Nafarroako Unibertsitate Publikoa
Universidad Pública de Navarra

Universidad Pública de Navarra
Trabajo de fin de grado
Ingeniería Informática

Asistente virtual interactivo y multidioma destinado al turismo cultural

Julio de 2018
Autor: Rubén Miguélez Briñas

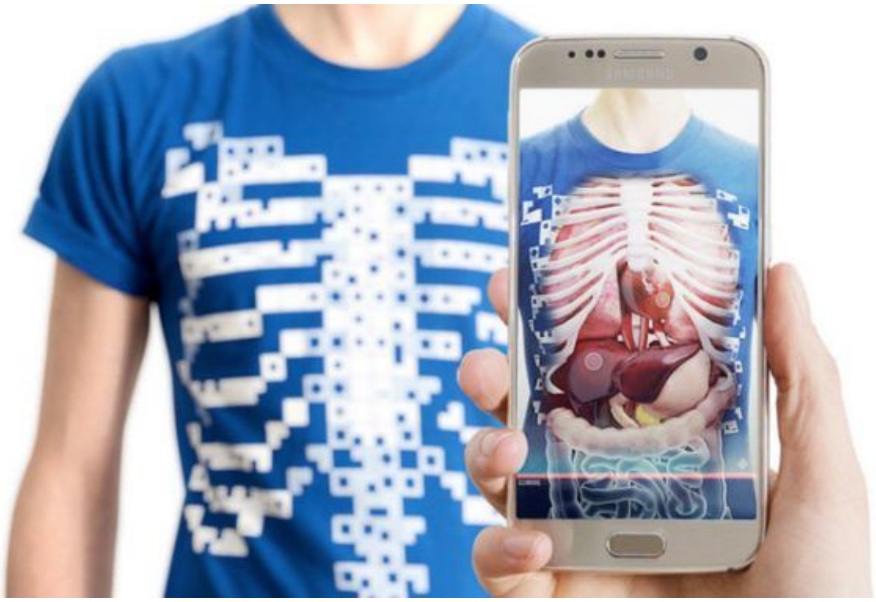
ÍNDICE

Introducción

Hoy en día la tecnología va avanzando muy rápido y con ella la forma de programar y las herramientas que utilizamos. Es muy importante definir bien qué tipo de desarrollo vas a llevar a cabo, hacia qué plataforma va a ir principalmente dirigida tu aplicación, qué material o equipo de trabajo vas a necesitar para la correcta implementación de la misma o incluso si vas a hacer uso de alguna tecnología novedosa y cuál o cuáles.

Existen tantas opciones que barajar, tanto que estudiar y contemplar... Mi proyecto se basa en dos pilares: La formación y la interacción.
Sobre estos puntos, empezaremos a estudiar y pensar en todo lo anterior.

Existen una gran variedad de aplicaciones didácticas, cuyo fin es el de formar en algún aspecto a las personas, sobre todo en el ámbito de los idiomas. La mayor parte de éstas van dirigidas a web o móvil, pero puedes encontrarlas en cualquier otra plataforma. También cabe destacar que cada vez más aplicaciones de este tipo utilizan nuevas tecnologías como la realidad aumentada. Aplicaciones móvil como Coursera o Udemy también están disponibles en web, para poder usarlas en cualquier plataforma. Son aplicaciones exclusivamente de formación y gratuitas, en este caso algo más tradicionales sin un uso de tecnologías novedosas. Por otro lado tenemos aplicaciones como Virtuali-Tee.



Es una aplicación para el estudio de la anatomía y se basa principalmente en la realidad aumentada como herramienta visual para la formación. De esta manera vemos distintas opciones y sobre todo como va avanzando el mundo de la formación hacia algo más visual, también puede ser auditivo, táctil o incluso quién sabe si algún día por olor o sabor. [1]

Lo que nos sugiere esto último es básicamente la interacción directa. Participar en el proceso de aprendizaje sin ser un actor externo, esto es muy importante y por eso es uno de los pilares básicos de mi proyecto. Hoy en día, la interacción es uno de los recursos más usados en el mundo de la programación, el simple hecho de que las pantallas empiezan a ser en su mayor parte táctiles, el usuario sin ser realmente consciente interactúa en todo momento con los dispositivos. Pero hay que llevarlo más allá, hacer que el usuario sea el protagonista y no la aplicación.

Es muy fácil escuchar anuncios en la televisión sobre asistentes virtuales en casa, los cuales mediante tu voz te ponen música, te ayudan, realizan búsquedas en internet, etc. En el que tu voz predomina, por encima de la aplicación, es el usuario el que posee el control y se siente participante directo.

Otro de los ejemplos claro son las televisiones inteligentes, a las que mediante la voz, le puedes pedir repeticiones, que grabe un programa etc.

Parece que se ve claro que uno de los componentes esenciales de la interacción actualmente es la voz. El reconocimiento y síntesis de voz es una de las claves para lograr una interacción plena.

En este caso particular, al unir ambos componentes obtenemos que la formación debe basarse en la interacción. No es lo mismo que una persona vea un video informativo, a que la misma persona sea capaz de pedir ese video, preguntar, etc. La experiencia formativa es más interesante si te sientes parte de ella.

En este proyecto, necesitamos dar información sobre un tema concreto, pero esta información se consigue mediante imágenes, presentaciones power point e información general que se puede conseguir mediante preguntas. Y todo esto es necesario que pueda ser obtenido utilizando la voz como instrumento de interacción, también deberá poderse adaptar a las necesidades del usuario, pudiendo cambiarse el idioma para la correcta comprensión de todos.

Además de la interacción vocal, ha de tener también una visual, esto quiere decir que se dispondrá de un interfaz gráfico en el que un avatar ("humano") sea el encargado de hablarte y fijar un contacto visual así como escuchar, de esta manera la inmersión es mayor.

A continuación veremos una tabla comparativa con distintas aplicaciones, como las que hemos ido nombrando que tienen un fin parecido al de mi proyecto. Para ello debemos fijar las características deseables: Interacción por voz, formación, interacción visual con el avatar, multidioma, capaz de adaptarse a las necesidades del usuario, mostrar distintos tipos de información y por último el objetivo.

	Tv inteligente	Aura (Movistar)	Coursera	Mi proyecto
Interacción por voz	X	X		X
Formación			X	X
Interacción visual				X
Multidioma	X	X	X	X
Adaptarse a las necesidades	X	X		X
Distintos tipos de información	X			X
Objetivo	Ofrecer la funcionalidad de una tv por voz	Asistir virtualmente por voz. (Música, búsquedas en Google…)	Ofrecer cursos de formación	Ofrecer información y formar mediante la voz sobre el patrimonio

Como se puede observar fácilmente, estas características son usadas notablemente en este tipo de aplicaciones. Y, si bien es cierto que hay bastantes programas similares a mi proyecto en cuanto a características y tecnologías usadas… El hecho de haber puesto el foco en un tema tan importante como es el patrimonio histórico y cultural de una zona, hace que sea una idea novedosa, y única. Ahora mismo en Navarra no existe algo parecido en relación a este tema, y sería muy fácilmente equiparable la aplicación con el trabajo que desarrolla un guía turístico.

Con todo esto, tenemos una ligera idea de cual es mi proyecto, pero vamos a concretarlo más.

La idea es programar un asistente virtual, el cual mediante un sistema de reconocimiento de voz, va a interpretar y entender nuestras palabras, para efectuar una u otra tarea, en base a lo que le hayamos pedido. Nuestro asistente va a poder mostrarnos fotos, enseñarnos y explicarnos presentaciones power point y responder preguntas dentro de su conocimiento. Todo ello, como ya he dicho, utilizando únicamente la voz del usuario. El sistema utilizará la síntesis de voz, para que el avatar sea capaz de respondernos con voz también.

Saliendonos un poco del tema de interacción por voz, se creará un interfaz gráfico donde aparecerá el avatar (con forma de humano) y al cual vamos a poder ver proyectado a tiempo real, a la vez que ejecuta las órdenes bajo demanda. A todo esto hay que añadir la posibilidad de leer los subtítulos en distintos idiomas, para la correcta comprensión del grupo que esté en cada momento de visita, este punto es muy importante ya que la gente que visita y se empapa de la cultura de un lugar pueden ser perfectamente extranjeros.

Es muy importante que el sistema funcione a tiempo real, los comandos por voz y las respuestas deben venir seguidos, así como la respuesta por voz del avatar.

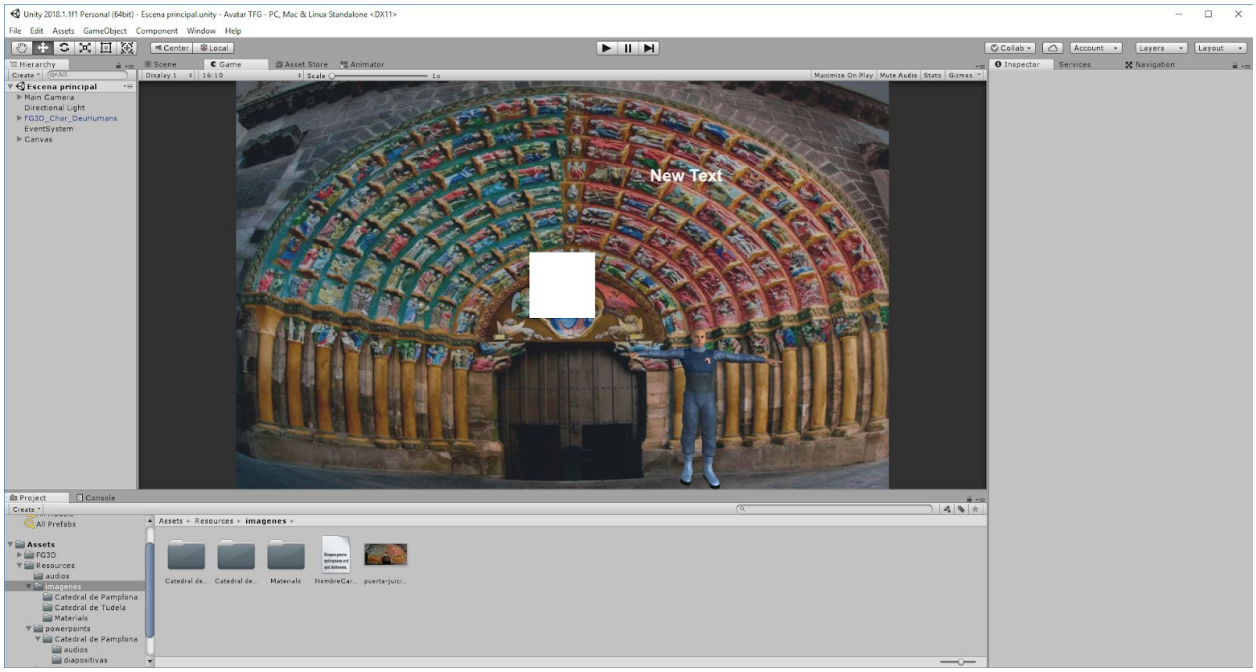
Al tratarse de un tema tan concreto como lo es el patrimonio de Navarra, el avatar puede llegar a ser capaz de tener un conocimiento mucho más elevado que los guías turísticos o incluso expertos, por lo que podría ser un recurso incluso para gente más avanzada en la materia y no solamente turistas que se encuentren visitando iglesias.

Herramientas utilizadas

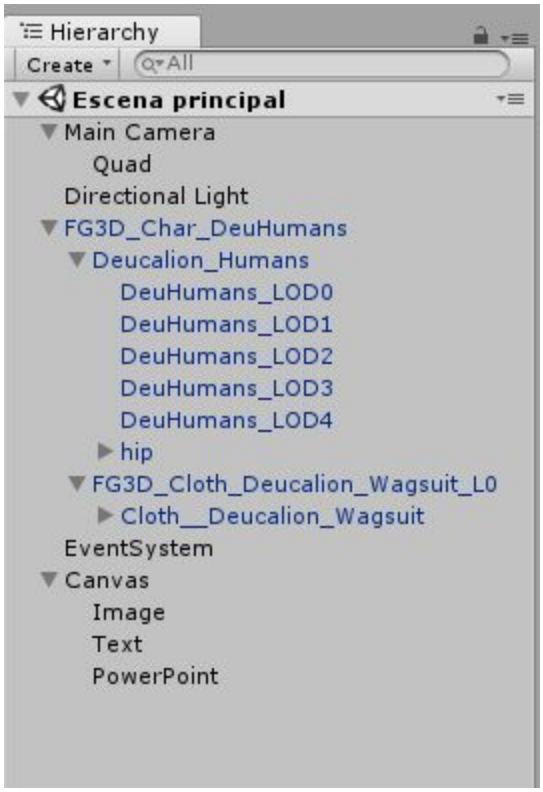
Unity

Unity es un motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, OS X, Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas. Es gratuito en su licencia básica, no obstante requiere una suscripción mensual si tu empresa sobrepasa unos límites de facturación. Lo que brinda Unity es un entorno de desarrollo muy fácil de usar y en el que resulta muy fácil crear un interfaz gráfico vistoso. Unity puede usarse junto con programas de diseño como: Blender, 3ds Max, Maya, Softimage, Modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks y Allegorithmic Substance.[2] Los cambios realizados a los objetos creados con estos productos se actualizan automáticamente en todas las instancias de ese objeto durante todo el proyecto sin necesidad de volver a importar manualmente. No obstante, mi proyecto no se basa en el diseño, sino en la programación. Dar funcionalidad y hacer que todo funcione como debería.

Esto es Unity:



Como cualquier entorno de desarrollo tiene distintas ventanas y una principal, en la cual se trabaja montando la escena y haciendo pruebas. Casi todo es tema de diseño en esa pantalla principal. Ahora explicaré brevemente las ventanas más importantes y cuales utilizo yo en el desarrollo del proyecto principalmente.



Una de las más importantes es la jerarquía de objetos, aquí ves todos los elementos que intervienen en tu escena. Todos los objetos que no estén aquí no se verán en la escena que estamos montando. En mi caso tenemos la cámara, que estará estática siempre apuntando hacia el avatar. Luz, da un toque más real al humano, aporta esas sombras y reflejos que imitan a la realidad. También disponemos del avatar (FG3D_Char_DeuHumans)*, el cual tiene distintos componentes que lo hacen ser humano. También dispongo de un Canvas, en el cual hay distintos elementos que serán activados cuando se lo digamos al avatar, como son las imágenes, power points y subtítulos (Text).

*El modelo 3D lo he descargado de la Asset Store de forma gratuita, es otra de las herramientas que incorpora Unity, muy necesarias para la gente que no sabemos de diseño 3D y necesitamos modelos. Hay tanto gratuitos como de pago.

El sistema de ficheros tiene un papel muy importante en Unity. Tener las carpetas bien organizadas, todo correctamente separado y fácilmente manipulable es de gran ayuda cuando te encuentras en un proyecto como este.

En mi caso, necesito un orden exacto y un sistema de carpetas muy bien organizado, ya que el avatar realiza muchas búsquedas entre los distintos recursos, para escoger imágenes, leer ficheros, etc.

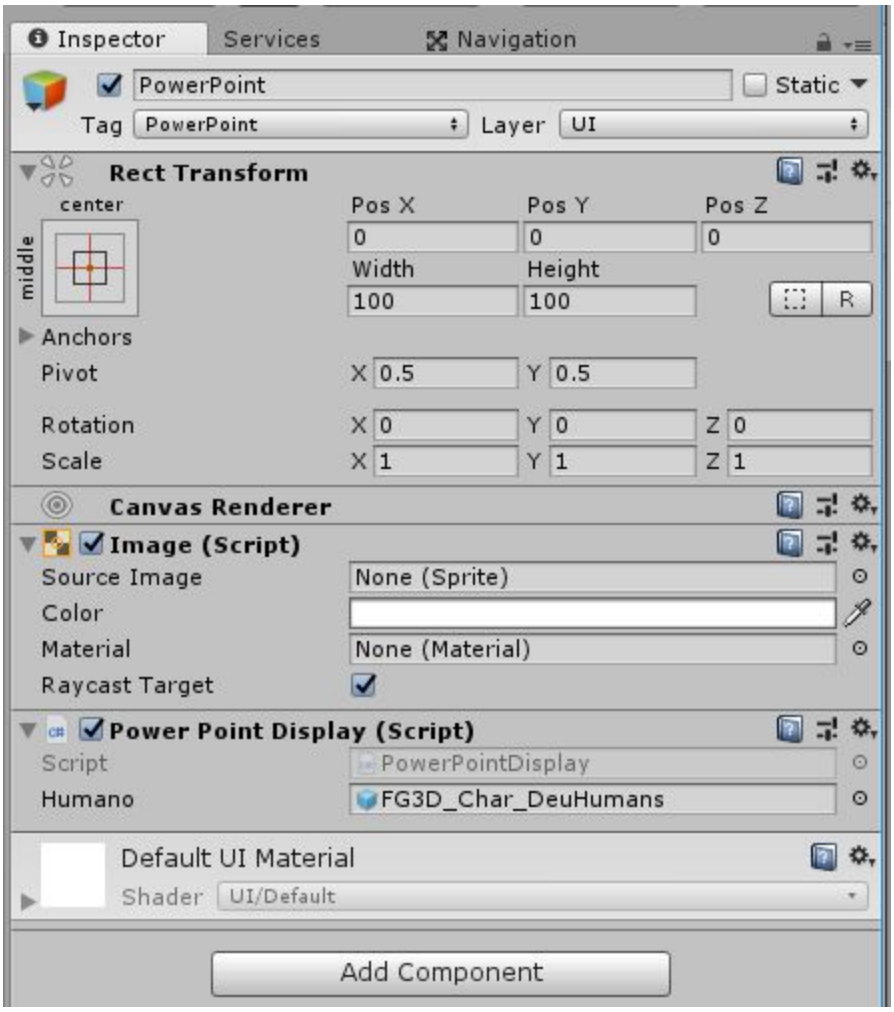
Como se puede ver en la imagen, cada cosa debe tener su lugar, tenemos por un lado los scripts necesarios para todo el funcionamiento, las escenas separadas, y todos los recursos organizados en la carpeta Resources, aquí podemos encontrar todo lo necesario para que el avatar pueda buscar y proyectar cada una de las cosas necesarias, así como hablar y responder preguntas.

Mi sistema de ficheros necesita un orden muy estricto y un acceso a Resources rápido y sencillo. La verdad es que Unity hace que esto sea muy sencillo.

El acceso a los recursos se hace por medio de la programación dentro de esos Scripts de los que hablaba antes.

En Unity principalmente se programa en C#, un lenguaje basado en objetos parecido a Java, aunque más adelante hablaremos de ello.

Otra de las cosas que debemos saber de Unity es que los objetos tienen componentes. Cada objeto puede tener una gran variedad de componentes.



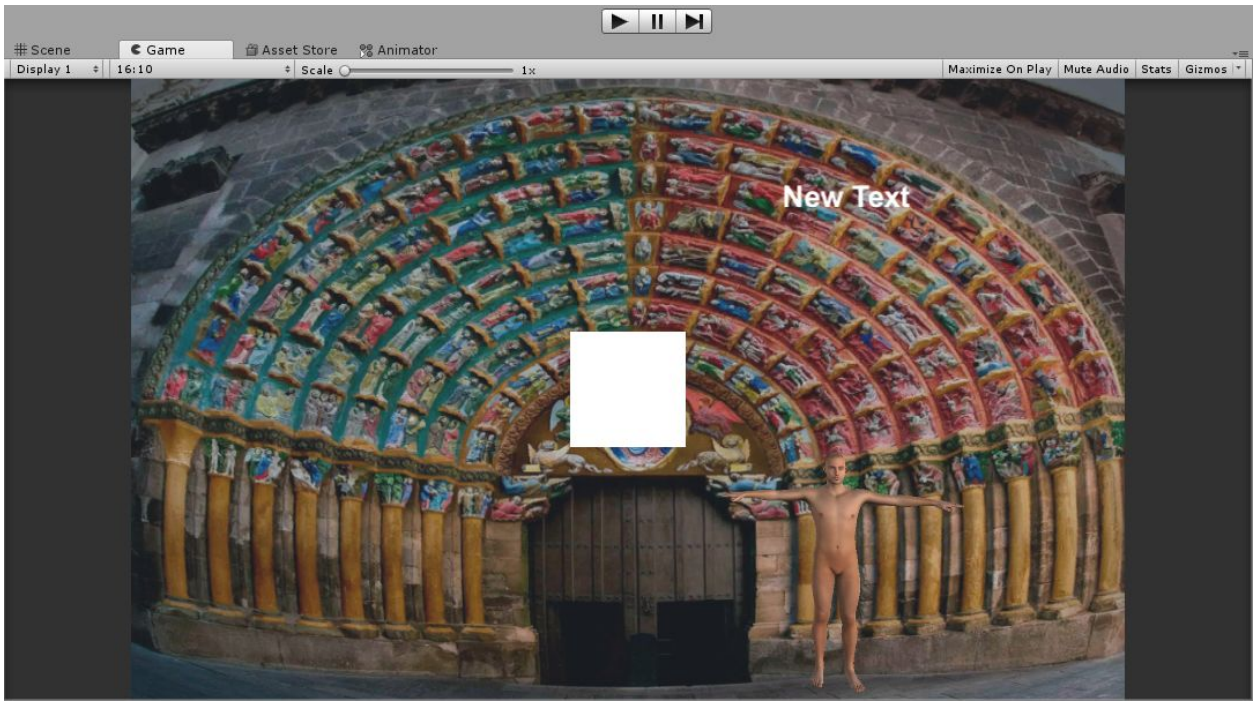
En este caso, he seleccionado el objeto “PowerPoint”, que se encontraba dentro del Canvas y mayormente es un elemento del interfaz del tipo Imagen. Este tiene

cuatro componentes (Rect Transform, Canvas Renderer, Image y Power Point Display).

Como se puede ver, el uso de los scripts es muy distinto al uso que hacemos en otros entornos de desarrollo, donde teníamos proyectos con distintos scripts en cada uno, etc. Aquí un script da funcionalidad a un objeto, y esto se consigue haciendo que ese script sea una componente del mismo.

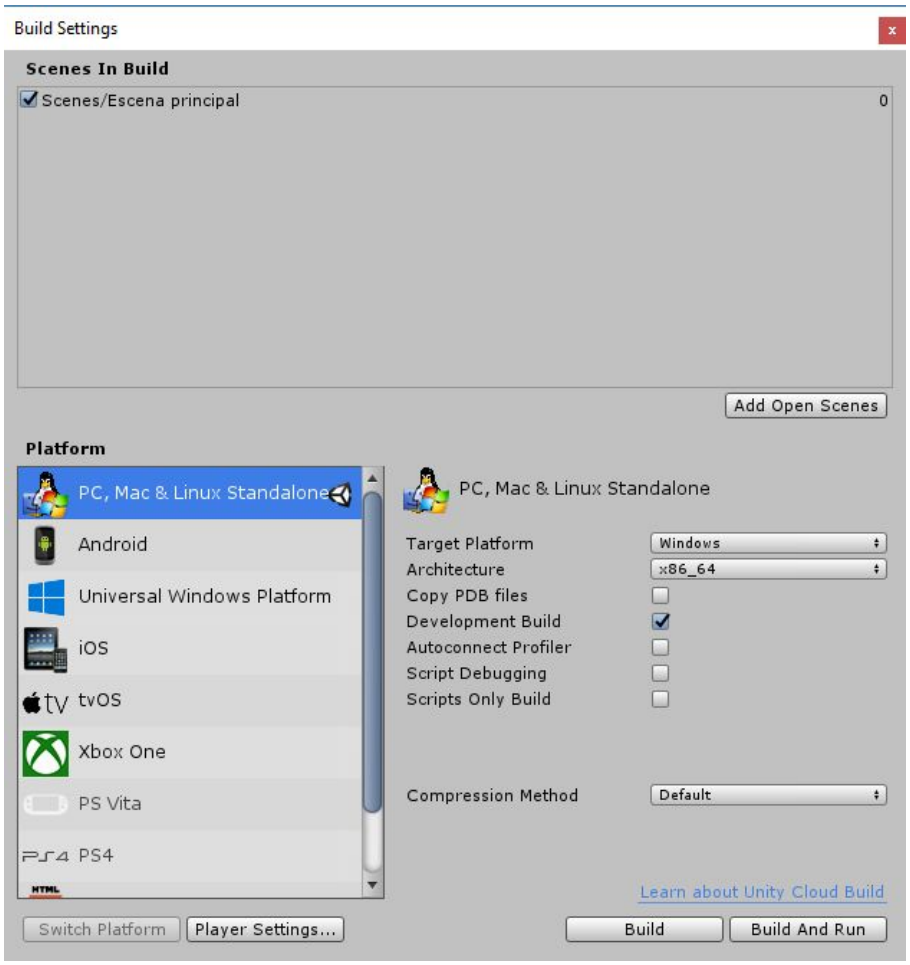
No obstante, sigue siendo muy fácil acceder a otros scripts que aportan más funcionalidad. En C# en Unity, se puede acceder a cada componente de los diferentes objetos de manera muy sencilla. Y de esa manera a sus scripts o mismamente su posición en la escena.

Como apreciamos en la foto, tengo un Script llamado “Power Point Display”, este es un componente más del objeto PowerPoint, pero se puede ver que tiene una variable pública Humano, y esta hace referencia a FG3D_Char_DeuHumans, que como hemos visto antes era el avatar. Por lo que puedo acceder a las componentes del avatar fácilmente, y así hacer que se ejecute algún método de un script que tenga el avatar, sin necesitar tenerlo también en el objeto PowerPoint. Hay que intentar evitar la duplicidad de componentes (en este caso, scripts).



Esta ventana es muy útil, es en la cual puedes ver la escena montada a tiempo real, con los cambios que le vayas haciendo. También es donde se hacen pruebas, al hacer click en el botón de Play puedes comenzar la ejecución del programa y ver si funciona lo que has estado desarrollando, o simplemente para ver como queda una animación o algún elemento de diseño en la escena. Los objetos se pueden arrastrar directamente y moverlos, rotarlos o transformarlos de cualquier manera en la venta de Scene.

Como se ve, durante una prueba, puedes parar la ejecución y reanudarla. Unity también dispone de una consola, con la cual puedes debugear viendo los distintos Logs y ver excepciones o errores que surgen durante la ejecución, así como errores de compilación que impidan ejecutar la aplicación. Como había comentado anteriormente Unity nos brinda facilidades para exportar nuestra aplicación a casi cualquier plataforma.



Aquí vemos la pantalla de exportación a ejecutable para Windows, pero como se observa a la izquierda, puedes seleccionar distintas plataformas, incluso para consolas como Xbox One. Lo que hay que tener muy claro es que la programación puede variar un poco entre unas plataformas y otras, sobre todo el uso de periféricos y entradas de cualquier tipo. Así como los ajustes individuales para cada plataforma, hay que tener muy claro a cual va dirigido, y en el caso de que vaya a ser más de una, tenerlo claro durante todo el desarrollo. Para evitar problemas mayores a la hora de generar el programa final.

En mi caso desde un principio Windows era la plataforma elegida, sobre todo para usarlo en un ordenador (bien sea portátil o sobremesa), y así conectarlo a un proyector para que todo el mundo pueda ver bien al avatar.

Google Text-To-Speech

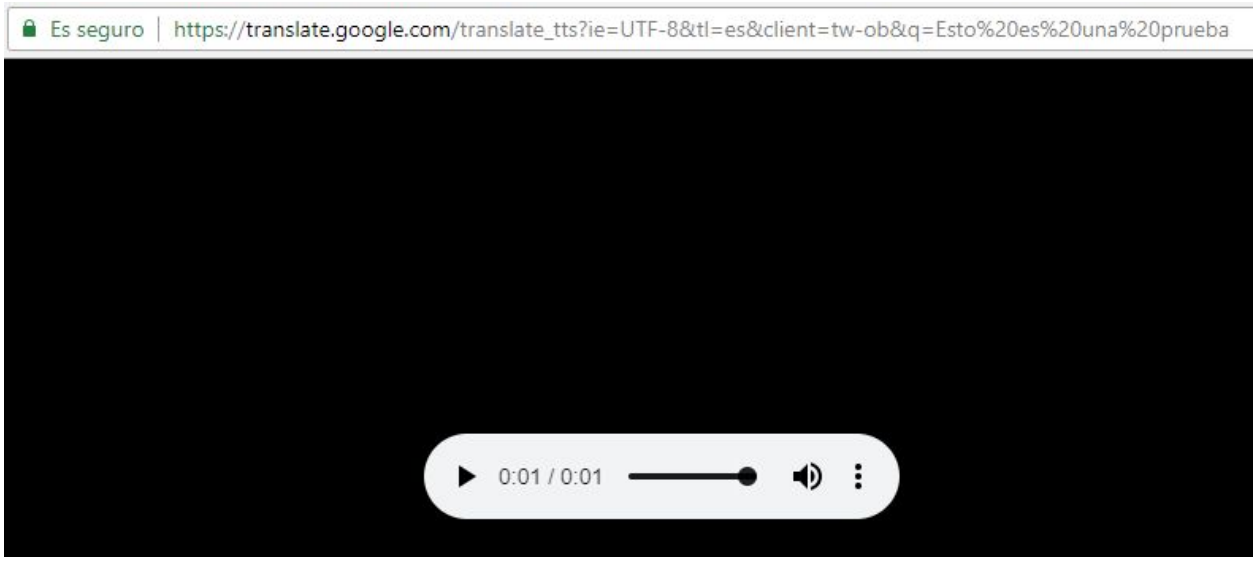
Es una herramienta de síntesis de voz de Google. Resulta muy útil, ya que es capaz de hablar en español y reproducir cualquier frase. La herramienta permite, a partir de una frase o conjunto de palabras, reproducir los sonidos, generando así lenguaje natural fácilmente entendible.[3]

Esta herramienta tiene un alto grado de importancia en mi proyecto, es la encargada de darle vida al avatar y hacer que hable como si fuera un humano. De esa manera la interacción es aún más pura, como si estuvieras hablando con un experto en la materia en vez de un simple robot proyectado.

Se puede decir que es también una herramienta gratuita, aunque como todo, tiene sus limitaciones. A partir de los 4 millones de caracteres reproducidos, Google empieza a cobrar la cantidad de 4 dólares americanos cada millón de caracteres. Realmente es complicado llegar a los primeros 4 millones, aunque con un uso habitual, o una distribución de los programas por distintas Iglesias o centros de turismo podrían hacer que esa cifra se alcance muy rápidamente.

A parte de la parte para desarrolladores, este servicio cuenta con una aplicación web, en la cual puedes realizar pruebas y ver cómo sonaría tu texto.

Es muy útil, sobre todo como digo para la realización de pruebas y me ha ayudado a aprender más sobre el tratamiento de URLs en código.



Como se puede observar, simplemente cambiando los parámetros de la url obtenemos unos u otros resultados. En el ejemplo de la imagen la url es :

https://translate.google.com/translate_tts?ie=UTF-8&tl=es&client=tw-ob&q=Esto%20es%20una%20prueba

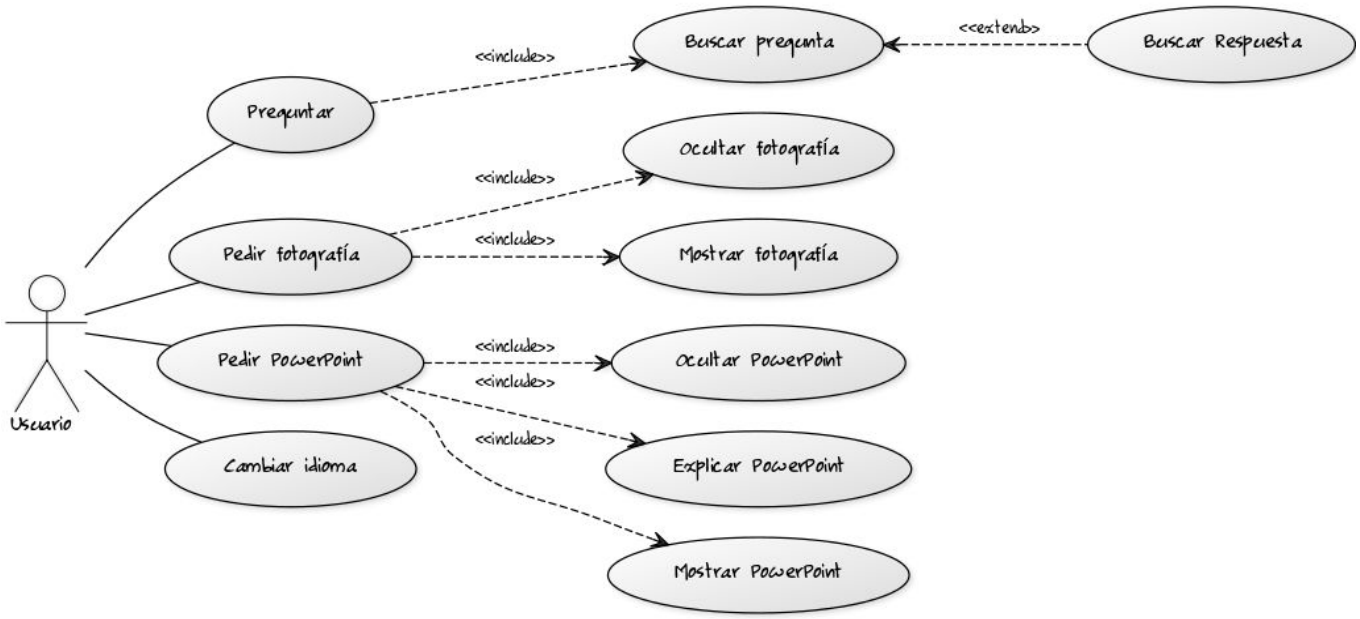
En el campo "tl" se puede cambiar el idioma, actualmente está en español. Y el campo más importante es "q", en el cual tenemos que introducir el texto que queremos que sea transformado a voz. En nuestro caso, nos dirá : "Esto es una prueba".

Se puede observar en la imagen que el interfaz es muy sencillo, simplemente el audio con las opciones de darle Play, modificar el volumen y por último descargar el audio. De esta manera podemos obtener audios de voz de absolutamente todas las frases que queramos. Es otra opción, pero esta ya no funciona con una API para desarrolladores etc.

Realmente está pensado para que uses el servicio de "pago" que te proporcionan y que he mencionado anteriormente, con facilidades para programadores y sin preocuparte de las peticiones por URL o descargar audios.

Requisitos funcionales del sistema

Para tener una idea más clara de las funcionalidades de mi sistema, he optado por un diagrama de casos de uso. Estos casos de uso van a representar la funcionalidad del sistema y el hecho de mostrarlo como un diagrama nos da una idea visual muy rápida de todo.



Como vemos, el usuario puede preguntar, pedir fotografías, pedir PowerPoints y cambiar el idioma. Todo ello lo va a poder realizar utilizando su voz para hacer las peticiones, por lo que el sistema ha de reconocer la voz e interpretarla para saber en cada momento que es lo que quiere el usuario.

Las cuatro interacciones que puede realizar el usuario son activadas por comandos de voz, o simplemente frases completas. Mientras que el resto de casos de uso, aportan funcionalidad interna que hace que todo funcione como debería sin que el usuario sepa qué está pasando.

En el capítulo de las implementaciones relevantes, desarrollaré más a fondo de qué trata cada funcionalidad, y como he ido desarrollandolas.

Implementaciones relevantes

Algoritmo Distancia de Levenshtein

Cabe destacar que todas las implementaciones se han realizado utilizando C# como lenguaje. Es un lenguaje muy utilizado por los desarrolladores de videojuegos y con una programación orientada a objetos.

En primer lugar, uno de los fundamentos básicos de mi proyecto es conseguir entender y analizar lo que el usuario ha dicho.

En el lenguaje natural, decir: “Cuales son las iglesias más bonitas de Navarra” y “Qué iglesias te parecen más bonitas en Navarra”, pueden interpretarse de la misma manera y obtener una misma respuesta. En cambio, para el ordenador, tener esas dos frases no le aporta nada. Son dos frases distintas con palabras distintas.

Por lo tanto, necesitamos un algoritmo que nos ayude a determinar si dos frases son suficientemente similares como para poder tomar la misma decisión.

Mi cabeza giraba un poco en torno a esta idea, no vamos a pensar en lo que ha dicho el usuario al 100%, si no que nos vamos a centrarnos en intentar averiguar de entre las cosas que conoce el avatar, que ha podido decir el usuario.

Siempre, teniendo en cuenta que el avatar puede no saber de lo que dice el usuario. Quiero decir, que no necesariamente a todo lo que dice el usuario se le atribuye su “frase paralela”.

Para ello, después de documentarme sobre el tema, llegué a la conclusión de implementar el algoritmo de la distancia de Levenshtein.

La distancia de Levenshtein, distancia de edición o distancia entre palabras es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra, se usa ampliamente en teoría de la información y ciencias de la computación. Se entiende por operación, bien una inserción, eliminación o la sustitución de un carácter. Esta distancia recibe ese nombre en honor al científico

ruso Vladimir Levenshtein, quien se ocupó de esta distancia en 1965.[4] Es útil en programas que determinan cuán similares son dos cadenas de caracteres, como es mi caso.

Se trata de un algoritmo de tipo bottom-up, común en programación dinámica. Se apoya en el uso de una matriz (n + 1) × (m + 1), donde n y m son las longitudes de las cadenas.

Para mi caso particular, tendríamos s como la frase formulada por el usuario, y t sería cada una de las preguntas por ejemplo que puede responder el avatar. Por lo tanto podríamos saber a cuál de esas preguntas se está haciendo referencia.

```
        int n = s.Length;
        int m = t.Length;
        int[,] d = new int[n + 1, m + 1];


        for (int i = 0; i <= n; d[i, 0] = i++) ;
        for (int j = 1; j <= m; d[0, j] = j++) ;

        for (int i = 1; i <= n; i++)
        {
            for (int j = 1; j <= m; j++)
            {
                int cost = (t[j - 1] == s[i - 1]) ? 0 : 1;
                int min1 = d[i - 1, j] + 1;
                int min2 = d[i, j - 1] + 1;
                int min3 = d[i - 1, j - 1] + cost;
                d[i, j] = Math.Min(Math.Min(min1, min2), min3);
            }
        }
        return d[n, m];
    }
}
```

Con este algoritmo, como he comentado anteriormente, obtendríamos una matriz “d”, que se va poblando con el mínimo número de cambios necesarios para que las dos frases sean iguales. De esa manera d[n,m] guarda el mínimo número de cambios necesarios para que sean iguales, una vez que están completamente recorridas las dos cadenas.

El algoritmo es muy sencillo, en la imagen he omitido las partes de comprobación de nulos, o cadenas vacías, para que se vea más claro el algoritmo.

Lo primero que se hace es calcular los tamaños de los strings, de esa manera al tenerlo almacenado en variables, va a ser más sencilla la programación y la comprensión.

El siguiente paso es poblar la matriz, empezamos por la primera fila y primera columna que son las más sencillas. Para que una cadena sea igual al primer elemento de la otra, simplemente cada carácter que avanzamos de la cadena va a necesitar un cambio más, que sería el de eliminación.

Para poblar el resto de la matriz vamos a tener que ir observando los resultados anteriores, también debemos saber si los caracteres en ese índice son iguales, de esa manera no necesitaríamos cambio ni eliminación.

De esa manera se va poblando la tabla, siempre escogiendo el mínimo número de cambios de entre los que vamos dejando, por arriba, a la izquierda y en diagonal a los indices en los que nos encontremos en cada momento.

		F	A	C	I	L
	0	1	2	3	4	5
D	1	1	2	3	4	5
I	2	2	2	3	3	4
F	3	2	3	3	4	4
I	4	3	3	4	3	4
C	5	4	4	3	4	4
I	6	5	5	4	3	4
L	7	6	6	5	4	3

Con esta imagen es más sencilla la comprensión, sobre qué es lo que hace el algoritmo paso a paso, rellenando la matriz.

Para la comparación de la entrada de voz con las posibles preguntas, se utiliza este algoritmo tantas veces como preguntas conozca el avatar. Las preguntas se almacenan en un fichero de texto, por lo que va viendo las similitudes a la vez que recorre el fichero. De esa manera se queda con la pregunta más similar a lo que el usuario le ha preguntado.

Y por ello viene la segunda parte de las implementaciones relevantes.

Responder preguntas

Es un método simple, el cual realiza la función de ver qué pregunta es la que tiene que ser respondida, utilizando la clase LevenshteinDistance, la cual dispone de un método Compute, con el cual comparamos los dos strings. Esto se realiza en el bucle hasta que no haya más preguntas en nuestro fichero. De entre todas las preguntas, solo seleccionamos la que menor número de cambios tenga.

Más adelante, hay dos opciones, que nos convenza la similitud, o que pensemos que no son suficientemente similares como para pensar que el usuario quiere esa respuesta. Si no son suficientemente similares, se lo hacemos saber al usuario diciendole que no entendemos esa pregunta. En caso contrario, tenemos que tener claro, el idioma en el que se encuentra el avatar en ese momento, ya que puede ser cambiado en todo momento durante su ejecución.

Tendríamos que abrir el fichero específico para ese idioma, en el cual se encuentran las respuestas a las preguntas. Seleccionamos la respuesta deseada, usando el índice de la pregunta y lo mostramos en los subtítulos, y también como audio, el avatar nos dice la respuesta.

```
private void responderAPregunta(string text)
{
    int minimaSimilitud = 100;
    int preguntaSimilar = 1;
    int similares;
    fraseActual = -1;
    TextAsset ficheroTexto = (TextAsset)Resources.Load("preguntas/preguntas", typeof(TextAsset));
    string[] preguntas = System.Text.Encoding.Default.GetString(ficheroTexto.bytes).Split('\n');
    for(int i=0;i < preguntas.Length; i++)
    {
        similares = LevenshteinDistance.Compute(text.ToLower(), preguntas[i]);
        print(preguntas[i]);
        print(similares);
        if (similares < minimaSimilitud)
        {
            minimaSimilitud = similares;
            preguntaSimilar = i+1;
        }
    }

    if(minimaSimilitud <= 20) {
        if (idioma.Equals("español"))
            ficheroTexto = (TextAsset)Resources.Load("preguntas/respuestas_es", typeof(TextAsset));
        if (idioma.Equals("inglés"))
            ficheroTexto = (TextAsset)Resources.Load("preguntas/respuestas_en", typeof(TextAsset));
        if (idioma.Equals("francés"))
            ficheroTexto = (TextAsset)Resources.Load("preguntas/respuestas_fr", typeof(TextAsset));
        if (idioma.Equals("japonés"))
        {
            ficheroTexto = (TextAsset)Resources.Load("preguntas/respuestas_jap", typeof(TextAsset));
            string[] respuestas = System.Text.Encoding.UTF8.GetString(ficheroTexto.bytes).Split('\n');
            subtitulos.text = respuestas[preguntaSimilar - 1];
        }
        else {
            string[] respuestas = System.Text.Encoding.Default.GetString(ficheroTexto.bytes).Split('\n');
            subtitulos.text = respuestas[preguntaSimilar - 1];
        }

        subtitulos.enabled = true;

        audioSource.clip = Resources.Load("preguntas/" + preguntaSimilar) as AudioClip;
        audioSource.Play();

    }
    else
    {
        hablarYSubtitular("NoEntiendoPregunta",7);
    }
}
```

Organización de recursos

Por esto, comentaba en el capítulo de Unity, la necesidad de tener los recursos muy bien ordenados, tenemos que hacer muchos accesos a distintas carpetas y ficheros durante la ejecución en tiempo real, y tener una estructura clara de carpetas, subcarpetas y ficheros, nos hace que la programación pueda ser mucho más rápida y fluida.

Los nombres de los ficheros también son importantes, tanto para poder compararlos directamente con lo que ha dicho el usuario, como para poder acceder a carpetas por el valor que tiene una variable en cualquier momento, etc.

Mi sistema de ficheros, no es realmente una implementación relevante, pero sí que es totalmente necesario a la hora de realizar cualquier tipo de implementación en este proyecto en particular. Voy a explicar la arquitectura de la carpeta de recursos, y como he gestionado los nombres de cada fichero, carpeta, etc.

Dentro de la carpeta de recursos, tenemos todos los tipos de acciones que puede ejecutar el avatar. Esto quiere decir, tantas carpetas como acciones puede realizar, en mi caso: Audios, imágenes, powerpoints y preguntas.

La carpeta de audios no tiene nada especial, simplemente algún audio general de respuesta rápida del avatar.

En cuanto a la carpeta de imágenes, tendríamos una carpeta por cada edificio o zona cultural, y un fichero de texto NombreCarpetas, con el nombre de todas ellas, para realizar una búsqueda más rápida y no buscar carpeta a carpeta.

Dentro de cada carpeta, encontramos una estructura similar, en este caso en vez de carpetas, hay imágenes de ese edificio o zona, en el que su nombre define brevemente qué muestra la foto. Y un fichero de texto, NombreImagenes, que al igual que antes, nos sirve para buscar la foto que deseamos de una manera más rápida.

En cuanto a la implementación para mostrar una imágen, se va a entender fácilmente por qué tengo esta estructura.

Cuando el usuario pide una imagen:

"Enseñame una fotografía de la fachada de la Catedral de Pamplona"

Lo que va a hacer el programa es ver si en esa entrada de voz, el usuario ha nombrado alguna zona o edificio del cual tenemos imágenes, eso se hace mirando el fichero de texto NombreCarpetas, el cual dispone de todos los edificios de los que tenemos alguna foto para mostrar.

Si es así, entraría en esa carpeta y haría una búsqueda similar intentando encontrar qué es lo que quiere sobre ese edificio, en este caso la fachada.

Al recorrer el fichero de texto: Catedral de Pamplona/NombreImagenes.txt , vemos si tenemos alguna imagen de lo que el usuario ha pedido, y en caso afirmativo se muestra, y en caso contrario se le dice al usuario que no es posible encontrar esa foto, para que vuelva a intentarlo si lo necesita.

Siguiendo con la estructura de las carpetas y sus respectivas implementaciones, tenemos la carpeta PowerPoints, la cual en su primera instancia es igual que la de imágenes. Encontramos tantas carpetas como PowerPoints tengamos y un fichero de texto con sus nombres. El cambio viene dentro de la carpeta.

El avatar debe poder explicarnos el power point por su cuenta, y esto realmente necesita una implementación bastante elaborada para que él solo, vaya hablando, pasando diapositivas y poniendo los subtítulos correspondientes.

Para ello, en la carpeta relativa al power point que va a explicar tenemos: Una carpeta de audios, una carpeta de diapositivas, y ficheros de textos con el texto que va a decir en cada diapositiva.

Los audios están nombrados al igual que las diapositivas, de 1 a N, siendo N el número de diapositivas a explicar. De esta manera, cuando estemos exponiendo la diapositiva número 2, tendremos que estar utilizando también el audio número 2.

El fichero de texto con el idioma correspondiente, tiene también tantas líneas como diapositivas haya, de ese modo cada vez que hay un salto de línea, estaremos indicando al avatar que necesita cambiar de diapositiva.

Por último, la carpeta de preguntas, está explicado en la implementación de la distancia de Levenshtein. Simplemente un fichero de preguntas, y con la misma estructura que ese fichero de preguntas tenemos un fichero de respuestas por cada idioma que necesitemos.

Problemas afrontados

Uno de los problemas más grandes que he tenido en el proyecto ha sido la inexperiencia. La falta de conocimiento de las herramientas y tecnologías que he tenido que utilizar, han resultado en ir generando problemas que eran muy difícil de gestionar.

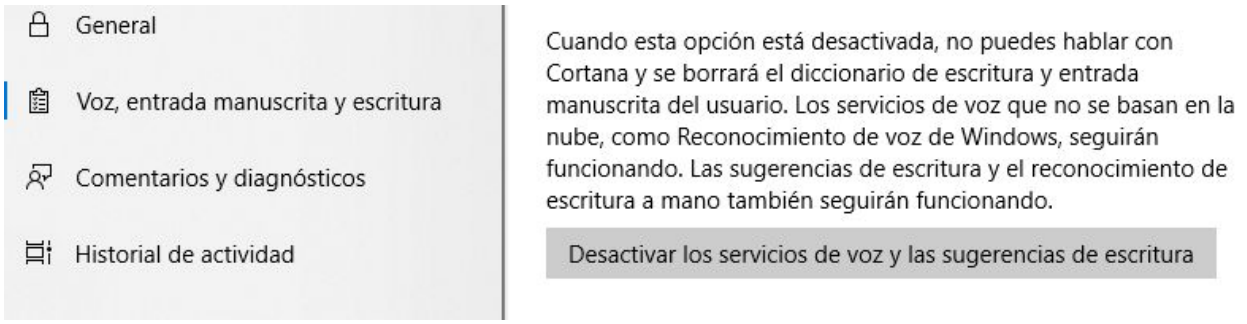
Un conocimiento bastante mínimo de Unity, me ha hecho que muchas veces navegar entre los posibles componentes y elementos, sea una odisea. Así como el lenguaje de programación C#, si bien es cierto que es muy parecido a Java, la realidad es que he tenido que realizar muchas búsquedas, investigación, etc. para poder continuar muchas veces con la programación.

Las librerías para C# de Unity también eran completamente novedosas, y realmente muy útiles. Pero son tan amplias, que es muy complicado poder llegar a aprender todo lo que realmente puedes hacer.

Aunque en ese aspecto, realmente la necesidad de documentarme, buscar por mi cuenta etc. ha hecho que aprenda rápidamente, pero también ha hecho que cuando me encontraba con un problema, la solución no se me ocurriera y fuera mucho más lenta su solución.

En cuanto a problemas “reales”, cabe destacar como he dicho, que juntandolos con el poco conocimiento del terreno, los problemas se multiplicaban por diez.

El primero de todos, viene con el reconocimiento de voz. Windows posee una librería de reconocimiento de voz, pero pone muchas trabas. Solamente para empezar a utilizarla, sin programar nada, hay que cambiar algunas configuraciones de Windows, y cambiar algunos de los ajustes del jugador en Unity, cosas muy específicas pero que en ningún lado nos las transmite Microsoft.



Para el correcto funcionamiento de la aplicación en cualquier ordenador, habría que activar estos ajustes. Tanto los servicios de voz y sugerencias de escritura, como permitir que el programa acceda al micrófono.

Otro de los problemas, sobre este tema, es que no era para nada intuitivo. Y la API, sirve de ayuda, pero no te sirve para entender el comportamiento ni cómo estructurar bien el código para que funcione. Ya que hay que sobreescribir cuatro métodos de la librería e inicializarlo de manera correcta, por lo que fue un poco enrevesado solo el hecho de empezar a poder hacer alguna prueba.

El siguiente problema ha sido uno de los más grandes de todo el proyecto: La síntesis de voz, al empezar con la implementación, comencé usando la clase SpeechSynthesizer. Esto ya me obligó a necesitar tener Cortana activada en mi ordenador, siempre ejecutada etc. Pero esto iba a hacer que no todas las máquinas pudieran soportar la aplicación, por lo que tenía que optar por otra opción.

Es entonces cuando opté por hacer streaming directamente de la web TTS de Google, después de muchos intentos, muchas pruebas y ver que no se podía... Sin realmente poder entender el por qué no funcionaba, tuve que cambiar de estrategia de nuevo.

Después de pasar por tres implementaciones distintas, todas ellas fallidas (En parte por inexperiencia y en parte por querer un sistema compatible), opté por el sistema de audios, como he comentado previamente en el documento. Una solución más sencilla, realmente compatible con cualquier máquina y además sin necesidad de una conexión a internet.

Gestión del proyecto

Esta es una de las partes más importantes de todo mi proyecto. Cabe destacar, que aunque yo haya sido el diseñador y desarrollador durante todo el proyecto, Jesús Villadangos también ha formado parte del proyecto. La idea original de este proyecto es suya, y por eso hemos tenido reuniones periódicas y comunicación por e-mail.

En cuanto a gestión del proyecto, he optado por una de las metodologías ágiles más comunes hoy en día: Scrum. Es cierto, que quizás tenemos una idea preconcebida de que Scrum es para equipos grandes, y que se organicen entre ellos para sacar el trabajo lo antes posible. Bueno, pues no necesariamente tiene que ser así, también es una forma de organizarse aunque no dispongas de un equipo directo.

Queda claro que un miembro por sí solo no hace un equipo. Sin embargo, la posibilidad que encuentre ayuda en el futuro es muy alta y aunque estemos solos necesitamos una manera de organizar el trabajo, poder ver el avance y la velocidad con que vamos resolviendo tareas. SCRUM es un buen compañero para todo esto.[5]

Para mi gestión particular, he seguido esta metodología con la ayuda de una hoja de cálculo. Ahora pondré algunas imágenes sobre la gestión, haciendo referencia al resumen del proyecto, con los integrantes del equipo. También el backlog del producto, muy importante para posteriormente dividir en tareas más pequeñas e ir haciendo el desarrollo del producto por partes.

Y por último, la pila de tareas del Sprint número 1, que en mi caso es el único Sprint, ya que iba a tener que trabajar solo en el desarrollo, al menos hasta que terminara este primer Sprint. Una vez finalizado el trabajo de fin de grado, el proyecto se quedaría abierto para poder trabajar, quizás incorporar más gente, más Sprints, etc.

Fechas de proyecto		
	Inicio Proyecto	Fin Proyecto
Prevista	4/6/2018	1/9/2018
Real	4/6/2018	27/7/2018

Recursos internos		
	Rol	Responsabilidad
Jesús Villadangos	Analista	Análisis, revisión y verificación
Rubén Miguélez	Scrum Master	Gestión del proyecto, diseño, desarrollo y verificación

Observaciones a tener en cuenta
Diseño y desarrollo íntegro por Rubén Miguélez
Idea original de Jesús Villadangos

Este es el resumen del proyecto. Como se ve, en un principio el proyecto estaba pensado para realizarse en todo el verano, pero por circunstancias personales, se tuvo que adelantar la fecha de fin de proyecto.

En cuanto a los integrantes del equipo de trabajo, se encuentra Jesús Villadangos como Analista. Es el encargado de revisar que el trabajo se ha cumplido, verificar y analizar el proyecto, también hace un poco el rol de cliente, al ser suya la idea. Y Rubén Miguélez, el Scrum Master y desarrollador. Es el encargado de gestionar el proyecto y hacer que se cumplan las tareas previstas, también diseñar y desarrollar el proyecto, y verificar que todo se hace de la manera correcta.

El backlog del producto se hace para tener una perspectiva de todo lo que se quiere hacer y tener claras las prioridades del cliente. Ayuda a que el equipo sea más autodisciplinado y respete las prioridades. También permite que el cliente pueda introducir cambios durante la vida del proyecto.[6]

Ayuda a manejar la incertidumbre durante el proyecto porque empuja a describir con más detalle las historias más importantes y a relativizar la importancia de detallar historias de menor prioridad. Es más ligero que un documento de requisitos exhaustivo.

El product backlog se compone de historias de usuario: Las historias de usuario deben ser:

Independientes unas de otras: De ser necesario, combinar las historias dependientes o buscar otra forma de dividir las historias de manera que resulten independientes.

Negociables: La historia en sí misma no es lo suficientemente explícita como para considerarse un contrato, la discusión con los usuarios debe permitir esclarecer su alcance y éste debe dejarse explícito bajo la forma de pruebas de validación.

Valoradas por los clientes o usuarios: Los intereses de los clientes y de los usuarios no siempre coinciden, pero en todo caso, cada historia debe ser importante para alguno de ellos más que para el desarrollador.

Estimables: Un resultado de la discusión de una historia de usuario es la estimación del tiempo que tomará completarla. Esto permite estimar el tiempo total del proyecto.

Pequeñas: Las historias muy largas son difíciles de estimar e imponen restricciones sobre la planificación de un desarrollo iterativo. Generalmente se recomienda la consolidación de historias muy cortas en una sola historia.

Verificables: Las historias de usuario cubren requerimientos funcionales, por lo que generalmente son verificables. Cuando sea posible, la verificación debe automatizarse, de manera que pueda ser verificada en cada entrega del proyecto.

	Historia de Usuario
2	
3	Como analista quiero un interfaz gráfico con avatar
4	Como analista quiero que el avatar interactúe por voz
5	Como analista quiero que el avatar responda preguntas
6	Como analista quiero que se puedan mostrar fotografías
7	Como analista quiero que se puedan mostrar power points
8	Como desarrollador quiero usar Unity
9	Como desarrollador quiero que el programa sea ejecutable en cualquier ordenador (Sistema operativo)
10	Como analista quiero que el programa sea multi idioma
11	
12	

	Imp	Sprint	Estimado	Realizado	Notas
2					
3	100	1	30	27	
4	90	1	40	45	
5	80	1	40	45	
6	70	1	30	23	
7	60	1	41	40	
8	40	1	18	25	
9	20	1	26		No conseguido.
10	20	1	20	25	
11					
12					
13					
14					

Como se puede observar, las historias de usuario están ordenadas por prioridad, todas pertenecen al sprint uno, y la mayoría son necesidades del analista, que en este caso, como también es el que hace rol de cliente, es lógico que exija qué tipo de cosas quiere que tenga su aplicación. Yo, como desarrollador, mis necesidades son más bien prácticas que realmente aportar funcionalidad.

Se puede observar que hay variaciones entre la estimación y la realidad. Esto siempre ocurre, pero cuantos más proyectos gestionas, la idea es conseguir buenas estimaciones. Para no perder dinero.

En cuanto a hacer un programa ejecutable en cualquier máquina, simplemente no es que haya fallado la estimación, sino que no ha sido posible. Con las herramientas y tiempo del que disponía no era viable esa opción.

Así que todas esas cosas, cambian el curso del proyecto. De una estimación que teníamos, a utilizar un montón de horas más y encima no haber podido solucionarlo.

El hecho de haber abortado la misión, es que era una historia de usuario hecha por el desarrollador, con una prioridad muy muy baja. Por eso es muy importante fijar las prioridades bien. Es más importante poder realizar todo lo que el cliente pedía con unas altas prioridades, que cosas que no aportan funcionalidad, que tampoco importan tanto. Hay que saber gestionar el tiempo y aceptar una derrota. Si no puede salir muy caro el atrevimiento.

La próxima parte de la gestión es quizás la más importante de todas, hay que especificar todas las tareas que se han de desarrollar en cada historia de usuario.

Las tareas deben ser suficientemente laboriosas como para tener que trabajar bastante tiempo en ellas y que aporte algo extra al proyecto.

En mi caso he dividido las historias de usuario en tareas pequeñas, pero es cierto que algunas quizás son demasiado complejas y laboriosas.

ID	Nombre	Estimado
ID0001	Como analista quiero un interfaz gráfico con avatar	30
ID0001_T1	Definir los componentes de Unity	5
ID0001_T2	Diseñar/Encontrar un modelo 3D para el avatar con animaciones	13
ID0001_T3	Proponer y crear la escena	6
ID0001_T4	Implementar el interfaz responsivo actualizando los componentes por código	6
ID0002	Como analista quiero que el avatar interactúe por voz	40
ID002_T1	Informarse sobre el reconocimiento de voz	5
ID002_T2	Implementar reconocimiento de voz en C#	10
ID002_T3	Informarse sobre servicios TTS (Text to Speech)	5
ID002_T4	Implementar síntesis de voz usando C#	10
ID002_T5	Realizar pruebas de reconocimiento de voz	5
ID002_T6	Realizar pruebas de síntesis de voz	5
ID0003	Como analista quiero que el avatar responda preguntas	40
ID003_T1	Planificar sistema de ficheros preguntas-respuestas	5
ID003_T2	Implementar un algoritmo para determinar similitud entre la entrada y lo esperado	18
ID003_T3	Implementar el modo de responder una pregunta dada	12
ID005_T4	Realizar pruebas	5
ID0004	Como analista quiero que se puedan mostrar fotografías	30
ID004_T1	Planificar un sistema de carpetas con todas las imagenes	5
ID004_T2	Implementar un método para obtener la imagen deseada	9
ID004_T3	Implementar un método para mostrar la imagen enmarcada en un canvas	6
ID004_T4	Implementar un método para ocultar la imagen	5
ID005_T5	Realizar pruebas	5
ID0005	Como analista quiero que se puedan mostrar power points	41
ID005_T1	Planificar sistema de carpetas para mostrar los power points	5
ID005_T2	Implementar un método para obtener el power point deseado	9
ID005_T3	Implementar un método para que el avatar explique el power point	7
ID005_T4	Implementar la manera de pasar diapositivas cuando el avatar deje de explicar	5
ID005_T5	Implementar un modo de mostrar el power point en un canvas	5
ID005_T6	Implementar un método para ocultar el power point	5
ID005_T7	Realizar pruebas	5
ID0006	Como desarrollador quiero usar Unity	18
ID006_T1	Informarse sobre la herramienta y su funcionamiento	3
ID006_T2	Documentarse sobre C# como lenguaje de programación en Unity	5
ID006_T3	Definir los componentes a utilizar	10
ID0007	Como desarrollador quiero que el programa sea ejecutable en cualquier ordenador (Sistema operativo)	26
ID007_T1	Buscar alternativas a la síntesis de voz de Microsoft, Android etc.	4
ID007_T2	Buscar alternativas al reconocimiento de voz de Microsoft	4
ID007_T3	Implementar un sistema de síntesis de voz compatible en todos ordenadores	7
ID007_T4	Implementar un sistema de reconocimiento de voz compatible en todos ordenadores	7
ID007_T5	Realizar pruebas en distintas máquinas	4
ID0008	Como analista quiero que el programa sea multi idioma	20
ID008_T1	Definir idiomas	3
ID008_T2	Implementar el método para cambiar de idioma	6
ID008_T3	Implementar un modo para mostrar por pantalla los subtitulos en el idioma deseado	7
ID008_T4	Realizar pruebas en distintos idiomas	4

Estás son todas las tareas que había que realizar en mi proyecto. Una gran cantidad de las tareas consistían en buscar, investigar, documentarse, etc. Esto es, como he dicho antes, porque no conocía el terreno. Y había que ser previsor a la hora de gestionar el proyecto, sin este tipo de tareas, el proyecto habría llevado mucho más tiempo del que realmente ha costado.

Mi manera de proseguir fue la siguiente:
En orden de historias de usuario más prioritarias a menos, cogía sus tareas en orden, de una en una. Hasta que no se acababa una tarea, no se empezaba con la siguiente. De esa manera, iba por partes, resolviendo problemas pequeños y siempre ampliando lo anterior, pero sin necesidad de modificarlo.

Un problema grande en este tipo de gestión es intentar coger más tareas de las que puedes gestionar, aunque dos tareas parezcan similares, el Scrum Master se ha encargado de buscar tareas que entre ellas sean distintas. Por lo que coger más de una tarea por similitud, puede hacer que al final tardes más tiempo en terminarlas, que el que realmente necesitarías si vas poco a poco.

En mi caso, la realización de pruebas en cada historia de usuario es una parte muy importante y costosa. Lleva mucho tiempo probar, verificar y analizar que esa historia de usuario está acabada. Realmente es una de las partes más importantes, ya que si das una historia de usuario por acabada con fallos, eso puede dar como resultado un grave problema de tiempo, tratando de identificar los problemas más adelante y volviendo a rehacer esas tareas. Ahora hablaremos de ello.

Pruebas realizadas

Como he comentado, las pruebas son parte imprescindible del proyecto… Tanto es así que es una de las tareas que más tiempo requiere.

En las principales implementaciones el uso de las pruebas era crucial para poder seguir avanzando con el proyecto, por ejemplo después de la implementación del reconocimiento de voz, era necesario probar el programa, con distintas voces, tonos e incluso micrófonos. Así como realizar pruebas periódicas durante el desarrollo de las tareas.

Las pruebas de reconocimiento de voz, trataban de probar diferentes palabras, un vocabulario más técnico, cambiar el acento de la voz. Todo ello para cerciorarse de que realmente funciona, y va a funcionar, en todos los casos.

En cuanto a las de síntesis, había que cerciorarse de que el avatar fuera capaz de pronunciar todo tipo de palabras, tecnicismos, palabras acentuadas, sin acentuar, nombres propios etc.

El problema que había con la síntesis de voz, era la imposibilidad de que hiciera otra cosa mientras hablaba. Esto quiere decir, que si mientras está hablando tiene que mostrar algo o reconocer algo de lo que yo digo, eso no lo va a hacer. Y eso era un punto importante en el proyecto. Con lo que las pruebas van más allá de simplemente hacerle al avatar decir unas cuantas frases.

Cabe destacar, que durante todo el desarrollo de las pruebas, estaba ejecutando una versión de depuración, con la que observar por consola qué partes del código se ejecutaban, donde falla, etc. Es muy parecida a la consola de Unity, y salen todos los errores de una manera muy similar. De esa manera la depuración es más sencilla y se acotan los errores en un tiempo más corto.

Pruebas de ejecución

Voy a mostrar el proceso de vida de la aplicación, desde que es ejecutada y como reacciona a las distintas frases que yo le voy diciendo.

Ejecutamos la aplicación:

Vemos claramente el interfaz, en el cual se ve el fondo que es la puerta del juicio de la Catedral de Tudela, y un humano, el cual es nuestro avatar.

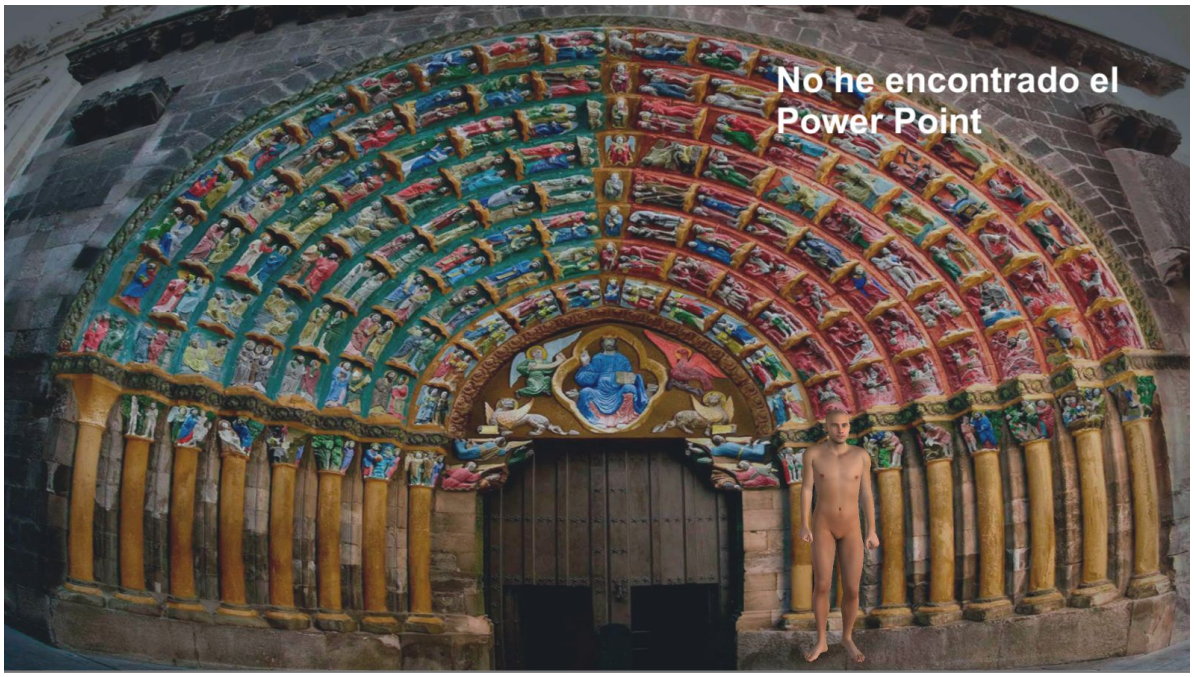
Ahora vamos a hacer que reaccione a algo que le decimos:

En este caso vamos a decirle : “Avatar, muestrame una foto de la fachada, de la Catedral de Pamplona”:

Seguidamente le decimos: “Cambia el idioma a japonés”:

Vamos a poner el idioma en español todo el rato, ya que como no hay audio y son solamente fotos demostrativas, en Japonés no nos vamos a enterar de que está diciendo el avatar: “Cambia el idioma a español de nuevo”.

Y ahora le pedimos que nos muestre un power point: “Avatar, ¿nos puedes mostrar el power point de la catedral de Tudela?”

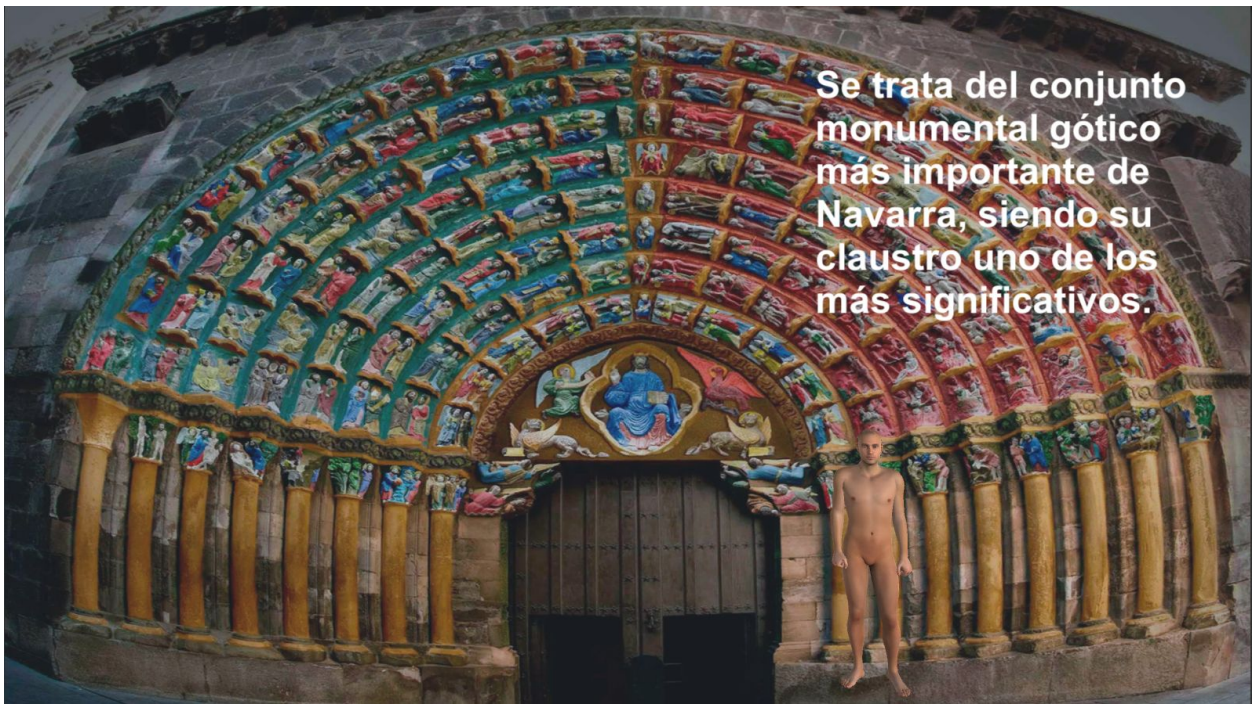


“¿Y el power point de mi tfg nos puedes enseñar?”. Una vez que le pedimos esto nos muestra por pantalla el power point deseado y nos dice que nos puede explicar el power point. Al decirle que si, empezaría a hablar sobre ese power point y a pasar diapositivas por su cuenta, pero siempre podríamos explicarlo nosotros y que vaya pasando diapositivas.

"Sí, explicanos el power point, por favor". Lo va explicando diapositiva a diapositiva, hasta el final:

Como sabemos, también responde preguntas:
“Avatar, ¿te puedo hacer una pregunta?”, a lo que el avatar responde: “Sí, adelante”.

“¿Cual es el valor artístico de la Catedral de Pamplona?”:



No lo he comentado, pero actualmente se puede cambiar entre 4 distintos idiomas: Español, inglés, francés y japonés.

Todo lo que el avatar dice, se puede leer con subtítulos en otro idioma distinto, esa es la gran ventaja de ser multidioma.

Conclusiones y líneas futuras

Se puede concluir fácilmente que ha sido un trabajo duro, sobre todo de preparación y enfrentamiento con nuevos retos, herramientas, tecnologías.

La parte del reconocimiento de voz, no es para nada intuitiva y es muy difícil de gestionar.

Si pudiera volver a empezar, posiblemente metería más horas a documentarse sobre el reconocimiento de voz usando C#, ya que tiene que haber muchas otras alternativas al reconocimiento de Microsoft, y esto ayudaría a terminar la única historia de usuario que no se ha podido completar.

Pienso que Unity es un gran entorno de trabajo, tiene muchísimas características que todavía no he podido probar, y realmente ha sido una gran ayuda para la realización del proyecto. Porque la verdad es que sin Unity, se me habría complicado mucho tanto el tema de diseño como la gestión de los recursos mediante el sistema de ficheros definido anteriormente.

He aprendido mucho, sobre gran variedad de temas que desconocía totalmente, y trabajar en un proyecto real, con un fin, ha sido una gran motivación.

El apoyo de Jesús Villadangos, y su idea original, han sido buenos aliados que han hecho que el proyecto avance más rápido de lo esperado. Agradezco su dedicación y esfuerzo en el proyecto.

Como última conclusión, quería destacar la intensidad del proyecto, en el aspecto del tiempo, literalmente los días se redujeron a la mitad de lo esperado, pero las horas que había que meter en el proyecto eran las mismas. Con lo que ha sido una carrera contrarreloj, que muchas veces no sale bien. Con más tiempo, y sin tanta prisa, podrían haberse terminado todas las tareas previstas en la gestión del proyecto, y posiblemente muchas de las cosas que vamos a ver a continuación.

Lo bueno de este proyecto, es que se ha generado un esqueleto muy fuerte. La idea original es destinar al avatar a la promoción del turismo en Navarra, pero sinceramente, esto se puede exportar a muchísimos más campos: Deporte, tecnología, profesorado, etc.

Lo que he creado es muy versátil, y lo que realmente importa es la funcionalidad. No tanto la estética o lo que realmente diga el avatar. El concepto de aplicación es lo que la hace tan fácilmente exportable.

En un futuro, también se podría hablar con un diseñador gráfico o modelador 3D. Para realmente obtener el resultado que queremos, que podría ser perfectamente un monje con túnica del siglo XVI. Pero para ello se necesita más gente trabajando en este proyecto.

Una de las cosas que me gustaría hacer es independizarlo totalmente del Sistema operativo, estaba en los planes, pero por el sistema de priorización al final no se pudo hacer. Puede que esto quiera decir que hay que cambiar la raíz y buscar otras formas de reconocimiento de voz, pero se puede hacer perfectamente con ganas y tiempo.

Cuantos más idiomas, mejor, incluso podría hacerse que pueda mostrar subtítulos en cualquier idioma del mundo. Habría que documentarse mucho, pero la verdad es que sería un gran punto a favor en cuanto a exportar el producto también a otros países. Y si pudiera también hablar en cualquier idioma, ya sería perfecto.

Sinceramente creo que el proyecto tiene mucho futuro, y se puede seguir investigando en esta dirección. Hay muchas cosas que se pueden hacer, y muchas otras que están por descubrir.

Lo importante es que ya está hecho el esqueleto o el maniquí, ahora solo hay que saber vestirlo.

Bibliografía

[1] "Introducción - Virtuali-Tee" KickStarter, https://www.kickstarter.com/projects/curiscope/virtualitee. Accedido el día 13 de Julio de 2018

[2] "Herramientas utilizadas - Unity." Unity, https://unity3d.com/es/learn. Accedido el día 13 de Julio de 2018.

[3] "Herramientas utilizadas - Google Text-To-Speech" Google Cloud, https://cloud.google.com/text-to-speech/. Accedido el día 14 de Julio de 2018.

[4] "Implementaciones relevantes - Distancia de Levenshtein" Wikipedia, https://es.wikipedia.org/wiki/Distancia_de_Levenshtein. Accedido el día 15 de Julio de 2018.

[5] "Gestión del proyecto - Scrum." Wikipedia, https://en.wikipedia.org/wiki/Scrum_(software_development). Accedido el día 17 de Julio de 2018.

[6] "Gestión del proyecto - Backlog del producto" Scrum Institute, https://www.scrum-institute.org/The_Scrum_Product_Backlog.php. Accedido el día 17 de Julio de 2018.