de un desplazamiento por coordenadas. Acto seguido, se leen las coordenadas del clic y se comprueba que se puede llegar hasta la posición descrita por las mismas. Si no es posible, la función termina sin desplazar al personaje.

En caso de poder llegar hasta el punto, se comprueba si este se encuentra dentro del *NavMesh*. Si no es así, se calcula un punto válido lo más cerca posible al original. Por último, se actualiza la posición de desplazamiento. El agente de navegación se encargará de realizar el desplazamiento del personaje.

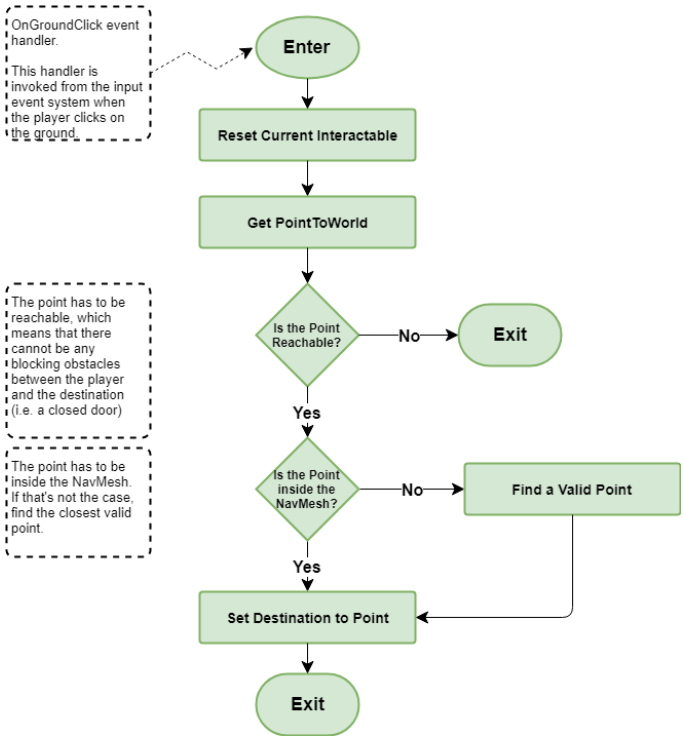


Ilustración 13. D.F. Función de Movimiento

La *Ilustración 14* muestra una representación del *NavMesh* (en azul) en la habitación donde comienza el jugador. Se puede observar cómo está delimitada la zona 'caminable', sorteando los diferentes obstáculos repartidos por la habitación.

Ilustración 14. Visualización del NavMesh (Azul)

El agente de navegación, mostrado en la *Ilustración 15*, es el encargado de desplazar al personaje a través del *NavMesh*. Dicho agente se compone de un colisionador en forma de cilindro que representa, a grandes rasgos, el volumen del personaje. Además, contiene información sobre la velocidad, velocidad angular, aceleración y diferentes parámetros para sortear obstáculos.

Cuando el agente recibe un nuevo destino, calcula un camino a través del *NavMesh* mediante vectores de recorrido. Una vez que se termina el cálculo, el agente inicia el desplazamiento del personaje a través de los vectores de recorrido. Dicho desplazamiento está representado en la *Ilustración 15* por un vector de dirección.

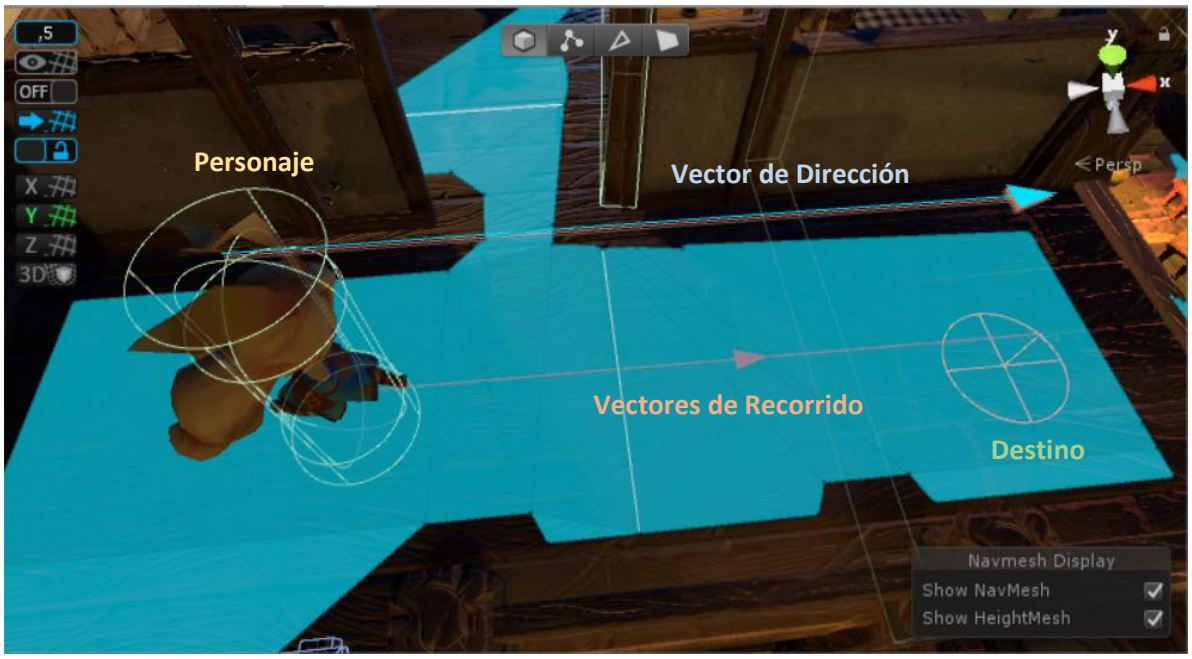


Ilustración 15. Agente de Navegación (NavigationAgent)

3.1.2 Interacción

Se entiende como interacción a la acción que se produce entre el personaje y una entidad, distinta de la zona de navegación, al hacer clic sobre ella. Dichas entidades, llamadas *interaccionables*, están compuestas por objetos (un mapa, monedas de oro, etc.) o individuos (el posadero) que son relevantes para el desarrollo del juego.

A nivel lógico, se tratan de objetos compuestos por una serie de propiedades con información acerca de la interacción. Cuentan al menos con la posición donde se ha de producir la interacción (donde se parará el personaje y en qué dirección mirará), y una colección de reacciones por defecto. Esta colección se usará en caso de que no exista otras condiciones o que estas no se cumplan. Por último, estos objetos pueden contar con colecciones adicionales, y sus reacciones. La *Ilustración 16* muestra el diagrama de la clase *Interactable*.

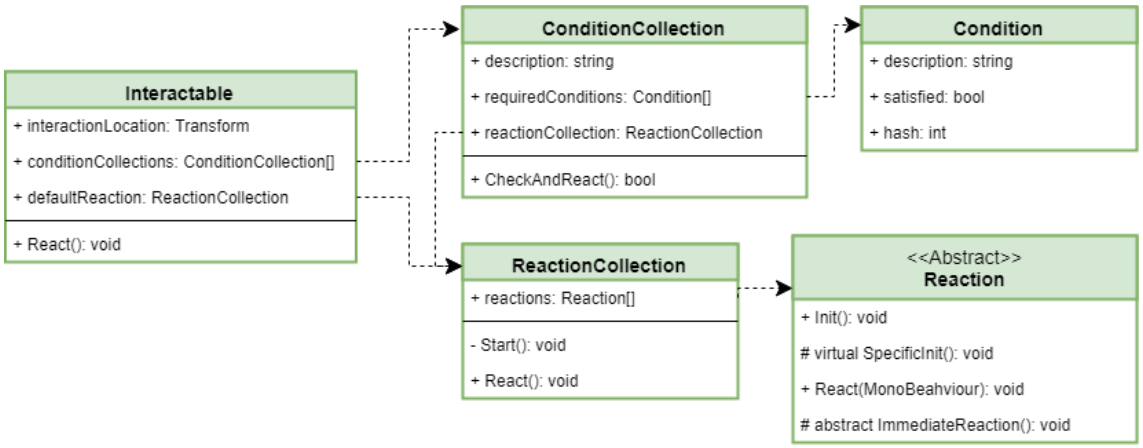
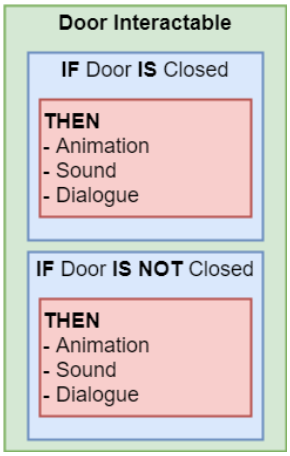


Ilustración 16. Diagrama de la Clase Interactable

Al producirse una interacción, se invoca la función *React* del objeto *Interactable*. Esta función se encarga de comprobar si existen colecciones de condiciones el vector *conditionCollections*. En caso afirmativo, invoca la función *CheckAndReact* de la clase *ConditionCollection*. Esta función comprueba si se cumplen todas las condiciones descritas en el vector *requiredConditions*. De cumplirse todas las condiciones, se invoca la función *React* de la clase *ReactionCollection*. Esta función se encarga de recorrer el vector *reactions* y ejecutar todas las reacciones que contiene. En caso de que no se cumplan las condiciones de alguna de las colecciones, se usa *defaultReaction* como la colección de reacciones a ejecutar por defecto.



Tomando como ejemplo el caso de una puerta interaccionable, se pueden distinguir dos condiciones (abierta o cerrada). Al interaccionar el personaje con la puerta, se evalúan sus dos condiciones. Si la puerta está abierta, se ejecutan todas las reacciones asociadas a dicha condición, como reproducir una animación, un sonido, etc. El caso contrario tiene su propio conjunto de reacciones independientes.

Este sistema es fácilmente extensible, con múltiples reacciones y condiciones. Por ejemplo, es posible usar solo la reacción por defecto en caso de querer abrir un cofre, sin tener que cerrarlo después.

Otro uso del que se hablará en el Capítulo 5, es el del empleo de objetos interaccionables como balizas de navegación e información. Esto es posible debido a que estos objetos no son visibles dentro del juego y pueden hacerse no cliqueables, por lo que es posible integrarlos sin que se entorpezca la navegación mediante el ratón.

La *Ilustración 17* muestra el diagrama de flujo de la función de interacción. Siguiendo el ejemplo para la función de movimiento, cuando el sistema de eventos registra un clic en el disparador del interaccionable, se invoca a la función *OnInteractableClick*. Lo primero que se hace es comprobar que se puede llegar hasta la posición donde se encuentra el interaccionable. Si no es posible, la función termina sin desplazar al personaje ni ejecutar ninguna acción.

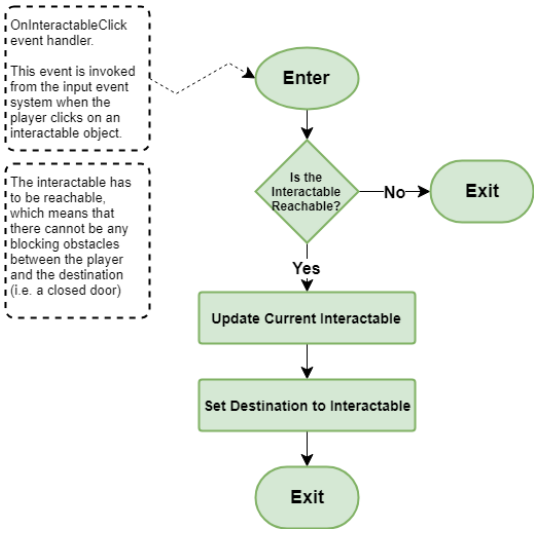


Ilustración 17. D.F. función de interacción

En caso de poder llegar hasta el objeto, se actualiza la variable que contiene al interaccionable actual y la posición de desplazamiento. A partir de aquí, el agente de navegación se encargará de mover al personaje hasta la posición de interacción, donde se disparará el evento de interacción.

3.1.3 Interfaz

Unity cuenta con su propio sistema para gestionar la interfaz de usuario mediante el uso del ratón. El juego cuenta con un menú principal que permite iniciar el juego, salir y realizar ajustes en el volumen general, música, efectos y ambiente. Más adelante se ampliará con ajustes para mejorar la accesibilidad.

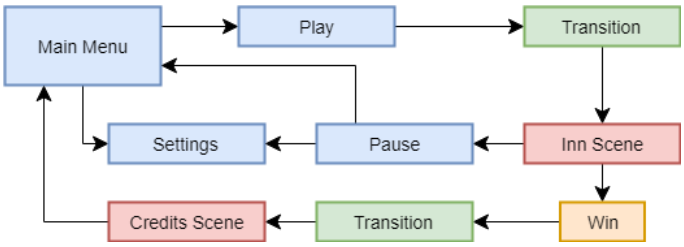


Ilustración 18. Diagrama de Navegación entre la UI y las Escenas del Juego

La *Ilustración 18* muestra el diagrama de navegación dentro del juego. Desde el menú principal se puede acceder a la pantalla de configuración o empezar una nueva partida. Una vez que empieza la partida, es posible acceder al menú de pausa pulsando la tecla de escape. En este menú se puede acceder a la configuración, volver al menú principal o resumir la partida. Por último, una vez que se llega a la condición de victoria y se muestran los créditos se vuelve al menú principal.

3.2 ESPECIFICACIONES

El juego se ha actualizado a la versión 2007.3.1f1 [20] de Unity3d. Para las librerías de C# de scripting en tiempo de ejecución, se utilizará la versión experimental 4.6 y C# 6 [21]. La razón de utilizar una versión experimental es la necesidad de hacer uso de un conjunto de librerías no disponibles en la versión 3.5.

Adicionalmente, esta actualización permite el uso de métodos asíncronos en lugar de corrutinas. Es especialmente útil a la hora de realizar peticiones HTTP para comunicarse con los diferentes servicios cognitivos. Además, se eliminan dos problemas que tienen las corrutinas, como es la imposibilidad de devolver valores y la dificultad de gestionar errores.

Como métodos de entrada/salida, se utilizará un ratón y un teclado como dispositivos por defecto. Para la versión cognitiva, se utilizará un micrófono para grabar los comandos de voz del usuario y un mando para activar la grabación de dichos comandos.

CAPÍTULO 4: LIBRERÍAS

A la hora de hacer uso de los servicios cognitivos de la plataforma Azure, es necesario diseñar una librería que se encargue de la comunicación cliente-servidor y que sea compatible con Unity3D. Esto último es especialmente importante pues los juegos son esencialmente sistemas síncronos, por lo que el hilo de renderizado no puede esperar a que una operación de entrada-salida termine.

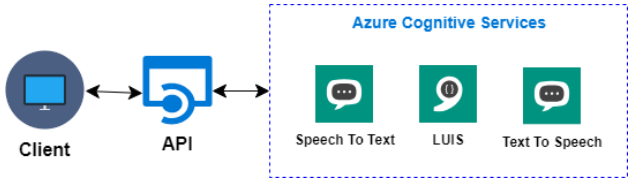


Ilustración 19. Servicios Cognitivos

4.1 DISEÑO

Tal y como se describió anteriormente, se busca usar los servicios de voz y lenguaje para controlar el juego mediante comandos de voz. Para ello, se necesita transcribir el comando dando por el jugador, extraer su intención y devolver una respuesta auditiva. Esta librería se encargará de la comunicación cliente-servidor, así como la grabación de los comandos de voz y reproducción del audio de respuesta. De la interpretación y toma de decisiones se hará cargo la propia lógica del juego, como se verá más adelante.

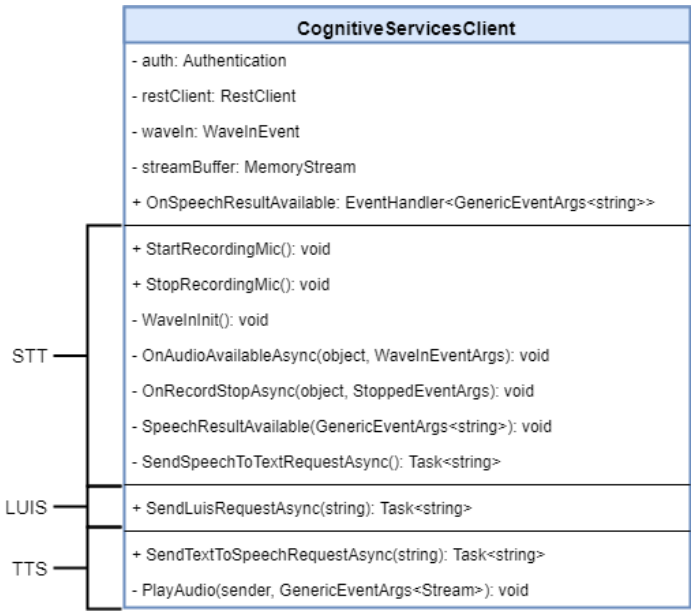


Ilustración 20. Diagrama de la Clase CognitiveServicesClient

El cliente mostrado en la *Ilustración 20* será el encargado de realizar las funciones descritas. Estas operaciones han de realizarse de forma asíncrona, de tal forma que no interfieran con Unity.

4.1.1 Voz a Texto

Los requisitos de esta función son los de grabar la voz del usuario, transmitirla al servidor para su transcripción y devolver el resultado en forma de evento. La comunicación se realizará mediante una petición HTTP a una API REST. El fragmento de audio enviado no puede contener más de 10 segundos de voz y 14 segundos en total. La API solo devuelve resultados completos, no resultados parciales o aproximados.

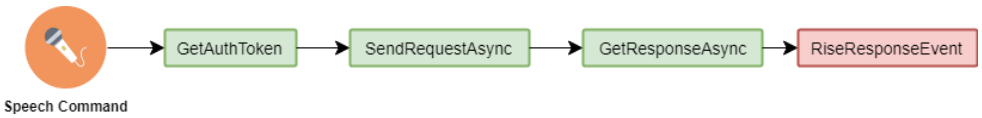


Ilustración 21. Voz a texto

Para la grabación del audio, se hace uso de la librería *NAudio* [22] para C#. Esta librería se encarga de gestionar la entrada de micrófono y el buffer de grabación. En la *Ilustración 20*, el método *WaveInInit* se encarga de la inicialización del cliente *WaveIn*, del buffer de grabación del micrófono y de la suscripción de los métodos *OnAudioAvailable* y *OnRecordStop* a los eventos *DataAvailable* y *RecordingStopped* respectivamente.

Cuando el buffer de grabación se llena con 100ms de grabación, se invoca al método *OnAudioAvailable*. Este método se encarga de escribir el audio en un buffer de memoria para ser enviado al servidor. Adicionalmente comprueba que no se supera el límite de longitud del audio. De ser así, se llama al método *StopRecording* para detener la grabación. La *Ilustración 22* muestra diagrama de flujo de este método.

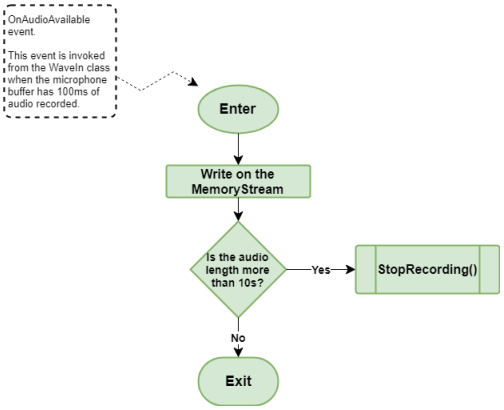


Ilustración 22. D.F. OnAudioAvailable

Una vez que se detiene la grabación, se comprueba que no se ha producido ningún error y se invoca al método *SendHttpRequest*. En la *Ilustración 21* se representa un diagrama simplificado. Este método construye la cabecera de la petición usan un toquen de autorización y especificando los parámetros del audio enviado. Cuando se recibe la respuesta, se llama al método *SpeechResultAvailable*. Este método se encarga de invocar el evento *OnSpeechResultAvailable* con la respuesta como argumento. Por último, se reinicia el buffer de memoria. La *Ilustración 23* muestra el diagrama de flujo del método *OnRecordStop*.

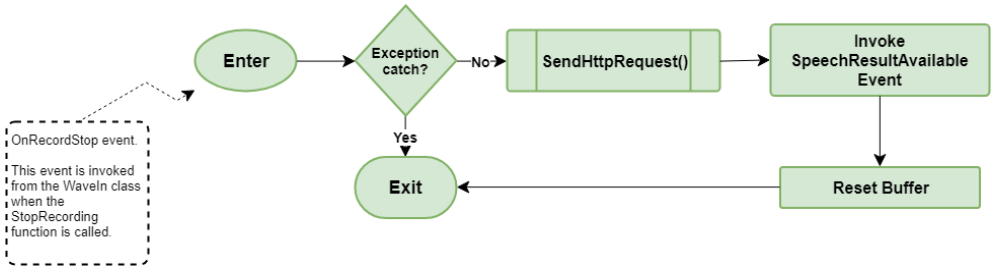


Ilustración 23. D.F. OnRecordStop

4.1.2 LUIS

Esta función se encarga de enviar la transcripción del comando del usuario para que LUIS lo procese y extraiga su intención. La petición se enviará a través de HTTP a una API REST. La *Ilustración 24* muestra un diagrama simplificado. La petición se ha de hacer especificando el identificador de la aplicación de LUIS que se desea usar. El resultado es devuelto como un objeto *JSON* serializado.

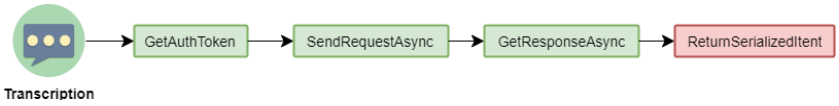


Ilustración 24. LUIS

El método *SendLuisRequestAsync* envía una petición asíncrona al servidor con la transcripción del comando del usuario como parámetro de entrada. Una vez se recibe la respuesta, esta es devuelta en forma de cadena serializada. La deserialización se realizará dentro de la lógica del videojuego, por lo que se expondrá más adelante.

4.1.3 Texto a Voz

Esta función se encarga de enviar una cadena de texto con la respuesta que ha de recibir el usuario para ser convertida a audio y reproducida. La petición se envía a través de HTTP a una API REST. En la petición se especifica el idioma, el género de la voz y la voz que se quiere emplear, así como el tipo de audio que se quiere recibir. La *Ilustración 25* muestra un diagrama simplificado.

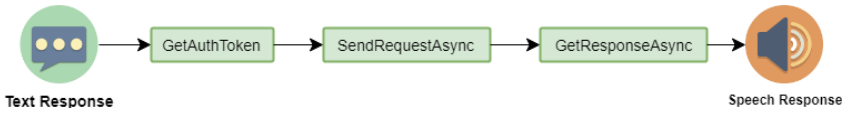


Ilustración 25. Texto a voz

El método *SendTextToSpeechRequestAsync* inicializa un sintetizador encargado de enviar la petición al servidor. Esta clase expone dos eventos llamados *OnAudioAvailable* y *OnError*. Y primero es invocado una vez que se ha recibido el audio y está listo para ser consumido por lo que se suscribir el método *PlayAudio*. El segundo es invocado en caso de encontrar un error por lo que se suscribirá el método *ErrorHandler*. En la *Ilustración 26* se muestra el diagrama de clase de *Synthesize*.

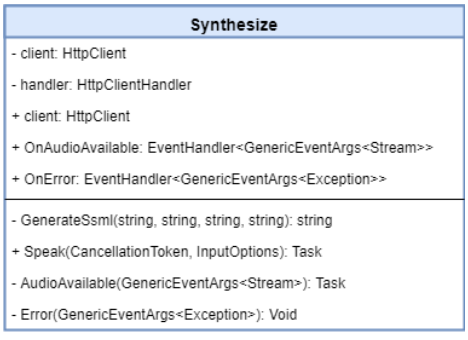


Ilustración 26. Diagrama de Clase Synthesize

Cuando se llama al método *Speak* de la clase *Synthesize*, se validan las cabeceras y se envía la petición al servidor. Si el servidor contesta sin errores, se almacena el audio recibido y se llama a *AudioAvailable* que se encargara de invocar al evento *OnAudioAvailable* pasando como parámetro el *Stream* recibido. En caso de error, se llama a *Error* que se encargará de invocar al evento *OnError*.

Al ser invocado, el método *PlayAudio* crea un objeto de tipo *SoundPLayer* que recibe como entrada el *Stream* de audio recibido. Acto seguido se reproduce el audio de forma síncrona. Cabe destacar que la clase *PlayAudio* pertenece a las librerías de .NET y no está integrada con el sistema de audio de Unity3D. Esto es debido a una incompatibilidad con la versión de Unity3D utilizada en el proyecto.

4.2 COMPATIBILIDAD

Como ya se mencionó anteriormente en el apartado de 3.2 Especificaciones, Unity plantea una serie de restricciones a la hora de poder utilizar librerías en .NET. Unity utiliza C# como lenguaje de scripting y C++ para la implementación de su núcleo. Unity serializa los scripts en C# y los pasa al núcleo para ejecutarlos. Esto quiere decir que las librerías de .NET han de ser en su mayoría adaptadas para poder funcionar en Unity.

Otro punto que destacar es que Unity utiliza como compilador *Mono* [23]. La especificación del compilador usada por Unity suele ir retrasada en su versión comparada con .NET. Esto se hace evidente al intentar hacer uso de los clientes HTTP y los métodos asíncronos, que no están disponibles en la versión 3.5 que trae Unity por defecto [24].

Además, Unity no es *thread safe* [25], por lo que los objetos y métodos propios de Unity se han de mantener fuera de los propios hilos. Esto complica el hacer uso de esos hilos para procesar, por ejemplo, el audio recibido del servicio de texto a voz.

Para la implementación de las funciones de comunicación con la API de Azure y de sintetización de voz, se han usado los ejemplos propuestos por Microsoft en su Git oficial [26]. Esto soluciona en parte los problemas de compatibilidad entre las librearías oficiales de Azure y Unity3D.

CAPÍTULO 5: INTERACTIVE QUEST

Una vez que las librerías están definidas, es el momento de analizar las necesidades de accesibilidad del juego e implementarlas. Este punto es crucial pues se busca mejor mejorar la accesibilidad sin que ello afecte al funcionamiento del propio juego. Por ejemplo, el jugador ha de ser capaz de interactuar con el juego usando tanto comandos de voz como con el ratón. Con esto se busca que cualquier jugador, independientemente de si cuanta con algún tipo de minusvalía, pueda beneficiarse de la mejora de la accesibilidad.

5.1 TABLA DE ACCESIBILIDAD

Usando una guía de accesibilidad tal y como se expuso en el apartado *1.2 Clasificación y Niveles de Accesibilidad en Videojuegos* en el Capítulo 1, se busca seleccionar que mejoras en la accesibilidad se van a implementar, así como su nivel de integración. El resultado obtenido se muestra en la *Tabla 3*.

Tabla 3. Estudio de la Accesibilidad

Movilidad	***Nivel 1***	
	Nivel 2	Click-to-Move / Mouse-to-Move
	Nivel 3	
Audición	***Nivel 1***	Subtítulos Cerrados[1]
	Nivel 2	Tamaño de Fuente Ajustable Color de Fuente Ajustable
	Nivel 3	Ruido Ambiente
Visión	***Nivel 1***	Color de Fuente Ajustable Tamaño de Fuente Ajustable Marcado de Enemigos[2]
	Nivel 2	
	Nivel 3	Texto-a-Voz
Cognitivo	***Nivel 1***	
	Nivel 2	Menús Intuitivos
	Nivel 3	Marcado de Enemigos[3]

En la parte de **movilidad**, no se introducen componentes adicionales pues el juego ya de por si implementa un sistema de *point-and-click* para realizar las acciones de movimiento e interacción.

En la parte de **audición**, el juego contará con subtítulos, aunque no cerrados. Además, permitirá el ajuste del tamaño de fuente y del color de esta. Por último, el juego hace uso de sonidos ambiente y efectos como pasos para ayudar a crear una imagen metal de lo que está pasando.

[1] Solo subtítulos.
[2] Efecto visual sobre los objetos interaccionables.
[3] Efecto visual sobre los objetos interaccionables.

En el apartado de **visión**, se añade el marcado de enemigos. Esto se aplica a la adición de pequeñas ayudas visuales para reconocer que objetos son interaccionables. Por último, se implementará un sistema de texto-a-voz, aunque esta parte será mucho más amplia.

A nivel **cognitivo** se implementarán menús intuitivos para facilitar la interactuación con el jugador.

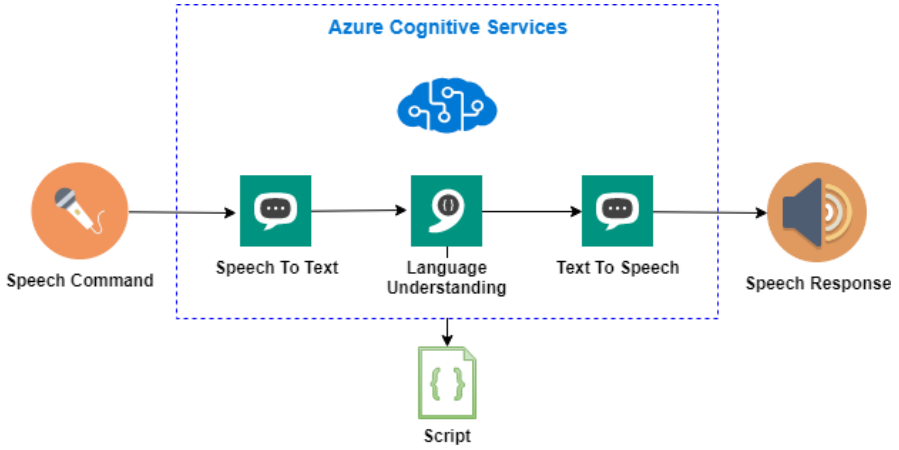


Ilustración 27. Proceso de Transcripción, Interpretación, Acción y Realimentación.

Adicionalmente a lo mostrado en la *Tabla 3*, el jugador podrá solicitar información espacial respecto a su posición actual y localizaciones a las que puede desplazarse, e información sobre los objetivos a cumplir. Esto es importante pues en ausencia de imagen, el jugador ha de ser capaz de ubicarse en el escenario y comprender en todo momento que tiene que hacer.

5.2 Actualización de la Interfaz E/S

Es necesario actualizar las interfaces de entrada/salida y ajustar las acciones que puede realizar el usuario. Se necesita buscar acciones equivalentes al sistema de entrada mediante ratón. Tiene sentido desplazarse a una posición en el escenario dada por un vector (x,y,z), correspondiente a un clic de ratón, no siendo así dando esas mismas coordenadas por un comando de voz. En la *Tabla 4* se representan algunas de las posibles alternativas entre ambos modos.

Tabla 4. Alternativas de Accesibilidad

Información Audiovisual	Alternativa Cognitiva
Información de posición espacial	Nodo de posición con descripción.
Navegación por coordenadas	Navegación por balizas predeterminadas.
Información de objetivos	Lectura de objetivos.
Interactuación con diálogos	Presentación de opciones de dialogo mediante voz.
Ajustes mediante menús	Ajuste de parámetros relevantes mediante voz.

En cuanto a la interfaz, se mostrará la transcripción del comando dado por el jugador con su índice de confianza, la intención reconocida y las entidades que participan en la misma, y por último los subtítulos con el dialogo correspondiente a la respuesta dada por el juego. En la *Ilustración 28* se puede ver un ejemplo de la interfaz descrita.

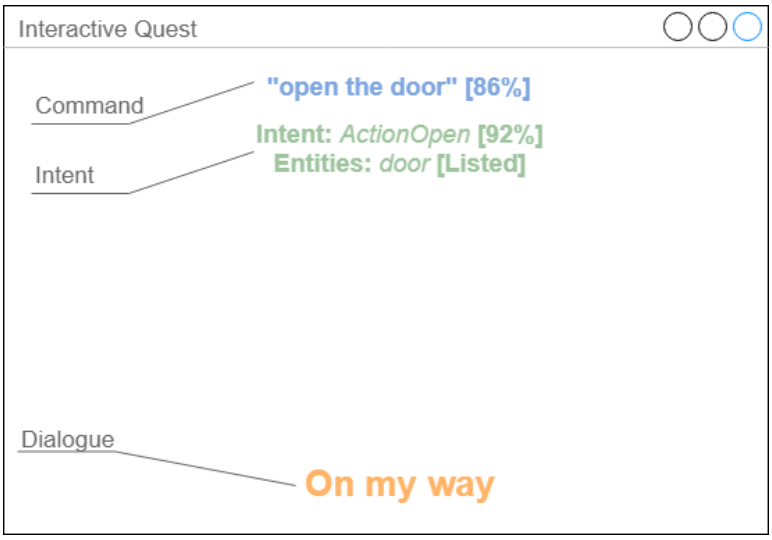


Ilustración 28. Ejemplo de Adaptación de la Interfaz

Para dar un comando de voz, tal y como se muestra en la *Ilustración 29*, el jugador mantendrá pulsado un botón mientras habla. Cuando el jugador suelte el botón o pasen 10 segundos desde que empezó a hablar, se enviará el comando al servidor para ser interpretado, tal y como se expuso anteriormente en el apartado de 4.1.1 Voz a Texto.

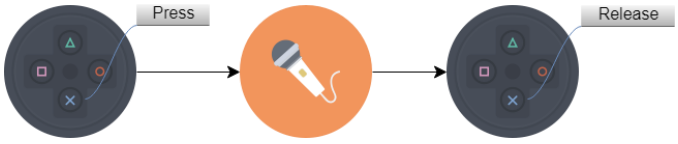


Ilustración 29. Proceso de Introducción de un Comando de voz

Una vez que se procese el comando, se transmitirá una respuesta hablada y escrita al usuario con el resultado de dicho comando.

5.3 Diseño del Cliente

Para hacer uso de las librerías descritas en el apartado 4.1 Diseño del Capítulo 4, es necesario deserializar las respuestas por parte del servidor. Dicha información viene en forma de un objeto *JSON*. En el caso de la transcripción del comando del usuario, el servidor devuelve los n-mejores resultados en forma de vector. Cada resultado cuenta con un índice de confianza e información acerca del léxico, intencionalidad (no confundir con la intención procesada por LUIS), etc. La *Ilustración 30* muestra el diagrama de clase respuesta.

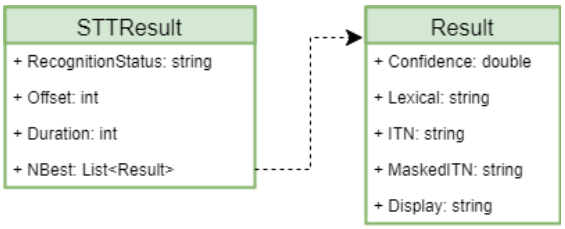


Ilustración 30. Diagrama de la Clase STTResult (Respuesta)

En el caso de LUIS, tal y como se muestra en la *Ilustración 31*, el resultado devuelto por el servidor siempre contiene la consulta y la intención con la puntuación más alta. Además, se incluye una lista con todas las posibles entidades reconocidas y su puntuación (índice de confianza). Adicionalmente, si alguna de las entidades es de tipo compuesto, se devuelven datos concernientes a dicha entidad.

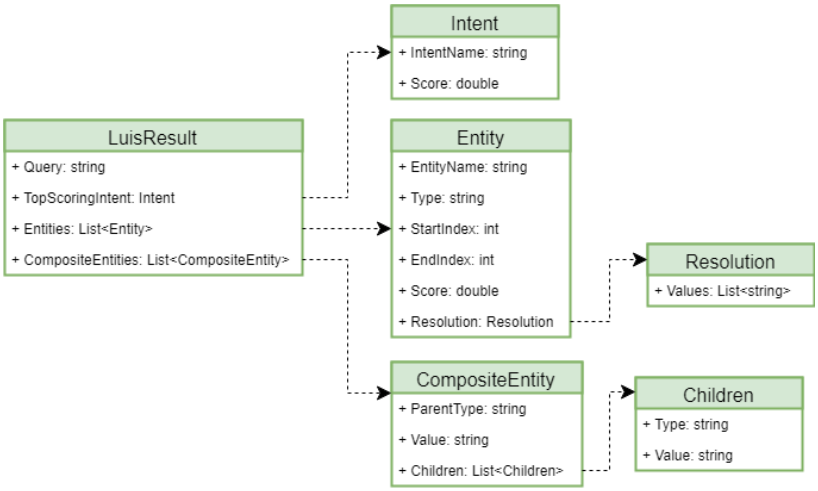


Ilustración 31. Diagrama de la Clase LuisResult (Respuesta)

La intención consta de un nombre y la puntuación recibida en base al modelo entrenado en LUIS. Esta puntuación es importante, pues se utilizará como discriminante a la hora de decidir si la intención ha sido identificada correctamente o por el contrario ha sido un falso positivo.

La clase entidad se compone de un nombre, un tipo en referencia al tipo de entidad (ver *Tabla 2*) y los índices que delimitan la entidad dentro de la cadena de texto. Adicionalmente, si la entidad es de un tipo al que a la que se le ha aplicado aprendizaje automático, se devolverá una puntuación (entidades de tipo simple). En caso contrario se devolverá un valor de tipo resolución, que contiene el nombre normalizado de la entidad. La siguiente figura muestra un conjunto de entidades de tipo lista. Suponiendo que la entidad reconocida es '*backpack*', el valor de la resolución será '*Bag*'.

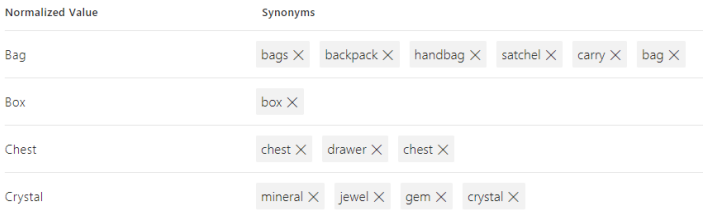


Ilustración 32. Entidad Interactable de Tipo Lista

Las entidades compuestas devuelven el tipo de la entidad padre (nombre que recibe la entidad compuesta), el valor al que hace referencia a la parte del enunciado que compone a dicha entidad, y una lista de hijos con todas las entidades participantes, su tipo y su valor. En la *Ilustración 33* se muestran dos ejemplos de entidades compuestas de tipo *VolumeSetting*.

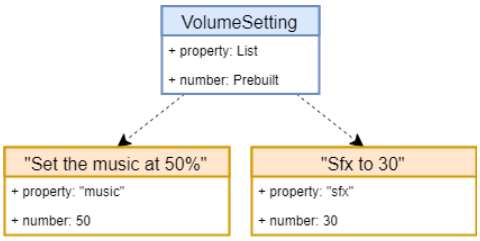


Ilustración 33. Entidad Compuesta 'VolumeSetting'

5.4 Modelo y Aplicación para LUIS

Antes de que LUIS puede interpretar los comandos del jugador, es necesario diseñar una aplicación que siga el modelo del juego expuesto en el apartado 3.1 Conceptos Básicos. La *Tabla 5* y la *Tabla 6* muestran una lista de intenciones y entidades relevantes para el jugador.

Tabla 5. Intenciones y Entidades

Acciones		
Que quiere el jugador	**Intención**	**Entidades**
Cerrar puertas; cofres…	ActionClose	Interactable
Abrir puertas…	ActionOpen	Interactable
Sí; no; tal vez…	ActionResponse	Response
Coger al mapa, el oro…	ActionTake	Interactable
Hablar con el posadero…	ActionTalk	Interactable
Menús		
Volver al menú anterior…	MenuBack	Command
Pausar; detener el juego…	MenuPause	Command
Jugar; empezar…	MenuPlay	Command
Salir…	MenuQuit	Command
Continuar…	MenuResume	Command
Guardar configuración	MenuSave	Command
Ir a menú de configuración	MenuSettings	FontSetting - VolumeSetting
Información		
¿Dónde estoy?	NavigationInfo	-
Ve al comedor; baja las escaleras…	NavigationMove	Location
¿Dónde puedo ir?	NavigationWaypointInfo	-
¿Qué tengo que hacer?	QuestInfo	-

Tabla 6. Desglose de Entidades

Entidad	Tipo	Hijos	Ejemplos
Interactable	Lista	-	Bag, Map, Innkeeper…
Response	Lista	-	Yes, No, Ok…
Command	Lista	-	Back, Pause, Play…
Location	Lista	-	Room, Stairs…
Number	Predefinida	-	1, 2, Ten…
Parameter	Lista	-	Red, Green, Blue, Size
Property	Lista	-	Music, Dialogue…
FontSetting	Compuesto	Number; Parameter; Property	Dialogue font size to 22
Keyword	Simple	-	Pay
VolumeSetting	Compuesto	Number; Property	Music at 50%

Este conjunto de intenciones cubre todas las acciones que puede realizar el jugador mediante comandos de voz, adaptándose al tipo de entrada. La necesidad de usar entidades de tipo lista viene dada por la normalización y la incertidumbre al interpretar la orden del jugador. Un jugador puede decir "*coge el oro del cofre*" mientras otro pude decir "*agarra las monedas*". Son enunciados distintos pero su intención es la misma. Si no se normalizan los nombres de las entidades, la lógica a la hora de tomar decisiones se vuelve muy complicada pues por lo general hay que hacer referencia a objetos del juego con nombres concretos. Al utilizar entidades de tipo lista, es posible añadir sinónimos a una misma palabra y devolver siempre su valor normalizado.

Para la parte de configuración, las entidades compuestas presentan más ventajas pues se requieren varias piezas de información para dar sentido al comando. Por ejemplo, para cambiar el volumen, se necesita especificar si se trata de la música o los efectos especiales y a qué nivel (porcentaje) se quiere fijar el volumen.

5.5 Toma de Decisiones

Una vez que se obtiene la intención del jugador, es el momento de tomar una decisión sobre qué acción se debe realizar. El primer factor a tener en cuenta a la hora de tomar una decisión es el índice de confianza de la predicción. Ya bien sea debido a una orden dada por el jugador de forma accidental o una interpretación incorrecta de la misma, dichas acciones han de ser filtradas y descartadas. Hay que encontrar un índice de confianza que permita realizar un filtrado eficiente sin generar falsos negativos. En este caso, es preferible aceptar un falso positivo que descartar un comando valido. Un índice de confianza de entre el 70% y el 80% es un buen punto de partida.

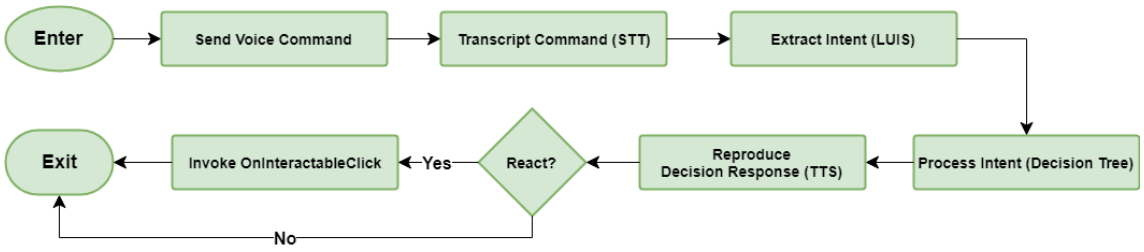


Ilustración 34. D.F. de Toma de Decisiones

En la *Ilustración 34* se muestra es procesamiento de la orden del jugador, desde que se envía el comando de voz hasta que se procesa su intención y se devuelve una respuesta al jugador.

5.5.1 Desarrollo del Árbol de Decisiones

Una vez que la intención recibida se valora como un caso positivo, es necesario interpretarla y actuar si fuera necesario. Para resolver este problema, se cuenta con la implementación de un árbol de decisión. Estos árboles son un modelo de predicción que toma un conjunto de datos y construye un diagrama de construcciones lógicas. [27]

En los arboles de decisión, las decisiones que se eligen son lineales. A medida que se van eligiendo unas opciones se van descartando otras, lo que implica que no hay marcha atrás. En la *Ilustración 35* se puede ver un ejemplo parcial del árbol de decisión perteneciente al dominio del juego.

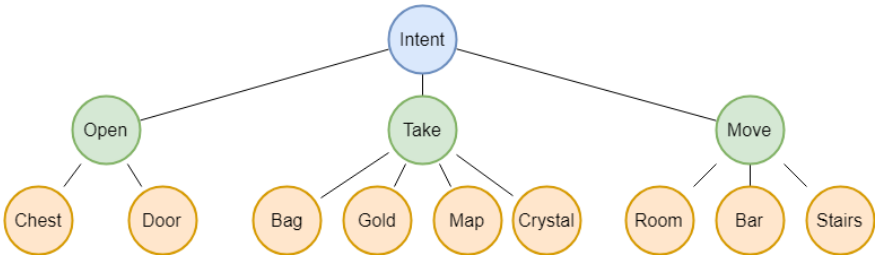
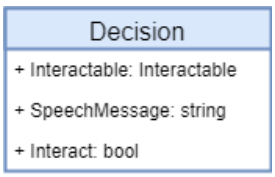


Ilustración 35. Ejemplo de Árbol de Decisión

A la hora de tomar una decisión, se elige en primer lugar la acción correspondiente a la intención y después la entidad sobre la que se va a aplicar dicha acción, si procede. No todas las acciones tienen entidades hijo, como por ejemplo la entidad *MenuQuit*, encargada de salir del juego.



Una vez que se ha identificado la acción y la entidad, la lógica del juego evalúa que tipo de decisión se debe tomar. Para ello, se genera un objeto de tipo *Decision* que contiene información sobre la interacción. Este objeto siempre contiene un mensaje de respuesta para el jugador, que comunica el resultado de la decisión. Esto puede ser una decisión positiva o un mensaje de error en caso de no poder realizar dicha acción. También se indica si se debe o no realizar una interacción y sobre que interaccionable se debe de ejecutar dicha interacción en caso positivo.

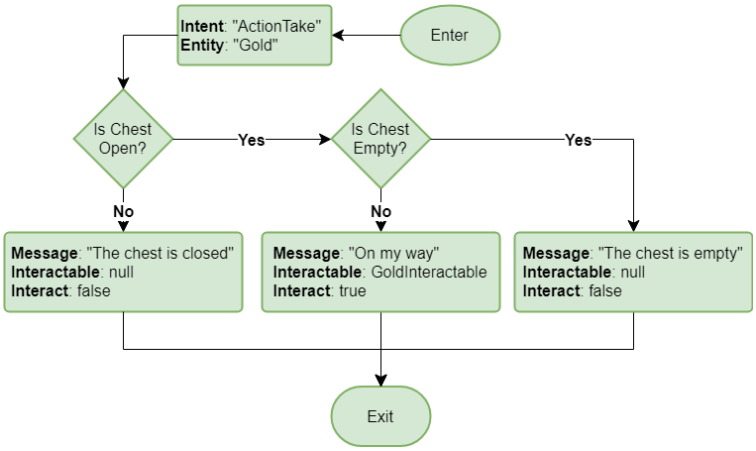


Ilustración 36. Decisión sobre la Acción "TakeGold"

En la *Ilustración 36*, la intención ha sido identificada como '*ActionTake'* y la entidad como '*Gold'*. Antes de tomar una decisión, hay que realizar una serie de preguntas asociadas con la lógica del juego. En el caso de que el cofre esté cerrado o vacío, no se producirá interactuación. Si se cumplen todas las condiciones entonces se asigna como interaccionable al objeto que este asociado a la entidad. Independientemente del resultado, se envía un mensaje que será mostrado por pantalla y hablado. Este mensaje es la realimentación al jugador de su comando.

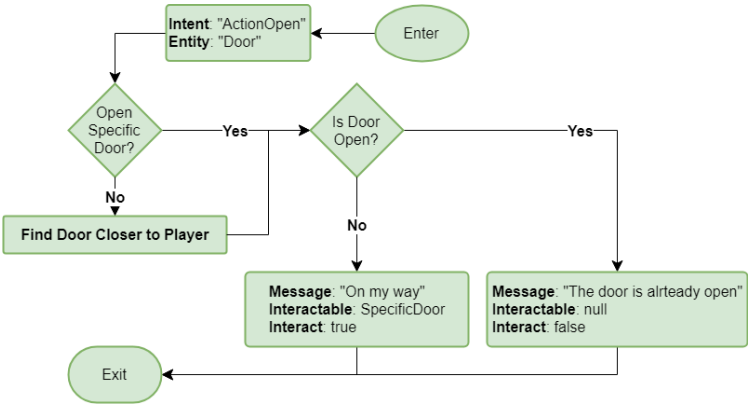


Ilustración 37. Decisión sobre la Acción "OpenDoor"

En la *Ilustración 37*, se hace referencia a la entidad '*Door'*. En este caso, al existir más de una puerta puede darse el caso de que el usuario no haya especificado que puerta quiere abrir. La solución planteada es la búsqueda espacial de la puerta más cercana al usuario y la asignación de dicha puerta como la puerta como la que se quiere abrir. Este sistema de búsqueda por proximidad también es empleado a la hora de encontrar nodos de información y posicionamiento. Otra opción es presentarle al usuario con una lista con las puertas más cercanas a su posición para que pueda elegir.

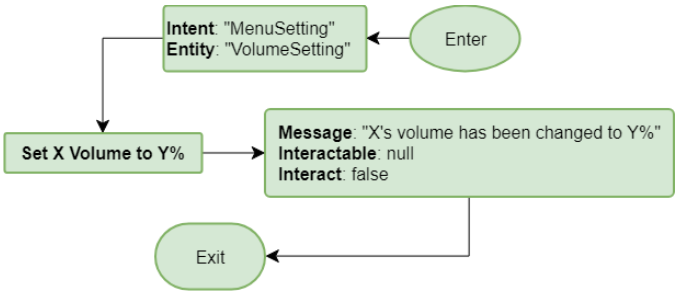


Ilustración 38. Decisión sobre la Acción "SetVolume"

En el último caso de la *Ilustración 38*, se pretende cambiar el volumen del juego a un porcentaje especifico. En este caso, se aplica la configuración y se devuelve un mensaje indicando que la operación ha sido correcta. Nunca se interacciona con un objeto del escenario.

5.5.2 CONTEXTUALIZACIÓN

El juego se compone de 3 escenas, tal y como se mostraba en la *Ilustración 18* (El menú principal es en sí una escena). El sistema de toma de decisiones ha de tener presente es que escena se encentra y contextualizar de la acción. Si el jugador quiere abrir una puerta, tiene sentido que suceda en el escenario y no en el menú de configuración o en la pantalla de créditos. De igual forma, hay otras acciones que son globales, como el poder ajustar el volumen del juego.

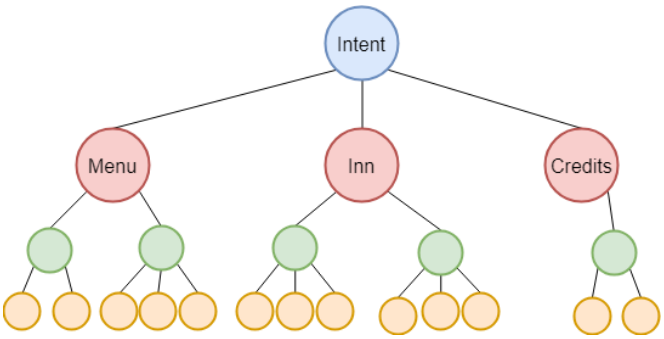


Ilustración 39. Árbol Contextualizado

En el árbol de la *Ilustración 39*, se introduce un nuevo nivel de nodos encargados de filtrar la intención según la escena en la que se encuentre le jugador en ese momento. Esto permite extender la funcionalidad del árbol y adaptarlo al tipo de escena activa en ese momento.

Capítulo 6: Conclusiones y Proyectos Futuros

Este sistema sin duda pone de manifiesto la capacidad de la computación cognitiva para dar respuestas a problemas donde la computación clásica encuentra dificultades. No solo el usuario es capaz de comunicarse de una forma bastante fluida con el juego, interactuando con él sin necesidad si quiera de ver una sola imagen, sino que presenta múltiples ventajas para el resto de los jugadores, como por ejemplo el ajuste de volumen sin necesidad de acceder a un menú. Usuarios con diferentes tipos de minusvalías pueden encontrar beneficios en el sistema.

6.1 Objetivos Alcanzados

El jugador ha sido capaz de interactuar con el juego sin necesidad de recibir información visual o usar el ratón como dispositivo de entrada. La aplicación ha sido capaz de identificar la intención del jugador y traducirla en una acción dentro del juego. La latencia de las conexiones ha sido relativamente pequeña y el tiempo que ha tenido que esperar el jugador antes de recibir respuesta del sistema ha sido breve, de menos de 3 segundos.

El jugador ha recibido información en todo momento sobre funcionamiento y estado del juego. Esto se ha logrado mediante un sistema automático de información, que es activado tras un periodo de inactividad prolongado y transmite al jugador instrucciones básicas para el uso de comandos de voz. El jugador es capaz de conocer el estado del juego en cualquier momento gracias a las respuestas contextualizadas a preguntas como ¿Dónde estoy? O ¿Qué puedo hacer?

6.2 Dificultades

No obstante, el sistema presenta una serie de carencias y dependencias que limitan su usabilidad. En primer lugar, el sistema depende de una conexión a internet para su funcionamiento. Una pérdida de conexión o un problema con el servidor, y sistema quedaría inutilizado.

Si bien el sistema se amolda bastante bien al tipo de juego empleado, otros tipos de juegos serían prácticamente imposibles de adaptar a este tipo de tecnología, como por ejemplo los juegos de acción.

La integración con Unity3D es en ciertos casos precaria, con diferentes problemas de compatibilidad entre sistemas y numerosos problemas con las funciones asíncronas, descritas en el apartado *4.2 Compatibilidad* del Capítulo 4. Sería deseable poder hacer uso de los sistemas de entrada/salida de audio de Unity, en lugar de depender de librerías externas.

Por último, el servicio de reconocimiento de audio es propenso a introducir errores en el sistema. Una transcripción deficiente de los comandos del jugador provoca, en muchos casos, una sensación de confusión e incertidumbre en este. Estos errores de transcripción suelen acabar en errores de

interpretación que son comunicados al jugador. En el peor de los casos se pueden producir falsos positivos en la interpretación, lo que puede dar lugar a errores más graves, y a agravar aún más el estado de confusión del jugador.

Sin embargo, cabe destacar que durante la redacción de esta memoria (junio de 2018) la plataforma de servicios de voz empezó a ofrecer modelos de reconocimiento de voz personalizables. Esto sin duda mejorará la calidad de las transcripciones de audio al permitir entrenar modelos personalizados.

6.3 Proyectos Futuros

Un sistema futuro debería contar con un procesamiento cognitivo local para reducir el tiempo de respuesta y la fiabilidad de este. La *Ilustración 40* muestra la propuesta del nuevo modelo. Se plantea un modelo de procesamiento cognitivo mixto. Las redes neuronales son entrenadas en la nube, ya que se cuenta con una alta capacidad computacional, y los modelos resultantes los aplicados y procesados a nivel local. Este tipo de procesamiento ya es utilizado en smartphones mediante el uso de NPUs [28] (Neural Processing Units) o aceleradores de IA, dedicados a procesar este tipo de modelos.

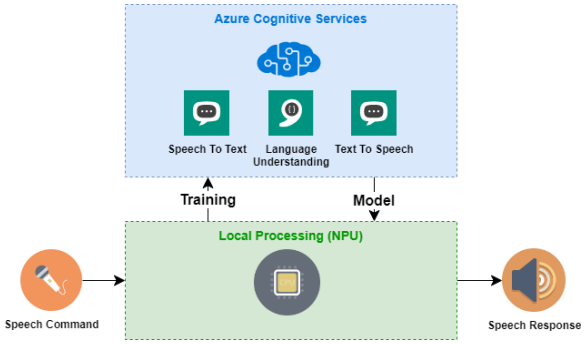


Ilustración 40. Procesamiento Cognitivo Local

Otro tipo de proyecto más ambicioso es de la creación de un ayudante basado en IA. Este ayudante no es un mero nexo entre el jugador y el juego, sino un compañero que guía y facilita al jugador la interacción con el juego. El jugar podrá ajustar el grado de asistencia en función de las dificultades de accesibilidad que presente.

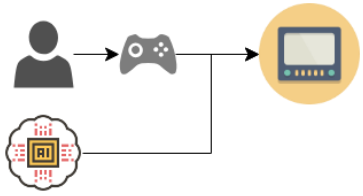


Ilustración 41. AI Companion

En la *Ilustración 41* se muestra un ejemplo de este tipo de asistente. En este supuesto, el jugador necesita realizar una acción mediante el uso de un mando, pero encuentra dificultades debido a una minusvalía. El asistente sería capaz de analizar la situación y asistir al jugador, complementando las acciones de este. Este asistente no se limita solo a operaciones de entrada, sino que también puede aportar información acerca del propio juego y hacer sugerencias si así fuera necesario.

BIBLIOGRAFÍA

[1] Real Patronato Sobre Discapacidad, «siis.net,» 15 Noviembre 2005. [En línea]. Available: https://www.siis.net/docs/ficheros/200701120002_24_0.pdf.

[2] Microsoft, «Azure Cognitive Services,» [En línea]. Available: https://azure.microsoft.com/es-es/services/cognitive-services/.

[3] Unity, «Unity User Manual (2017.3),» [En línea]. Available: https://docs.unity3d.com/Manual/index.html.

[4] Synthexis, «Cognitive Computing,» 2018. [En línea]. Available: https://synthexis.com/cognitive-computing/.

[5] Wikipedia, «Wikipedia,» 2018. [En línea]. Available: https://es.wikipedia.org/wiki/Accesibilidad.

[6] AbleGamers, «AbleGamers,» [En línea]. Available: https://www.includification.com/. [Último acceso: 2018].

[7] ONU, «Convention on the Rights of Persons with Disabilities (CRPD),» [En línea]. Available: https://www.un.org/development/desa/disabilities/convention-on-the-rights-of-persons-with-disabilities.html. [Último acceso: 2018].

[8] Comisión Europea, «European Accessibility Act,» [En línea]. Available: http://ec.europa.eu/social/main.jsp?catId=1202. [Último acceso: 2018].

[9] Game Accessibility Guidelines, «Game Accessibility Guidelines,» [En línea]. Available: http://gameaccessibilityguidelines.com/. [Último acceso: 2018].

[10] N. Willson, «A Beginners Guide to Cognitive Services,» 9 Mayo 2016. [En línea]. Available: https://www.linkedin.com/pulse/idiots-guide-cognitive-services-nigel-willson/. [Último acceso: 2018].

[11] IBM, «Watson,» [En línea]. [Último acceso: 2018].

[12] Google, «Google Cloud,» [En línea]. Available: https://cloud.google.com/products/machine-learning/. [Último acceso: 2018].

[13] Microsoft, «Microsoft Speech API overview,» [En línea]. Available: https://docs.microsoft.com/es-es/azure/cognitive-services/speech/home.

[14] Microsoft, «Luis API documentation,» [En línea]. Available: https://docs.microsoft.com/es-es/azure/cognitive-services/luis/home.

[15] Microsoft, «Luis Project,» [En línea]. Available: http://www.luis.ai.

[16] N. Esposito, «A Short and Simple Definition of What a Videogame Is,» 2005. [En línea]. Available: http://www.utc.fr/~nesposit/publications/esposito2005definition.pdf.

[17] Unity3d, «Adventure Game Tutorial,» 2018. [En línea]. Available: https://unity3d.com/es/learn/tutorials/projects/adventure-game-tutorial.

[18] Unity3d, «NavMesh,» [En línea]. Available: https://docs.unity3d.com/Manual/nav-BuildingNavMesh.html. [Último acceso: 2018].

[19] Unity3d, «NavMesh Agent,» [En línea]. Available: https://docs.unity3d.com/Manual/class-NavMeshAgent.html. [Último acceso: 2018].

[20] Unity3d, «Unity3d 2017.3.1.f.1,» [En línea]. Available: https://unity3d.com/es/unity/qa/patch-releases/2017.3.1f1.

[21] Unity3d, «Unity 2017,» [En línea]. Available: https://blogs.unity3d.com/es/2017/07/11/introducing-unity-2017/. [Último acceso: 2018].

[22] M. Heath, «NAudio Git,» [En línea]. Available: https://github.com/naudio/NAudio. [Último acceso: 2018].

[23] Mono Project, «Mono Project,» [En línea]. Available: https://www.mono-project.com/.

[24] S. Vermeulen, «Async-Await instead of coroutines in Unity 2017,» [En línea]. Available: http://www.stevevermeulen.com/index.php/2017/09/using-async-await-in-unity3d-2017/. [Último acceso: 2018].

[25] StackOverflow, «Coroutines vs Threading,» [En línea]. Available: https://answers.unity.com/questions/357033/unity3d-and-c-coroutines-vs-threading.html.

[26] Microsoft Azure, «Azure Git Repository,» [En línea]. Available: https://github.com/Azure. [Último acceso: 2018].

[27] Wikipedia, «Árbol de Decisión,» [En línea]. Available: https://es.wikipedia.org/wiki/%C3%81rbol_de_decisi%C3%B3n. [Último acceso: 2018].

[28] University of California, «Neural Processing Unit,» [En línea]. Available: https://patents.google.com/patent/US8655815B2/en. [Último acceso: 2018].

[29] J. Skeet, C# In Depth rev. 4, Manning, 2018.

[30] RandomGuyDev, «http://randomguydev.com/survival-shooter-in-unity/,» [En línea]. [Último acceso: 2018].

ANEXO A: ÍNDICE DE ILUSTRACIONES

ANEXO B: ÍNDICE DE TABLAS

UNIVERSIDAD DE ALMERÍA

Resumen/Abstract

El presente Trabajo Fin de Máster plantea como objetivo principal de investigación mostrar la posibilidad de integrar servicios cognitivos en el ámbito de los videojuegos para mejorar la accesibilidad de estos. En concreto se estudiarán los servicios que permiten la comunicación humano-máquina mediante lenguaje natural.

El proyecto hará uso de los servicios cognitivos de voz y lenguaje para realizar la transcripción e interpretación de los comandos de voz del jugador.

Para demostrar el potencial de estos servicios y su integración, se adaptará un pequeño juego en Unity3D, donde el jugador deberá realizar una serie de tareas sin la necesidad de recibir información visual. El jugador se comunicará con el juego mediante su voz y recibirá información a través de su oído.

This Master's Thesis proposes the research of cognitive services and the possibility of integrating said services in the field of video games to improve their accessibility. The study focuses on the services that allow human-machine communication through natural language.

The project will use cognitive voice and language services to perform the transcription and interpretation of the player's voice commands.

To demonstrate the potential of these services and their integration, a small game will be adapted in Unity3D, where the player must perform a series of tasks without the need of visual feedback. The player will communicate with the game using voice commands and receive spoken information.