ESCOLA UNIVERSITÀRIA D'ENGINYERIA
TÈCNICA DE TELECOMUNICACIÓ LA SALLE

TREBALL FINAL DE MÀSTER

MÀSTER EN ENGINYERIA DE XARXES I TELECOMUNICACIONS

# INTEGRACIÓN
# DE UN CONVERSOR TEXTO A VOZ
# EN UNA PLATAFORMA DE MUNDOS VIRTUALES

ALUMNE

Oier Fuentes Mendizabal

PROFESSOR PONENT

Ignasi Iriondo Sanz

# ACTA DE L'EXAMEN
# DEL TREBALL FINAL DE MÀSTER

Reunit el Tribunal qualificador en el dia de la data, l'alumne

D. Oier Fuentes Mendizabal

va exposar el seu Treball Final de Màster, el qual va tractar sobre el tema següent:

INTEGRACIÓN
DE UN CONVERSOR TEXTO A VOZ
EN UNA PLATAFORMA DE MUNDOS VIRTUALES

Acabada l'exposició i contestades per part de l'alumne les objeccions formulades pels Srs. membres del tribunal, aquest valorà l'esmentat Treball amb la qualificació de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENT DEL TRIBUNAL

*Bihotz, antzinako bihotz,*
*Ez al zara zaharregi,*
*Eta ilun, eta itsu, barregarri.*

# Agradecimientos

Quiero dar las gracias primeramente a mi tutor Ignasi Iriondo por toda su ayuda. Todo mi agradecimiento también para el equipo de ENNE en Iruñea: Amalia, Dani, Carlos, Izai, Andrés y Alberto. Con ellos he convivido durante estos meses en la oficina y sin ellos este trabajo no hubiera podido realizarse.

Muchas gracias también a esos que siempre están ahí y a los que por desgracia no pueden estar entre nosotros.

Un abrazo a mi familia más cercana.

# Prefacio

Este proyecto ha sido desarrollada en la empresa ENNE Entertainment Studios, situada en el Centro de I+D Jeronimo de Ayanz de Iruñea. Con ella, se finalizan los estudios del Master Universitario en Telecomunicaciones en la Escuela de Ingeniería y Arquitectura La Salle de Barcelona.

La tesis cuenta con la supervisión de Amalia Ortiz (responsable de soluciones I+D+i) por parte de ENNE e Ignasi Iriondo por parte de La Salle.

El trabajo ha sido planificado, diseñado y concretado entre Junio y Septiembre de 2010.

Iruñea, Septiembre 2010. Oier Fuentes Mendizabal.

# Abstracto

El objetivo de esta tesis es la integración de un conversor de texto a voz dentro de una plataforma de mundos virtuales enfocada al juego multijugador masivo en línea.

El proyecto requiere diversas tareas. Primero es necesario un conocimiento de los mundos virtuales, y más concretamente de las plataformas. Tras un primer análisis de la plataforma, el proyecto se centra en las funcionalidades relativas a las comunicaciones, tales como el chat o la voz. Analizadas estas funcionalidades, se estudian las necesidades para la extensión de la plataforma, y más concretamente se detecta la necesidad de un conversor de texto a voz.

Sabiendo los requisitos mínimos que ha de satisfacer el conversor, se elabora un estado del arte y se analizan las diversas opciones a integrar en la plataforma. Tras la elección del sistema, se lleva a cabo la integración, previo análisis de la arquitectura del sistema y estudio de la viabilidad.

Finalmente se concreta la integración, se desarrollan tres aplicaciones que utilizan el servicio y se realizan diversos tests con el fin de evaluar y validar el resultado final.

También son estudiadas diversas líneas futuras para continuar trabajando, tras el análisis de los resultados.

# Sumario

El desarrollo de una plataforma de mundos virtuales es una de las tareas a realizar de ENNE Entertainment Studios en su centro de I+D de Pamplona. La plataforma sobre la que se trabaja es realXtend (reX), una herramienta de código abierto sobre el cual se pretende crear un producto extendiendo sus funcionalidades.

Algunas de estas extensiones están relacionadas con las comunicaciones, por lo que en primer lugar reX es analizado para ver la situación de las comunicaciones en la plataforma y detectar posibles implementaciones o mejoras.

Tras una evaluación de este estudio, se plantea la integración de un conversor de texto a voz dentro de realXtend para lo cual hay que estudiar tanto la forma de integrar el conversor texto a voz (Text To Speech, TTS) como la elección propia de un TTS que satisfaga las necesidades de ENNE.

El estado del arte realizado dentro de los TTS de código libre, define el software sobre el que se trabajará: Festival Speech Synthesis System.

Esta herramienta es estudiada, creando una versión apropiada para la integración en reX. Por otro lado la arquitectura modular de realXtend (compuesto por un cliente llamado Naali y un servidor llamado Taiga) permite la creación de un nuevo módulo con servicios TTS que utiliza Festival, aprovechado para diversas funcionalidades.

Tras la integración del módulo se desarrollan tres aplicaciones aprovechando el sistema de mensajería instantánea (chat) propia del sistema, las notificaciones, y el sistema de componentes de reX. Para ello es necesario, tanto el módulo TTS como una parte gráfica de configuración en alguno de los casos.

Por último se valida y evalúa la integración tanto del módulo TTS como de las distintas aplicaciones, mediante un test realizado por 7 sujetos.

Tras extraer unas conclusiones, se plantean las líneas futuras a seguir así como la posibilidad de desarrollar más funcionalidades que aprovechen el módulo integrado en la plataforma.

# Índice general

# Capítulo 1

# Introducción

## 1.1. Motivación

Virtual, del latín (virtus): fuerza, virtud. Se le suele atribuir el significado de aquello que existe, sin ser real. El término, es utilizado hoy en día en infinitas ocasiones: realidad virtual, campus virtual, guerra virtual, impresora virtual... Con lo cual, y aunque parezca una contradicción, lo virtual existe, ya que virtual no significa irreal en el sentido de existencia verdadera y efectiva.

A lo largo de los últimos años, diversas aplicaciones relacionadas con esta acepción de la palabra han ido desarrollanose conjuntamente con la técnica. De esta manera, y ligado a la inteligencia artificial la proliferación de los mundos virtuales (virtual worlds, VW) ha sido vertiginosa.

En un mundo virtual, se puede simular un mundo o un entorno en tres dimensiones (3D), en los cuales un actor representado por un personaje (avatar) puede interactuar con el entorno o con otros participantes de manera semejante a la vida real. Estas simulaciones de la realidad pueden servir de plataforma para diversas funciones tanto posibles como imposibles (o improbables).

Los mundos virtuales pueden ofrecer innumerables oportunidades [12] [11] [2] [13], también en el sector empresarial. Una empresa, además de desarrollar estas opciones, puede aprovechar también los mundos virtuales para ella misma [5]. Podría, por ejemplo, hacer reuniones virtuales ahorrándose el coste, dinero y sacrificio de los viajes, juntando en un mismo espacio tridimensional a todos los afectados de forma que puedan interactuar, hablar, escribir, o compartir cualquier tipo de información como archivos, sonidos, videos...

Los campos en los cuales un mundo virtual puede ser explotado son muchísimos. Prácticamente cualquier situación real puede ser trasladada a un mundo virtual, consiguiendo además añadir funcionalidades inexistentes en la realidad. En el mundo de los contenidos digitales, este tipo de plataformas son muy interesantes para desarrollar productos de diferentes tipos.

ENNE Entertainment Studios se define como una productora de contenidos digitales orientados al entretenimiento y otros sectores. Entre sus líneas de negocios, está el sector de los videojuegos, y más concretamente el sector de los juegos multijugador masivos online (MMOG). Este tipo de videojuego puede desarrollarse utilizando plataformas de mundos virtuales, pudiendo ser estas comerciales o de código libre. La empresa parte de la experiencia de Puerta 34 [1], un innovador proyecto donde el jugador se mueve por diversas ciudades, dando a conocer el castellano con fines académicos, lúdicos y turísticos.

El propósito del centro de I+D de ENNE, es desarrollar una nueva plataforma basándose en los últimos avances técnicos y mediante software libre. Además de colaborar con la Comunidad de desarrolladores, ENNE trabajará en sus propias ideas desarrollándolas y adaptándolas para sus productos.

Tras un análisis de las diversas posibilidades, el software elegido para la creación de esta nueva plataforma fue realXtend. Este es un proyecto nuevo, en continuo desarrollo y con constantes modificaciones.

Aunque las funciones a desarrollar pueden ser muchas, este trabajo se centra en las aplicaciones relacionadas con las comunicaciones en los mundos virtuales. En un VW, los avatares pueden comunicarse de forma escrita, oral (mediante la voz, natural o artificial sintetizada) y visual (mediante gestos, imágenes, video) pudiendo ser estos métodos públicos (todo el mundo es participante), privados (participan sólo algunos), y retardados u off-line (a modo de buzón, por ejemplo). Es posible también la comunicación táctil u otro tipo de comunicación no tan desarrollada como las comentadas.

En el caso de ENNE, con una plataforma funcionando en el entorno educativo, hay comunicaciones que deben de realizarse para cumplir con las tareas asignadas. Esta comunicación o interacción, sea entre estudiantes, profesores u objetos, debería de ser realizada de las mayores formas posibles. Entre otras cosas, el estudiante debe trabajar su vocabulario, su pronunciación, su escritura, etc. Para ello, todas las herramientas de comunicaciones de la plataforma deben

[1] www.puerta34.com

de ser explotadas.

Además, los conocimientos lingüísticos pueden ser adquiridos por estas vías, mediante las cuales es posible ponerlos en práctica.

En la vía oral, la voz artificial es útil en este tipo de entornos. Esta voz, puede ser generada o sintetizada de forma artificial, haciendo que se genere voz a partir de una determinada orden o entrada, como por ejemplo un archivo de texto. Este tipo concreto de generar voz, es conocido como conversor de texto a voz (Text To Speech, TTS).

El TTS puede ser utilizado de diversas formas, pudiendo convertir texto de cualquier tipo (noticias, correo, cadenas de texto, libros) a formato audible. Este proceso además, requiere de varios pasos intermedios en los que se obtiene información de los fonemas. Los datos fonéticos pueden ser utilizados para animar un avatar, sincronizando sus labios con el habla sintetizada. No es menos importante otro tipo de aplicación como la lectura y manejo para discapacitados, o la combinación de sistemas TTS con reconocedores de voz, sistemas de traducción, o aplicaciones del tipo Call Center (servicio telefónico).

Una aplicación TTS puede ser la de generación de contenido educativo. Así, un profesor podría generar una lección auditiva partiendo de un texto escrito, haciendo que el estudiante pueda oír la lección e incluso ver al avatar de su profesor hablándole, en cualquier instante.

Otra aplicación para mejorar la comprensión y agilizar el oído, es la utilización del TTS dentro de un chat o mensajería instantánea del mundo para reproducir el texto por los altavoces.

Otras aplicaciones se presentan si se combinan los conversores de texto a voz con otros sistemas, como los conversores de voz a texto y los traductores a tiempo real. Con esta combinación podría crearse una comunicación multilingüe a tiempo real tanto de forma escrita como auditiva/hablada. Por otro lado, hay que tener claro que en cualquier lugar o uso donde sea empleado el texto como comunicación, el TTS puede proporcionar una voz sintética.

La comunicación de forma visual puede llevarse a cabo mediante animaciones en el avatar, como pueden ser un saludo, una sonrisa, o un gesto, o mediante vídeo a través de una cámara de vídeo. También puede considerarse comunicación visual la efectuada mediante alguna presentación, algún tipo de imagen o fotografía, o utilizando una pizarra.

Nuevos métodos de detección de gestos, o de ojos pueden proveer información

visual añadida.

## 1.2. Objetivo

El presente proyecto, se centra en el estudio de la plataforma realXtend y en la investigación y desarrollo de un TTS capaz de integrarse con la plataforma y que pueda ser explotado para diversas funcionalidades.

Además de esta integración con la creación de un nuevo módulo para la plataforma de mundos virtuales, la integración de cualquier aplicación que necesite el módulo TTS es explicada implementando tres aplicaciones, de tres formas diferentes: la aplicación del chat con TTS, los componentes TTS y el lector de notificaciones.

## 1.3. Contenido de la memoria

En la presente memoria se analizan las partes involucradas en el proyecto. Primeramente los mundos virtuales son explicados en el Capítulo 2, explicando brevemente la terminología utilizada y se describe la plataforma: reX. Se analizan las comunicaciones implementadas y la necesidad de un conversor de texto a voz. Por último se describe la arquitectura de la plataforma y se presenta el sistema modular.

En el Capítulo 3 se introduce la herramienta integrada, y se explica el porqué de la elección de *Festival Speech Synthesis System* como solución definitiva, tras el estado del arte efectuado en todas las alternativas de Código Libre.

En el Capítulo 4 se lleva a cabo un análisis de estas dos partes involucradas. Se amplían y desarrollan funcionalidades para *Festival* adaptándolas para la integración de reX. Se crea el módulo TTS partiendo de la creación de un módulo base y del diseño de módulos de comunicaciones implementados y desarrollando un servicio válido para todos los módulos. Como ejemplo de uso de esta integración, se crea un conjunto de aplicaciones diferentes para mostrar la utilidad del módulo: una aplicación TTS para el chat, añadiendo una nueva parte gráfica y otra funcional, un lector de notificaciones del mundo y un componente TTS que puede ser agregado a los distintos objetos del VW.

En el Capítulo 5 se evalúa y valida la integración del módulo y su uso en las aplicaciones descritas, para concluir con el Capítulo 6 donde se extraen unas

conclusiones y se presentan líneas futuras así como otras aplicaciones previstas.

# Capítulo 2

# Mundos virtuales

A lo largo de este capítulo se describen los conceptos básicos de los mundos virtuales y se presenta la plataforma de desarrollo de mundos virtuales realXtend.

Se muestran sus características actuales de cara al jugador y se analiza en detalle el estado de los canales de comunicaciones, entre los cuales está el TTS.

Antes de escoger el sistema TTS, en el Capítulo 3 se describe la arquitectura modular de realXtend para poder integrar ambos sistemas en el Capítulo 4

## 2.1. Definiciones

Es importante conocer la terminología utilizada en el entorno de mundos virtuales. A continuación se describen brevemente algunos conceptos clave en la materia:

- **Mundo Virtual**: Es la representación en tres dimensiones de un entorno donde uno o varios usuarios pueden realizar diversas actividades interactivas (como jugar, aprender, comprar o socializarse) a través de los personajes virtuales (avatares) que les representan.

- **MMOG (Massively multiplayer online game)**: Es un videojuego en el que miles de jugadores conectados en red juegan e interactúan simultáneamente en un mundo virtual, percibiendo la acción de los otros sobre el entorno.

- **Plataforma de desarrollo de mundos virtuales**: Herramienta o conjunto de herramientas de trabajo que facilitan la creación de mundos virtuales. En ambientes MMOG se les suele llamar Middleware, aunque también puede ser definida como SDK (Software Development Kit) o kit de desarrollo de software. Esta plataforma facilita la creación de mundos virtuales.

- **Modelado 3D**: El modelado es la fase en la que se crean las estructuras o geometrías de los objetos tridimensionales que van a formar el mundo virtual (escenario, objetos, avatares,...).

- **Animacion 3D**: La animación es la fase en la que se dota de movimiento a cada uno de los modelos 3D que forman el mundo virtual.

- **Assets**: Conjunto de elementos (modelos, texturas, animaciones, audios,...) que forman el mundo virtual.

- **Renderizado**: Proceso de generación de imágenes 2D a partir del mundo virtual 3D.

- **Realidad aumentada**: Tecnologías que permiten mezclar el mundo real con el virtual.

- **Avatar**: Es un personaje virtual tridimensional. El avatar puede tener dos tipos de roles en el mundo virtual. Por un lado, puede representar al usuario. En este caso, el usuario le dirige como a una marioneta. Por otro lado, puede ser autónomo, es decir, tiene un sistema de inteligencia artificial que le permite actuar. En el mundo del videojuego a los avatares autónomos también se les llama NPC (non player character).

- **Framework**: Se refiere a una estructura o marco conceptual que sirve como base para el desarrollo de algo, y lo convierte en útil. El framework indica como se construye y relaciona el programa, mediante su interfaz y sus herramientas. Es la estructura de un software, o de una plataforma de desarrollo de mundos virtuales, que facilita el desarrollo de la propia plataforma.

## 2.2. Introducción a los mundos virtuales

Una vez conocida la terminología, se puede entrar más en detalle a la composición de un mundo virtual y su desarrollo. Estos mundos inmersivos en 3D

que permitan interactuar a un número masivo de usuarios y que sea persistente (el mundo sigue existiendo independientemente de que estemos conectados al mismo) exigen ciertos requisitos que se explican a continuación:

1. **Modelado**: El juego se desarrolla en un espacio geográfico virtual que tiene que ser modelado junto a los avatares El modelado incluye formas, colores, esqueletos, posibilidad de movimientos, texturas, etc).

2. **Animación**: El mundo virtual ha de ser animado, dotando a los elementos de capacidad de movimiento de acuerdo a los principios definidos en el modelado (esqueletos, deformaciones,. . . .)

3. **Red**: El mundo virtual requiere que se conozca centralmente la posición y movimiento de todos los elementos y que se comunique a todos los usuarios/clientes para poder gestionar las interacciones entre los mismos. Esta actividad es gestionada por el motor de red que se ejecuta en un servidor central. La optimización de las comunicaciones entre clientes y servidores, dado el volumen de las mismas y las limitaciones de Internet, convierten al motor de red, en la pieza fundamental de las plataformas de desarrollo de Mundos virtuales.

4. **Renderizado/Gráfico**: Los usuarios/clientes deben ser capaces de visualizar los elementos del mundo virtual correspondientes a su punto de vista. Para ello necesita conocer las posiciones de todos los elementos y como representarlos en la pantalla según sus características. Esta actividad es gestionada por el motor gráfico. El motor gráfico se ejecuta en los clientes de manera que la experiencia del usuario sea ágil y satisfactoria.

5. Herramientas que faciliten la **administración y explotación** del juego: creación de usuarios, autenticación, gestión de backups, gestión del rendimiento y monitorización de los elementos de la plataforma, etc.

6. **Física**: Implementar los principios físicos que rigen el mundo virtual. Por ejemplo: colisiones, efecto del viento en los árboles, etc.

7. **Inteligencia Artificial (IA)**: Implementar la inteligencia artificial por la que se rigen todos los elementos no controlados por los usuarios. El elemento es cualquier cosa capaz de percibir su entorno (recibir entradas), procesar

tales percepciones y actuar en su entorno (proporcionar salidas), como por ejemplo un NPC.

8. **Lógica**: Implementar la lógica que definan las misiones, juegos, etc, que componen el objeto de la aplicación.

Para facilitar el desarrollo de todos estos elementos comunes para todos los mundos virtuales, se emplean plataformas de desarrollo de (algunas veces referenciadas como *middleware*, SDK o *framework*). Estas plataformas facilitan el trabajo de desarrollo al proveer de un entorno donde tan solo es necesario parametrizar y personalizar los mundos virtuales, aumentando la productividad de los creadores que de esta forma requieren un nivel técnico mucho menor.

Algunos ejemplos de middlewares son Multiverse, Hero Engine, Big World o NetDog + C4 entre los privativos y RedDwarf o realXtend entre los de código libre.

## 2.3. RealXtend

realXtend es una plataforma de desarrollo o *middleware* de Mundos Virtuales con licencia Open Source desarrollada principalmente en Oulu, Finlandia. Como el propio equipo de reX describe [1], el valor de los Mundos Virtuales no reside en la plataforma, sino en el contenido. La comunidad reX es Open Source, pretendiendo alcanzar el mayor numero de usuarios y desarrolladores ya sea mediante socios o usuarios independientes que formen una comunidad sólida y con futuro. Así, mediante una base desarrollada de manera colaborativa, se pueden crear aplicaciones propias invirtiendo el tiempo en el desarrollo del mundo virtual gracias al middleware.

realXtend es un proyecto nuevo, en desarrollo continuo y con cambios y actualizaciones constantes. Sus antecedentes, son sobre todo dos: Second Life y OpenSim. Second Life nace en 2003, constando de un cliente o visor Open Source y servidores de pago. En 2007 nace la alternativa libre y abierta a estos servidores, con el nombre de OpenSimulator, o OpenSim. Ese mismo año nace realXtend, partiendo del cliente de Second Life integrando nuevas funcionalidades y modificando el motor de renderizado.

[1] www.realxtend.org

En 2009 se decide reformular toda la plataforma y partiendo de una arquitectura nueva se crea la segunda generación de realXtend. Así en 2010, nacen Naali (zorro ártico) y Taiga (bosque boreal), cliente y servidor respectivamente. Naali tiene una licencia Apache 2 y, aún teniendo menos funcionalidades que el visor anterior, consta de un sistema modular mucho más extensible en el futuro. Taiga es una continuación del servidor basado en OpenSim, con licencia BSD. Tanto el cliente como el servidor son altamente configurables al estar compartiendo un mismo nucleo, pero pudiendo utilizar unos u otros servicios gracias a su arquitectura modular.

El hecho de que el proyecto sea tan reciente, tiene tanto su lado positivo como su lado negativo. Por un lado, está sujeto a grandes cambios y, al no tener una base totalmente definida dado que hay mucho por desarrollar, puede considerarse un proyecto algo inestable. Sin embargo, el software está evolucionando y su comunidad se está haciendo solida. Por ello, al haberse planteado nuevamente todo desde 0 aprovechando los últimos avances técnicos y con una visión futura clara, hace que realXtend vaya a ser la plataforma de desarrollo más importante de los VW.

Las versiones actuales de Naali y Taiga son la 0.3 y la 1.3 respectivamente. A continuación se describen brevemente las características disponibles en estas versiones.

Después se analiza el estado de las comunicaciones en Naali, y finalmente se explica la arquitectura de realXtend, centrandose en el sistema modular.

### 2.3.1. Características

Hay que tener en cuenta, que Naali, además de ser una plataforma de desarrollo de VW escrita en C++, es también un cliente para el jugador o usuario final.

Mucha de su implementación es llevada a cabo mediante librerías externas como Ogre, PoCo, Boost y Qt. Conviene explicar algunos conceptos de Qt ya que el diseño gráfico se realiza en él y el sistema de señales y slots (ranuras) es muy utilizado en todo el proyecto para la comunicación.

Qt es un framework de desarrollo multiplataforma utilizado sobretodo para el desarrollo de interfaces gráficas de usuario (GUI) aunque puede ser utilizado para desarrollar otro tipo de software como consola o servidores.

Utiliza C++ con muchas extensiones no-estándar que son preprocesadas antes de compilar el código. Está preparado para trabajar con bases de datos, para XML, manejo de hilos, soporte de red y manejo de archivos.

Qt puede integrarse en Visual Studio y puede utilizarse mediante el software Qt Designer o Qt Creator para crear interfaces de forma gráfica. Estas interfaces generan su propio código y clases, basandose en widgets. Los widgets son las ventanas que contienen diferente contenido como pueden ser QChekbox, QPushbutton, QCombobox... Información extra y ejemplos de implementaciones pueden verse en su página web[1].

En una aplicación Qt, la comunicación entre objetos se realiza mediante el mecanismo de señales y slots. Esta comunicación es altamente utilizada en todo el proyecto por lo que requiere de una descripción básica para la compresión del sistema.

El comportamiento de lo que ocurre cuando un evento es emitido depende de como se definen las conexiones entre los objetos. Un simple ejemplo es aquel en el que un programador quiere que el widget A haga algo cuando se pulse el botón y por lo tanto la señal **OnButtonClicked** sea emitida. Sin embargo, el widget B no debe hacer nada. Esto se puede manejar de forma muy sencilla, conectando señales de los widgets con slots o métodos que se quieran. De hecho, varias señales pueden activar un slot, y una señal puede activar diversos slots. A continuación se muestra un simple ejemplo:

```
1 // Ejemplo de conexión
2 QObject::connect(accountFileButton, SIGNAL(clicked()),
3  signalMapper, SLOT (map()));
4 // El elemento gráfico accountFileButton emite la señal clicked
      al ser pulsado
5 // lo cual activa el slot map() el cual puede ser un método.
```

Esta señal es emitida automáticamente al hacer clic. Sin embargo, las señales pueden ser emitidas en el código mediante `emit nombreSeñal`. Tal y como se ha comentado, la señal **clicked()** podría ser conectada a su vez a otro slot, y el mismo slot **map()** podría ser conectado a cualquier otra señal. La conexión se realiza en el elemento que emite la señal.

Si nos centramos en las funcionalidades que puede aprovechar este usuario, debemos de atender al desarrollo de Naali. Además de permitir jugar o interactuar

[1] http://qt.nokia.com/

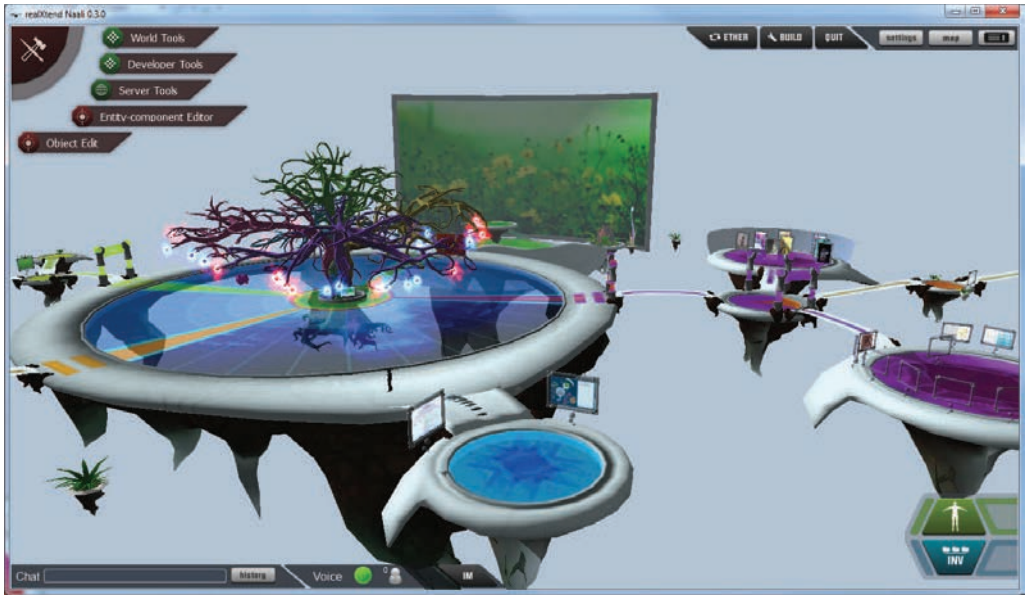


Figura 2.1: Pantalla de Naali 0.3.0 con los diferentes menús y botones. En la parte superior izquierda están las herramientas del mundo, de desarrollo, del servidor, y los editores de entidades y objetos. En la parte superior derecha están los botones para salir o cambiar de mundo, el acceso al constructor/editor, y el menú de configuración junto al mapa y la ventana de notificaciones. En la parte inferior está todo lo relacionado con las comunicaciones en la izquierda y el inventario y el editor de avatar en la parte derecha.

en el mundo, el cliente tiene la característica que permite editarlo y construirlo directamente. El jugador, que puede autenticarse y entrar, ya sea mediante su avatar OpenSim, realXtend o cuenta OpenID, accede a un interfaz con paneles, menús y botones tal y como se ve en la Figura 2.1.

El avatar puede ser modificado mediante un simple editor, donde se pueden añadir texturas o cambiar la composición del cuerpo [2]. También se puede acceder al inventario y subir archivos y utilizarlos en el mundo.

En cuanto a la construcción y edición del mundo, existe un editor de entorno donde tanto el terreno, el agua, el cielo, la niebla o el sol/luz pueden ser alterados. También dispone de un editor de mundo donde se pueden crear objetos, clonarlos, eliminarlos o modificarlos (mover, rotar, escalar, propiedades...). Esto puede verse en la Figura 2.2. También hay un editor de objetos que se puede lanzar desde el menú superior izquierdo, facilitando la adición de mallas.

[2] http://wiki.realxtend.org/index.php/Getting_Started_with_Naali#What_is_Naali

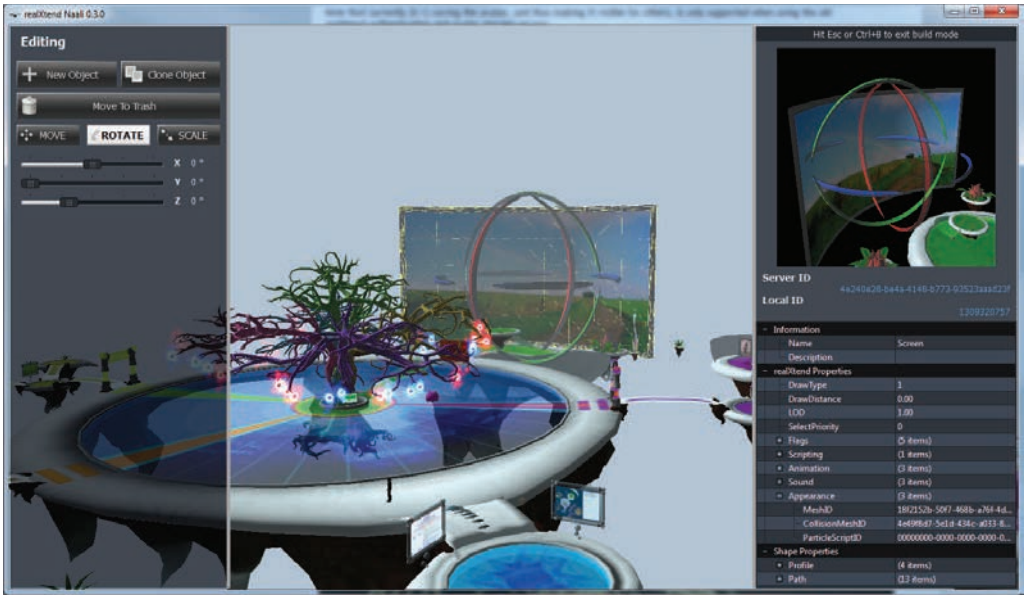


Figura 2.2: Creación de contenido mediante el constructor/editor de mundos

La navegación se efectúa tanto desde teclado como desde el ratón, y existen diversas formas de comunicación: In-World chat o chat interno, mensajería instantánea basada en Jabber/XMPP y voz por IP. En la siguiente sección 2.3.2 se analizarán de manera detallada todas las características relacionadas con las comunicaciones

Las características descritas a alto nivel van evolucionando en cada nueva versión. La ordenación de estas mediante módulos (una funcionalidad corresponde realmente a un módulo, independiente del framework tal y como se explica en la sección de arquitectura) permite que construyamos diferentes tipos de visores, dotando o privando de funcionalidades la misma base. Esto permite, a modo de ejemplo, disponer de una versión de Naali que no permita realizar cambios en el mundo, quitando toda la parte de edición. Podríamos tener por otro lado, la versión con las herramientas de construcción y edición, siendo estas herramientas tanto las desarrolladas como unas nuevas disponible únicamente para los creadores del juego.

### 2.3.2. Análisis de comunicaciones

A la hora de desarrollar un producto, hay un paso previo que consiste en analizar las funcionalidades desarrolladas y ver en qué punto están. Aunque hay

diversas áreas para el análisis como pueden ser las relacionadas con temas de renderizado, de arte, de animación, de físicas, de inteligencia artificial, modelado, etc, a continuación se realiza un análisis de las comunicaciones en realXtend.

### Introducción

La comunicación es un proceso de transferencia de información entre al menos dos entidades en un medio. Los humanos nos comunicamos de formas diferentes, entre ellas estarían el habla (sonoro, con matices como el tono o volumen), el medio escrito, y la comunicación no verbal (gestos, lenguaje corporal o posturas).

Situándonos en un mundo virtual, pretendiendo ser este una representación del real, cabría esperar que los avatares pudiesen comunicarse por medio de este tipo de comunicación, utilizando tecnologías diversas para conseguir ese fin.

Los métodos más populares serían:

- Chat, donde un grupo de gente se comunica mediante la escritura en un lugar común, o privado. Es la forma escrita.
- Voz, utilizando el habla, ya sea mediante el micrófono o por voz sintética generada.
- Comunicación no verbal del avatar, donde pueda expresar mediante gestos o posturas información relativa a su persona.
- Video/imágenes, donde en lugar de ser el avatar sea el propio usuario quien controle los gestos (que pueda verse cara a cara con otros usuarios, mediante una cámara)
- Combinaciones: videoconferencia (voz + imagen), comunicación no verbal a traves de información de una imagen, chat a partir de voz....

En el caso de realXtend, el análisis de los canales de comunicación se centra en los aspectos de Chat, Voz y Videoconferencia, analizando su estado para la versiones 0.3.0 del visor Naali con el servidor Taiga en la versión 1.3.

### Estado de los canales en realXtend

El análisis se centra en el ver el estado actual de:

1. **Audio 3D**: Se considera sonido tridimensional aquel que proporciona sensación espacial, variando la señal dependiendo de la distancia y orientación de emisor y receptores.

2. **Chat In-World**: El chat propio del mundo, es aquel donde diferentes avatares pueden comunicarse mediante un canal público, abierto para todos los participantes del mundo.

3. **TTS**: El Text To Speech, o conversión texto a voz, produce habla partiendo de un texto, sea en modo fichero o cadena de caracteres. La voz puede ser simplemente reproducida o almacenada en un fichero. Puede utilizarse la información fonética para crear animación facial.

4. **Automatic Speech Recognition**: El reconocimiento automático permite comunicarse mediante la voz con una máquina interprete. Así mismo la voz puede ser entendida, traducida a texto y procesada para diferentes motivos como por ejemplo traducir el idioma de entrada y generar voz (TTS) en otro idioma o para dar órdenes.

5. **Correo Interno**: El correo interno brinda la posibilidad de enviar mensajes privados entre usuarios y disponer de un buzón propio. Estos mensajes pueden ser enviados aunque el receptor este offline/desconectado. Al conectarse, el mensaje estará en su buzón personal.

6. **Voz por IP In-World**: La voz por IP puede ser utilizada como el chat, dentro del mundo (In-World voice).

7. **Webcam/Videoconferencia**: Para utilizar imágenes reales en comunicaciones (videoconferencia) o animación y gesticulación de avatares.

8. **Mensajería Instantánea externa**: Servicios externos del tipo Messenger, Jabber, etc.

### Audio 3D

La implementación para el audio 3D está realizada y su funcionamiento es correcto. Utiliza OpenAL [3] al igual que otros muchos juegos (Unreal, Quake, Doom...).

[3] http://connect.creativelabs.com/openal/default.aspx

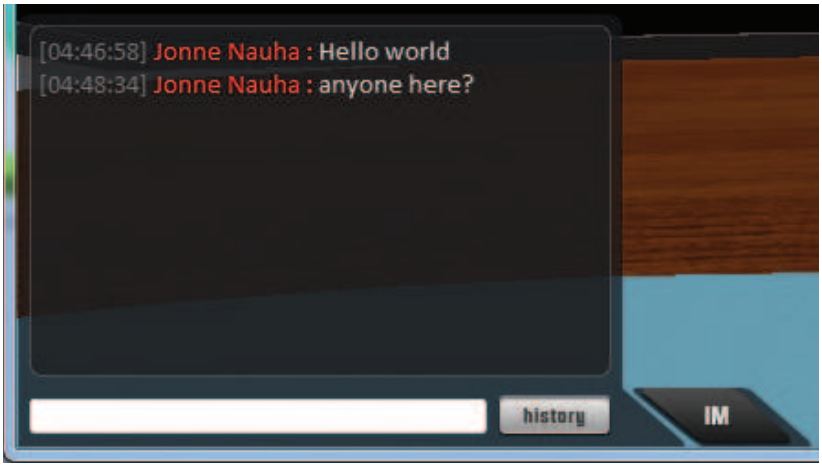


Figura 2.3: Interfaz gráfica de usuario del In-World chat, con el historial

La sensación 3D es correcta, así como la atenuación por distancia. No dispone de bloqueo de sonido por objetos/avatares, ni difracción, o cambios espectrales.

El sonido ha sido testeado con reproducción de ficheros de audio dentro de un mundo.

Un posible uso puede ser el utilizar contenido audible en un entorno de aprendizaje, por ejemplo para oír una exposición sobre un tema, o para utilizar sonidos ambiente, por ejemplo en un zoo donde el objetivo sea aprender los nombres de los animales.

### Chat In-World

El chat In-World viene implementado, con un interfaz de tamaño variable donde se puede introducir texto o ver el historial (donde se registra cualquier entrada de cualquier avatar) tal y como se observa en la Figura 2.3. También se muestra el texto encima del avatar que introduce un texto, mediante una burbuja. Así mismo, se puede crear un log (un fichero donde se almacena toda la conversación).

El chat opera en un entorno definido por una distancia (por defecto 30m) ajustable desde el servidor. Todos los avatares que se encuentren dentro de este radio, pueden tanto ver la conversación como participar en ella.

No dispone de funcionalidades como presencia (saber quién está conectado, o en que estado: ocupado, comiendo...), lista de amigos, o la posibilidad de realizar un chat privado (1 a 1, o grupos seleccionados).

La comunidad prevé implementar estas funciones para finales del 2010. Actualmente se pueden llevar a cabo de forma parcial mediante mensajería instantánea externa.

### Text To Speech

Actualmente no hay nada implementado en cuanto a TTS. No obstante, y aunque la comunidad no tiene intención de implementar esta función a corto plazo, aparece como requerimiento en la página de diseño [4]

La funcionalidad de un TTS dentro de un mundo virtual puede ser muy útil ya que cualquier texto puede ser pasado a formato audible. Por ejemplo, el chat podría ser oído, las notificaciones del mundo, cualquier contenido de texto como una lección, o una explicación de cualquier contenido...

### Automatic Speech Recognition

Actualmente no hay nada desarrollado, y tampoco tienen previsto realizar ningún trabajo en este ámbito.

### Mensajería instantánea externa

La mensajería instantánea (Por ejemplo XMPP, Jabber, AIM, ICQ, MSN, GTalk, Facebook...) está integrada en la plataforma mediante el protocolo XMPP, utilizado por alguno de este tipo de servicios. De esta forma, la comunicación con el resto de usuarios de estas redes es posible, sin que sea necesario que estén conectados mediante Naali, sino cualquier cliente que lo soporte.

En la Figura 2.4 puede observarse una conexión al servicio de Google, Gmail, con una lista de amigos con las que teoricamente se pueden establecer chats privados, escritos y de voz, así como realizar videoconferencias.

A la práctica no está todo implementado tal y como se describe en cada apartado.

[4] http://wiki.realxtend.org/index.php/NG_Design_Document/Technical_Requirements#Sound

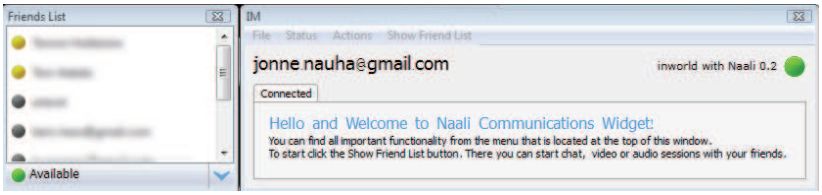


Figura 2.4: In-world voice activado

### Correo Interno

En la versión actual no existe esta funcionalidad, aunque en OpenSim la implementación de IM (presencia, lista, etc.) permite que un usuario envíe un mensaje a otro aunque este se encuentre desconectado [5]. Al conectarse, el usuario recibirá un mensaje con el texto que le ha sido enviado.

La implementación a finales de año de esta funcionalidad, traerá previsiblemente esta opción de correo interno, que de ser insuficiente, necesitaría un desarrollo más concreto creando bandejas de entrada y salida y un sistema más parecido al correo electrónico.

### Voz por IP In-World

La voz In-World está implementada mediante Mumble [6]. Mumble es un software de chat de voz pensado para juegos, de código abierto y baja latencia, ofreciendo una muy buena calidad.

Funciona correctamente aunque con una interfaz algo escasa, y un funcionamiento extraño a veces. Al igual que con el In-World chat, a cierta distancia la voz del resto de avatares deja de oírse (aunque esta opción podrá ser deshabilitada próximamente). Pero estando cerca, se pueden oír a diferentes avatares en 3D siempre y cuando estos activen el micrófono desde una pequeña GUI y no se los silencien mediante el checkbox.

En la siguiente Figura 2.5 se puede observar como el chat de voz está activado (el botón está de color verde) y se aprecia la lista de participantes en una nueva ventana.

La comunidad ha creado módulos, separando los actuales de comunicaciones así como adecuando el visor para no depender del cliente de Mumble (necesario hasta la versión 0.2.1).

[5] http://opensimulator.org/wiki/Offline_Messaging
[6] http://mumble.sourceforge.net/

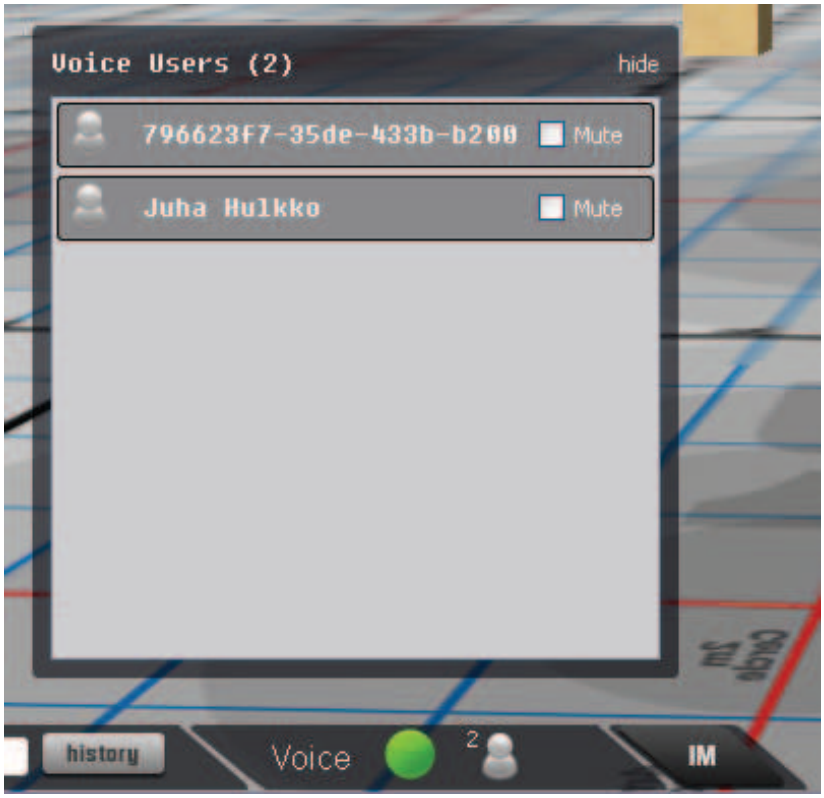


Figura 2.5: In-world voice activado.

En el roadmap tienen previsto hacer el servidor mumble escalable para tener una o más regiones por servidor, y para utilizar varios servidores. Mantienen cuestiones abiertas como el filtrado de streams innecesarios (por ejemplo dos usuarios que por distancia no se oyen pero generan audio). El acceso al servidor no es controlado de momento, de forma que cualquier conexión proveniente de fuera de un cliente Naali puede convivir con las conexiones de realXtend.

En este caso, además de mejorar la interfaz gráfica conviene añadir funcionalidades, en la misma forma que en el In-World chat, para crear conversaciones privadas, salas, etc. Estas funcionalidades están soportadas en Mumble, pero su integración no se ha llegado a completar en realXtend. También se comenta [7] la idea de añadir Lipsync.

En las primeras versiones, se podían utilizar algunas de estas funciones mediante mensajería instantánea externa [8] aunque en la actualidad falla mucho, y las comunicaciones de voz se centran en Mumble.

[7] http://wiki.realxtend.org/index.php/Technology_for_multi_user_voice_chat
[8] http://www.youtube.com/watch?v=4iiE66hY-RU

### Imagenes/Vídeo

No existe el In-world video o similar, aunque existía la videoconferencia mediante el protocolo Jaber/XMPP/Jingle mediante mensajería externa. En la actualidad no funciona.

En este punto comentar que Ludocraft, una de las empresas involucradas en realXtend, trabaja en el tema de animación facial a través de la captura y seguimiento mediante webcam. [9]

### Conclusión

Tras el análisis de diversas áreas, es hora de detectar los aspectos menos desarrollados con el objetivo de priorizar qué aspectos a desarrollar se abordarán en el proyecto

La conclusión que se puede extraer del análisis de comunicaciones es que al ser un proyecto tan nuevo, aún hay mucha funcionalidad por implementar. Hay aspectos que no se han comenzado a elaborar como pueden ser el tema del TTS o del ASR, y otros que poco a poco van cogiendo forma como pueden ser el chat, la voz por IP o el tema del vídeo, heredados del visor anterior. Por lo tanto, es previsible que en estos campos, el desarrollo se vaya cerrando en próximas versiones. Sin embargo, no se incluirán aún ni el TTS ni el ASR.

Ambos servicios pueden ser muy útiles a la hora de aprovechar las posibilidades de los mundos virtuales. Si nos centramos en un entorno de aprendizaje, un ASR puede servir para practicar un idioma, como corrector de pronunciación, para el dictado de textos, para la ejecución de ordenes al avatar (siéntate, camina, etc.)

A su vez, un TTS puede ser muy útil para escuchar textos, cuentos, ejercicios, conversaciones o cualquier tipo de texto.

Como la comunidad tenía prevista la inclusión de algún tipo de TTS, y las aplicaciones como la generación de contenido audible a partir de un texto son prioridades para la plataforma de ENNE, la integración con un sistema TTS es detectada como una de las necesidades prioritarias.

[9] http://www.mail-archive.com/realxtend@googlegroups.com/msg00786.html

### 2.3.3. Arquitectura modular

Para saber como se puede integrar cualquier servicio externo al proyecto realXtend, como puede ser un TTS, es necesario conocer la arquitectura de la plataforma.

En primer lugar, hay que separar la parte cliente Naali del servidor Taiga. Cualquier integración puede llevarse tanto en el lado cliente, en el servidor o en ambos, según sea necesario.

#### Naali

El visor Naali pretende ser el estándar entre las plataformas de Mundos Virtuales. El diseño requiere robustez, flexibilidad y ser fácilmente extensible. Por ello, este diseño modular es la base de su arquitectura [1].

La modularidad permite añadir nuevas características y funcionalidades a los objetos existentes, o modificar el comportamiento de ellos sin la necesidad de cambiar otras partes del sistema. Estas características son requeridas a menudo por terceros con alguna necesidad especifica. Con un sistema modular, esta tarea queda reducida a la implementación del plug-in o módulo y sus correspondientes submódulos, sin la necesidad de estudiar todo el sistema.

Basándolo en plug-ins, activar, desactivar, o intercambiar una funcionalidad es más sencillo. Esto permite ofrecer un servicio añadido sustituyendo uno de los módulos que pueda ser insuficiente para un determinado uso, por otro módulo de una forma sencilla y cómoda. Pero esto no quiere decir que el sistema sea simple, ni que implementar funcionalidades sea fácil. Las nuevas partes deben de comunicarse con la parte existente de forma claramente definida y con robustez. Esto se hace a través de un mecanismo de mensajes y eventos, tal y como se ve en la Figura 2.6

En el mundo de los juegos, la eficiencia es otro de los puntos fuertes de diseño, incluso más en juegos inmersivos, expansivos y muy potentes visualmente. Sin embargo, en plataformas de Mundos Virtuales tiene más peso el hecho de poder construir una plataforma para muchas aplicaciones de diferentes tipos. Un equilibrio entre ambos conceptos es requerido. El hecho de que el sistema sea modular implica una bajada de eficiencia, que se compensa por el beneficio que acarrea la adición o eliminación de funcionalidades.

[1] http://wiki.realxtend.org/index.php/NG_Design_Document/Viewer_Architecture/Framework

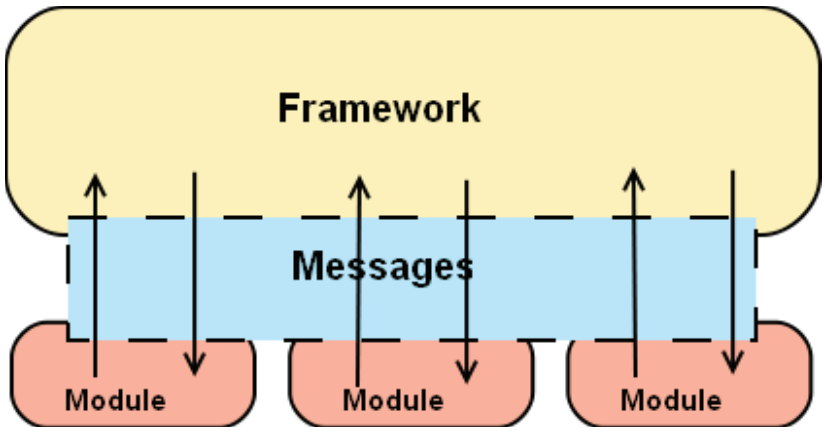


Figura 2.6: Interacción entre diferentes módulos por medio de mensajes con el framework por medio. Cuando un módulo quiere comunicarse con otro, el framework hace de intermediario y se comunica con las partes implicadas. Esto hace que un módulo no tenga porque conocer nada del otro, ya que todo el servicio que ofrece este, lo ofrece el framework por él. Imagen tomada desde la página de realXtend.

Para mejorar la eficiencia, el diseño tiene en cuenta que los procesadores son y serán multinúcleo, permitiendo ejecuciones de tareas en paralelo.

La plataforma por tanto es modular en los términos arriba expresados, pero además, es modular al estar basado en componentes (ECA: Entity, Component, Attribute), como muchos juegos MMOG. Aquí, cualquier objeto o entidad del mundo virtual, es un ente vacío que conforma su identidad dependiendo de los componentes que le sean agregados y los atributos de estos. Se pueden agregar diferentes componentes a las entidades, creando una red como puede verse en la Figura 2.7. De esta forma, se crean unos componentes ligeros, con complejidad reducida y por lo tanto produciendo menos fallos o *bugs* al poder analizar los componentes por separado.

Un aspecto negativo de esta arquitectura es que se introduce una mayor carga y se requiere la comprobación de presencia de los componentes.

La implementación de módulos y componentes está relacionada, y su implicación mayor es que el conjunto es un sistema muy flexible y adaptable que permite crear soluciones muy adaptadas a las necesidades.

Un módulo puede cubrir diferentes necesidades. Por un lado, puede tener una determinada función, y cubrir una necesidad concreta de forma que no sea utilizada más que por el core del software.

Por otro lado, un nuevo módulo (A) puede servir a otros módulos, si ofrece

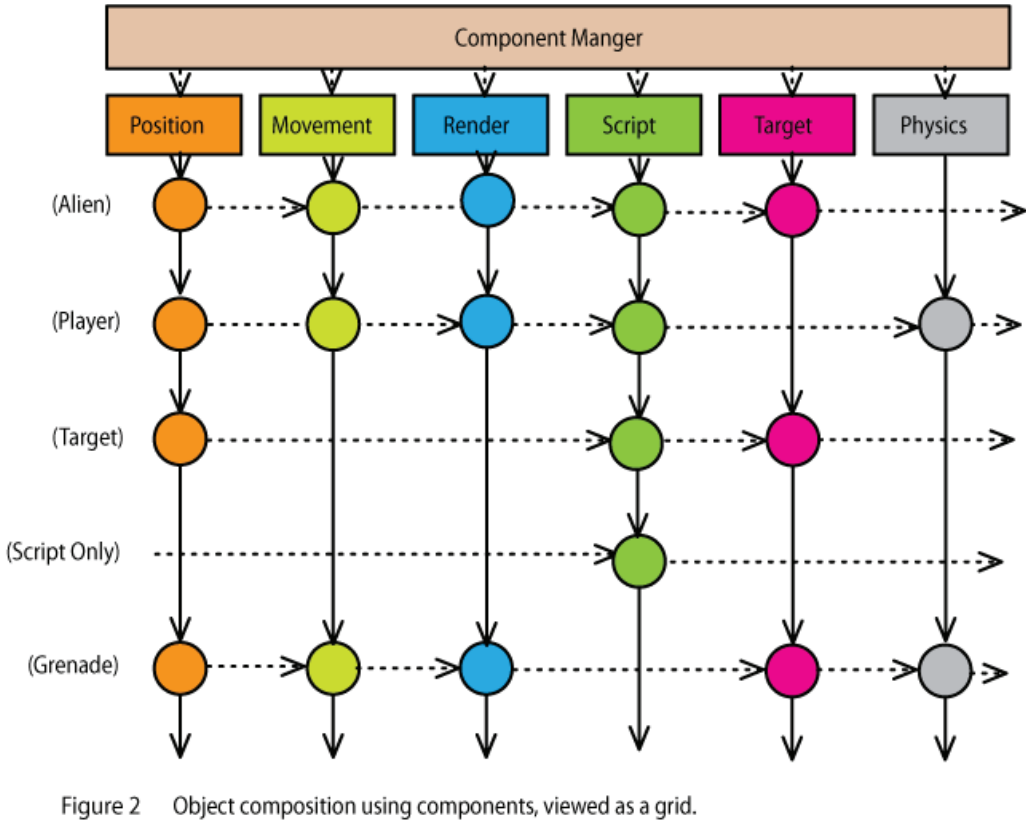


Figura 2.7: Una entidad, puede contener un conjunto de componentes con unos determinados atributos, que hacen que esta entidad tome personalidad propia. Imagen tomada desde la página de realXtend.

sus servicios. Para ello, es necesario que estos servicios sean registrados en el framework. De esta forma, el framework conoce las posibilidades de este módulo y puede ofrecerlas al resto de módulos.

Cuando otro módulo (B) necesite utilizar el módulo A, preguntará al framework si dispone del servicio de A. Si es así, el framework ofrece este servicio, y el módulo B puede acceder a los métodos que A le ha ofrecido al framework sin depender de él.

De esta forma, se crean unas interfaces para cada módulo que ofrece servicios, registrados con el manejador de servicios o service manager y con una interfaz heredada de la clase ServiceInterface.

En la Figura 2.8 puede verse un ejemplo. En este caso se observa como los módulos 'cuelgan'de sus interfaces (un archivo de cabecera donde están definidos

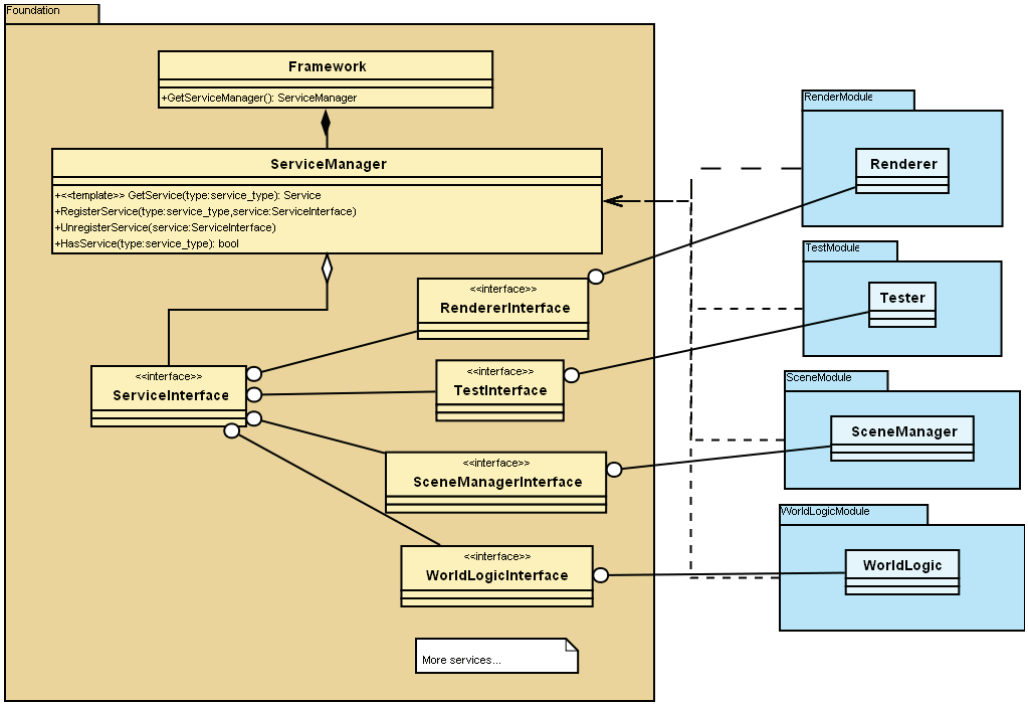


Figura 2.8: Diagrama de servicios. El ServiceManager se encarga de registrar los servicios que ofrecen los diferentes módulos. Estas interfaces se crean para cada módulo, y es en estos donde se encuentran los métodos implementados. Todos los servicios registrados son accesibles por otros módulos. Imagen tomada desde la página de realXtend.

todos los métodos). De esta forma, el framework contiene todos los servicios aunque estos están implementados en los módulos.

Además de utilizar los módulos mediante sus servicios, también se puede acceder a ellos directamente o utilizar su API, pero esto crea dependencias y conocer el funcionamiento del módulo por lo que tiene que evitarse si es posible.

Los componentes son ligeros contenedores de comportamiento y datos específicos de entidades. Se recomienda que los componentes minimicen la cantidad de comportamiento a definir dejando que los módulos controlen el comportamiento según los datos que contiene el componente.

El framework está diseñado de forma que puedan ser introducidos nuevos componentes a los objetos del mundo y reemplazar los viejos fácilmente. Todos los componentes tienen una interfaz común (ComponentInterface) donde se implementa las funciones y herramientas comunes a todos los componentes, necesarias para su utilización.

Si se quieren utilizar los módulos mediante ellos, pueden ser declarados y añadir sus funcionalidades a entidades.

De esta forma, es responsabilidad de los módulos registrarlos en el framework por medio del *component factory*. El framework incluye macros para facilitar el registro de los componentes de forma implícita, logrando que este registro se de cuando el módulo sea inicializado.

Una vez hecho esto, los componentes pueden ser agregados a entidades y estas gozan de los servicios ofrecidos por la interfaz del módulo.

### Taiga

Taiga no es una plataforma construida desde cero como Naali. Realmente, más que una plataforma es un framework basado en OpenSim. OpenSim es un proyecto OpenSource construido mediante ingeniería inversa del servidor de Second Life, implementado en C#. A lo largo de su desarrollo, se ha convertido en un servidor con diferentes protocolos de Mundos Virtuales. Su diseño está basado también en módulos y dispone de una comunidad muy fuerte.

RealXtend utiliza de momento un protocolo basado en SLUDP (Second Life UDP) y el módulo ModreX, que permite utilizar servidores OpenSim mediante realXtend aprovechando el potencial de Ogre3D, scripting en Python, y otras opciones. Además de los proyectos de código abierto OpenSim y ModreX, Taiga incorpora servicios de autenticación o integración (inventario). En cuanto al protocolo, el grupo kyoryoku (en el cual se encuentra reX) está investigando el futuro de los protocolos para los Mundos Virtuales, con lo que pueden producirse cambios en un futuro.

Cable Beach, implementado en C# e integrado en OpenSim es un estándar que permite conectar usuarios, mundos virtuales, servicios de confianza y terceros. La identidad es tratada de forma separada teniendo que ser autorizado antes de utilizar datos propios del usuario y sin enviar datos confidenciales a terceros desconocidos (*untrusted*). Se puede considerar como un protocolo de entrada de identificación, organizando grupos de forma segura, escalable, y fácil de administrar.

La autenticación utilizando OpenID, estandar de identidad distribuida, también está contemplada, tal y como se muestra en la Figura 2.9.

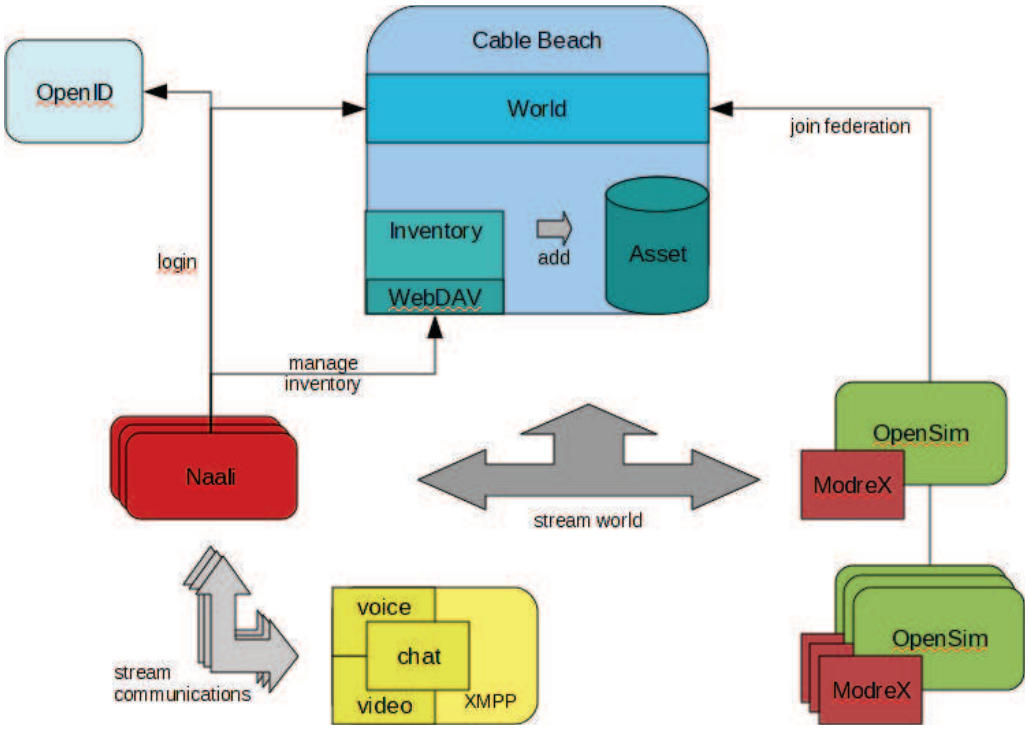


Figura 2.9: El visor Naali en comunicación con el servidor Taiga, compuesto de OpenSim+Modrex, CableBeach y OpenID. También ser observa la comunicación con otros servicios como el chat o la voz por XMPP. Imagen tomada desde la página de realXtend.

# Capítulo 3

# Conversor texto a voz

## 3.1. Introducción

Como el propio nombre indica el text-to-speech-system *TTS System* (sistema conversor de texto a voz) genera una voz artificial o sintética partiendo de una entrada de texto. Este texto puede ser obtenido de diversas fuentes, como páginas web, ficheros de texto u otros.

EL sistema TTS, requiere tanto de procesado del lenguaje como de procesado acústico-linguístico. Esto hace que el conversor se descomponga en dos partes: el *front-end* o *Natural Language Processing* que partiendo del texto genera una representación lingüistica fonética y el *back-end* o *Digital Signal Processing* que parte de esta representación para sintetizar voz, mediante procesador de señal. Este diagrama de bloques puede verse en la Figura 3.1.

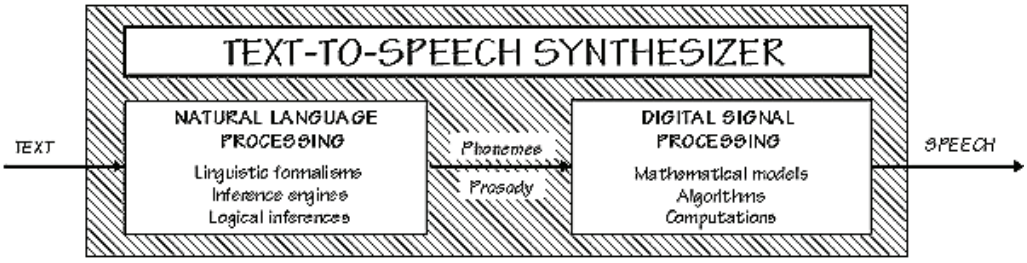


Figura 3.1: Diagrama de bloques básico de un TTS

### 3.1.1. Front-end

El front-end o procesador de lenguaje natural es el primer bloque del TTS. Por lo tanto, parte del texto que tiene que procesar y sigue una serie de acciones hasta conseguir una información de salida válida para el segundo bloque. Al estar separados, estos bloques pueden funcionar de forma independiente permitiendo así que un front-end determinado funcione con diferentes back-ends o viceversa.

Lo primero que debe hacer el front-end es convertir partes problemáticas como números y abreviaturas en palabras equivalentes. Este proceso se llama a menudo normalización de texto o pre-procesado. Una vez hecho esto, se asigna una transcripción fonética a cada palabra y se divide y marca el texto en varias unidades prosódicas, como frases y oraciones. El proceso de asignar transcripciones fonéticas a las palabras recibe el nombre de conversión de texto a fonema (TTP en inglés) o de grafema a fonema (GTP en inglés). La combinación de transcripciones fonéticas e información prosódica constituye la representación lingüística fonética.

Esta salida puede pasar directamente al siguiente bloque o puede ser almacenada en un archivo de una forma determinada por el procesador de señal, usualmente con la extensión .PHO.

### 3.1.2. Back-end

El back-end, se encarga de crear la onda sintetizada tomando como entrada la representación simbólica. Mediante modelos y algoritmos de una determinada tecnología de síntesis es capaz de generar una voz artificial con ciertas características (más o menos limitada por la tecnología) que puede ser a menudo reproducida o almacenada como fichero de audio.

## 3.2. Tecnologías de síntesis

Centrándonos en el back-end, existen diferentes tecnologías para la síntesis de voz. Existen dos tecnologías principales para generar habla: síntesis concatenativa y síntesis de formantes.

### 3.2.1. Síntesis concatenativa

La síntesis concatenativa se basa en la concatenación de segmentos de voz grabados. Generalmente, la síntesis concatenativa produce los resultados más naturales. Sin embargo, las diferencias entre la variación natural del habla y las técnicas automatizadas de segmentación de formas de onda resultan en defectos audibles, que conllevan una pérdida de naturalidad.

Hay tres tipos básicos de síntesis concatenativa: por selección de unidades, de difonos[1] y específica para un dominio.

#### Síntesis por selección de unidades

Utiliza una base de datos de voz grabada (más de una hora de habla grabada). Durante la creación de la base de datos, el habla se segmenta en algunas o todas de las siguientes unidades: fonemas, sílabas, palabras, frases y oraciones. Se crea un índice de las unidades en la base de datos basada en parámetros acústicos de la segmentación como la frecuencia fundamental, el pitch, la duración, la posición en la sílaba y los fonemas vecinos. En tiempo de ejecución, el objetivo deseado se crea determinando la mejor cadena de candidatos de la base de datos.

La selección de unidades permite una elevada naturalidad ya que prácticamente no requiere procesamiento digital de las señales de voz del corpus. El principal problema de calidad puede ocurrir en la concatenación de segmentos de voz, ya que pueden aparecer artefactos debidos a discontinuidades en algunas características del habla, como el espectro o la entonación. Algunos sistemas utilizan técnicas de procesamiento de la señal para suavizar este tipo de discontinuidades.

De hecho, la salida de la mejor selección de unidades es a menudo indistinguible de la voz humana real, especialmente en contextos en los que el sistema ha sido adaptado. Por ejemplo, un sistema de síntesis de voz para dar informaciones de vuelos puede ganar en naturalidad si la base de datos es construida a base grabaciones de informaciones de vuelos, pues será más probable que aparezcan unidades apropiadas e incluso cadenas enteras en la base de datos. Sin embargo, la máxima naturalidad a menudo requiere que la base de datos sea muy amplia, llegando en algunos sistemas a los gigabytes de datos grabados.

[1] Un difono representa el sonido que abarca desde la mitad de la realización de un fonema hasta la mitad de la realización del fonema siguiente. La síntesis consiste, entonces, en la concatenación de segmentos de señal en el tiempo, siendo los segmentos difonos.

#### Síntesis de difonos

La síntesis de difonos usa una base de datos mínima conteniendo todos los difonos que pueden aparecer en un lenguaje dado. El número de difonos depende de la fonotáctica del lenguaje: el castellano tiene unos 800 difonos, el alemán unos 2500. En la síntesis de difonos, la base de datos contiene un sólo ejemplo de cada difono. En tiempo de ejecución, la prosodia de una oración se sobreimpone a estas unidades mínimas mediante procesamiento digital de señales, como codificación lineal predictiva lineal (LPC)[1], PSOLA o MBROLA [8] [9].

La calidad del habla resultante es generalmente peor que la obtenida mediante selección de unidades pero más natural que la obtenida mediante sintetización de formantes. La síntesis de difonos adolece de los defectos de la síntesis concatenativa y suena robótica como la síntesis de formantes.

Esta tecnología tiene pocas ventajas a parte del pequeño tamaño de la base de datos, así que su uso en aplicaciones comerciales experimenta un declive, aunque continúa usándose en investigación porque hay unas cuantas implementaciones libres.

#### Síntesis específica para un dominio

La síntesis específica para un dominio concatena palabras y frases grabadas para crear salidas completas. Se usa en aplicaciones donde la variedad de textos que el sistema puede producir está limitada a un particular dominio, como anuncios de salidas de trenes o información meteorológica.

Esta tecnología es muy sencilla de implementar, y se ha usado comercialmente durante largo tiempo: es la tecnología usada por aparatos como relojes y calculadoras. La naturalidad de estos sistemas puede ser muy grande, porque la variedad de oraciones está limitada y corresponde a la entonación y la prosodia de las grabaciones originales. Sin embargo, al estar limitados a unas ciertas frases y palabras de la base de datos, no son de propósito general y sólo pueden sintetizar la combinación de palabras y frases para los que fueron diseñados.

### 3.2.2. Síntesis de formantes

La síntesis de formantes no usa muestras de habla humana en tiempo de ejecución. En lugar de eso, la salida se crea usando un modelo acústico. Parámetros

como la frecuencia fundamental y los niveles de ruido se varían durante el tiempo para crear una forma de onda o habla artificial.

Muchos sistemas basados en síntesis de formantes generan habla robótica y de apariencia artificial, y la salida nunca se podría confundir con la voz humana. Sin embargo, la naturalidad máxima no es siempre la meta de un sintetizador de voz, y estos sistemas tienen algunas ventajas sobre los sistemas concatenativos.

La síntesis de formantes puede ser muy inteligible, incluso a altas velocidades, evitando los defectos acústicos que pueden aparecer con frecuencia en los sistemas concatenativos. La síntesis de voz de alta velocidad es a menudo usada por los discapacitados visuales para utilizar computadores con fluidez. Por otra parte, los sintetizadores de formantes son a menudo programas más pequeños que los sistemas concatenativos porque no necesitan una base de datos de muestras de voz grabada. De esta forma, pueden usarse en sistemas embebidos, donde la memoria y la capacidad de proceso son a menudo exiguas. Por último, dado que los sistemas basados en formantes tienen un control total sobre todos los aspectos del habla producida, pueden incorporar una amplia variedad de tipos de entonaciones, que no sólo comprendan preguntas y enunciaciones.

### 3.2.3. Otras tecnologías

Dentro de otras tecnologías, menos usadas, dispondríamos de la articulatoria, la hibrída y la basada en modelos ocultos de Markov (HMM en inglés).[2]

La síntesis articulatoria se basa en modelos computacionales del tracto vocal y el proceso de articulación. Pocos de los modelos son suficientemente avanzados o eficientes computacionalmente para ser usados en sistemas comerciales de síntesis de voz. Una excepción notable *gnuspeech*, que proporciona una conversión texto a voz articulatoria completa mediante una analogía de guía de onda o línea de transmisión de los tractos vocal y nasal humanos, controlados por el Modelos de Región Distintiva de Carré que está basado en el trabajo de Gunnar Fant y otros del laboratorio Stockholm Specch Technology Lab del Royal Institute of Technology sobre el análisis de la sensibilidad de formantes. Este trabajo mostró que los formantes en un tubo resonante pueden ser controlados por sólo ocho parámetros que corresponden a los articuladores disponibles en el tracto vocal humano natural.

[2] http://hts.sp.nitech.ac.jp/

La síntesis híbrida aúna aspectos de las síntesis concatenativa y de formantes para minimizar los defectos acústicos cuando se concatenan segmentos.

En la síntesis HMM, el habla, contenida por su espectro de frecuencias (tracto vocal), frecuencia fundamental (fuente vocal), y la duración (prosodia) se modelan simultáneamente por modelos ocultos de Márkov. Las formas de onda se generan desde estos modelos ocultos de Márkov mediante el criterio de máxima verosimilitud.

## 3.3. Análisis y estado del arte

### 3.3.1. Análisis de requisitos

A la hora de analizar las diferentes alternativas de conversores texto a voz, hay que fijar ciertos requisitos para realizar una comparativa. En cuanto a estos requisitos, hay algunos indispensables y otros a valorar. Los requisitos mínimos que se imponen son:

- Plataforma de código libre o licencia similar.
- Inglés y castellano soportados.
- Sistema operativo Windows soportado.

Por otro lado, se también se tiene en cuenta y se analiza:

- Tecnología de síntesis y calidad.
- Arquitectura (Cliente/Servidor).
- Lenguaje de programación.
- Sistemas Operativos soportados.
- Número de voces y su calidad.
- Opciones de salida (principalmente WAV y PHO)
- Disponibilidad de API o facilidad de acceso.

Partiendo de estos requisitos, los productos encontrados han sido:

1. eSpeak: `http://espeak.sourceforge.net/`.

2. Festival: http://festvox.org/festival/.
3. Festival-Lite (Flite): http://www.speech.cs.cmu.edu/flite/index.html.
4. Mbrola: http://tcts.fpms.ac.be/synthesis/mbrola.html (aunque es solo el sintetizador, es importante analizarlo al ser compatible con casi todos los TTS)
5. EULER: http://tcts.fpms.ac.be/synthesis/euler/.
6. FreeTTS: http://sourceforge.net/projects/freetts.

Por otro lado, al no cumplir los requisitos, han quedado fuera:

- MARY: http://mary.dfki.de/ No esta disponible en Castellano de momento.
- NeXTenS: http://nextens.uvt.nl/ Holandés, basado en Festival/Mbrola.
- HTS-Engine: http://hts.sp.nitech.ac.jp/ No es un TTS, se puede usar con Festival o MARY.

### 3.3.2. TTS OpenSource

#### eSpeak

eSpeak es un un sintetizador de voz de Código Abierto (GNU General Public License 3), que puede ser utilizado tanto en Linux como Windows (aunque se ha podido utilizar en MAC y Solaris).

Utiliza síntesis de formantes, obteniendo como resultado un programa compacto y reducido. Aunque la voz es clara y puede ser utilizada a altas velocidades, no es tan natural como los sistemas concatenativos (los cuales utilizan grabaciones reales).

Tiene una sola voz en castellano (por defecto masculina) y 9 voces inglesas (por defecto masculinas) si bien eSpeak puede exportar a *.PHO para las voces en ingles o castellano de Mbrola. Las voces propias de eSpeak pueden ser modificadas (variando los parámetros) consiguiendo de cada voz 5 variantes masculinas (+m1, +m2...) y 4 femeninas (+f1....).

Está disponible como línea de comandos (partiendo de una cadena de texto, un archivo de texto, HTML o entrada con mnemónicos fonéticos) y puede reproducir

el audio directamente o guardar el resultado en formato WAV. También dispone de otras versiones como librería (DLL), versión SAPI5, o código fuente en C.

Para utilizar eSpeak, hace falta descargar la versión adecuada desde http://espeak.sourceforge.net/download.html y compilar o instalar. En versión línea de comandos, la estructura a seguir es:

```
1 espeak -v idioma "texto"
2 espeak idioma -w archivo.wav "texto"
3 espeak -v idioma -f archivo.txt
4 espeak -v es -w "ejemplo-espeak.wav" -f ejemplo.txt -s 160 -g 5
```

Pudiendo obtenerse el resultado audible, o en formato WAV (o ambos a la vez). Dispone de muchas opciones, como ajustes de nivel, pitch, duración, pausas, entrada de archivos en UTF8, 8 bits o 16 bits, interprete de SSML, compilación de normas y diccionarios, o la escritura de archivos *.pho.

Para utilizar voces Mbrola, hace falta descargar las bases de datos de las voces Mbrola y descomprimir el archivo en la carpeta espeak-data de espeak. Por otro lado, hace falta Mbrola, para leer los archivos *.pho.

Una vez instaladas las voces se puede exportar tanto texto como archivos siguiendo el siguiente comando –phonout y utilizando las voces de Mbrola.

```
1 espeak -v mb-de2 --phonout=mypho.pho "hello"
```

Una muestra (la única disponible en la web oficial) es 'The Raven'de Edgar Allen Poe: http://espeak.sourceforge.net/samples/raven.ogg

### Festival

Festival Speech Synthesis System es un TTS[4] de Código Abierto desarrollado por la Universidad de Edinburgo en C++. Es un sistema modular y soporta tanto Castellano como Inglés. Ofrece síntesis concatenativa de difonos aunque se ofrecen diferentes opciones como selección de unidades, HTS (HMM-based Speech Synthesis System) , Clustergen o voces Mbrola.

Acaban de sacar la versión 2.1 tras unos años sin actualizar. De todas formas, el proyecto no ha parado durante este tiempo ya que sobre todo han seguido con el proyecto FestVox para la creación de nuevas voces. Algunas de estas voces son de buena calidad, habiendo incluso voces en castellano notablemente mejores que la que trae por defecto tanto masculinas como femeninas [3]. Además de FestVox

[3] http://forja.guadalinex.org/repositorio/frs/?group_id=21

(donde hoy en día están las noticias y la comunidad), que está en constante desarrollo, Festival se sitúa como un buen front-end al ser compatible con el otro gran proyecto de voces como es Mbrola.

El proyecto incluye la documentación completa para desarrollar sistemas de síntesis de voz con varios APIs, siendo un entorno ideal para el desarrollo e investigación de las técnicas de síntesis de voz.

El proyecto está escrito en lenguaje C++ y está implementado como un interprete de comandos el cual puede conectarse con diversos módulos y aplicaciones.

Además existen librerías para el desarrollo de aplicaciones en los lenguajes Java y C++ , así como un interfaz para el editor de textos Emacs.

Las herramientas y la documentación completas para la utilización de nuevas voces en el sistema están disponibles en el proyecto FestVox. El proyecto FestVox pretende hacer de la construcción de voces sintéticas nuevas un proceso más sistemático y mejor documentado, haciendo lo posible para que cualquiera pueda construir nuevas voces. El Proyecto FestVox es distribuido bajo la misma licencia de software libre, similar a la Licencia MIT-X11.

Hay una versión Demo online (en inglés) en la siguiente dirección: http://www.cstr.ed.ac.uk/projects/festival/onlinedemo.html.

El software se ofrece en código fuente que ha de ser compilado, o utilizar una versión compilada para Windows http://www.eguidedog.net/doc_build_win_festival.php. También funciona en Linux y Mac OS. Además se ofrece como librería C++, interprete de Scheme, Shell, Java y Emacs.

Festival puede implementarse como cliente o como servidor (http://festvox.org/docs/manual-1.4.3/festival_toc.html) aunque en la página no recomiendan usar Festival en modo servidor en entornos no seguros porque parece que puede tener fallos de seguridad.

La utilización [4] de Festival es similar a Mbrola o Speak. Partiendo de una versión compilada, se instala Festival en un directorio al cual se puede acceder desde la linea de comandos o haciendo clic en el ejecutable. Aunque tiene ayuda (-h) la utilización básica es escribir

```
1 (SayText "Texto a decir")
```

Para cambiar el idioma es necesario instalar los que se quieran en la carpeta *voices* del directorio de instalación y escribir en Festival :

[4] Tutorial de la versión antigua: http://www.cstr.ed.ac.uk/projects/festival/manual

```
(voice_nombredelavoz)
```

Donde el nombre de la voz para utilizar la voz por defecto en castellano es:

```
(voice_el_diphone)
```

Para instalar voces hace falta descargar el archivo de la voz deseada y descomprimirlo en la carpeta de las voces (`http://festvox.org/packed/festival/2.0.95/` por ejemplo ofrece varias voces en inglés, o `http://festvox.org/packed/festival/1.95/` ofrece `el_diphone` en castellano (`festvox_ellpc11k.tar.gz`).

Para leer un archivo de texto se escribe:

```
(tts "archivo.txt" nil)
```

### Flite

Esta es una versión reducida de Festival (Festival-Lite) pensada tanto para servidores como para sistemas embebidos, siendo una alternativa a Festival, utilizando voces de FestVox.

Está escrito en C, precisamente para reducir espacio y agilizar el software y se distribuye como software libre.

### Mbrola

Mbrola (Multiband Resynthesis OverLap-Add) es un sintetizador basado en la concatenación de difonos, produciendo una salida de audio de 16 bits (WAV, AIFF, RAW). Al no aceptar texto como entrada, necesita una lista de fonemas junto con la información prosódica, por lo que es un sintetizdor o back-end (archivo *.pho).

La licencia de Mbrola permite su utilización libre para usos no comerciales y no militares, por lo que no es exactamente una licencia de Software Libre. El proyecto lo lidera el TCTS Lab de la Faculté Polytechnique de Mons (Bélgica) y su objetivo es obtener un sintetizador para muchas lenguas, ayudando además la investigación en la síntesis de voz, especialmente en la generación prosódica.

En cuanto a los idiomas, la lista es muy extensa, pero centrándonos en el inglés y castellano, tenemos 4 voces en inglés (3 de EEUU, siendo una de ellas femenina, y 1 de Inglaterra) y 2 en castellano (ambas masculinas).

- Demo en inglés británico: `http://tcts.fpms.ac.be/synthesis/mbrola/demo/en1.wav`

- Demos en castellano:
  http://tcts.fpms.ac.be/synthesis/mbrola/demo/es1.wav
  http://tcts.fpms.ac.be/synthesis/mbrola/demo/es2.wav

El funcionamiento es sencillo, partiendo de el archivo binario. La versión de Windows puede ser descargada desde http://tcts.fpms.ac.be/synthesis/mbrola/bin/pcdos/mbr301d.zip aunque existen versiones para otros SO como MAC, Linux o Symbian.

El código fuente no está disponible, ya que la intención del proyecto es que terceros adapten sus bases de datos de difonos al formato Mbrola. Sin embargo, si se quiere obtener el código para una aplicación comercial, se puede firmar un contrato con la empresa Babel Technologies.

Una vez se descarga el binario, es necesario descargar las bases de datos de los idiomas de interés desde http://tcts.fpms.ac.be/synthesis/. Los idiomas se descomprimen en el mismo directorio de Mbrola.

El propio *readme.txt* explica su funcionamiento, aunque básicamente se trata de situarnos en la carpeta de Mbrola en la consola y ejecutar el comando:

```
1 mbrola diphone_database command_file1 command_file2 ...
    output_file
```

Un ejemplo podría ser:

```
1 mbrola fr1/fr1 fr1/TEST/bonjour.pho bonjour.wav
```

Donde se utiliza la voz francesa número 1, y el archivo .pho bonjour, situado en la carpeta fr/TEST/. Mbrola dispone de diversas funciones, accesibles desde el comando `mbrola -h`, del cual se desprenden las siguientes opciones:

```
1 mbrola [-i] [-e] [-c CC] [-v VR] [-f FR] [-t TR]
2 [-l VF] [-R RL] [-C CL] basededatos pho_file* output_file
3 >i = Print the database information if any
4 >e = No fatal error on unkown diphone
5 >CC= Comment Char, escape sequence for a comment
6 >VR= Volume Ratio, float ratio applied to ouput samples
7 >FR= Frequency Ratio, float ratio applied to pitch points
8 >TR= Time Ratio, float ratio applied to phone durations
9 >VF= Voice Freq, target freq for voice quality
10 >RL= Phoneme renaming list of the form a A b B ...
11 >CL= Phoneme cloning list of the form a A b B ...
```

Todos estos valores pueden ser modificados directamente dese los archivos *.pho añadiendo por ejemplo las secuencias:

```
1 ;; F=0.8 (se multiplica la frecuencia fundamental por 0.8)
2 ;; T=1.2 (se multiplica el tiempo por 1.2)
```

Otra opción es utilizar el paquete Mbrola Tools, el cual contiene:

- Mbrola.dll - Mbrola Engine
- MbrPlay.dll - Easy-to-use Interface to the engine
- Mbroli - A .pho Player
- PhoPlayer - A .pho Script Player Control Panel A control panel for managing the Mbrola Databases installed on the computer
- MbrEdit - Another .pho Player (written in VB, sources included)
- C and VB source codes - Interface to the DLLs
- Documentation

Cabe decir que Mbrola está vivo, aunque no tiene una comunidad como tal. Han desarrollado y siguen desarrollando diferentes aplicaciones como MBROLING, XLANG, EmoFilt, u otras utilidades. [5]

Una lista de los programas compatibles con Mbrola puede encontrarse en: http://tcts.fpms.ac.be/synthesis/mbrola.html

### Euler

Euler es un proyecto complementario a Mbrola para crear un TTS completo, aprovechando otros proyectos como Festival. La intención es crear un sistema abierto (que no lo es, al utilizar en muchos de sus módulos la licencia que utiliza Mbrola), multi-lingüe (utiliza las voces Mbrola), y multi-plataforma (de momento Windows y Linux, con planes para MAC, aunque el proyecto está estancado...).

Utiliza partes de Festival, tal y como se ha dicho, y está escrito en C++ con una síntesis de difonos. Se proporciona el código fuente y parece ser que existe únicamente en modo cliente, y sin API (falta información).

Según parece la intención era crear un sistema módular muy completo y compacto pero no hay novedades desde hace años (Versión 2.0 beta).

[5] http://tcts.fpms.ac.be/synthesis/

#### FreeTTS

Esta es una herramienta escrita en Java y basada en Flite, y con lo cual en permanente contacto con el equipo de Festival y FestVox. Puede utilizar tanto estas voces como voces Mbrola.

Es un proyecto desarrollado por Sun, y puede ser utilizado en versión cliente o servidor, así como web o API. Funciona en Solaris, MAC, Linux y Windows aunque depende de tener Java.

Se facilita el código fuente y disponen de foros activos en SourceForge. Puede obtener salida en WAV, y al soportar Mbrola parece que debería de disponer de salida en .pho aunque no hay información al respecto.

#### Cuadro resumen

Una vez analizados los diferentes TTS de código abierto, viendo en cada uno de ellos como es su estado actual en cuanto a los requisitos tanto obligatorios como valorables, todos estos datos se han resumido en el Cuadro 3.1

Cabe destacar que todos los sistemas han sido revisados en entre Junio y Septiembre de 2010, y que los enlaces facilitados también se corresponden con ese periodo de fechas.

### 3.3.3. TTS Comerciales

Aun no cumpliendo con los requisitos, en la página `http://ttssamples.syntheticspeech.de/` puede encontrarse un listado actualizado el 2010 creado por Felix Burkhardt con una tabla comparativa de diversos softwares comerciales. Los disponibles en castellano son Acapela, Natural Voices, Loquendo , RealSpeak (Nuance), SVOX y los TTS desarrollados por IBM.

## 3.4. Elección: Festival Speech Synthesis System

Finalmente, el sistema escogido es Festival. Festival permite utilizar prácticamente todo tipo de tecnología de síntesis, y dispone de un ambicioso proyecto abierto y colaborativo de creación de voces: FestVox. Su arquitectura permite la instalación tanto en el cliente como en el servidor, y su salida en PHO permitirá que pueda ser utilizada para la animación labial.

| | Festival | Flite | eSpeak |
|---|---|---|---|
| **Síntesis** | Selección de unidades, difonos (LPC, mbrola), HMM, Clustergen, Festvox | Selección de unidades, difonos (LPC, mbrola), HMM, Clustergen (Festvox) | Formantes, Compatible algunos idiomas con mbrola |
| **Arquitectura** | Cliente / Servidor | Servidor | Cliente |
| **Lenguaje** | C++ | ANSI C | C |
| **OS** | W/L/M | W/WC/L/P | W/L/(M/S) |
| **Voz Inglés** | mbrola/festvox/HTS/... | Festvox (convirtiendolas) | 9 x 9 modificaciones + voces Mbrola |
| **Voz Castellano** | mbrola/festival/JuntadeAndalucia/otras | Festvox (convirtiendolas) | 1 x 9 modificaciones + voces Mbrola |
| **WAV** | Si | Si | Si |
| **PHO** | Si | ? | Si |
| **Licencia** | X11 / MIT / Free Software | X11 / MIT / Free Software | OpenSource |
| **API** | Shell level, Scheme command interpreter, librería C++, Java, y Emacs | Librería C | .dll, SAPI5 |
| **URL** | cstr.ed.ac.uk/projects/festival | speech.cs.cmu.edu/flite | espeak.sourceforge.net |
| | **mbrola** | **Euler** | **FreeTTS** |
| **Síntesis** | Síntesis de difonos | Síntesis de difonos | Festvox + mbrola |
| **Arquitectura** | Cliente / Servidor * | Cliente | Cliente / Servidor |
| **Lenguaje** | C | C++ | Java |
| **OS** | Todos (bin) | W/M*/U | W/S/M/L |
| **Voz Inglés** | 4 | mbrola | festival |
| **Voz Castellano** | 4 | mbrola | festival |
| **WAV** | Si | ? | Si |
| **PHO** | No | ? | ? |
| **Licencia** | Libre NO Comercial | Libre NO Comercial | OpenSource BSD |
| **API** | - | - | Si, JAPI |
| **URL** | tcts.fpms.ac.be/synthesis | tcts.fpms.ac.be/synthesis/euler | sourceforge.net/projects/freetts |

Cuadro 3.1: Cuadro resúmen de los diferentes TTS Open Source

Al ofrecer voces de calidad, y adaptarse a las necesidades de licencia cumpliendo el resto de requisitos, Festival será la base del TTS integrado en la plataforma de mundos virtuales.

Será necesario añadir funciones adaptadas a las necesidades de las diferentes aplicaciones posibles de realXtend. Festival debe proporcionar de manera sencilla la opción de sintetizar un texto proveniente tanto de una cadena de caracteres como de un fichero, con opción de que la salida sea únicamente audible, en formato WAV o en formato PHO. Además la conmutación entre diferentes idiomas tiene que ser sencilla y transparente, pudiendo ser alterada dicha configuración para cada declaración.

# Capítulo 4

# Implementación

## 4.1. Modo de integración

Antes de comenzar la integración es necesario saber como va a ser llevada a cabo. Hay diversas opciones y todas deben de ser evaluadas antes de continuar adelante.

Lo primero que hay que definir es la arquitectura de la implementación. Hay que decidir si integrar el TTS en el lado cliente o en el servidor.

Si la integración se lleva a cabo en el lado cliente se obtiene una mayor carga en el cliente tanto de peso/espacio como de procesado. Por contra, se libra al servidor y a la red de esta carga y no es necesaria la comunicación con el servidor para obtener la voz sintetizada (streaming). En esta arquitectura, el servidor no conoce el módulo TTS.

Naali, está diseñado para implementar funcionalidades mediante módulos. Taiga aún no está preparado con esta visión, por lo que también puede resultar más sencilla la integración en el lado cliente. Además, ya se ha comenzado a trabajar con un nuevo servidor-cliente experimental (Tundra [1]) por lo que parece más estable trabajar sobre Naali.

Otro punto desfavorable para la integración en el cliente es que en el caso de que el sistema TTS Open Source Festival sea sustituido por uno privativo, se ha de pagar una licencia para cada usuario final.

Aunque festival puede trabajar tanto como cliente o como servidor, el conver-

[1] http://github.com/realXtend/naali/tree/tundra

sor se integra en el cliente, dejando abierta la puerta para una integración futura en el servidor cuando este quede más definido, si fuera necesario.

Una vez decidida esta parte, hay que ver como integrar exactamente Festival Speech Synthesis System. Tal y como se observa en el Capítulo 3, Festival puede ser integrado mediante librerías, o se puede crear un ejecutable adaptado utilizando su API.

El hecho de integrar las librerías de Festival no es un problema ya que las licencias son compatibles, pero por otro lado una integración basada en un ejecutable puede traer ciertas ventajas. Por un lado, si en un futuro se viese la necesidad de reemplazar el TTS escogido por otro sistema, sea libre o comercial, bastaría con sustituir el ejecutable, sin tener que tocar absolutamente nada del código de realXtend.

ENNE planea esta posibilidad en el futuro, con lo que la integración se lleva en el lado cliente y por medio de un ejecutable.

En resumen, esta implementación se divide en dos partes: Festival TTS y realXtend. En la parte del TTS, es necesario crear un binario que cumpla con todas las funciones requeridas por realXtend. El TTS debe aceptar como entrada tanto un fichero, como una cadena de caracteres y debe sintetizar el texto, en forma de fichero de audio (WAV), directa al hardware de sonido y salida fonética (PHO). También debería de poder controlarse y modificar los idiomas de manera sencilla.

El TTS deberá de tener voces en castellano e inglés, con la máxima calidad posible, por lo que exigirá una búsqueda de voces y su agregación al proyecto.

Por otro lado, en realXtend, hay que crear un módulo TTS, así como un servicio que pueda ser utilizado por cualquier módulo que quiera una aplicación TTS. En esta interfaz habrá que definir los diferentes métodos, siendo estos los mismos que se han desarrollado en el ejecutable. En el módulo únicamente se hará la llamada al ejecutable de forma que si este es modificado, sólo haya que cambiar el nombre de la aplicación.

Para esto, primeramente se crea un módulo vacío como ejemplo y explicación práctica de los módulos, para después crear el TTS. Para demostrar el uso de sus servicios, se desarrollan dos aplicaciones, siendo una de ellas simple e independiente y otra con modificaciones mayores tanto a nivel gráfico como de programación.

Mediante la creación de las aplicaciones se puede comprobar el funcionamiento

de la integración, tanto en la parte de Festival o lo relacionado con sus librerías (voces), como en el módulo o servicio TTS. Al desarrollar la aplicación después del módulo y el ejecutable, habrá mejoras detectadas en el momento de prueba de la aplicación, que son explicadas en cada apartado.

Para finalizar esta introducción es importante describir la comunicación con la comunidad durante este proceso de integración. La comunidad dispone de dos listas de distribución [2] [3] mediante las que se puede poner en contacto con ellos. Otra vía más directa son los canales de IRC (Internet Relay Chat) situados en el servidor *freenode*, #realxtend y #realxtend-dev.

A lo largo de la implementación la comunicación con la comunidad ha sido fluida, en especial con el responsable de comunicaciones Matti Kuonanoja.

## 4.2. Festival Speech Synthesis System

### 4.2.1. Extensión de funcionalidades

Festival Speech Synthesis System ha sido desarrollado bajo un entorno UNIX. Aunque se ha compilado para diversas plataformas, y poco a poco se va mejorando, actualmente sigue concebido para UNIX. No obstante, el uso de sus librerías y la compilación es posible aunque laboriosa.

La compilación de un binario ha sido efectuada basandose en el trabajo de Xavi Gonzalvo [4] con unas librerías adaptadas para Windows basandose en el código de Festival 1.95.

De esta forma, se logra un ejecutable que puede ser utilizado de la forma en la que se ha explicado en el estado del arte de los TTS del Capítulo 3. Este, por defecto ofrece la posibilidad de sintetizar texto mediante el comando (SayText "texto"). También pueden ser configuradas las voces mediante (voice_nombre_base_de_datos) y la ruta donde se encuentra la librería (libdir).

No obstante, esto no es suficiente y hace falta tanto expandirlo como adaptarlo siguiendo una nomenclatura exacta. El interfaz que hay que crear ha de satisfacer las siguientes necesidades:

1. El archivo de entrada puede ser texto o puede ser un fichero
2. La salida puede ser audible (A), WAV (W) o fonética en formato PHO (P)

[2] http://groups.google.com/group/realxtend?pli=1
[3] http://groups.google.com/group/realxtend-dev
[4] http://sites.google.com/site/xavigonzalvo/festival-1-95-win32

3. El idioma puede ser elegido de forma sencilla (EN1, ES2...)

Lo necesario en este caso es analizar el texto, o *parsear* las opciones que se presentan. De esta forma desde la línea de comandos se puede escribir por ejemplo:

```
festival.exe –TEXTO 'Esto es un texto' –A
```

De forma que esta llamada es analizada, obteniendo el texto a sintetizar, y la opción A, sinónimo de que la síntesis ha de ser audible.

Esto permite que si en lugar de festival, se quiere utilizar otro programa, bastaría con crear un ejecutable para ese programa con las mismas funciones, haciendo que funcione de misma manera si se le llama con:

```
programa.exe –TEXTO 'Esto es un texto' –A
```

Para ello, se definen las siguientes marcas o etiquetas:

- -T "Texto": El texto entrecomillado inmediatamente posterior a -T es la entrada de texto a sintetizar.
- -F archivo.txt: El archivo de texto cuya ruta/nombre va después de -F es la entrada a procesar.
- -A: La salida ha de ser audible, haciendo que Festival sintetice la entrada y envíe la señal al sistema de audio.
- -W rutaYarchivo: Se almacena la salida en formato WAV en el lugar indicado en rutaYarchivo
- -P rutaYarchivo: Se almacena la salida en formato PHO en el lugar indicado en rutaYarchivo
- -IDIOMA: La voz correspondiente es seleccionada.

El código añadido puede verse en el Apéndice A. Únicamente se ha añadido la función modificada. Si se desea el original, hay que recurrir a las direcciones citadas en el Capítulo 3.

Si se desea añadir una nueva voz, que cumpla un mínimo de calidad, basta con implementar el código en ese fichero y añadir las voces en la carpeta de las librerías. Esta carpeta junto el binario, son las que hay que copiar en el proyecto de realXtend.

```
C:\festival_dev\festival\main_win32\Release>festival -help
Usage: festival Usage:
festival  <options> <file0> <file1> ...
In evaluation mode "filenames" starting with ( are evaluated inline
Festival Speech Synthesis System: 1.9.5:dev 2006
-q              Load no default setup files
--libdir <string>
                Set library directory pathname
-b              Run in batch mode (no interaction)
--batch         Run in batch mode (no interaction)
--tts           Synthesize text in files as speech
                no files means read from stdin
                (implies no interaction by default)
-i              Run in interactive mode (default)
--interactive
                Run in interactive mode (default)
--pipe          Run in pipe mode, reading commands from
                stdin, but no prompt or return values
                are printed (default if stdin not a tty)
--language <string>
                Run in named language, default is
                english, spanish and welsh are available
--server        Run in server mode waiting for clients
                of server_port (1314)
--script <ifile>
                Used in #! scripts, runs in batch mode on
                file and passes all other args to Scheme
--heap <int> {1000000}
                Set size of Lisp heap, should not normally need
                to be changed from its default
-v              Display version number and exit
--version       Display version number and exit

*************************************************************
*****    ************    MORE OPTIONS    ************ *********
*************************************************************

The options are:
     Input:   File or Text.
     Output:  Audible, Phonetic information, Wave.

-F <string>       <string> filename is the input file
-T <string>       <string> is an input text between quotation marks
-A                Audible output is ON
-P <string>       Phonetic file is saved in <string> filename
-W <string>       WAV Output is saved in <string> filename
-ES1              Castillian male voice is chosen
-ES2              Castellian female voice is chosen

C:\festival_dev\festival\main_win32\Release>
```

Figura 4.1: Menu de ayuda del festival modificado. Se pueden observar las nuevas funcionalidades añadidas en la parte inferior.

Un ejemplo de las nuevas funcionalidades, con dos voces castellanas (masculina ES1 y femenina ES1) puede verse en la siguiente captura de pantalla, mostrada en la Figura 4.1. La idea puede verse gráficamente en la Figura 4.2

### 4.2.2. Voces iniciales

La prioridad en la integración es la posibilidad de sintetizar texto en castellano y en inglés. Para ello se han tenido que localizar e implementar voces de calidad en estos idiomas.

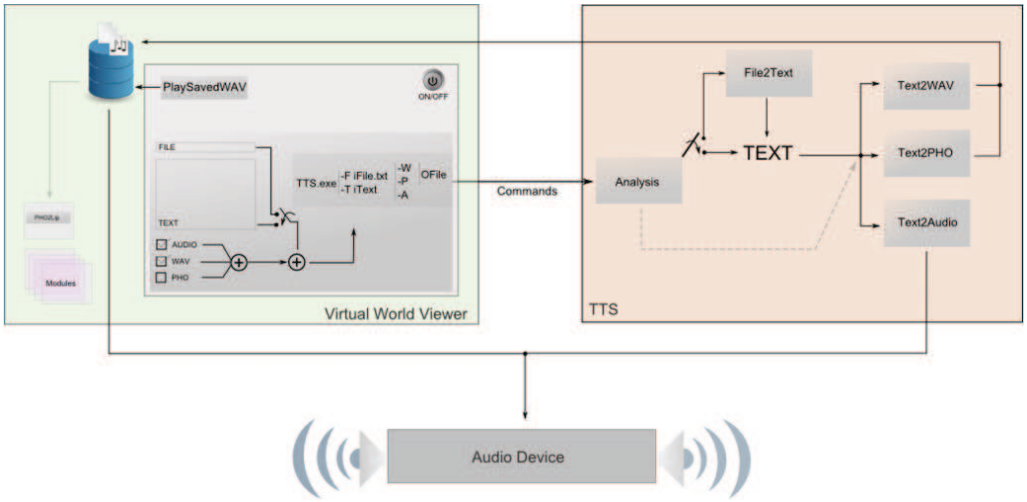


Figura 4.2: El ejecutable ampliado se muestra dentro del recuadro de la parte derecha. Acepta como entrada texto o un archivo, y ofrece diferentes salidas. Desde el cliente de mundos virtuales se construirá la llamada, y dependiendo de esta, Festival ofrecerá sus servicios. La parte de la izquierda es orientativa, y el diseño es especifico de la aplicación.

| Etiqueta | Idioma | Genero | Base de datos |
|---|---|---|---|
| -ES1 | Castellano | Masculino | JuntaDeAndalucia_es_pa_diphone |
| -ES2 | Castellano | Femenino | JuntaDeAndalucia_es_sf_diphone |
| -EN1 | Inglés US | Masculino | voice_nitech_us_awb_arctic_hts |
| -EN2 | Ingles US | Femenino | voice_nitech_us_slt_arctic_hts |

Cuadro 4.1: En esta tabla se ven las asignaciones de las etiquetas con las voces correspondientes y los nombres de sus bases de datos

### Castellano

Las posibilidades para utilizar voces en castellano pasan por utilizar la voz por de difonos de Eduardo López (el_diphone, la de defecto de Festival en castellano), utilizar voces mbrola, o encontrar otras voces.

La voz por defecto de Festival tiene una alta inteligibilidad pero muestra muy poca naturalidad. Aunque sea una voz funcional, es una voz muy limitada ya que no se incluyen diversas variaciones, y se basa en cubrir las necesidades básicas a una calidad razonable. El total de difonemas es 662 y fueron grabados en un entorno no especializado.

La *addenda* de la voz de Eduardo sólo contiene determinados símbolos y caracteres especiales. Por *addenda* se entiende un conjunto de palabras (y sus trans-

| VOTOS RECIBIDOS POR CADA VOZ PARA LA NATURALIDAD | % |
|---|---|
| ***PA*** | ***50,83*** |
| ***SF*** | ***23,33*** |
| EL | 22,5 |
| Espeak | 3,33 |

| VOTOS RECIBIDOS POR CADA VOZ PARA LA INTELIGIBILIDAD | % |
|---|---|
| ***PA*** | ***58,33*** |
| ***SF*** | ***25*** |
| EL | 14,17 |
| Espeak | 2,5 |

| PUNTUACIÓN MEDIA POR VOZ | |
|---|---|
| ***PA*** | ***6,23*** |
| ***SF*** | ***5,45*** |
| EL | 4,21 |
| Espeak | 2,56 |

Figura 4.3: Resultados del test subjetivo llevado a cabo por Guadalinex, obtenido de [10]

donde se observael incremento de naturalidad e intelegibilidad de las voces.

cripciones) que quedan fuera del léxico compilado. Este conjunto de palabras, por tanto, puede ser modificado dinámicamente [3].

La opción de utilizar voces mbrola, requiere que se tenga que integrar también mbrola. Además estas voces son similares a la de Eduardo.

Por último si se buscan otras voces desarrolladas, sólo son accesibles las voces impulsadas por la Junta de Andalucia para el proyecto Guadalinex [5].

El desarrollo de estas voces trajo muchas mejoras respecto a la voz de Eduardo. Al incluir variaciones, se graban 1090 difonos grabados en un entorno especializado. Guadalinex incluyo tanto una voz masculina (Pedro) como una femenina (Silvia), y se incluyo una *addenda* de 418 términos. Todas estas mejoras son detalladas en [10].

De ese mismo documento, se pueden obtener los resultados de un experimento subjetivo realizado por el mismo equipo, donde se comparan sus propias voces con la de Eduardo y la de eSpeak. En la Figura 4.3 se muestra el resumen.

Por ello, las voces de Pedro y Silvia son las que se implementan, accesibles desde `http://forja.guadalinex.org/projects/hispavoces/`

[5] `http://www.guadalinex.org/que-es-guadalinex`

#### Inglés

En el caso de las voces en inglés la disponibilidad es mucho mayor ofreciendo incluso voces de calidad para distintos acentos. En el paquete festival se incluyen diversas voces tanto masculinas como femeninas que pueden ser suficientes.

Un aspecto que se tiene en cuenta para elegir las voces es el espacio que ocupan, sin olvidar la calidad. La síntesis basada en modelos ocultos de Markov permite precisamente obtener voces de calidad ocupando muy poco. Básicamente están las voces CMU ARTIC [6] y las HTS [7].

Las voces CMU ARTIC ocupan al rededor de 150 MB porque están basadas en bases de datos formadas por CLUNITS (Selección de unidades, síntesis concatenativa). Por otro lado, las voces HTS utilizan la síntesis de Markov, por lo que ocupan mucho menos (unos 2 MB).

Todas las voces CMU ARTIC han sido convertidas a voces HTS, por lo que son estas las que se utilizan, pudiendo descargarse todas ellas desde `http://hts.sp.nitech.ac.jp/?Voice%20Demos`

En concreto las voces son las comentadas en el Cuadro 4.1.

### 4.2.3. Adición de nuevas voces

Tras una primera implementación, y el desarrollo de una aplicación, se ha pretendido extender la librería mediante otras voces de calidad con dos objetivos:

1. Mostrar la sencillez de adición de nuevos idiomas.

2. Añadir funcionalidad y ofrecer el servicio a más usuarios, para que la propia comunidad pueda probar el producto.

Además de las iniciales, se han incluido voces en catalán y suomi. Estos idiomas añadidos pueden verse en el Cuadro 4.2. Como puede observarse, también hay nuevas voces en inglés.

Si surge la necesidad de añadir una nueva voz, hay que seguir dos pasos:

1. Añadir la base de datos a la librería $\backslash lib \backslash voices$

2. Modificar el código para que acepte el nuevo idioma y lo utilice.

[6] `http://www.festvox.org/cmu_arctic/`
[7] `http://hts.sp.nitech.ac.jp/`

| Etiqueta | Idioma | Genero | Base de datos |
|---|---|---|---|
| -EN3 | Inglés US | Masculino | voice_nitech_us_jmk_arctic_hts |
| -EN4 | Inglés US | Femenino | voice_nitech_us_clb_arctic_hts |
| -EN5 | Inglés US | Masculino | voice_nitech_us_rms_arctic_hts |
| -EN6 | Ingles US | Masculino | voice_nitech_us_bdl_arctic_hts |
| -CAT1 | Catalán | Masculino | voice_upc_ca_pep_hts |
| -CAT2 | Catalán | Femenino | voice_upc_ca_ona_hts |
| -FI | Suomi | Masculino | voice_hy_fi_mv_diphone |

Cuadro 4.2: Tabla con voces añadidas: 4 en inglés, 2 en catalán y una en finlandes.

Modificar el código implica añadir una nueva etiqueta entre las opciones y ver si esta está en la llamada al programa para en tal caso utilizar dicho idioma en la síntesis.

Las voces catalanas y la finlandesa son explicadas en las próximas secciones.

### Catalán

Actualmente el proyecto Festcat [8] [6] contiene hasta 10 voces masculinas y femeninas y tanto formadas por CLUNITS como las equivalentes en HTS.

Para la integración, se ha seguido el consejo de Sergio Oller del equipo de Festcat, escogiendo la voz de Ona como femenina, y Pau como masculina.

### Suomi

Actualmente sólo hay una voz disponible en Suomi en `http://www.ling.helsinki.fi/suopuhe/index.shtml`. Como realXtend es un proyecto con sede en Finlandia, se incorpora esta voz para que la Comunidad pruebe la integración en su propia lengua.

### Otras voces

Otras voces HTS pueden encontrarse en: `http://hts.sp.nitech.ac.jp/?Voice%20Demos` Entre ellas están el Japones, Portugues, Xhosa o Zulu entre otras.

[8] `http://www.talp.upc.edu/festcat/`

## 4.3. realXtend

### 4.3.1. Módulo TTS

realXtend está diseñado para poder añadir módulos de una forma medianamente sencilla. El manejador de módulos es el encargado de detectar todos los módulos y cargarlos de forma ordenada al iniciar Naali. Para que el manejador detecte el módulo, es necesario construirlo de forma adecuada siguiendo ciertos pasos.

Antes de crear el módulo TTS, se crea un nuevo módulo vacío que sirve como guía para cualquier tipo de funcionalidad.

#### Creación de un módulo vacío

La creación de un nuevo módulo debe de seguir ciertos pasos. Estos están explicados en la página de documentación de realXtend[1], pero a continuación se complementa esta información con ejemplos.

El primer paso para crear un nuevo módulo es descargar el código de Naali [2] y crear el proyecto. Para ello es necesario descargas las dependencias y realizar algunos cambios detallados en la página http://wiki.realxtend.org/index.php/Building_Naali_from_source_trunk.

Una vez creado el proyecto para Visual Studio, se puede observar que la solución contiene diversos proyectos. Estos proyectos son los diferentes módulos, siendo el módulo principal el viewer, o visor.

Primeramente es necesario crear un nuevo proyecto, teniendo en cuenta que ha de ser añadido a la solución. Este proyecto ha de estar físicamente en una carpeta con el mismo nombre dentro de la ruta de Naali, de igual forma al resto de módulos. Para este ejemplo, se crea el modulo EmptyModule.

Para generar el proyecto en Visual Studio, CMake[9] lee los archivos *CMakeList.txt* y *CMakeOptionalModules.txt*. Estos archivos contienen los nombres de los módulos obligatorios y opcionales que son añadidos a la solución. En este caso, el módulo vacío es opcional, por lo que hay que añadir en el archivo *CMakeOptionalModules.txt* la siguiente línea:

```
1 add_subdirectory (EmptyModule)
```

[1] http://realxtend.org/doxygen/

[2] http://github.com/realXtend/naali

[9] CMake es una familia de herramientas diseñada para construir, probar y empaquetar software

En caso de que no se quiera añadir a la solución uno de los módulos, es suficiente dejarlo comentado añadiendo el símbolo almohadilla (#) delante de su línea.

Además de este CMake general, es necesario que el propio módulo tenga su *CMakeList.txt*.

Por otro lado es necesario crear un fichero de texto (un XML) en el directorio de trabajo. Este XML es el archivo que leerá el manejador de módulos para ordenar la carga y descarga de módulos, observando las dependencias. Este archivo, llamado *EmptyModule.xml*, en caso de no tener ninguna dependencia se muestra así:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <config>
3     <entry>EmptyModule</entry>
4 </config>
```

En caso de disponer de alguna dependencia, se añade una línea después de la entrada del nombre del estilo:

```
1     <dependency>OgreRenderingModule</dependency>
```

En este momento el módulo existe, y el manejador de módulos sabe que tiene que cargarlo y descargarlo. Sin embargo, esta carga y descarga ha de ser creada. Para ello se crean los ficheros *EmptyModule.h* y *EmptyModule.cpp* dentro del proyecto. Es aquí donde se crea una nueva clase que hereda del interfaz de módulos (Foundation::ModuleInterface). También es necesario implementar algunos de los métodos como:

- Load(): Es ejecutado al iniciar el visor. Es donde se declaran los componentes que ofrece el módulo con la macro de declaración de componentes *DECLARE_MODULE_EC(ComponentName)*.
- Initialize(): Una vez cargados, se inician. Aquí es donde se registra el servicio en el framework si el módulo tiene una interfaz que ofrecer.
- Unitialize(): Aquí se eliminaría el registro del servicio, y todo le pertinente al fin de uso del módulo.
- HandleEvent(): Mediante este módulo se pueden manejar los eventos de red, ya que estos son enviados a todos los módulos. El módulo puede funcionar de esta forma cuando ocurre algún evento de red.

- Update(): Para funciones sincronizadas.

Hay que comentar que la programación en realXtend debe seguir unas guías de programación descritas por la comunidad en: `http://www.realxtend.org/doxygen/progconv.html` Los archivos base son *EmptyModule.cpp* y *EmptyModule.h* donde hay que cumplir este convenio de programación junto con partes de código que han de ser añadida de forma obligatoria, para que todo funcione correctamente:

```
1 extern "C" void POCO_LIBRARY_API SetProfiler(Foundation::
    Profiler *profiler);
2 void SetProfiler(Foundation::Profiler *profiler)
3 {
4     Foundation::ProfilerSection::SetProfiler(profiler);
5 }
6
7 using namespace Namespace;
8
9 POCO_BEGIN_MANIFEST(Foundation::ModuleInterface)
10     POCO_EXPORT_CLASS(EmptyModule)
11 POCO_END_MANIFEST
```

Al proyecto hay que añadirle los archivos *StableHeaders.cpp/.h* e incluirlos en el EmptyModule.

De esta forma, tenemos un módulo vacío que puede hacer algo, en el momento de cargarse, inicializarse, recibir mensajes de red, o actualizarse.

Ahora hay que añadir una API al módulo y desarrollar un servicio o interfaz para que otros módulos accedan a él mediante el framework.

Para crear la API, se crea un archivo cabecera llamado *EmptyModuleApi.h*.

Para crear el servicio, es necesario crear una Interfaz en un archivo de cabecera dentro del proyecto Interfaces. Esta interfaz, es el la hoja de presentación de este módulo para el resto, y es donde se definen los métodos accesibles. En este ejemplo, este archivo es *EmptyModuleInterface.h*.

Estos métodos han de ser implementados en los ficheros *EmptyModuleService.h/cpp*.

Se puede acceder al servicio a través del framework utilizando el nombre del módulo, o un identificador. Estos identificadores se encuentran en una lista, dentro del fichero *ServiceInterface.h*. Hace falta añadir el identificador en él:

```
namespace Service
{
        enum Type
        {
            ST_Renderer,
            ST_Physics,
            ...
        ST_EmptyModule
        };
}
```

Con este último paso se concluye la creación de un módulo vacío con servicio.

### Creación del módulo TTS

La creación del módulo resulta sencilla partiendo de la plantilla diseñada en el anterior apartado. En este punto hay que tomar decisiones e implementar los métodos.

El módulo, debe ofrecer su servicio al resto de módulos. Por otro lado, este módulo es totalmente independiente, de forma que no debe incluir ninguna dependencia. Por último hay que barajar la posibilidad de añadir un componente, y declararlo.

#### Módulo

En el módulo, únicamente se registra el servicio y se declara el componente por lo que su configuración, visible en el Apéndice B es muy simple y similar al módulo vacío.

#### Servicio

Las funcionalidades del módulo se escriben en el servicio. Este servicio será el que utilice el ejecutable modificado, por lo que las llamadas deberán de seguir la estructura descrita.

Hay que crear una tabla o estructura donde se almacenen todos los idiomas disponibles, limitando el uso del módulo a las voces de las librerías:

```

struct AvailableVoice
{
```

```
4    Voice ES1;
5    Voice ES2;
6    Voice EN1;
7    Voice EN2;
8    Voice EN3;
9    Voice EN4;
10   Voice EN5;
11   Voice EN6;
12   Voice CAT1;
13   Voice CAT2;
14   Voice FI;
15 };
16
17 static const AvailableVoice Voices = {"-ES1","-ES2","-EN1","-
       EN2","-EN3","-EN4","-EN5","-EN6","-CAT1","-CAT2","-FI"};
```

De esta forma, si se añade una nueva voz, hay que añadir una línea a esta estructura y definir su valor.

Por otro lado, los servicios que ha de proporcionar el módulo son:

- Convertir texto a voz sintetizada, proveniente de una cadena de caracteres o de un archivo de texto
- Convertir ese mismo texto a WAV
- Obtener información fonética en formato PHO partiendo de esa misma información
- Configurar el idioma de síntesis

Para ello se definen los siguientes métodos, incluidos en el Apéndice B:

```
1       virtual void text2Speech(QString message, TTS::Voice
           voice);
2     virtual void text2WAV(QString message, QString
          pathAndFileName, TTS::Voice voice);
3     virtual void text2PHO(QString message,QString
          pathAndFileName, TTS::Voice voice);
4
5     virtual void file2Speech(QString pathAndFileName, Voice
          voice);
```

```
6       virtual void file2WAV(QString pathAndFileNameIn, QString
            pathAndFileNameOut, Voice voice);
7       virtual void file2PHO(QString pathAndFileNameIn, QString
            pathAndFileNameOut, Voice voice);
```

Tal y como puede observarse, se ha creado el namespace TTS, dentro del cual se encuentra todo lo referente al módulo TTS.

#### Componente

Si se desea dotar a las entidades de funcionalidades TTS, es imprescindible crear un componente. Mediante la adición de este componente a un objeto de la escena, podemos hacer por ejemplo, que al hacer clic, se reproduzca un texto en un determinado idioma (un atributo del componente).

#### Mejoras

El código descrito en el Apéndice B es el código final. Sin embargo, se detectaron problemas al probar la primera aplicación del chat que sirvieron para desarrollar la última versión.

Los problemas fueron:

1. Naali quedaba congelado mientras se utilizaba Festival.
2. El hecho de que el mensaje tuviese comillas o corchetes, hacia que el mensaje no se pasara correctamente.
3. Pocos idiomas
4. La codificación era alterada, no respetando las tildes.
5. La librería por defecto tenía que estar en la carpeta raíz C:\

La adición de voces se solventa añadiendo las nuevas en la estructura, tal y como se refleja en el código. Por otro lado, la codificación se veía alterada al cambiar el formato de QString a strings, por lo que finalmente se opta por trabajar con QStrings.

Para las comillas, se implementa una comprobación para saber si hay comillas en el mensaje. Si las hay, estas son sustituidas por comas (,) teniendo en cuenta que en una conversación si se pretenden añadir comillas a algo, se suele parar para remarcar esa palabra.

Estas comillas únicamente se eliminan y son sustituidas en el momento de sintetizar la frase, por lo que realmente siguen en el texto.

Para modificar la librería, ya existe la opción –libdir de Festival que permite ubicar las voces donde interese. En el caso de la aplicación, se crea una carpeta llamada festival donde se almacenan las voces.

Por último, Festival era priorizado dejando a Naali congelado mientras se sintetizaba una frase. Por otro lado, de esta forma las frases o los textos son sintetizados en cola, ya que la ejecución no continua hasta que termina la síntesis. La solución es crear un proceso independiente para cada llamada a festival. Esto puede ser realizado desde windows añadiendo `start /B` antes de la llamada al ejecutable. Esto hace que el proceso se ejecute "por detrás"(*background*) y no interfiera en Naali. De esta forma, la síntesis se efectúa en paralelo, pudiendo utilizar Festival de manera simultanea con diferentes textos.

## 4.4. Aplicación 1: TTS para el Chat

En esta sección se presenta una aplicación completa donde se da uso al servicio TTS. El caso del chat, es uno de los posibles usos más evidentes para este tipo de servicio, mediante el cual se puede conseguir un chat hablado.

Para definir esta primera aplicación hay que analizar primero el funcionamiento del In-world chat. En la misma fase, se descubre el funcionamiento e interacción entre las interfaces gráficas (GUI) y los módulos así como el flujo del mensaje desde el instante en que un usuario lo escribe hasta que el otro lo visualiza.

Tras la finalización de esta aplicación se detectan algunas carencias o errores, parte de las cuales ya se han comentado en anteriores apartados. Es por ello, que esta aplicación sirve además como test para lo realizado anteriormente.

### 4.4.1. Análisis del In-World Chat

El flujo del mensaje desde el momento en el que se introduce en el cuadro de texto hasta que todos los participantes lo visualizan sigue distintos procesos. Supongamos que existe una comunicación entre los usuarios A y B. Si A escribe un mensaje a B, esto es lo que sucede:

- A introduce el texto, y pulsa enter.
- El texto es mostrado sobre la cabeza de A tanto para A como para B


Figura 4.4: Captura de pantalla de dos usuarios donde uno de ellos escribe un mensaje. Este mensaje se visualiza mediante una burbuja sobre el usuario y en el lado inferior izquierda.

- El texto se añade al historial de chat, y se ve en pequeño encima de la línea de texto del chat también en el visor de A como en el de B.

Para entenderlo gráficamente, en la Figura 4.4 se pueden observar este proceso.

La ventana en la que se introduce el texto, es un Widget creado con Qt. Más concretamente se trata del *CommunicationWidget*, mostrado en la Figura 4.5 una vez abierto con el software Qt Designer.

Este software permite diseñar las ventanas o GUI de Qt de una forma gráfica. Sin embargo, tal y como se aprecia en la Figura 4.5 el resultado final (ver Figura 4.4) no es exactamente el mismo. Esto es así porque el diseño puede ser programado en los correspondientes códigos *CommunicationWidget.cpp/.h*.

El widget de comunicaciones (CommunicationWidget) incluye diversas partes. En la parte izquierda esta el ChatLine donde se introduce el texto y lo relacionado con el In-World Chat, por otro lado, en la parte central esta lo relacionado con la Voz por IP o Mumble. Finalmente en la parte derecha está la funcionalidad de la mensajería instantánea (IM) externa. La carga de estos servicios depende de si los módulos correspondientes están o no añadidos en el proyecto.

El texto, una vez introducido se envía a la sesión de chat. La sesión se encarga de emitir una señal, recibida por el proveedor que se encarga de enviar el mensaje a la red. Antes de enviarlo, este mensaje se construye con el formato adecuado.

Los mensajes de red llegan a los usuarios, de forma que la sesión de chat detecta un nuevo mensaje y emite una señal de ”nuevo mensaje recibido”. Esta

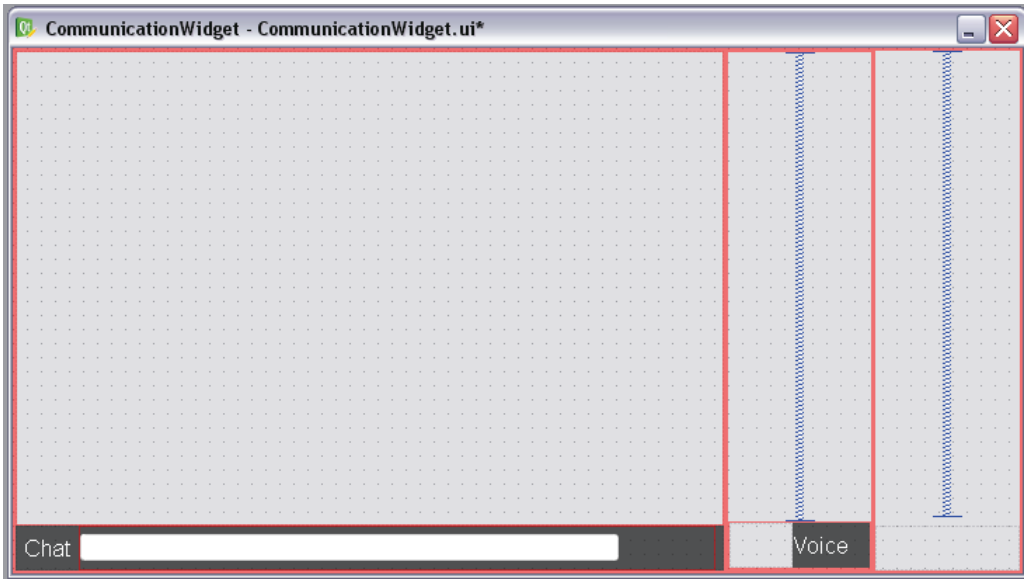


Figura 4.5: CommunicationWidget. Contiene todas las funcionalidades de comunicaciones: In-World Chat, VoIP e IM. En la imágen no se observan los botones ni el contenido porque este se genera de forma dinámica y contiene texturas cargadas mediante hojas de estilo (CSS)

señal, genera diferentes procesos: muestra la burbuja con el mensaje, muestra el texto en pantalla, lo añade al historial y lo almacena en un log si el logging está activado.

De esta forma, tanto A como B, ven el mensaje cuando les llega por la red. Nótese que incluso A, envía el mensaje a la red y lo muestra al recibirlo de vuelta.

Es conveniente tener localizado en el código todo este proceso para saber donde y como se puede tratar este mensaje. Básicamente el mensaje es accesible desde el CommunicationWidget (módulo Ui, User Interface), o desde el módulo InWorldChat.

### 4.4.2. Diseño de la aplicación

El diseño de la aplicación se plantea como una extensión del widget de comunicaciones. La idea es añadir un nuevo botón que lance un nuevo widget llamado TTSChatWidget, de manera similar a como se lanza el widget de IM o la lista de participantes de Mumble.

Este botón únicamente será mostrado si el servicio TTS está disponible. De esta manera, si un usuario tiene desactivado este módulo, no dispondrá de esta

aplicación, mientras que otro usuario con el módulo activado podrá disfrutar de ella.

Además de los widget, es necesario acceder al servicio TTS implementado en el módulo y enviarle el mensaje y el idioma que debe utilizar para sintetizar el texto. El método que hay que utilizar es:

```
1 text2Speech(QString message, TTS::Voice voice)
```

Con lo que hay que obtener la voz del usuario, configurada a través del TTS-ChatWidget. La configuración debe permitir la activación o desactivación del TTS, y la elección de la voz de síntesis.

En este punto, Naali no dispone de un método de distinción del mensaje propio del de los demás, aunque está implementado a medias. Tras comunicación con la comunidad de reX, se acuerda desarrollar esta parte conjuntamente, por lo que el diseño tiene en cuenta la posibilidad de distinguir el mensaje propio del resto.

La voz de síntesis, o la base de datos, es escogida a partir de la información de idioma utilizado y genero del avatar. Actualmente, los avatares no pueden incluir la información de genero en su perfil. Esta prevista una remodelación de todo el sistema de avatares donde pueden ser añadidos este tipo de datos. De momento, se utiliza una clase de configuración donde se almacena esta información.

La activación o desactivación del servicio exige un flag/booleano también almacenado en la clase de configuración. Esta clase será la que se actualice mediante el TTSChatWidget.

Los idiomas de los que se ha de disponer en principio son el castellano y el inglés, por lo que la aplicación es pensada para poder ser utilizada con voces inglesas y castellanas, tanto masculinas como femeninas.

A continuación se describe la implementación paso a paso: parte gráfica e implementación.

### 4.4.3. Implementación parte gráfica

#### Botón

El botón es añadido en el CommunicationWidget. Mediante Qt Designer, se pueden desagrupar los diferentes componentes y construir una nueva zona para el botón TTSButton. En la Figura 4.6 se ve el resultado de añadir una nueva columna con una zona para el botón.

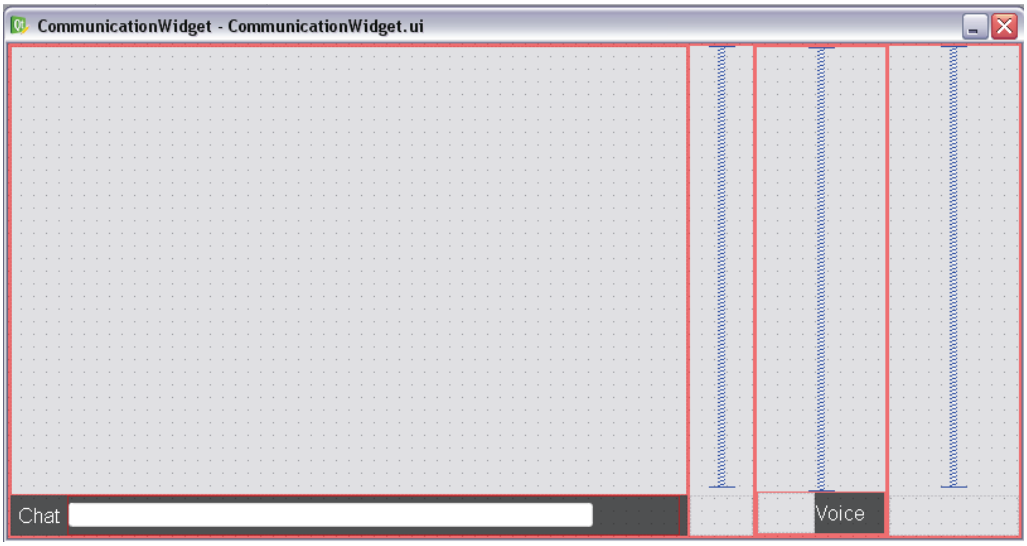


Figura 4.6: El nuevo widget reserva un espacio para el botón TTS.

El diseño de esta columna parte del diseño del resto de columnas. Ello incluye un layout vertical general sobre el que se crean dos espacios: el superior, en el que se sitúa un espaciador (spacer) vertical y el interior donde se crea el widget ttsContentWidget. Este widget contiene un pushButton llamado ttsButton sin contenido.

Al botón, en principio vacío, se le puede añadir una textura y un fondo. El fondo se añade en el contenedor (ttsContentWidget) y la textura del botón al pushButton. Esto se hace mediante hojas de estilo (stylesheet) añadiendo el siguiente código en el CommunicationWidget:

```
1  QWidget#ttsContentWidget {
2    background-color: transparent;
3    background-image: url('./data/ui/images/chat/TTS_bground.png'
         );
4    background-position: top left;
5    background-repeat: no-repeat;
6  }
7
8
9  QPushButton#ttsButton {
10   border: 0px;
11   background-color: transparent;
12   background-image: url('./data/ui/images/chat/
```


Figura 4.7: Los tres estados del botón, normal, con el ratón encima y al hacer clic. La diferencia es la opacidad.

```
        uibutton_TTS_normal.png');
13    background-position: top left;
14    background-repeat: no-repeat;
15  }
16
17  QPushButton#ttsButton::hover {
18    border: 0px;
19    background-image: url('./data/ui/images/chat/
        uibutton_TTS_hover.png');
20  }
21
22  QPushButton#ttsButton::pressed {
23    border: 0px;
24    background-image: url('./data/ui/images/chat/
        uibutton_TTS_click.png');
25  }
```

Tal y como se observa, las imágenes al situar el ratón sobre el botón, hacer clic y en su estado normal, son diferentes aunque se encuentran en la misma carpeta de imágenes.

Esta imagen es un texto (TTS), y sus tres estados se muestran en la Figura 4.7

#### TTSChatWidget

El diseño de este widget incluye tanto para el avatar del cliente como para los demás:

- Activación o desactivación para la voz sintética.
- Elección del idioma en el que se escribe.
- Elección de genero del avatar.

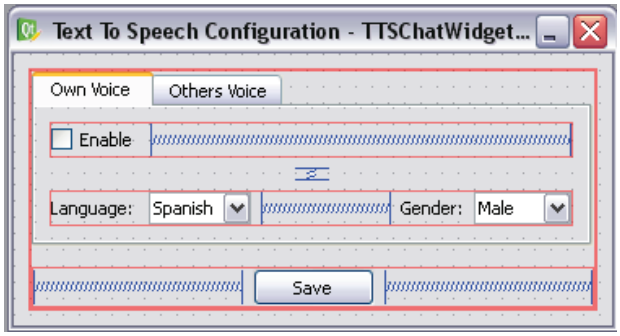


Figura 4.8: Diseño del TTSChatWidget basado en pestañas. Se puede seleccionar el idioma y el genero del avatar.

Se opta por un diseño de pestañas, replicando la información para la voz propia y la del resto de avatares. El resultado de este diseño puede verse en la Figura 4.8

Para ello se han utilizado 2 checkbox (activación TTS), 4 combobox (idioma y genero), espaciadores, y el pushbutton para guardar y cerrar el widget.

La carga de idiomas, se realiza desde Qt Designer, añadiendo los items Spanish y English a uno de ellos y Male y Female al otro.

### 4.4.4. Implementación funcionalidad

En este punto hay que dotar de funcionalidad tanto al botón como al widget.

El botón y el widget sólo han de ser cargados si el servicio TTS está disponible. Por otro lado, al hacer clic en el botón, se ha de abrir el widget de configuración. Al presionar el botón de guardar, la información de configuración ha de ser actualizada.

Por último hay que añadir la llamada al TTS y el paso del mensaje y la configuración.

#### Inicialización:

El CommunicationWidget, al ser iniciado oculta directamente el botón mediante HideTTSChatControls(). Por otro lado, comprueba si el servicio de comunicaciones está disponible. Esta inicialización, se efectúa al iniciar el módulo Ui. El módulo Ui, se inicia después del TTS y del de comunicaciones, por lo que si el módulo existe, el servicio estará registrado y el CommunicationWidget lo detectará.

Si encuentra el servicio, se efectúan algunas conexiones entre señales y slots. La señales, son como su propio nombre indica unos mensajes que se envían cuando ocurre algo. De esta forma, la emisión de la señal, si está conectada con un slot o ranura, que es un método, se ejecuta.

Partes del código se muestran a continuación, donde se ve la conexión entre una señal y un slot.

```
1 if (framework_ && framework_->GetServiceManager())
2 {
3   Communications::ServiceInterface *comm =
4    Framework_->GetService<Communications::ServiceInterface>();
5   if (comm)
6    {
7      ...
8        connect(comm, SIGNAL(InWorldChatAvailable()),
9       SLOT(InitializeInWorldTTS()));
10   }
11 }
```

Tal y como se observa, se conecta el servicio de comunicaciones, con la inicialización del In-World TTS. Esto se hace así, porque la señal InWorldChatAvailable es enviada cuando se crea la sesión de chat. Esta sesión, es creada una vez el usuario se conecta al mundo.

La inicialización del TTS primeramente comprueba si el servicio TTS está o no disponible. En caso de no hacerlo, no hace nada. Por el contrario, si hay servicio, conecta la señal TextMessageReceived proveniente de la sesión de chat con el slot SpeakIncomingMessage. De esta forma, cuando un nuevo mensaje es recibido por la red, se emite la señal que avisa al slot de la llegada de este mensaje.

En este punto, se añade el botón a la parte visual del CommunicationWidget y se crea el TTSChatWidget. Este widget, necesita ser configurado, realizando conexiones entre señales y slots, detallado en el parágrafo *Obtención y almacenamiento de la información*.

La conexión entre el botón y la aparición y desaparición del widget también se realiza aquí.

Por último, se añade a escena el widget de configuración, oculto. A continuación el código:

```
1 void CommunicationWidget::InitializeInWorldTTS()
2 {
```

```
3   //se pide el servicio
4   tts_service_ = framework_->GetService<TTS::
        TTSServiceInterface>();
5   if (!tts_service_)
6     return;
7   //si hay, se conecta la sesión de chat con el slot
        SpeakIncomingMessage y se le pasa el mensaje
8   connect(in_world_chat_session_,
9     SIGNAL(TextMessageReceived(const Communications::
          InWorldChat::TextMessageInterface&)),
10    SLOT(SpeakIncomingMessage(const Communications::InWorldChat
          ::TextMessageInterface&)) );
11
12  //Inicialización del botón
13  ShowTTSChatControls();
14
15  //Si hubiese quedado el widget abierto en otra sesión o se
        crea duplicado
16   //se elimina
17  if (TTS_chat_widget)
18    SAFE_DELETE(TTS_chat_widget);
19
20  //Se crea el widget de configuración
21  TTS_chat_widget = new Communications::TTSChat::TTSChatWidget
        ();
22  //Se crea el objeto para almacenar la información
23  tts_config_ = new Communications::TTSChat::TTSChatConfig();
24  //Se configura el widget con las conexiones pertinentes
25  TTS_chat_widget->ConfigureInterface(tts_config_);
26  //Se conecta el botón TTS para mostrar/ocultar el widget de
        configuración
27  connect(ttsButton, SIGNAL(clicked()), SLOT(
        ToggleTTSChatWidget()));
28  connect(TTS_chat_widget, SIGNAL(TTSstateChanged()),SLOT(
        UpdateTTSChatControls()));
29  Foundation::UiServiceInterface *ui =
30    framework_->GetService<Foundation::UiServiceInterface>();
31  if (ui)
32  {
33    //Se añade el widget a escena, pero oculto. Se mostrará al
```

```
        pulsar el botón TTS.
34      tts_proxy_ = ui->AddWidgetToScene(TTS_chat_widget);
35      tts_proxy_->AnimatedHide();
36    }
37 }
```

**Mostrar u ocultar el TTSChatWidget:**

Un widget, puede ser añadido a la escena mediante el servicio del UiModule mediante AddWidgetToScene. También se puede añadir un widget al menú, pero no es el caso.

Al añadirlo a escena, directamente sale visible. Es por eso que hay que ocultarlo al añadirlo. En este caso, esta activación la gobierna el botón TTS.

En el caso del botón, la adición del widget es automática (CommunicationWidget) pero el propio botón puede ser ocultado o mostrado de la siguiente manera:

```
1 void CommunicationWidget::ShowTTSChatControls()
2 {
3   this->ttsContentWidget->show();
4   this->ttsButton->show();
5 }
6 void CommunicationWidget::HideTTSChatControls()
7 {
8   this->ttsContentWidget->hide();
9   this->ttsButton->hide();
10 }
```

Una vez se pulsa el botón, y gracias a la conexión realizada en la inicialización, se llama a ToggleTTSChatWidget. Este método comprueba que el widget existe, y lo muestra si está oculto y viceversa.

**Obtención y almacenamiento de la información:**

La información contenida en los checkbox y en el TTSChatWidget ha de ser analizada y utilizada para la síntesis.

Para almacenar esta información se crea una clase en el propio widget (TTSChatConfig). Al inicializar el In-World TTS se crea una nueva configuración, y

esta por defecto coge el idioma castellano para las voces y los checbox se inician desactivados. Esto se aplica tanto gráficamente como en el objeto.

La actualización de esta información se realiza mediante una conexión del botón de guardado y un slot creado para actualizar la configuración. Este slot (saveChanges()), definido el en TTSChatWidget, coge el valor de los QCheckbox y los QCombobox y los almacena. El funcionamiento, es el siguiente:

```
1 QString currentLanguage,currentGender;
2 currentLanguage=ui.ownLangComboBox->currentText();
3 currentGender=ui.ownGendComboBox->currentText();
4 if (currentLanguage=="Spanish")
5 {
6   if (currentGender=="Male")
7     tts_config_->setOwnVoice(TTS::Voices.ES1);
8   if (currentGender=="Female")
9     tts_config_->setOwnVoice(TTS::Voices.ES2);
10 }
```

De esta forma, la base de datos de la voz es escogida dependiendo de los valores de genero y idioma seleccionados.

**Uso del servicio:**

El botón se muestra si existe el módulo TTS y hay una sesión de chat. Esto permite mostrar el widget de configuración, que contiene una configuración por defecto. Los cambios se actualizan en el objeto de configuración.

Por otro lado, la conexión con la llegada de un mensaje está hecha, por lo que únicamente hace falta llamar a festival, con la configuración deseada en el momento en el que llega el mensaje.

Esto se hace en dos pasos. Primeramente en SpeakIncomingMessage se obtiene el mensaje de red y se parsea, obteniendo el mensaje limpio. En ese momento se obtiene antes la voz desde el objeto de configuración, utilizando el servicio TTS mediante:

```
1   tts_service_->text2Speech(msg, voice);
```

Sin embargo, hay que hacer una comprobación añadida para ver si sintetizar el texto o no. Primeramente hace falta saber si el mensaje lo ha escrito uno mismo, o pertenece a otro avatar. De esta forma, habrá que coger una voz u otra desde

la configuración. Además, hay que comprobar si el usuario ha activado la síntesis para ese caso. Esta comprobación se lleva a cabo de la siguiente forma:

```
1 if((message.IsOwnMessage() && tts_config_->isActiveOwnVoice())
    || (!message.IsOwnMessage() && tts_config_->
    isActiveOthersVoice()))
```

En tal caso se utiliza el servicio TTS y la síntesis de voz puede escucharse a través del sistema de sonido del equipo.

### 4.4.5. Mejoras

#### Voces

Con esta implementación se han extendido las funcionalidades del chat y se han detectado problemas y carencias relacionados con el módulo, y con el TTS. Una de ellas era la disponibilidad de pocas voces. El añadido de nuevas librerías, afecta a la configuración que se almacena en la clase correspondiente. Pero esta modificación, implica grandes cambios en el widget de configuración. Ahora se disponen de varias voces para el inglés y únicamente una para el finlandes. Con ello, la carga de los combobox ha de ser dinámica dependiendo del idioma que sea seleccionado.

La solución es eliminar los ítems desde Qt Designer y añadir desde código (TTSChatWidget) las opciones. Para ello se crea el método TTSChatWidget::reloadItems() que es ejecutado cada vez que se modifica el idioma. Este añade o elimina ítems del combobox de voces.

#### Muestra de voz

Otra mejora es el botón "demo". Este botón, una vez clicado permite escuchar una muestra de la voz seleccionada. De esta forma, el usuario puede escoger su voz. Esta modificación junto al resto pertenecientes al TTSChatWidget puede verse en la figura 4.9.

Para reproducir las muestras (previamente creadas con el ejecutable y almacenadas en la misma carpeta), se crea un reproductor de audio en Qt, se conecta a la salida de audio y se indica el archivo a reproducir.

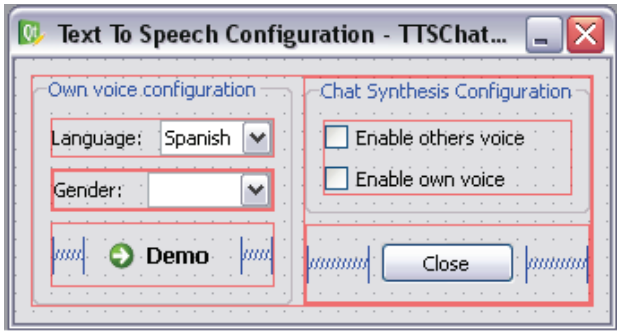


Figura 4.9: Diseño final del TTSChatWidget. El combobox de la voz se carga dinámicamente dependiendo del idioma seleccionado. Se ofrece la posibilidad de oír una muestra de voz para escoger la que más guste. Se eliminan las pestañas ya que la voz de los demás es obtenida a partir de la configuración que cada usuario tiene.

### Configuración de voces

Tal y como se observa en la Figura 4.9 la pestaña de la voz de los otros avatares es eliminada. En este caso se rediseña totalmente esta parte. El hecho de asignar una voz igual a todos los avatares que no sean uno mismo es una gran limitación. Como cada usuario dispone de la configuración propia, por lo que únicamente hay que compartir esta información. Una primera opción es que el propio avatar disponga de un campo de voz, o idioma y genero, de igual manera a como tiene un nombre. Actualmente esto no está disponible, y comentándolo con la comunidad reX, aún pasará bastante tiempo hasta que el editor de avatares y el avatar contenga este tipo de información.

Otra opción es utilizar un archivo XML y almacenar ahí esa información. Cada avatar, dispone de un XML donde se define su forma, sus texturas, etc. Este archivo, llamado descripción de avatar se almacena en el servidor y es distribuido a todos los usuarios conectados al mundo. El XML se construye en el servidor, y no incluye este tipo de información, aunque se podría utilizar un campo llamado *propiedades*. No obstante esta solución requiere modificaciones en la base de datos y en el servidor. Estos cambios, para un caso tan especifico no parecen justificados, y menos aún cuando en el futuro el avatar podrá tener un perfil completo donde se pueda escoger el idioma.

Finalmente se opta por una solución no tan óptima pero válida hasta que los avatares incluyan la información de voz. El mensaje transportará información de voz, justo antes de este y con las etiquetas <voice> y </voice>. Por ejemplo, un

usuario que tenga configurada su voz como catalana femenina enviará antes del mensaje la información: <voice>-CAT2</voice>.

Las implicaciones son que esta información ha de ser eliminada antes de mostrarla en el chat y únicamente utilizada para la llamada de festival, en el caso de que no sea nuestro mensaje (si es nuestro, disponemos un objeto de configuración).

El mensaje es enviado en CommunicationWidget::SendMessageRequested() dentro de CommunicationWidget.cpp, donde se obtendrá la información de la configuración y se añadirá la cabecera. Esto se realizará si se dispone de módulo TTS (si no, no hay configuración).

Por otro lado, el mensaje llega al método *SpeakIncomingMessage*. Aquí hay que separar el mensaje de la voz, antes de llamar al servicio *text2speech*. Esto se realiza de la siguiente manera:

```
1        QString msg;
2        QRegExp rxlen("^<voice>(.*)</voice>(.*)$");
3        int pos = rxlen.indexIn(message.Text());
4        QString Qvoice;
5        TTS::Voice voice;
6
7        if (pos > -1)
8        {
9          Qvoice = rxlen.cap(1);
10         msg = rxlen.cap(2);
11       }
12
13       voice=Qvoice.toStdString();
14
15       tts_service_->text2Speech(msg, voice);
```

Lo que queda es eliminar esta información para que no se muestre ni en la burbuja ni en el chat. Esto se hace en dos sitios, por lo que en ambos, antes de utilizar el texto se comprueba si hay servicio TTS y si lo hay se buscan y eliminan las etiquetas y la voz.

### Cambio del botón de estado

Se trata de una mejora totalmente visual. El botón del TTS, cambia de color (de opacidad) según está sin seleccionar, seleccionado, o pulsado. El usuario


Figura 4.10: En la figura se ven los tres estados del botón. En el estado inicial el TTS está inactivo tanto para los mensajes propios como para los externos. En el segundo caso, el botón es amarillo a causa de que uno u otro está activado. En el caso de que ambos mensajes de chat sean sintetizados el botón se muestra verde, correspondiente al tercer caso en la figura.

está muy acostumbrado a este proceso y su aplicación es hoy en día básica, ya que se ayuda al usuario diciéndole: estás sobre el botón, lo has pulsado. Con esto, el usuario sabe del estado de su acción.

Una vez se ha utilizado, el botón TTS tiene el mismo estado que al principio. Puede resultar interesante hacer que este botón, refleje además el estado de la configuración del TTSChatWidget. De esta forma, se podría saber sin entrar en la configuración, si el TTS del chat está activado o no.

La idea es que si no hay activación alguna (esta se puede dar para el propio mensaje o para el resto de mensajes) el botón se muestra de la forma que ha sido diseñado. En cambio cuando el TTS se activa en alguna (y solo una) de sus formas, el botón pasa a colorearse de amarillo. Cuando incluso la otra opción de síntesis es activada, el botón pasa a ser verde.

Esto se hace, cambiando de manera dinámica la hoja de estilo del widget. En la Figura 4.10 pueden verse los tres estados.

La forma de hacer esto, es mediante señales y slots. La señal es enviada en cuanto cambia el estado de uno de los checkbox. De hecho, se conecta la señal de cambio de los checkbox del TTSChatWidget con un slot, que únicamente envía la señal de modificación. Esta señal, esta conectada al slot UpdateTTSChatControls del CommunicationWidget. Este método es el encargado de leer los checkbox, y actualizar el estilo con una u otra imagen.

## 4.5. Aplicación 2: Lector de notificaciones

Naali dispone de un lector de notificaciones, llamado NotificationManager situado en el módulo Ui (Interfaz de usuario). Este manejador, muestra en la parte superior derecha de la ventana un mensaje cada vez que ocurre una notificación. Por ejemplo en la Figura 4.11 se ve el caso en el que un usuario entra en el mundo.


Figura 4.11: Diseño final del TTSChatWidget. El combobox de la voz se carga dinámicamente dependiendo del idioma seleccionado. Se ofrece la posibilidad de oír una muestra de voz para escoger la que más guste. Se eliminan las pestañas ya que la voz de los demás es obtenida a partir de la configuración que cada usuario tiene.

Las notificaciones suceden cuando ocurren eventos de este tipo en el mundo.

El propósito de esta aplicación es que la notificación sea efectuada mediante síntesis de voz. Esta aplicación se lleva a cabo implementando un módulo sin servicio. Este módulo es muy simple, y únicamente hace la conexión entre ambos servicios.

Sin embargo, antes de crear el módulo hay que conseguir que el *NotificationManager* emita una señal desde su interfaz. Esto requiere que la interfaz *UiServiceInterface* ha de ser extendida añadiendo la señal:

```
1 void Notification(const QString &message);
```

La generación del mensaje se hace con la señal *ShowNotificationCalled* desde el NotificationManager. Hay que crear una nueva conexión que emita la señal de la interfaz cuando *ShowNotificationCalled* sea lanzado:

```
1     connect(owner_->GetNotificationManager(), SIGNAL(
          ShowNotificationCalled(const QString&)),
2       this, SIGNAL(Notification(const QString&)));
```

Esta modificación se hace en el servicio de escena (UiSceneService). Esto es necesario para seguir la filosofía de realXtend, de forma que si el manejador de

notificaciones es sustituido, nada se vea alterado. De esta forma únicamente es necesario conocer la interfaz UiServiceInterface.

La emisión de esta señal, ha de ser conectada con el servicio TTS. Hay diferentes opciones para la creación de esta aplicación. Una es incluirla en el mismo módulo TTS, pero esto va en contra de la arquitectura modular. También se puede incluir en la zona de la primera aplicación, añadiendo un checkbox para activar o desactivar la aplicación e incluso escoger la voz (el idioma es el inglés).

Otra opción es crear un módulo independiente que puede ser activado o desactivado desde la lista de módulos, sin ningún servicio y que su función sea conectar la señal con el servicio TTS.

Se opta por esta opción y se crea el módulo *NotificationTTS*. En la inicialización de este módulo, se obtiene tanto el servicio de interfaz de usuario como el del conversor de texto a voz. Se hace la conexión de la señal con el slot Notification2Speech, implementado en el modulo NotificationTTS:

```
1 void NotificationTTS::Initialize()
2 {
3   if (framework_ &&  framework_->GetServiceManager())
4    {
5     Foundation::UiServiceInterface *ui = framework_->
6         GetService<Foundation::UiServiceInterface>();
7       if (ui)
8     connect(ui, SIGNAL(Notification(const QString&)),
9         SLOT(Notification2Speech(const QString&)));
10    }
11  }
```

Este método coge el mensaje desde la señal emitida (en este caso como argumento se incluye el mensaje), y utiliza el servicio TTS (text2Speech) añadiéndole la voz por defecto EN4.

```
1 void NotificationTTS::Notification2Speech(const QString &
    message)
2 {
3   tts_service_ = framework_->GetService<TTS::
      TTSServiceInterface>();
4   if (!tts_service_)
5     return;
6   tts_service_->text2Speech(message, TTS::Voices.EN4);
7 }
```

## 4.6. Aplicación 3: Componente TTS

El componente, no es una aplicación en sí. No obstante, crear un componente TTS puede proporcionar servicio TTS a una entidad de la escena. Este puede ser el caso, de añadir un texto a un objeto, y que este objeto 3D reproduzca el texto al hacer clic en él. Si en lugar de un objeto se genera un NPC con el componente agregado y con animación generada con el texto, se podría hacer que un "profesor"diese una clase con el contenido de ese texto. También se podría programar el NPC con diversos textos.

Es por ello que el desarrollo de un componente es explicado en la sección de aplicaciones, aunque la aplicación concreta puede ser la de generación de contenido a través de una entidad de escena, dentro de muchas otras posibilidades.

### 4.6.1. Creación del componente EC_TtsVoice

Al igual que los módulos heredan de la clase ModuleInterface, los componentes lo hacen de ComponentInterface, donde ya hay implementados algunos de los métodos.

En el fichero de cabecera hay que declarar la macro:
*DECLARE_EC(EC_TtsVoice)*
Esta macro contiene los métodos necesarios para crear y registrar los componentes de manera automática.

El componente debe tener los métodos necesarios para el manejo de los atributos y las funciones que proporcionará a la entidad mediante el uso del servicio del módulo.

En el caso del componente TTS, se crea la posibilidad genere habla sintética a partir de texto (SpeakMessage) y su atributo es la voz (la base de datos).

En la clase del módulo TTS, al que pertenece el componente (o que éste utiliza), se debe declarar la macro DECLARE_MODULE_EC (EC_TtsVoice) en la inicialización del módulo (load()). De ésta forma, el componente está listo para ser creado tantas veces y para tantas entidades como se desee.

Para crear el componente, una vez realizado todo lo anterior, se debe tener acceso al framework desde la clase en la que lo creemos. Es recomendable comprobar primero si se puede crear de la siguiente forma:

```
1 if ( Framework->GetComponentManager()->CanCreate( EC_TtsVoice::
    TypeNameStatic() ) )
```

Seguidamente se crea el componente:

```
1     boost::shared_ptr< EC_TtsVoice > component = Framework->
2   GetComponentManager()->CreateComponent( EC_TtsVoice::
        TypeNameStatic() )
```

Y se puede añadir a la entidad que lo tendrá desde el propio código:

```
1 if(framework_->GetService<TTS::TTSServiceInterface>())
2 {
3   entity->AddComponent(owner_->GetFramework()->
        GetComponentManager()->
4    CreateComponent(EC_TtsVoice::TypeNameStatic(),"voz"));
5 }
```

Para acceder al componente desde cualquier punto del código necesitamos un puntero a la entidad y realizar la siguiente asignación:

```
1 boost::shared_ptr< EC_TtsVoice > component = entity->
     GetComponent<EC_TtsVoice>();
```

Una vez recuperado podemos utilizar cualquiera de los métodos que implementa, como por ejemplo:

```
1 avatar_voice_->SpeakMessage(msg, avatar_voice_->GetMyVoice());
```

### 4.6.2. Ejemplo de uso

Una vez existe el componente, es posible utilizarlo desde Naali directamente mediante el editor de entidades (Entity-component editor). Al seleccionar una entidad, se pueden ver todos los componentes que tiene y modificarlos. También es posible añadir nuevos. La Figura 4.12 puede verse este proceso.

Si añadimos el componente EC_TtsVoice a una entidad, se puede modificar tanto la voz como el texto a sintetizar. En el ejemplo de la Figura 4.13 se ve como la entidad ha cogido el componente y se configura tanto la voz como el texto. Si la entidad es "touchable", es decir, puede ser clicada, el componente TTS utilizará el servicio TTS cuando se haga clic, y sintetizará el texto para el usuario.

Esto se hace en el método RexLogicModule::EntityClicked obteniendo el componente y utilizándolo de manera similar a como se usan los servicios:

```
1     boost::shared_ptr<EC_TtsVoice> tts_voice = entity->
          GetComponent<EC_TtsVoice>();
```

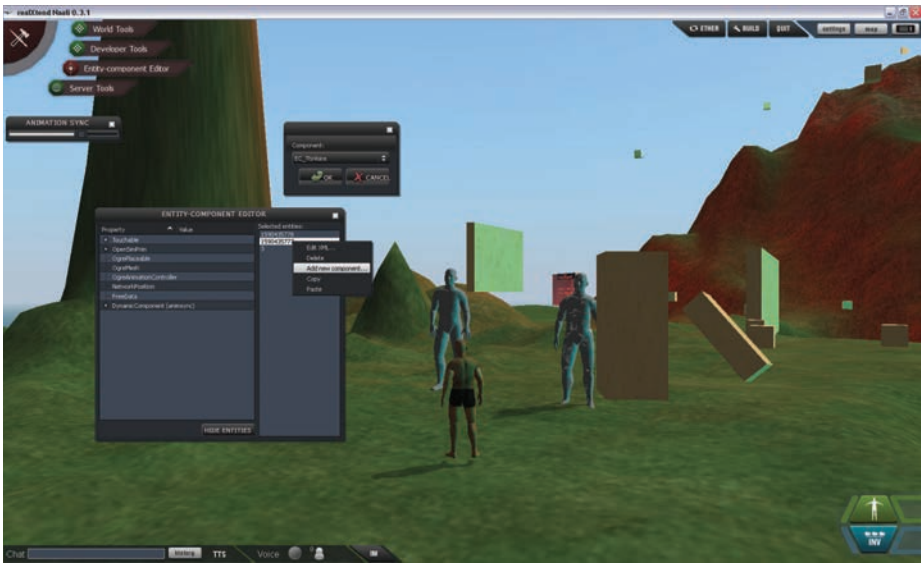


Figura 4.12: Se pueden añadir componentes a través del editor de componentes, para cada entidad. Pueden ser modificados y almacenados. Estos cambios son visibles por todos los usuarios que dispongan de los componentes.

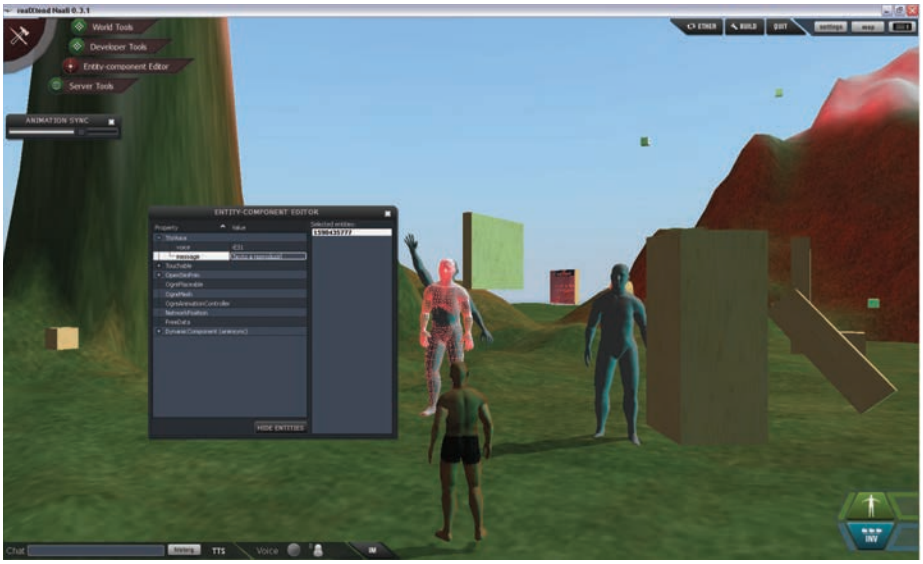


Figura 4.13: Una vez añadido el componente a la entidad, este puede ser configurado, modificando tanto la voz como el texto a sintetizar.

```
2     if (tts_voice)
3           tts_voice->SpeakMessage();
```

El propio componente tiene sus atributos, de donde coge tanto el texto como el idioma y la base de datos a utilizar.

# Capítulo 5

# Validación y evaluación

En este capítulo se realiza un test para validar la implementación y sus mejoras. En concreto se comprueba el funcionamiento del módulo TTS, el servicio, y las tres aplicaciones desarrolladas.

Se trata de comprobar su funcionamiento y detectar errores para obtener una conclusiones y posibles líneas de trabajo. Para ello se ha creado un test que consta de una parte objetiva (validación) y una subjetiva (evaluación). A continuación se describe la muestra utilizada, y las fases del test.

Esta prueba se completa con una comprobación de las funcionalidades no utilizadas por las aplicaciones pero disponibles en el servicio.

Finalmente se muestra un resumen con los resultados de todas las fases. Los datos pueden consultarse en el Apéndice C

## 5.1. Muestra

El test es efectuado en Septiembre de 2010 en el departamento de I+D de ENNE, en Iruñea. Los participantes son los desarrolladores de la empresa junto con Amalia Ortiz, responsable de soluciones I+D.

En total, son 7 personas con conocimientos en los mundos virtuales, con estudios universitarios y experiencia en la materia. Únicamente una es femenina y las edades están comprendidas entre los 20 y los 35 años.

Los datos de sus equipos se muestran en el Cuadro 5.1.

| ID | Equipo | Sistema Operativo | RAM | Procesador |
|---|---|---|---|---|
| 1 | Fijo | Windows XP | 3.25 GB | Intel Xeon 2.33GHz |
| 2 | Fijo | Windows XP | 3.25 GB | Intel Xeon 1.6GHz |
| 3 | Fijo | Windows Vista 64 | 4 GB | Intel Core Quad 2.33GHz |
| 4 | Fijo | Windows XP | 3.25 GB GB | Intel Core Quad 2.4GHz |
| 5 | Fijo | Windows Vista 64 | 4 GB | Intel Core Quad 2.33GHz |
| 6 | Fijo | Windows XP | 3.5 GB GB | Intel Core Quad 2.33GHz |
| 7 | Portatil | Windows Vista 32 | 4 GB | Pentium Dual Core 2GHz |

Cuadro 5.1: Datos de los equipos de los 7 participantes del test.

## 5.2. Test

La metodología del test es descrita a continuación. Primeramente cada participante ha de rellenar una ficha con información relativa al equipo utilizado.

El test puede ser consultado en el Apéndice C.

### 5.2.1. Validación

La validación tiene como objetivo comprobar que las implementaciones funcionan correctamente. La validación se plantea como técnica, describiendo como usuarios a un equipo de perfil técnico. Esta validación se subdivide en dos fases.

En la primera fase, todos los participantes tienen activado el módulo TTS y las 3 aplicaciones. Todos ellos se conectan al mismo mundo virtual dentro de un servidor interno lanzado desde uno de los equipos y se mueven en él durante 10 minutos. En este tiempo, el usuario ha de probar todas las aplicaciones TTS, comunicándose a través del chat, probado diferentes voces, o añadiendo componentes a entidades.

Durante este periodo además se coordina una prueba de carga con todos los usuarios escribiendo a la vez en el chat con el TTS activado.

Tras concluir el tiempo, los participantes responden a un total de 32 preguntas. La respuesta ha de ser afirmativa si todo funciona o añadir un comentario en caso contrario.

La segunda fase es similar a la primera, sin embargo, en este caso sólo parte de los participantes tienen el módulo activo. En concreto 3 usuarios mantienen la configuración, 2 utilizan el cliente sin modificaciones y 2 utilizan el cliente modificado con el módulo desactivado.

Tras 10 minutos, todos han de contestar a otras preguntas, diferentes si se tiene activado el módulo TTS (6 preguntas) o está desactivado (5).

### 5.2.2. Evaluación

La tercera fase del test es la evaluación. Como el propio nombre indica en este caso el usuario debe cumplimentar un cuestionario, valorando diferentes aspectos dentro del margen 0-10 siendo 10 la máxima puntuación.

Este cuestionario se rellena una vez han terminado las dos anteriores fases.

Se sabe que este test no es una evaluación de usabilidad válido, simplemente se aprovecha la validación técnica para hacer una pequeña entrevista subjetiva a los usuarios que han participado en el test de pruebas

### 5.2.3. Comprobación de funcionalidad no utilizada

Las aplicaciones disponibles no utilizan todas las funcionalidades del servicio TTS. Para comprobar su correcto funcionamiento se efectúa un test a parte mediante la modificación de una de las aplicaciones.

En concreto, quedan por utilizar:

1. Reproducción de un archivo de texto (file2Speech)
2. Almacenamiento en formato WAV de un texto (text2WAV)
3. Almacenamiento en formato WAV de un archivo (file2WAV)
4. Almacenamiento en formato PHO de un texto (text2PHO)
5. Almacenamiento en formato PHO de un archivo (file2PHO)

Para comprobar su funcionamiento se crea un fichero de texto *ejemplo.txt* con un texto aleatorio y se añade a la aplicación del componente:

```
1 void EC_TtsVoice::SpeakMessage()
2 {
3   if(!ttsService_)
4     ttsService_ = framework_->GetService<Tts::
          TtsServiceInterface>();
5
6   ttsService_->text2Speech(message_.Get(),voice_.Get().
       toStdString());
```

```
7    ttsService_->text2PHO(message_.Get(),"nombrepho", voice_.Get
         ().toStdString());
8    ttsService_->text2WAV(message_.Get(),"nombrewav", voice_.Get
         ().toStdString());
9
10   ttsService_->file2Speech("ejemplo.txt",voice_.Get().
         toStdString());
11   ttsService_->file2PHO("ejemplo.txt","nombrephoejemplo",voice_
         .Get().toStdString());
12   ttsService_->file2WAV("ejemplo.txt","nombrewavejemplo",voice_
         .Get().toStdString());
13 }
```

## 5.3. Resultados y Análisis

A continuación se muestra un resumen de los resultados para pasar a analizarlos. Todos estos datos están disponibles en el Apéndice C.

Los resultados se muestran junto a las preguntas con un determinado color. En el caso de las primeras dos fases (validación) el color verde significa que todos han coincidido en la respuesta afirmativa. El rojo significa que todos han contestado que no a la pregunta y el amarillo es utilizado cuando se ha contestado de forma diferente.

En la tercera fase (evaluación) el color verde indica que el resultado del cuestionario promedio es superior a 7.5 puntos sobre 10. El color rojo significa un resultado inferior a 5, y amarillo entre ambos resultados.

### 5.3.1. Validación

La primera fase de validación consta de 32 preguntas, representadas en la Figura 5.1. Estas preguntas son contestadas tras un periodo de 10 minutos donde 7 participantes utilizan el cliente Naali modificado con las tres aplicaciones.

Una vez terminado este periodo, cada participante contesta si está de acuerdo con la pregunta o si no. Como puede observarse en la Figura 5.1 28 de las 32 funcionalidades son validadas correctamente.

En el primer bloque (FI.1-FI.10) se comprueba el correcto funcionamiento de la aplicación de chat, sus checkbox, la actualización y envío de la voz en el mensaje y el parseo y eliminación de la etiqueta de voz en los distintos puntos

donde se muestra el mensaje. Como se ve en los resultados promedio, queda todo validado.

El segundo bloque (FI.11-FI.14) se centra en preguntas que están íntimamente relacionadas con el corpus de voz, y por lo tanto con la base de datos de prueba (fechas, caracteres, etc). En este punto se detectan algunos caracteres, raramente utilizados y que a veces no se corresponden con el idioma (por ejemplo, ñ en inglés, ·, ¿? o $^{oa}$). También se han detectado cortes en la voz femenina en castellano cuando hay varias consonantes juntas.

El tercer bloque de preguntas (FI.15-FI.22) se centra en aspectos más visuales (botón y sus estados/colores) y en algunas de las mejoras implementadas, como el botón de demo y la carga dinámica de idiomas. En este apartado no hay ningún problema para ninguno de los usuarios aunque se remarca la necesidad de resaltar algo más los colores de los botones para que su efecto quede más claro.

El cuarto bloque (FI.23-F.I28) está relacionado con pruebas de carga. La mejora de llamar Festival en el *background* funciona y el audio se recibe en un tiempo adecuado de forma sincronizada con la burbuja. La carga de los equipos está por debajo del 60 % en alguno de los casos y por encima en otros. Si se observan los datos del Apéndice C hay un usuario que no ha contestado a las preguntas. Esto es así porque el servidor estaba en su equipo. También se ve que en algún caso la carga ha sobrepasado este 60 %. Hay que comentar que en este punto los 7 participantes escribían frases constantemente, obligando a lanzar instancias de Festival cada instante.

A continuación (FI.29-FI.32) se validan completamente tanto la aplicación de componentes como la de notificaciones.

Los resultados de la segunda parte del test se muestran en la Figura 5.2. Tres de los usuarios continuaron la conversación con el módulo TTS activado y como se ven en los resultados la diferencia fue que no oía la síntesis de los usuarios sin TTS aún teniendo la opción activa.

Los otros cuatro participantes tienen el TTS desactivado en esta fase, aunque dos de ellos utilizaron el cliente Naali sin modificar y otros dos utilizaron el cliente modificado con el módulo inactivo. El resultado para ambos es diferente. Con el cliente modificado, las etiquetas de voz no se ven ni en la burbuja, ni en el historial, ni en la línea de chat, siendo el efecto totalmente invisible para ellos.

En el caso de utilizar el cliente por defecto de reX, las etiquetas no se eliminan y se muestran junto con el texto en el caso de la burbuja y el texto que

| | FASE I: Módulo TTS activado | |
|---|---|---|
| FI.1 | Se escucha audio de los demás si se activa su síntesis | |
| FI.2 | No se escucha audio del otro si no tengo activada su síntesis | |
| FI.3 | Se escucha tu propio audio si lo tienes activado | |
| FI.4 | No se escucha tu propio audio si no lo tienes activado | |
| FI.5 | El receptor de tu audio lo escucha con tus opciones de voz | |
| FI.6 | Cuando modifico las opciones de voz, el siguiente texto enviado, el receptor lo recibe con las nuevas opciones de voz | |
| FI.7 | Leo correctamente mi burbuja cuando hablo | |
| FI.8 | Leo correctamente su burbuja cuando habla | |
| FI.9 | Leo correctamente la línea de chat que aparece/desaparece | |
| FI.10 | Leo correctamente el historial del chat | |
| | | |
| FI.11 | Cuando escribo &%$·ºHolá!¿!98 el sintetizador lo reproduce bien | |
| FI.12 | Cuando escribo 15/2/2007 el sintetizador lee la fecha correctamente. | |
| FI.13 | Cuando escribo 14:30 el sintetizador lee la hora correctamente | |
| FI.14 | No he escrito ningún texto que el sintetizador no pudiera sintetizar correctamente | |
| | | |
| FI.15 | Cuando tengo activa la opción de TTS escuchar todo el botón está verde | |
| FI.16 | Cuando tengo activa la opción de TTS escuchar a todos menos a mí, el botón está amarillo | |
| FI.17 | Cuando tengo activa la opción de TTS escuchar a nadie, el botón está blanco | |
| FI.18 | La modificación del color se efectúa incluso sin cerrar el widget | |
| FI.19 | Cuando pongo el ratón encima del botón, cambia la opacidad | |
| FI.20 | Cuando hago click en el botón, cambia la opacidad | |
| FI.21 | Las voces a elegir cambian de forma dinámica al cambiar de idioma | |
| FI.22 | Se escucha la demo de la voz al hacer click en el botón. | |
| | | |
| FI.23 | El cliente no se congela cuando se sintetiza voz | |
| FI.24 | Se recibe el audio en un tiempo adecuado | |
| FI.25 | Se recibe el audio de forma sincronizada con la burbuja del texto sobre el avatar | |
| FI.26 | La carga de mi PC está dentro del límite (menor del 60%) | |
| FI.27 | Cuando todos los participantes están escribiendo constantemente (cada uno una frase) el sistema lo soporta | |
| FI.28 | Cuando todos los participantes están escribiendo constantemente (cada uno una frase) se escuchan los audios a la vez | |
| | | |
| FI.29 | Las notificaciones de entra/sale usuario se sintetizan | |
| | | |
| FI.30 | Se puede añadir un componente TTS a una entidad | |
| FI.31 | Se puede configurar su voz y el texto | |
| FI.32 | Se puede reproducir el texto al hacer clic en la entidad | |

Figura 5.1: Resumen de la primera fase donde pueden leerse las preguntas en la columna izquierda y el resultado en colores en la columna derecha. Verde significa que todos han contestado afirmativamente, rojo que todos negativamente y amarillo que ha habido ambos tipos de respuesta.

| | FASE II | |
|---|---|---|
| FII.1 | No oigo su voz pero sí la mía cuando tengo esa opción activada | |
| FII.2 | No oigo ni su su voz ni la mía cuando tengo esa opción activada | |
| FII.3 | Cuando tengo activa la opción de TTS escuchar todo el botón está verde | |
| FII.4 | Cuando tengo activa la opción de TTS escuchar a todos menos a mí el botón está amarillo | |
| FII.5 | Cuando tengo activa la opción de TTS escuchar a nadie, el botón está blanco | |
| FII.6 | Se ha detectado alguna diferencia respecto al chat con un usuario con TTS | |
| | | |
| FII.7 | Leo correctamente mi burbuja cuando hablo | |
| FII.8 | Leo correctamente su burbuja cuando habla | |
| FII.9 | Leo correctamente la linea de chat que aparece/desaparece | |
| FII.10 | Leo correctamente el historial del chat | |
| FII.11 | Puedo chatear sin problema con un usuario | |

Figura 5.2: Resumen de la segunda fase donde pueden leerse las preguntas en la columna izquierda y el resultado en colores en la columna derecha. Puede verse que 3 de los 7 participantes han tenido TTS activado y 4 no, estando estos últimos con el fondo azul. Verde significa que todos han contestado afirmativamente, rojo que todos negativamente y amarillo que ha habido ambos tipos de respuesta

se desvanece. El historial, elimina estas etiquetas aunque la información de voz queda ahí.

Por lo demás la comunicación puede realizarse correctamente.

### 5.3.2. Evaluación

Las preguntas formuladas y el promedio de los resultados se muestra en la Figura 5.3. En verde se muestran las respuestas con valoraciones más positivas, que como se ve, son la gran mayoría de las 15.

Las preguntas respondidas por los usuarios de perfil técnico muestran datos muy positivos de todas las aplicaciones y de la integración del módulo TTS. Sin embargo, es importante tener en cuenta que la muestra se compone de un equipo de desarrolladores, y que no se trata del jugador o usuario final.

Las preguntas con menor aceptación indican que al usuario no le gusta ver sólo texto al chatear y que prefiere escuchar su voz en una conversación sintetizada, evitando la parcialmente sintetizada (únicamente se sintetiza una de las voces).

La utilidad del lector de notificaciones aprueba la prueba pero no tiene mucha aceptación.

| | FASE III: Cuestionario | |
|---|---|---|
| FIII.1 | Te ha parecido intuitiva la forma de saber si tienes el TTS activado o no? | 8,71 |
| FIII.2 | ¿Te parece fácil de usar esta funcionalidad? | 9,29 |
| FIII.3 | ¿Crees que escuchar la voz de las personas con las que estas chateando te ayuda a seguir mejor la conversación? | 7,71 |
| FIII.4 | ¿Te gusta escuchar la voz de las personas con las que chateas? | 7,71 |
| FIII.5 | ¿Te gusta ver sólo texto cuando estás chateando? | 4,29 |
| FIII.6 | En una conversación con voz sintetizada, ¿Crees que es más fácil seguir la conversación cuando escuchas tu voz? | 7,71 |
| FIII.7 | En una conversación con voz sintetizada, ¿Crees que es más fácil seguir la conversación cuando no escuchas tu voz? | 4,71 |
| FIII.8 | ¿Crees que escuchar la voz de las personas con las que estas chateando es más entretenido? | 8,43 |
| FIII.9 | Cuando se en una conversación dos usuarios hablan a la vez, ¿crees que el sistema lo gestiona de forma correcta? | 8,71 |
| FIII.10 | ¿Te parece fácil entender el audio sintetizado en castellano? | 8,43 |
| FIII.11 | ¿Te parece fácil entender el audio sintetizado en inglés? | 8,57 |
| | | |
| FIII.12 | ¿Te parece útil que las notificaciones sean sintetizadas? | 6,57 |
| | | |
| FIII.13 | ¿Crees que añadir componentes de síntesis a entidades es útil? | 9,29 |
| | | |
| FIII.14 | En general, ¿Como valoras la integración del módulo TTS y sus aplicaciones? | 8,71 |
| FIII.15 | ¿Ves útil la implementación del módulo TTS en la plataforma de VW? | 9,71 |

Figura 5.3: Resultado promedio de la tercera fase del test, donde se responde al cuestionario con las preguntas formuladas en la columna izquierda. A partir de 7,5 el fondo es verde, por debajo de 5 es rojo y entremedio es amarillo.

| | |
|---|---|
| Reproduce un archivo | Sí |
| Guarda a wav un texto | Sí |
| Guarda a wav un archivo | Sí |
| Guarda a PHO un texto | Sí |
| Guarda a PHO un archivo | Sí |

Figura 5.4: Resultado de la prueba manual, añadiendo en el componente código para probar el servicio TTS al completo.

### 5.3.3. Comprobación de funcionalidad no utilizada

La funcionalidad no utilizada por las aplicaciones se comprueba en esta fase del test. Esto es efectuado de forma manual tal y como se ha detallado, y el resultado se puede ver en la Figura 5.4.

El resultado es que el texto puede almacenarse sin problemas en un fichero WAV, y también funciona el servicio de guardar información fonética (PHO). En caso de utilizar un fichero de texto, las tres pruebas (File2Speech, File2WAV y File2PHO) dan un resultado positivo, y quedan validados.

# Capítulo 6

# Conclusiones y líneas futuras

## 6.1. Conclusiones

El objetivo de este proyecto es integrar un sistema TTS dentro de una plataforma de creación de mundos virtuales. Para ello se ha hecho un estado del arte de los sistemas TTS OpenSource escogiendo Festival Speech Synthesis System y modificándolo para extenderlo y adaptar sus funcionalidades y sus librerías a la plataforma de mundos virtuales.

La integración de este sistema modificado se ha llevado a cabo en el cliente Naali, mediante un módulo y su correspondiente servicio. Se ha utilizado este servicio para la creación de tres aplicaciones, siendo todas diferentes: Chat TTS (parte gráfica, extensión de código de la comunidad y creación de nuevo código), Lector de notificaciones TTS (creación de un módulo sin servicio, utilizando otros servicios) y el Componente TTS que puede ser agregado a entidades de la escena.

Finalmente se ha testeado toda la funcionalidad implementada, sea utilizada por aplicaciones o no.

Mediante este trabajo se han obtenido diversas conclusiones en todos los ámbitos donde se ha trabajado, que pasan a ser comentados a continuación.

En cuanto los sistemas TTS, los comerciales parten con mucha ventaja respecto los de código libre, principalmente a causa de sus corpus de voz. Como ventaja de Festival, hay que decir que se muestra como un sistema muy completo, superior al resto de código abierto, con un proyecto llamado FestVox muy prometedor, pero que necesita mejorar y crear voces de calidad.

Otra ventaja de Festival es que es un sistema relativamente sencillo de utilizar, aunque haya que realizar diversas modificaciones en sus librerías para compilarlo en Windows. Estos problemas desaparecerán en futuras versiones de Festival, pero a día de hoy permanecen ahí.

En cuanto a las voces, las HTS presentan una relación de espacio/calidad muy buena, ocupando realmente un espacio muy reducido para ofrecer una buena calidad. En una integración en el lado cliente este es un aspecto importante, y aún más en equipos portátiles o móviles.

Sin embargo, una de las desventajas que presentan las voces libres son sus grandes limitaciones, relacionadas con caracteres raramente utilizados en la comunicación escrita, letras no utilizadas en determinados idiomas, o al tener una pobre *addenda*.

En cuanto a los mundos virtuales y sus plataformas, hay que remarcar que su futuro es muy prometedor y sin duda realXtend será una referencia tanto para la comunidad de código abierto como para las empresas. Su comunidad es muy potente y el desarrollo es continuo. La apuesta por esta plataforma parece haber sido una elección acertada.

Además de tener futuro, su arquitectura modular basada en componentes permite la creación y adición de implementaciones de una forma sencilla. Los conocimientos adquiridos en este terreno han permitido crear un módulo base, un módulo TTS con un servicio propio, y otro módulo con funcionalidad, así como un componente nuevo y entender otros módulos y flujos y modificarlos.

El contacto con la plataforma también ha propiciado un aprendizaje en lenguajes de programación y en entornos de creación de contenido y documentación.

La comunicación con la comunidad ha sido constante, utilizando sus listas de distribución y canales IRC, abriendo un canal de comunicación entre ENNE y realXtend de forma que en el futuro más inmediato el código y el desarrollo pueda ser compartido con ellos. Para lograr esto, se han aprendido las convenciones de programación de reX, así como las de documentación.

Tal y como se menciona en el Capítulo 5 el test de validación y evaluación es efectuado por un equipo técnico y debería de realizarse con jugadores del mundo habituales. Los resultados obtenidos muestran que el usuario prefiere escuchar su voz si se sintetiza la de los demás y que no le gusta leer únicamente el texto.

Los participantes valoran muy positivamente la integración del sistema TTS y sus posibles usos debido a que los MV se utilizan principalmente para conectar

a un conjunto de personas.

Los test de validación muestran que todas las funcionalidades implementadas funcionan de la manera esperada, cumpliendo con los objetivos propuestos al comienzo del proyecto.

Se ha visto que la voz presenta ventajas en aplicaciones de socialización [12] (donde los usuarios no quieren utilizar su voz real), de aprendizaje [11][5] (mejora la retención de datos), de turismo [2] (permite crear avatares guía con capacidad de habla), de ayuda (personas con dificultades físicas y de visión [13], etc...

Hay que recalcar que es importante mejorar las voces, su *addenda* y su análisis, incluyendo abreviaturas extendidas como por ejemplo: xq n t vnes? (¿Por qué no te vienes?).

Sin embargo, disponer de un módulo TTS integrado en la plataforma de mundos virtuales permite crear contenido audible de forma muy económica. Generar audios con locutores profesionales supone un coste importante además de ser muy poco flexible. Una aplicación de creación de contenido con TTS permitiría crear y modificar contenido de forma rápida y dinámica a un coste prácticamente nulo.

Resulta útil el hecho de sintetizar el texto del mundo también para no estar pendiente todo el rato de esa aplicación, en caso de que se tengan que realizar otras tareas, o en el caso de que en el propio mundo se este realizando alguna acción que impida estar leyendo el texto.

Con lo cual y como conclusión final, hay que remarcar que gracias a este proyecto la comunidad realXtend ha extendido la funcionalidad de su plataforma con la síntesis de voz. Se trata de una funcionalidad que no poseen muchas plataformas y que sin embargo puede dar a los mundos virtuales un valor añadido.

## 6.2. Líneas futuras

En un futuro, se deberían de disponer de voces de mayor calidad y no tan limitadas. Esto permitirá que la interacción sea más natural y no canse ni sorprenda al jugador. Además la inclusión de emociones o expresiones mediante el uso de un sistema de marcado como el estándar Virtual Human Markup Language (VHML)[14] o mediante la obtención y sincronización de gestos a través de una cámara permitirá una inmersión mucho mayor combinada con sistemas 3D sobre los que ya trabaja realXtend con sistemas CAVE (Cave Automatic Virtual Environment) [7].

El hecho de crear un servicio TTS que va mas allá de su uso en las aplicaciones permite obtener información fonética y en forma de archivo WAV que no han sido explotadas en este proyecto. Se deberían estudiar diversos caminos, como puede ser la combinación de la información fonética generada con la sincronía de los labios de un avatar o de un NPC, el posible uso de archivos WAV y su gestión, así como su posible compresión en formatos como OGG o MP3.

Habría que estudiar entonces la posibilidad de utilizar los servicios que se usan, en nuevas aplicaciones.

El Chat TTS es muy completo pero puede ser mejorado sobre todo en su envío de información de la voz a utilizar para la síntesis. Lo ideal es que esta información no viaje por la red en cada mensaje, aunque sea dinámica, y corresponda a información del propio avatar. La futura configuración del avatar debería de permitir añadir y modificar esta información, aunque habría que estudiar cual es el mejor método: utilizar un fichero XML, un componente de TTS, configuración propia del avatar...

Otra mejora puede ser activar o desactivar la síntesis para un determinado avatar o establecer comunicaciones privadas. Esto podría efectuarse de la misma manera en la que está la voz por IP (lista de participantes) o con la próxima implementación de chat privado dentro del mundo.

El lector de notificaciones debería de poder activarse o desactivarse, y no actuar cuando uno se conecta al mundo (notifica de todos los usuarios conectados). Se podría añadir que la notificación se efectuase únicamente cuando entre o salga un determinado avatar.

El componente es una aplicación a explotar, ya que se puede crear contenido en el mundo de una forma muy fácil. Sin embargo, necesita una interfaz intuitiva o un menú, así como la posibilidad de utilizar ficheros de texto para por ejemplo, agregárselos a un NPC, que gracias a la información fonética nos pueda dar una clase magistral. La interfaz debería de listar las opciones de idioma de manera similar a la aplicación de chat en lugar de tener que editarse a mano.

También se pueden extraer algunas conclusiones de los tests de evaluación y validación realizados. Estos han sido ejecutados tanto en servidores externos como dentro de la LAN, y convendría realizar nuevamente pruebas de carga en ambos, y observar si el problema de saturación es únicamente debido al TTS o viene intrínseco en el In-World Chat.

El funcionamiento bajo diferentes sistemas operativos y distintas versiones ha

sido satisfactorio, sin embargo, hay que estudiar la portabilidad a otras plataformas, incluyendo las móviles, para que Festival funcione correctamente.

Más adelante, la implementación en el lado del servidor puede ser otra vía de estudio sobre todo por si se decide utilizar un sistema conversor de texto a voz privativo, para evitar las licencias. Además si la librería de voces aumenta, el peso para el cliente puede ser elevado. Si se utilizase el módulo TTS en el servidor, podrían utilizarse corpus de voz completos sin darle tanta importancia al espacio y centrándose en la calidad.

# Bibliografía


[1] B.S Atal and N. David. *On synthesizing natural-sounding speech by linear prediction.* ICASSP, pp 44-47, 1979. [cited at p. 32]

[2] Helmut Berger, Michael Dittenbach, Dieter Merkl, Anton Bogdanovych, Simeon Simoff, and Carles Sierra. *Opening new dimensions for e-Tourism.*, volume 11. Springer-Verlag, virtual reality edition, 2007. [cited at p. 1, 89]

[3] A. W. Black and K. A. Lenzo. *Building Synthetic Voices.* Language Technologies Institute, Carnegie Mellon University., http://festvox.org/bsv/, 2007. [cited at p. 49]

[4] Alan W. Black and Paul A. Taylor. *The Festival Speech Synthesis System: System documentation.* Human Communciation Research Centre, University of Edinburgh, Scotland, http://www.cstr.ed.ac.uk/projects/festival.html, 1997. [cited at p. 36]

[5] Robert J. Bloomfield. *Worlds for Study: Invitation - Virtual Worlds for Studying Real-World Business.* Cornell University - Samuel Curtis Johnson Graduate School of Management, 2007. [cited at p. 1, 89]

[6] Antonio Bonafonte, Lourdes Aguilar, Ignasi Esquerra, Sergio Oller, and Asunción Moreno. *Recent work on the FESTCAT database for speech synthesis.* I Joint SIG-IL / Microsoft Workshop on Speech and Language Technologies for Iberian Languages (Porto Salvo, Portugal), http://www.lsi.upc.edu/~nlp/papers/bonafonte09_sltech.pdf, 2009. [cited at p. 51]

[7] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. *Surround-screen projection-based virtual reality: the design and implementation of the CAVE.* 1993. [cited at p. 89]

[8] T. Dutoit and H. Leich. *MBR-PSOLA: Text-To-Speech synthesis based on an MBE re-synthesis of the segments database.* Speech Communication, 13 (3-4), pp. 435-440., 1993. [cited at p. 32]

[9] Moulines E. and Charpentier F. *Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones.* Speech Communication, vol. 9, pp. 453-467, December, 1990. [cited at p. 32]

[10] Guadalinex. *Desarrollo de las voces sintéticas de Guadalinex.* http://forja.guadalinex.org/frs/download.php/155/DesarrolloVocesSinteticasGuadalinex.pdf, 2007. [cited at p. 49, 127]

[11] Stacy Kluge and Liz Riley. *Teaching in Virtual Worlds: Opportunities and Challenges*, volume 5. Issues in Informing Science and Information Technology, http://proceedings.informingscience.org/InSITE2008/IISITv5p127-135Kluge459.pdf, 2008. [cited at p. 1, 89]

[12] Paul R. Messinger, Eleni Stroulia, Kelly Lyons, Michael Bonea, Run H. Niud, Kristen Smirnova, and Stephen Perelgut. *Virtual worlds - past, present, and future: New directions in social computing.*, volume 47 , Issue 3. Elsevier Science Publishers B. V, 2009. [cited at p. 1, 89]

[13] Vicki Hanson Shari Trewin, Mark Laff and Anna Cavender. *Exploring Visual and Motor Accessibility in Navigating a Virtual World.*, volume 2 of *2*. ACM Transactions on Accessible Computing (TACCESS), 2009. [cited at p. 1, 89]

[14] VHML. *Virtual Human Markup Language Specification.* http://www.interface.computing.edu.au/documents/VHML/vhml.pdf, 2001. [cited at p. 89]

# Appendices

## Apéndice A

# Modificaciones del Festival Speech Synthesis System

```
1 static void festival_main(int argc, char **argv)
2 {
3     EST_Option al;
4     int stdin_input,interactive;
5     EST_Litem *p;
6     EST_StrList files;
7     int real_number_of_files = 0;
8     int heap_size = FESTIVAL_HEAP_SIZE;
9
10 /* <AÑADIDO>*/////////////////
11
12   EST_String texto; // Cadena donde se pasa el texto, de la
         cadena o del archivo.
13   EST_Wave wave; // Para almacenar el wave antes de guardarlo
14   bool flagAWP=false; // Flag para saber si hay un texto O
         archivo, y mirar las opciones -A -W -P
15
16 /* </AÑADIDO>*/////////////
17
18     if (festival_check_script_mode(argc,argv) == TRUE)
19     {
20   // Need to check this directly as in script mode args are
```

```
21  // passed for analysis in the script itself
22  return;
23    }
24
25    parse_command_line(argc, argv,
26  EST_String("Usage:\n")+
27  "festival␣␣<options>␣<file0>␣<file1>␣...\n"+
28        "In␣evaluation␣mode␣\"filenames\"␣starting␣with␣(␣are␣
              evaluated␣inline\n"+
29  "Festival␣Speech␣Synthesis␣System:␣"+ festival_version +"\n"+
30  "-q␣␣␣␣␣␣␣␣␣␣␣Load␣no␣default␣setup␣files\n"+
31  "--libdir␣<string>\n"+
32        "␣␣␣␣␣␣␣␣␣␣␣␣␣Set␣library␣directory␣pathname\n"+
33        "-b␣␣␣␣␣␣␣␣␣␣␣Run␣in␣batch␣mode␣(no␣interaction)\n"+
34  "--batch␣␣␣␣␣␣␣Run␣in␣batch␣mode␣(no␣interaction)\n"+
35  "--tts␣␣␣␣␣␣␣␣␣Synthesize␣text␣in␣files␣as␣speech\n"+
36  "␣␣␣␣␣␣␣␣␣␣␣␣␣no␣files␣means␣read␣from␣stdin\n"+
37  "␣␣␣␣␣␣␣␣␣␣␣␣␣(implies␣no␣interaction␣by␣default)\n"+
38  "-i␣␣␣␣␣␣␣␣␣␣␣Run␣in␣interactive␣mode␣(default)\n"+
39  "--interactive\n"+
40        "␣␣␣␣␣␣␣␣␣␣␣␣␣Run␣in␣interactive␣mode␣(default)\n"+
41  "--pipe␣␣␣␣␣␣␣␣Run␣in␣pipe␣mode,␣reading␣commands␣from\n"+
42  "␣␣␣␣␣␣␣␣␣␣␣␣␣stdin,␣but␣no␣prompt␣or␣return␣values\n"+
43  "␣␣␣␣␣␣␣␣␣␣␣␣␣are␣printed␣(default␣if␣stdin␣not␣a␣tty)\n"+
44  "--language␣<string>\n"+
45        "␣␣␣␣␣␣␣␣␣␣␣␣␣Run␣in␣named␣language,␣default␣is\n"+
46  "␣␣␣␣␣␣␣␣␣␣␣␣␣english,␣spanish␣and␣welsh␣are␣available\n"+
47  "--server␣␣␣␣␣␣Run␣in␣server␣mode␣waiting␣for␣clients\n"+
48  "␣␣␣␣␣␣␣␣␣␣␣␣␣of␣server_port␣(1314)\n"+
49  "--script␣<ifile>\n"+
50        "␣␣␣␣␣␣␣␣␣␣␣␣␣Used␣in␣#!␣scripts,␣runs␣in␣batch␣mode␣
              on\n"+
51  "␣␣␣␣␣␣␣␣␣␣␣␣␣file␣and␣passes␣all␣other␣args␣to␣Scheme\n"+
52  "--heap␣<int>␣{1000000}\n"+
53        "␣␣␣␣␣␣␣␣␣␣␣␣␣Set␣size␣of␣Lisp␣heap,␣should␣not␣
              normally␣need\n"+
54        "␣␣␣␣␣␣␣␣␣␣␣␣␣to␣be␣changed␣from␣its␣default\n"+
55  "-v␣␣␣␣␣␣␣␣␣␣␣Display␣version␣number␣and␣exit\n"+
56  "--version␣␣␣␣␣Display␣version␣number␣and␣exit\n"+
57  "\n"+
```

```
58 /* <AÑADIDO>*////////////////////////
59   "
         **********************************************************\
         n"+
60   "*****   ************    MORE OPTIONS   ************
         *********\n"+
61   "
         **********************************************************\
         n"+
62   "\n"+
63   "The options are:\n"+
64   "     Input:    File or Text.\n"+
65   "     Output:   Audible, Phonetic information, Wave.\n"+
66   "\n"+
67   "-F <string>     <string> filename is the input file \n"+
68   "-T <string>     <string> is an input text between quotation
         marks\n"+
69   "-A              Audible output is ON\n"+
70   "-P <string>     Phonetic file is saved in <string> filename\
         n"+
71   "-W <string>     WAV Output is saved in <string> filename\n"+
72   "-ES1    Castillian male voice\n"+
73   "-ES2    Castellian female voice\n"+
74   "-EN1    US English male voice 1\n"+
75   "-EN2    US English female voice 1\n"+
76   "-EN3    US English male voice 2\n"+
77   "-EN4    US English female voice 2\n"+
78   "-EN5    US English male voice 3\n"+
79   "-EN6    US English female voice 4\n"+
80   "-CAT1   Catalan male voice\n"+
81   "-CAT2   Catalan female voice\n"+
82   "-FI     Finnish male voice",
83 /* </AÑADIDO>*////////////////////////
84       files, al);
85
86
87   //Version
88     if ((al.present("-v")) || (al.present("--version")))
89     {
90     printf("%s: Festival Speech Synthesis System: %s\n",
```

```
91            argv[0],festival_version);
92      exit(0);
93      }
94    //Set library directory
95      if (al.present("--libdir"))
96      festival_libdir = wstrdup(al.val("--libdir"));
97      else if (getenv("FESTLIBDIR") != 0)
98      festival_libdir = getenv("FESTLIBDIR");
99
100   //Set size of Lisp heap
101   if (al.present("--heap"))
102     heap_size = al.ival("--heap");
103
104     festival_initialize(!al.present("-q"),heap_size);
105
106   //Language
107     if (al.present("--language"))
108     festival_init_lang(al.val("--language"));
109
110 /* <AÑADIDO> *//////////////////////////////////
111
112   if (al.present("-ES1"))
113     festival_eval_command("(
          voice_JuntaDeAndalucia_es_pa_diphone)");
114
115   else if (al.present("-ES2"))
116     festival_eval_command("(
          voice_JuntaDeAndalucia_es_sf_diphone)");
117
118
119   if (al.present("-EN1"))
120     festival_eval_command("(voice_nitech_us_awb_arctic_hts)");
121
122   else if (al.present("-EN2"))
123     festival_eval_command("(voice_nitech_us_slt_arctic_hts)");
124
125   else if (al.present("-EN3"))
126     festival_eval_command("(voice_nitech_us_jmk_arctic_hts)");
127
128   else if (al.present("-EN4"))
```

```
129     festival_eval_command("(voice_nitech_us_clb_arctic_hts)");
130
131   else if (al.present("-EN5"))
132     festival_eval_command("(voice_nitech_us_rms_arctic_hts)");
133
134   else if (al.present("-EN6"))
135     festival_eval_command("(voice_nitech_us_bdl_arctic_hts)");
136
137   if (al.present("-CAT1"))
138     festival_eval_command("(voice_upc_ca_pep_hts)");
139
140   else if (al.present("-CAT2"))
141       festival_eval_command("(voice_upc_ca_ona_hts)");
142
143   if (al.present("-FI"))
144     festival_eval_command("(voice_hy_fi_mv_diphone)");
145
146 /* </AÑADIDO> *//////////////////////////
147
148
149   for (p=files.head(); p != 0; p=next(p))
150     {
151     if (files(p) == "-")  // paul thinks I want the "-" -- I
          don't
152       continue;
153     real_number_of_files++;
154
155       if (al.present("--tts"))
156     {
157       if (!festival_say_file(files(p)))
158       festival_error();
159     }
160
161     else if (files(p).matches(make_regex("^(.*")))
162     {
163       if (!festival_eval_command(files(p)))
164       festival_error(); // fail if it fails
165     }
166     else if (!festival_load_file(files(p)))
167       festival_error();
```

```
168   }
169 /* <AÑADIDO> *////////////////////////////////
170
171
172   if (al.present("-T"))
173   {
174     flagAWP=true; //Mirar si hay -A -W o -P
175     texto=al.val("-T"); // Pasa el string a la variable texto
176
177     if (al.present("-F"))
178     {
179       flagAWP=false;
180       cout << "Error: No se puede leer un fichero y un texto a
              la vez";
181       exit(0);
182     }
183   }
184
185   if(al.present("-F"))
186   {
187     ifstream f(al.val("-F"), ifstream::in);
188     if (!f)
189     {
190        cout << "Error" << endl;
191        texto = "Error";
192     }
193     else
194     {
195       char line[30];
196       while(!f.eof()) {
197         f.getline(line,30,' ');
198         texto=texto+" "+line;
199       }
200     f.close();
201     flagAWP=true;
202     }
203     if (al.present("-T"))
204     {
205       flagAWP=false;
206       cout << "Error: No se puede leer un fichero y un texto a
```

```
            la vez";
207         exit(0);
208       }
209     }
210
211
212     if (flagAWP) //Si hay texto o archivo
213     {
214
215       if (al.present("-A"))
216       {
217         if (!festival_say_text(texto))
218         festival_error();
219       }
220       if (al.present("-W"))
221       {
222         festival_text_to_wave(texto,wave);
223         wave.save((al.val("-W")+".wav"),"riff");
224       }
225
226
227       if (al.present("-P"))
228       {
229         // NOTA: Se ha copiado lib/mbrola.scm en lib/init.scm
230
231         //  save_segments_mbrola UTT FILENAME)
232         //  Save segment information in MBROLA format in filename
              .
233         // The format is phone duration (ms) [% position F0
              target]*
234
235         festival_eval_command("(set! utt1 (Utterance Text \""+
              texto+"\"))");
236         festival_eval_command("(utt.synth utt1)");
237         festival_eval_command("(save_segments_mbrola utt1 \""+al.
              val("-P")+".pho""\")");
238       }
239       festival_wait_for_spooler();    // wait for end of audio
            output
240
```

```
241     exit(0);
242   }
243
244 /* </AÑADIDO> *///
      ////////////////////////////////////////////////////
245
246
247   // What to do about standard input and producing prompts etc.
248   if ((al.present("-i")) || (al.present("--interactive")))
249   {
250     interactive = TRUE;
251     stdin_input = TRUE;
252   }
253   else if ((al.present("--pipe")))
254   {
255     interactive=FALSE;
256     stdin_input = TRUE;
257   }
258   else if ((al.present("-b")) || (al.present("--batch")) ||
259      (al.present("--tts")))
260   {
261   interactive=FALSE;
262   stdin_input=FALSE;
263   }
264   else if (isatty(0))  // if stdin is a terminal assume
         interactive
265   {
266     interactive = TRUE;
267     stdin_input = TRUE;
268   }
269   else                        // else assume pipe mode
270   {
271     interactive = FALSE;
272     stdin_input = TRUE;
273   }
274
275     if (al.present("--server"))
276     festival_server_mode();          // server mode
277     else if ((al.present("--tts")) && (real_number_of_files ==
          0))
```

```
278     festival_say_file("-");          // text to speech from files
279     else if (stdin_input)
280     festival_repl(interactive);      // expect input from stdin
281
282     if (al.present("--tts"))
283     festival_wait_for_spooler();     // wait for end of audio
            output
284
285     return;
286 }
```

Listing A.1: main de festival

## Apéndice B

# Código del módulo

```
1  /**
2   *  For conditions of distribution and use, see copyright
        notice in license.txt
3   *
4   *  @file    TtsModule.h
5   *  @brief  TTS Module registers a TTS Service, integrating
        Festival Speech Synthesis
6   *      System with realXtend.
7   */
8
9  #ifndef incl_TtsModule_h
10 #define incl_TtsModule_h
11
12
13 #include "ModuleInterface.h"
14 #include "ModuleLoggingFunctions.h"
15 #include "Core.h"
16
17 #include <QObject>
18
19 #include "TtsModuleApi.h"
20 #include "TtsServiceInterface.h"
21 #include "TtsService.h"
22
23
```

```
24
25
26
27 class RexUUID;
28
29 namespace Foundation
30 {
31     class EventDataInterface;
32 }
33
34 namespace ProtocolUtilities
35 {
36     class ProtocolModuleInterface;
37     typedef boost::weak_ptr<ProtocolModuleInterface>
          ProtocolWeakPtr;
38 }
39
40 namespace Tts
41 {
42
43     class TTS_MODULE_API TtsModule :  public QObject, public
          Foundation::ModuleInterface
44     {
45         Q_OBJECT
46
47     public:
48         /// Default constructor.
49         TtsModule();
50
51         /// Destructor
52         virtual ~TtsModule();
53         /// Load components: EC_TtsVoice
54     /// It deletes tmp folder (if exists) and all the content
          and creates a new one \note This is need by Festival
          Speech Synthesis System
55         void Load();
56     /// Unload function: It removes tmp folder.
57     void UnLoad();
58     /// Initializes and registers the TTS service
59         void Initialize();
```

```
60      /// PostInitialize function. Empty for now.
61          void PostInitialize();
62      /// Unregisters and destroys the service.
63          void Uninitialize();
64
65          /// Returns name of this module. Needed for logging.
66          static const std::string &NameStatic() { return
                module_name_; }
67
68      //! Logging
69          MODULE_LOGGING_FUNCTIONS
70
71
72      private:
73          Q_DISABLE_COPY(TtsModule);
74
75          /// Name of the module.
76          static const std::string module_name_;
77      /// TTS service
78      TtsServicePtr tts_service_;
79      };
80  }  // end of namespace: Tts
81
82  #endif // incl_TtsModule_h
```

Listing B.1: TtsModule.h

```
1  /**
2   *  For conditions of distribution and use, see copyright
        notice in license.txt
3   *
4   *  @file   TtsModule.cpp
5   *  @brief  TTS Module registers a TTS Service, integrating
        Festival Speech Synthesis
6   *      System with realXtend.
7   */
8
9  #include "StableHeaders.h"
10
11 #include "EventManager.h"
12 #include "ModuleManager.h"
```

```
13 #include "CoreStringUtils.h"
14 #include "MemoryLeakCheck.h"
15
16 #include <QColor>
17
18
19 #include "TtsModule.h"
20 #include "EC_TtsVoice.h"
21
22 namespace Tts
23 {
24   const std::string TtsModule::module_name_ = std::string("
       TtsModule");
25
26   TtsModule::TtsModule() :
27     ModuleInterface(module_name_)
28   {
29
30   }
31
32   TtsModule::~TtsModule()
33   {
34   }
35
36   void TtsModule::Load()
37   {
38     DECLARE_MODULE_EC(EC_TtsVoice);
39
40     if (QDir("tmp").exists())
41       boost::filesystem::remove_all("tmp");
42
43     QDir().mkdir("tmp");
44   }
45
46   void TtsModule::UnLoad()
47   {
48     if (QDir("tmp").exists())
49       boost::filesystem::remove_all("tmp");
50   }
51
```

```
52   void TtsModule::Initialize()
53   {
54     tts_service_ = TtsServicePtr(new TtsService(framework_));
55     framework_->GetServiceManager()->RegisterService(Foundation
           ::Service::ST_Tts, tts_service_);
56   }
57   void TtsModule::PostInitialize()
58   {
59   }
60
61
62   void TtsModule::Uninitialize()
63   {
64     framework_->GetServiceManager()->UnregisterService(
           tts_service_);
65     tts_service_.reset();
66   }
67 } // end of namespace: Tts
68
69
70 extern "C" void POCO_LIBRARY_API SetProfiler(Foundation::
       Profiler *profiler);
71 void SetProfiler(Foundation::Profiler *profiler)
72 {
73     Foundation::ProfilerSection::SetProfiler(profiler);
74 }
75
76
77 using namespace Tts;
78
79 POCO_BEGIN_MANIFEST(Foundation::ModuleInterface)
80     POCO_EXPORT_CLASS(TtsModule)
81 POCO_END_MANIFEST
```

Listing B.2: TtsModule.cpp

```
1 // For conditions of distribution and use, see copyright notice
      in license.txt
2
3 #ifndef incl_Tts_TtsService_h
4 #define incl_Tts_TtsService_h
```

```
5
6 #include "Framework.h"
7
8 #include "TtsServiceInterface.h"
9
10
11 namespace Tts
12 {
13
14   class TtsService : public TtsServiceInterface
15   {
16     Q_OBJECT
17
18     public:
19
20     // Constructor
21     TtsService(Foundation::Framework* framework_);
22
23         //! Destructor
24         virtual ~TtsService();
25
26     virtual void Text2Speech(QString message, Tts::Voice voice)
         ;
27       virtual void Text2WAV(QString message, QString
           pathAndFileName, Tts::Voice voice);
28     virtual void Text2PHO(QString message,QString
         pathAndFileName, Tts::Voice voice);
29
30
31     virtual void File2Speech(QString pathAndFileName, Voice
         voice);
32         virtual void File2WAV(QString pathAndFileNameIn,
             QString pathAndFileNameOut, Voice voice);
33     virtual void File2PHO(QString pathAndFileNameIn, QString
         pathAndFileNameOut, Voice voice);
34
35     private:
36
37         //! Framework we belong to
38         Foundation::Framework* framework_;
```

```
39
40      //Voice voice_;
41
42      };
43 }
44 #endif
```

Listing B.3: TtsService.h

```
1 #include "StableHeaders.h"
2
3 #include "TtsService.h"
4
5
6 namespace Tts
7 {
8    TtsService::TtsService(Foundation::Framework* framework) :
9          framework_(framework)
10     {
11
12     }
13
14     TtsService::~TtsService()
15     {
16
17     }
18
19
20   void TtsService::Text2Speech(QString message, Voice voice)
21   {
22
23     std::stringstream commandoss;
24     std::string commandos,msg;
25     commandoss << "start /B festival.exe --libdir \"festival/
           lib\" ";
26
27     commandoss << voice;
28
29     commandoss << " -A -T \"";
30
31     msg=message.toStdString();
```

```
32      std::replace_if(msg.begin(),msg.end(),boost::is_any_of("\""
            ),',␣');
33      commandoss << msg;
34      commandoss << "\"";
35      commandos = commandoss.str();
36
37      system(commandos.c_str());
38    }
39
40    void TtsService::Text2WAV(QString message, QString
          pathAndFileName, Voice voice)
41    {
42      std::string msg;
43      msg=message.toStdString();
44
45      std::stringstream commandoss;
46      std::string commandos;
47      commandoss << "start␣/B␣festival.exe␣--libdir␣\"festival/
            lib\"␣";
48      commandoss << voice;
49      commandoss << "␣-W␣";
50      commandoss << pathAndFileName.toStdString();
51      commandoss << "␣-T␣\"";
52
53      std::replace_if(msg.begin(),msg.end(),boost::is_any_of("\""
            ),',␣');
54      commandoss << msg;
55      commandoss << "\"";
56      commandos = commandoss.str();
57
58      system(commandos.c_str());
59    }
60
61    void TtsService::Text2PHO(QString message, QString
          pathAndFileName, Voice voice)
62    {
63      std::string msg;
64      msg=message.toStdString();
65
66      std::stringstream commandoss;
```

```
67      std::string commandos;
68      commandoss << "start /B festival.exe --libdir \"festival/
            lib\" ";
69      commandoss << voice;
70      commandoss << " -P ";
71      commandoss << pathAndFileName.toStdString();
72      commandoss << " -T \"";
73
74      std::replace_if(msg.begin(),msg.end(),boost::is_any_of("\""
            ),',');
75      commandoss << msg;
76      commandoss << "\"";
77      commandos = commandoss.str();
78
79      system(commandos.c_str());
80  }
81
82  void TtsService::File2Speech(QString pathAndFileName, Voice
        voice)
83  {
84      std::stringstream commandoss;
85      std::string commandos,file;
86      commandoss << "start /B festival.exe --libdir \"festival/
            lib\" ";
87
88
89      commandoss << voice;
90
91      commandoss << " -A -F \"";
92
93      file=pathAndFileName.toStdString();
94
95      commandoss << file;
96      commandoss << "\"";
97      commandos = commandoss.str();
98
99      system(commandos.c_str());
100 }
101 void TtsService::File2WAV(QString pathAndFileNameIn, QString
        pathAndFileNameOut, Voice voice)
```

```
102  {
103    std::string fileIn;
104    fileIn=pathAndFileNameIn.toStdString();
105
106    std::stringstream commandoss;
107    std::string commandos;
108    commandoss << "start /B festival.exe --libdir \"festival/
           lib\" ";
109    commandoss << voice;
110    commandoss << " -W ";
111    commandoss << pathAndFileNameOut.toStdString();
112    commandoss << " -F \"";
113
114    commandoss << fileIn;
115    commandoss << "\"";
116    commandos = commandoss.str();
117
118    system(commandos.c_str());
119  }
120
121  void TtsService::File2PHO(QString pathAndFileNameIn, QString
       pathAndFileNameOut, Voice voice)
122  {
123    std::string fileIn;
124    fileIn=pathAndFileNameIn.toStdString();
125
126    std::stringstream commandoss;
127    std::string commandos;
128    commandoss << "start /B festival.exe --libdir \"festival/
           lib\" ";
129    commandoss << voice;
130    commandoss << " -P ";
131    commandoss << pathAndFileNameOut.toStdString();
132    commandoss << " -F \"";
133
134    commandoss << fileIn;
135    commandoss << "\"";
136    commandos = commandoss.str();
137
138    system(commandos.c_str());
```

```
139     }
140
141 }
```

Listing B.4: TtsService.cpp

## Apéndice C

# Test de validación y evaluación

| | FASE I: Módulo TTS activado | |
|---|---|---|
| FI.1 | Se escucha audio de los demás si se activa su síntesis | |
| FI.2 | No se escucha audio del otro si no tengo activada su síntesis | |
| FI.3 | Se escucha tu propio audio si lo tienes activado | |
| FI.4 | No se escucha tu propio audio si no lo tienes activado | |
| FI.5 | El receptor de tu audio lo escucha con tus opciones de voz | |
| FI.6 | Cuando modifico las opciones de voz, el siguiente texto enviado, el receptor lo recibe con las nuevas opciones de voz | |
| FI.7 | Leo correctamente mi burbuja cuando hablo | |
| FI.8 | Leo correctamente su burbuja cuando habla | |
| FI.9 | Leo correctamente la línea de chat que aparece/desaparece | |
| FI.10 | Leo correctamente el historial del chat | |
| | | |
| FI.11 | Cuando escribo &%$·"Holá!¿!98 el sintetizador lo reproduce bien | |
| FI.12 | Cuando escribo 15/2/2007 el sintetizador lee la fecha correctamente. | |
| FI.13 | Cuando escribo 14:30 el sintetizador lee la hora correctamente | |
| FI.14 | No he escrito ningún texto que el sintetizador no pudiera sintetizar correctamente | |
| | | |
| FI.15 | Cuando tengo activa la opción de TTS escuchar todo el botón está verde | |
| FI.16 | Cuando tengo activa la opción de TTS escuchar a todos menos a mí, el botón está amarillo | |
| FI.17 | Cuando tengo activa la opción de TTS escuchar a nadie, el botón está blanco | |
| FI.18 | La modificación del color se efectúa incluso sin cerrar el widget | |
| FI.19 | Cuando pongo el ratón encima del botón, cambia la opacidad | |
| FI.20 | Cuando hago click en el botón, cambia la opacidad | |
| FI.21 | Las voces a elegir cambian de forma dinámica al cambiar de idioma | |
| FI.22 | Se escucha la demo de la voz al hacer click en el botón. | |
| | | |
| FI.23 | El cliente no se congela cuando se sintetiza voz | |
| FI.24 | Se recibe el audio en un tiempo adecuado | |
| FI.25 | Se recibe el audio de forma sincronizada con la burbuja del texto sobre el avatar | |
| FI.26 | La carga de mi PC está dentro del límite (menor del 60%) | |
| FI.27 | Cuando todos los participantes están escribiendo constantemente (cada uno una frase) el sistema lo soporta | |
| FI.28 | Cuando todos los participantes están escribiendo constantemente (cada uno una frase) se escuchan los audios a la vez | |
| | | |
| FI.29 | Las notificaciones de entra/sale usuario se sintetizan | |
| | | |
| FI.30 | Se puede añadir un componente TTS a una entidad | |
| FI.31 | Se puede configurar su voz y el texto | |
| FI.32 | Se puede reproducir el texto al hacer clic en la entidad | |

Figura C.1: Resumen de la primera fase donde pueden leerse las preguntas en la columna izquierda y el resultado en colores en la columna derecha. Verde significa que todos han contestado afirmativamente, rojo que todos negativamente y amarillo que ha habido ambos tipos de respuesta.

| FASEI | | | | | | | |
|---|---|---|---|---|---|---|---|
| ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| FI.1 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.2 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.3 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.4 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.5 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.6 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.7 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.8 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.9 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.10 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| | | | | | | | |
| FI.11 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.12 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.13 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.14 | Sí | Sí | No | No | Sí | No | No |
| | | | | | | | |
| FI.15 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.16 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.17 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.18 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.19 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.20 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.21 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.22 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| | | | | | | | |
| FI.23 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.24 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.25 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| FI.26 | Sí | Sí | | Si | Sí | Si | No |
| FI.27 | Sí | Sí | | No | No | No | No |
| FI.28 | Sí | Sí | | Si | Sí | Sí | Sí |
| | | | | | | | |
| FI.29 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |
| | | | | | | | |
| FI.30 | Sí | Sí | Sí | Sí | Sí | Sí | |
| FI.31 | Sí | Sí | Sí | Sí | Sí | Sí | |
| FI.32 | Sí | Sí | Sí | Sí | Sí | Sí | Sí |

Figura C.2: Resultados completos de la primera fase del test de validación.

| | FASE II | |
|---|---|---|
| FII.1 | No oigo su voz pero sí la mía cuando tengo esa opción activada | |
| FII.2 | No oigo ni su su voz ni la mía cuando tengo esa opción activada | |
| FII.3 | Cuando tengo activa la opción de TTS escuchar todo el botón está verde | |
| FII.4 | Cuando tengo activa la opción de TTS escuchar a todos menos a mí el botón está amarillo | |
| FII.5 | Cuando tengo activa la opción de TTS escuchar a nadie, el botón está blanco | |
| FII.6 | Se ha detectado alguna diferencia respecto al chat con un usuario con TTS | |
| | | |
| FII.7 | Leo correctamente mi burbuja cuando hablo | |
| FII.8 | Leo correctamente su burbuja cuando habla | |
| FII.9 | Leo correctamente la linea de chat que aparece/desaparece | |
| FII.10 | Leo correctamente el historial del chat | |
| FII.11 | Puedo chatear sin problema con un usuario | |

Figura C.3: Resumen de la segunda fase donde pueden leerse las preguntas en la columna izquierda y el resultado en colores en la columna derecha. Puede verse que 3 de los 7 participantes han tenido TTS activado y 4 no, estando estos últimos con el fondo azul. Verde significa que todos han contestado afirmativamente, rojo que todos negativamente y amarillo que ha habido ambos tipos de respuesta

| FASEII | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| FII.1 | Sí | Sí | | | Sí | | | |
| FII.2 | Sí | Sí | | | Sí | | | |
| FII.3 | Sí | Sí | | | Sí | | | |
| FII.4 | Sí | Sí | | | Sí | | | |
| FII.5 | Sí | Sí | | | Sí | | | |
| FII.6 | No | No | | | No | | | |
| FII.7 | | | Sí | Sí | | Sí | Sí | |
| FII.8 | | | No | Sí | | Sí | No | |
| FII.9 | | | No | Sí | | Sí | No | |
| FII.10 | | | No | Sí | | Sí | No | |
| FII.11 | | | Sí | Sí | | Sí | Sí | |

Figura C.4: Resultados completos de la segunda fase del test de validación.

| | FASE III: Cuestionario | |
|---|---|---|
| FIII.1 | Te ha parecido intuitiva la forma de saber si tienes el TTS activado o no? | 8,71 |
| FIII.2 | ¿Te parece fácil de usar esta funcionalidad? | 9,29 |
| FIII.3 | ¿Crees que escuchar la voz de las personas con las que estas chateando te ayuda a seguir mejor la conversación? | 7,71 |
| FIII.4 | ¿Te gusta escuchar la voz de las personas con las que chateas? | 7,71 |
| FIII.5 | ¿Te gusta ver sólo texto cuando estás chateando? | 4,29 |
| FIII.6 | En una conversación con voz sintetizada, ¿Crees que es más fácil seguir la conversación cuando escuchas tu voz? | 7,71 |
| FIII.7 | En una conversación con voz sintetizada, ¿Crees que es más fácil seguir la conversación cuando no escuchas tu voz? | 4,71 |
| FIII.8 | ¿Crees que escuchar la voz de las personas con las que estas chateando es más entretenido? | 8,43 |
| FIII.9 | Cuando se en una conversación dos usuarios hablan a la vez, ¿crees que el sistema lo gestiona de forma correcta? | 8,71 |
| FIII.10 | ¿Te parece fácil entender el audio sintetizado en castellano? | 8,43 |
| FIII.11 | ¿Te parece fácil entender el audio sintetizado en inglés? | 8,57 |
| | | |
| FIII.12 | ¿Te parece útil que las notificaciones sean sintetizadas? | 6,57 |
| | | |
| FIII.13 | ¿Crees que añadir componentes de síntesis a entidades es útil? | 9,29 |
| | | |
| FIII.14 | En general, ¿Como valoras la integración del módulo TTS y sus aplicaciones? | 8,71 |
| FIII.15 | ¿Ves útil la implementación del módulo TTS en la plataforma de VW? | 9,71 |

Figura C.5: Resultado promedio de la tercera fase del test, donde se responde al cuestionario con las preguntas formuladas en la columna izquierda. A partir de 7,5 el fondo es verde, por debajo de 5 es rojo y entremedio es amarillo.

| FASEIII | | | | | | | |
|---|---|---|---|---|---|---|---|
| ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| FIII.1 | 10 | 6 | 8 | 10 | 7 | 10 | 10 |
| FIII.2 | 8 | 10 | 9 | 10 | 10 | 8 | 10 |
| FIII.3 | 7 | 10 | 7 | 7 | 7 | 9 | 7 |
| FIII.4 | 6 | 10 | 7 | 10 | 6 | 9 | 6 |
| FIII.5 | 5 | 0 | 8 | 5 | 6 | 2 | 4 |
| FIII.6 | 8 | 10 | 9 | 8 | 3 | 8 | 8 |
| FIII.7 | 4 | 0 | 8 | 6 | 9 | 2 | 4 |
| FIII.8 | 10 | 10 | 6 | 10 | 9 | 7 | 7 |
| FIII.9 | 8 | 10 | 8 | 10 | 8 | 7 | 10 |
| FIII.10 | 8 | 10 | 7 | 10 | 8 | 8 | 8 |
| FIII.11 | 9 | 10 | 7 | 10 | 8 | 8 | 8 |
| | | | | | | | |
| FIII.12 | 7 | 0 | 10 | 5 | 10 | 7 | 7 |
| | | | | | | | |
| FIII.13 | 10 | 7 | 10 | 9 | 9 | 10 | 10 |
| | | | | | | | |
| FIII.14 | 8 | 9 | 9 | 8 | 8 | 9 | 10 |
| FIII.15 | 9 | 10 | 9 | 10 | 10 | 10 | 10 |

Figura C.6: Resultados completos del cuestionario.

| | |
|---|---|
| Reproduce un archivo | Sí |
| Guarda a wav un texto | Sí |
| Guarda a wav un archivo | Sí |
| Guarda a PHO un texto | Sí |
| Guarda a PHO un archivo | Sí |

Figura C.7: Resultado de la prueba manual, añadiendo en el componente código para probar el servicio TTS al completo.

# Lista de Símbolos y Abreviaturas

| Abreviatura | Descripción |
|---|---|
| reX | realXtend |
| TTS | Text To Speech System |
| VW | Virtual World |
| MMOG | Massively Multiplayer Online Game |
| SDK | Software Development Kit |
| NPC | Non Player Character |
| IA | Inteligencia Artificial |
| XMPP | Extensible Messaging and Presence Protocol |
| IP | Internet Protocol |
| ASR | Automatic Speech Recognition |
| ECA | Entity, Component, Attribute |
| TTP | Text To Phoneme |
| GTP | Grapheme to Phoneme |
| API | Application Programming Interface |

# Índice de figuras

# Índice de cuadros

*Ilun, ez da dena ilun,*
*Begirazazu leihotik,*
*Ikustazu baso hori, berdatzen.*