

# Índice general

<b>I</b>	<b>Introducción</b>	<b>5</b>
<b>1.</b>	<b>Presentación del proyecto</b>	<b>7</b>
1.1.	Descripción del proyecto . . . . .	7
1.2.	Alcance . . . . .	7
1.3.	Acerca de esta documentación . . . . .	8
<b>2.</b>	<b>Objetivos propuestos</b>	<b>9</b>
2.1.	Objetivos propuestos . . . . .	9
<b>3.</b>	<b>Recursos educativos y nuevas tecnologías</b>	<b>11</b>
3.1.	Reflexiones acerca de la necesidad del uso de las nuevas tecnologías en la educación .	11
3.2.	Ejemplos de aplicación . . . . .	12
3.2.1.	Recursos educativos en la educación infantil y primaria . . . . .	12
3.2.2.	Recursos educativos en la educación especial . . . . .	13
<b>II</b>	<b>Estudio de las tecnologías de la interacción</b>	<b>19</b>
<b>4.</b>	<b>Síntesis de voz</b>	<b>21</b>
4.1.	Visión general de la tecnología de síntesis de voz . . . . .	21
4.2.	Orígenes . . . . .	21
4.3.	Tecnologías de síntesis . . . . .	22
4.3.1.	Síntesis concatenativa . . . . .	22
4.3.2.	Síntesis de formantes . . . . .	23
4.3.3.	Otros métodos de síntesis . . . . .	24
4.3.4.	Desafíos de la síntesis de voz . . . . .	24
4.4.	Sintetizadores de voz disponibles libremente . . . . .	26
4.4.1.	Edinburgh Speech Tools . . . . .	26
4.4.2.	Festival . . . . .	27
4.4.3.	Flite . . . . .	28
4.4.4.	FreeTTS . . . . .	29
4.4.5.	MBROLA . . . . .	30
4.4.6.	Microsoft Speech SDK 5.1 . . . . .	31
4.4.7.	Comparativas estadísticas de algunos sintetizadores . . . . .	31
4.5.	Sintetizadores de voz disponibles comercialmente . . . . .	32
4.5.1.	Fonix Speech . . . . .	33
4.5.2.	Acapela Mobility . . . . .	33
4.5.3.	Cepstral . . . . .	33
4.5.4.	AT&T Natural Voices . . . . .	34

<b>5. Uso de interfaces animadas y síntesis de voz</b>	<b>35</b>
5.1. Necesidad de interfaces de usuario amigables . . . . .	35
5.2. Desarrollo de las Interfaces de Usuario (IU) . . . . .	35
5.3. Interfaces de Usuario con personajes animados . . . . .	36
5.4. Elementos de conversación cara a cara con personajes animados . . . . .	37
5.5. Directrices para el diseño de la interacción con personajes . . . . .	39
5.5.1. Emplear sonido . . . . .	39
5.5.2. No ser exclusivo . . . . .	40
5.5.3. Proporcionar la apropiada realimentación . . . . .	40
5.5.4. Usar variaciones naturales . . . . .	41
5.5.5. Interacción social . . . . .	41
5.5.6. Uso de gestos . . . . .	42
5.5.7. Crear una personalidad . . . . .	42
5.5.8. Aspecto físico . . . . .	44
5.5.9. Observar el protocolo apropiado . . . . .	44
5.5.10. Usar la alabanza . . . . .	45
5.5.11. El personaje y el usuario como miembros del mismo equipo . . . . .	46
5.5.12. Considerar efectos de género . . . . .	47
5.6. Sonido como Interfaz de Usuario . . . . .	47
5.6.1. Reconocimiento del habla . . . . .	48
5.7. Adaptación a las Interfaces de usuario con habla . . . . .	49
5.8. Simular por parte del sistema una conversación natural . . . . .	50

### **III Desarrollo de la aplicación** **51**

<b>6. Análisis del sistema. Definición del problema.</b>	<b>53</b>
6.1. Consideraciones iniciales . . . . .	53
6.2. Metodología empleada . . . . .	53
6.3. Resultados de las entrevistas. . . . .	54
6.4. Definición de actores . . . . .	54
6.5. Diagramas de caso de uso . . . . .	55
6.6. Modelo de clases . . . . .	55
6.6.1. Diagrama inicial de clases . . . . .	56
6.7. Descripción de los casos de uso . . . . .	56
6.7.1. Jugar imágenes . . . . .	57
6.7.2. Jugar inglés . . . . .	58
6.7.3. Jugar cuento . . . . .	59
6.7.4. Consultar descripción de los personajes . . . . .	59
6.7.5. Demostración de gestos . . . . .	60
6.7.6. Demostración de personajes . . . . .	61
6.7.7. Solicitar ayuda . . . . .	61
6.7.8. Cambiar personaje . . . . .	62
6.7.9. Elegir opciones avanzadas del sistema . . . . .	62
<b>7. Diseño</b>	<b>63</b>
7.1. Descripción de los Casos de uso . . . . .	64
7.1.1. Solicitar ayuda . . . . .	64
7.1.2. Cambiar personaje . . . . .	65
7.1.3. Jugar imágenes . . . . .	66
7.1.4. Jugar inglés . . . . .	67

<b>ÍNDICE GENERAL</b>	<b>3</b>
7.1.5. Jugar cuento . . . . .	68
7.1.6. Consultar descripción de los personajes . . . . .	69
7.1.7. Demostración de gestos . . . . .	70
7.1.8. Demostración de personajes . . . . .	71
7.1.9. Elegir opciones avanzadas del sistema . . . . .	71
7.2. Diagrama de clases final . . . . .	72
7.3. Especificación de clases . . . . .	73
<b>8. Implementación</b>	<b>79</b>
8.1. Software utilizado . . . . .	79
8.1.1. Microsoft Agent 2.0 . . . . .	80
8.2. Hardware empleado . . . . .	81
<b>9. Pruebas</b>	<b>83</b>
9.1. Pruebas realizadas . . . . .	83
<b>IV Manual de usuario</b>	<b>87</b>
<b>10. Manual de usuario</b>	<b>89</b>
10.1. Descripción de la aplicación . . . . .	89
10.2. Guía de instalación . . . . .	89
10.2.1. Instalación del Ms Agent . . . . .	90
10.2.2. Instalación de la aplicación Veo y escucho . . . . .	90
10.3. Manual de usuario de la aplicación . . . . .	91
10.3.1. Bienvenida . . . . .	91
10.3.2. Imágenes . . . . .	92
10.3.3. Inglés . . . . .	94
10.3.4. Cuento . . . . .	95
10.3.5. Presentación . . . . .	96
10.3.6. Gestos . . . . .	97
10.3.7. Menu superior . . . . .	98
10.3.8. Cambiar personaje . . . . .	98
10.3.9. Descripción . . . . .	99
10.3.10.Opciones avanzadas de los personajes . . . . .	100
10.3.11.Consultar ayuda . . . . .	100
10.3.12.Acerca de . . . . .	101
<b>V Conclusiones</b>	<b>103</b>
<b>11. Conclusiones</b>	<b>105</b>
11.1. Dificultadas encontradas . . . . .	105
11.2. Objetivos alcanzados . . . . .	105
11.3. Posibles mejoras . . . . .	106
11.3.1. Reconocimiento del habla . . . . .	106
11.3.2. Desarrollo en aplicaciones web . . . . .	106
11.3.3. Permitir la introducción de nuevos elementos . . . . .	106

<b>A. Microsoft Agent</b>	<b>109</b>
A.1. Potencial de Microsoft Agent 2.0 . . . . .	109
A.1.1. ¿Respecto a qué referencias o antecedentes se puede comparar a Microsoft Agent?110	
A.1.2. ¿Para qué pueden ser usados los personajes proporcionados por Microsoft Agent?110	
A.2. Instalación . . . . .	111
A.3. Interfaz de Usuario de Microsoft Agent . . . . .	112
A.3.1. Ventana del personaje . . . . .	112
A.3.2. Menú contextual de comandos . . . . .	112
A.3.3. Icono del personaje en la barra de herramientas . . . . .	113
A.3.4. Ventana de Comandos de Voz . . . . .	113
A.3.5. Globo de Texto . . . . .	114
A.3.6. Ventana informativa del estado Listening . . . . .	114
A.3.7. Ventana de Opciones Avanzadas de Personajes . . . . .	114
A.3.8. Página de salida . . . . .	115
A.3.9. Página de la entrada de voz . . . . .	115
A.3.10. Página del Copyright . . . . .	115
A.3.11. Ventana de Propiedades del Personaje por Defecto . . . . .	116
A.4. Interfaz de programación de Microsoft Agent . . . . .	116
A.4.1. Acceso a los servicios de programación . . . . .	117
A.4.2. Control ActiveX . . . . .	117
A.4.3. Acceso directo a la Interfaz COM . . . . .	117
A.4.4. Modelo de Objetos de Microsoft Agent . . . . .	117
A.4.5. Puesta en marcha del Servidor AgentSvr.exe . . . . .	120
A.4.6. Cargando el personaje y los datos de animación . . . . .	120
A.4.7. Crear un recolector de eventos (Notification Sink) . . . . .	120
A.4.8. Servicios de Animación . . . . .	121
A.4.9. Cargar un personaje . . . . .	121
A.4.10. Cargar el Personaje por Defecto . . . . .	121
A.4.11. Animando un personaje . . . . .	122
A.4.12. Servicios de Entrada . . . . .	124
A.4.13. Cliente con entrada activa . . . . .	124
A.4.14. Soporte para Menú Contextual . . . . .	125
A.4.15. Soporte para entrada de habla . . . . .	126
A.4.16. Elección del motor de habla . . . . .	127
A.4.17. Eventos de entrada de habla . . . . .	127
A.4.18. Ventana de Comandos de Voz . . . . .	128
A.4.19. Ventana de Opciones Avanzadas de Personajes . . . . .	128
A.4.20. Servicios de Salida . . . . .	128
A.4.21. Soporte para síntesis de voz . . . . .	128
A.4.22. Soporte para salida de audio grabado . . . . .	128
A.4.23. Soporte para Globo de Texto . . . . .	129
A.4.24. Efectos de Sonido de las Animaciones . . . . .	129
<b>B. Panorámica General .NET</b>	<b>131</b>
B.1. Plataforma.NET . . . . .	131
B.2. El lenguaje C# . . . . .	134
B.3. Visual Studio.NET 2003 . . . . .	138
B.4. XML . . . . .	143

# **Parte I**

## **Introducción**



# Capítulo 1

## Presentación del proyecto

### 1.1. Descripción del proyecto

El proyecto tiene como primera parte el estudio de la síntesis de voz abordado desde varios puntos: estudio teórico de la formación de la voz y estudio de las distintas posibilidades disponibles en síntesis de voz.

La aplicación del proyecto consiste en el desarrollo de software de educación primaria, basada en tecnologías de síntesis de voz. Su valor pedagógico se centra en la labor de reconocimiento de items del entorno.

El núcleo de la aplicación serán tres juegos; uno de reconocimiento de imágenes, otro relacionado con la música y el idioma inglés y por último un cuentacuentos.

La síntesis de voz utilizada es la que nos proporcionan los agentes animados. La utilización de dichos agentes viene dada por la naturaleza de la educación primariza: ha de ser gráfica y amena.

El hecho de utilizar MsAgent hace necesario el incluir un estudio en profundidad de las capacidades que ofrece. Microsoft Agent proporciona un juego de servicios software programables que pueden usarse para complementar una interfaz de usuario clásica al facilitar la integración de personajes animados interactivos.

### 1.2. Alcance

La aplicación esta pensada para niños de educación primaria, niños que se acerquen a la aplicación con una intención de ocio, aunque tenga un trasfondo educativo.

Estos niños han de tener conocimientos básicos de lectura y escritura, así como de inglés para los niños de primaria que quieran utilizar el juego de la aplicación que esta en dicho idioma.

### 1.3. Acerca de esta documentación

Esta memoria pretende ser concisa con los contenidos teóricos y no crear una compilación de datos teóricos poco relacionados con el objeto de estudio. Las cuestiones teóricas relacionadas con el software o hardware utilizados se ceñirán a la utilidad que de ellos se pueda dar.

La memoria se organiza en partes, y esta a su vez en capítulos que también son divididos en puntos, de esta forma se establece un orden lógico de lectura por temas.

Las partes de la memoria son:

- **Parte I. Introducción:** breve presentación del tema que aborda el proyecto, junto con los objetivos del mismo, a la vez que se hace una pequeña descripción del contenido y estructura de la memoria.

También incluye una pequeña introducción al software educativo y a lo que supone.

- **Parte II. Estudio de las tecnologías de la interacción:** esta parte comienza con un estudio teórico de que es la síntesis de voz y como puede desarrollarse.

Posteriormente se hace un estudio de las alternativas disponibles ya sean comerciales o libres.

La última parte es un estudio sobre interfaces animadas y su relación con la síntesis de voz, que sirve como justificación para el hecho de haber utilizado en la aplicación un motor de síntesis de voz que venga inserto en un asistente animado y no uno aislado.

- **Parte III. Desarrollo de la aplicación:** Análisis, diseño, implementación y pruebas.
- **Parte IV. Manual del usuario:** guía pormenorizada de la instalación y el manejo de la aplicación.
- **Parte V. Conclusiones**
- **Apéndices:** Manual del Ms Agent, Plataforma.Net



## Capítulo 2

# Objetivos propuestos

### 2.1. Objetivos propuestos

A continuación se mencionan los objetivos propuestos:

- Comprender el proceso de síntesis de voz desde un punto de vista teórico.
- Conocer las distintas alternativas que se nos ofrece para la síntesis de voz y su evolución histórica.
- Conocer las posibilidades de la síntesis de voz en dispositivos móviles teniendo en cuenta las características intrínsecas como la escasez de memoria.
- Realización de una aplicación que sirva como introducción al mundo del PC para niños de educación primaria.
- Derivado de este objetivo,lograr cierta estimulación sensorial novedosa para niños que es la primera vez que manipulan un ordenador.
- Valor mas académicamente dicho,orientado a la adquisición de nuevos conceptos ya sean visuales,sonoros o lingüísticos.
- Conocer un entorno de trabajo específico como puede ser el desarrollo de aplicaciones bajo Visual Studio .Net.
- Utilizar una metodología de desarrollo de software que permita la creación de esta aplicación.
- Aplicar correctamente los conocimientos y técnicas adquiridos a lo largo de estos años.
- Conocer un entorno de trabajo específico como es el desarrollo de aplicaciones en un nuevo lenguaje C#,y una nueva forma de trabajar la reutilización de código y librerías proporcionadas por los fabricantes dentro de la filosofía de trabajo del grupo GIRO.



## Capítulo 3

# Recursos educativos y nuevas tecnologías

### 3.1. Reflexiones acerca de la necesidad del uso de las nuevas tecnologías en la educación

Por Tecnologías de la información y de la comunicación (TICs) se entiende un concepto difuso empleado para designar lo relativo a la informática conectada a Internet y, especialmente, el aspecto social de éstos. El acrónimo TIC'S se significa Tecnologías de la Información y de la Comunicación.

También se las suele denominar Ntic's (por Nuevas Tecnologías de la Información y de la Comunicación)

El concepto de tecnologías de información y comunicación presenta dos características típicas. Por una parte se usa frecuentemente en los debates contemporáneos, especialmente por la clase política. Por otra parte el término se sumerge en una borrosidad semántica ejemplar (en la primera década del siglo XXI, el término se usa con frecuencia para estar a la moda), que es por lo que posiblemente los políticos tengan tanto gusto por usarlo.

Parece pues necesario conectar el concepto a un conjunto de estructuras materiales, localizar el origen de la difusión de estas estructuras en el tiempo y en el espacio geográfico y delimitar el fenómeno del espacio virtual que estas estructuras hacen posible. Dentro de ésta definición general encontramos los siguientes temas principales:

- Sistemas de comunicación
- Informática
- Herramientas ofimáticas que contribuyen a la comunicación

La 'sociedad de la información' en general y las nuevas tecnologías en particular inciden de manera significativa en todos los niveles del mundo educativo. Las nuevas generaciones van asimilando de manera natural esta nueva cultura que se va conformando y que para la gente mayor conlleva muchas veces importantes esfuerzos de formación, de adaptación y de 'desaprender' muchas cosas que ahora 'se hacen de otra forma' o que simplemente ya no sirven. Los más jóvenes no tienen el peso experiencial de haber vivido en una sociedad 'más estática', de manera que para ellos el cambio y el aprendizaje

continuo para conocer las novedades que van surgiendo cada día es lo normal.

Precisamente para favorecer este proceso que se empieza a desarrollar desde los entornos educativos informales (familia, ocio...), la escuela debe integrar también la nueva cultura: alfabetización digital, fuente de información, instrumento de productividad para realizar trabajos, material didáctico, instrumento cognitivo.... Obviamente la escuela debe acercar a los estudiantes la cultura de hoy, no la cultura de ayer. Por ello es importante la presencia en clase del ordenador (y de la cámara de vídeo, y de la televisión...) desde los primeros cursos, como un instrumento más, que se utilizará con finalidades diversas: lúdicas, informativas, comunicativas, instructivas... Como también es importante que esté presente en los hogares y que los más pequeños puedan acercarse y disfrutar con estas tecnologías de la mano de sus padres.

La Era Internet exige cambios en el mundo educativo. Y los profesionales de la educación tienen múltiples razones para aprovechar las nuevas posibilidades que proporcionan las TIC para impulsar este cambio hacia un nuevo paradigma educativo más personalizado y centrado en la actividad de los estudiantes. Además de la necesaria alfabetización digital de los alumnos y del aprovechamiento de las TIC para la mejora de la productividad en general, el alto índice de fracaso escolar (insuficientes habilidades lingüísticas, matemáticas...) y la creciente multiculturalidad de la sociedad con el consiguiente aumento de la diversidad del alumnado en las aulas (casi medio millón de niños inmigrantes en 2004/2005 de los que una buena parte no dominan inicialmente la lengua utilizada en la enseñanza), constituyen poderosas razones para aprovechar las posibilidades de innovación metodológica que ofrecen las TIC para lograr una escuela más eficaz e inclusiva.

## **3.2. Ejemplos de aplicación**

### **3.2.1. Recursos educativos en la educación infantil y primaria**

El uso de las TIC, en los centros educativos se impone y sustituye a antiguos usos y recursos. El uso del ordenador y el software educativo como herramienta de investigación, manipulación y expresión tiene una cualidad muy motivadora y atractiva para el alumnado de los distintos niveles educativos.

El trabajo cotidiano con y en la informática permite al alumnado una intervención creativa y personal, mantener un ritmo propio de descubrimiento y aprendizaje, así como el acceso a la información más integral, permitiendo iniciar un proceso de universalización del uso y conocimiento de las TIC.

El profesor ha de adquirir un nuevo rol y nuevos conocimientos, desde conocer adecuadamente la red y sus posibilidades hasta como utilizarla en el aula y enseñar a sus alumnos sus beneficios y desventajas.

En la actualidad, los niños asumen con total normalidad la presencia de las tecnologías en la sociedad. Conviven con ellas y las adoptan sin dificultad para su uso cotidiano. En este sentido los docentes deben propiciar una educación acorde con nuestro tiempo realizando nuevas propuestas didácticas e introduciendo las herramientas necesarias para este fin.

Es a la edad de tres años cuando la mayoría de niños tienen el primer contacto con un centro escolar, y a diferencia de épocas anteriores, en las cuales no se otorgaba gran importancia a esta etapa de la educación Infantil, en la actualidad se considera relevante, ya que sienta las bases de futuros aprendizajes, se adquieren hábitos de conducta y de convivencia, se suceden grandes cambios de crecimiento intelectual, adquieren gran capacidad de aprendizaje, etc.

Estas y otras características permiten considerar que la acción educativa que se lleve a cabo en este período será fundamental en su posterior proceso evolutivo. Esta acción educativa debe plantearse la utilización del ordenador como recurso.

Los Programas de Educación Primaria tienen como objetivo ofrecer, a los padres y profesores de los niños de toda la etapa de primaria, un completo conjunto de actividades educativas interactivas especialmente diseñadas para completar y reforzar lo que los niños aprenden en el colegio de una forma útil, efectiva y muy amena.

Los programas están desarrollados a partir de la premisa, ampliamente demostrada, de que realizar actividades de ampliación y refuerzo es una forma ideal de completar la formación escolar de los niños y que permiten generar el hábito de dedicar un tiempo a estudiar todos o casi todos los días.

Las actividades de los programas de educación primaria están especialmente pensadas para que los niños adquieran y ejerciten los conocimientos y habilidades que necesitan para formarse y desarrollarse intelectualmente, de una forma efectiva y entretenida, en los siguientes campos:

- Matemáticas y cálculo
- Lenguaje, lectura
- Familiarización con el idioma inglés
- Capacidades cognitivas: atención, concentración, memoria...
- Habilidades: motricidad, coordinación viso-manual, razonamiento lógico...

Para potenciar su atractivo y efectividad, se presentan con un planteamiento lúdico y con un diseño gráfico atractivo y adaptado al gusto de las edades a las que van dirigidas. El objetivo es que los niños tengan la percepción de que están jugando (concepto de aprender jugando), potenciando de esta forma su atención y concentración y, por tanto, la calidad y efectividad del tiempo empleado. El uso periódico y ordenado de las actividades de los programas de educación primaria no solo favorece la formación integral del niño, sino que además permite potenciar aspectos tan importantes como:

- La práctica y desarrollo de las capacidades de atención, concentración y memoria, tan importantes para un correcto proceso de aprendizaje y maduración.
- El refuerzo de los contenidos escolares, especialmente la adquisición de vocabulario y lectura, el cálculo y el razonamiento lógico, el conocimiento del medio...
- Generar el hábito de dedicar tiempo al estudio.
- Adquirir auto-confianza y destreza en el uso de ordenadores y nuevas tecnologías

### **3.2.2. Recursos educativos en la educación especial**

La utilización de la informática en Educación Especial requiere adaptar nuestros ordenadores a toda una amplia gama de discapacidades.

En primer lugar, y dentro de lo que podemos denominar como hardware (soporte físico del ordenador) conviene tener en cuenta las modificaciones que debemos de llevar a cabo en nuestro ordenador como son las adaptaciones al teclado, commutadores e interruptores, digitalizador de voz, emulador de teclado, teclado de conceptos, ratón, emulador de ratón, etc.

Los recursos informáticos, a su vez, los vamos a clasificar teniendo en cuenta a las discapacidades a los que van dirigidos, como pueden ser:

- Discapacidad motórica.
- Discapacidad visual.
- Discapacidad auditiva.
- Discapacidad psíquica.

### Discapacidad motórica

La utilización de algunos de estos programas pueden ser controlados por el escáner o por el ratón; para ello los ordenadores cuentan con unos conmutadores que permiten a los deficientes motóricos utilizarlo como instrumento educativo insustituible tanto para los aprendizajes escolares como para el juego.

Se encuentra aquí, por lo tanto, una primera diferenciación a la hora de utilizar el ordenador; una, determinada por la propia utilización de éste para juegos y, otra segunda, relacionada con la naturaleza de estos programas, cuya principal característica es servir de acceso al ordenador. Ahora bien, esta presunta diferenciación no es tal si tenemos en cuenta que en ambos casos lo que se persigue es que la persona discapacitada, ya sea niño o mayor, pueda utilizar libremente las nuevas tecnologías de la comunicación.

En el primero de los casos, algunos programas para juegos permiten a los niños con deficiencias motóricas sentirse sujetos activos; por ejemplo, pueden jugar a una serie de juegos tan tradicionales como los barcos, los puzzles o el cuatro en raya.

Como hardware se puede utilizar el teclado de conceptos, digitalizador de voz, emulador de teclado, adaptaciones al teclado, sintetizador de voz, etc.

En cuanto a los diferentes programas que podemos utilizar son muchos, por lo que aquí nos vamos a limitar a mencionar alguno de ellos como:

1. **Plaphoons:** Este software está pensado para ser utilizado como un comunicador. Es ideal para ser utilizado por personas con discapacidad motórica que no pueden comunicarse mediante la voz.
2. **Kanghooru:** Se utiliza para realizar un barrido automático para cualquier programa.
3. **Teclado silábico:** El teclado fonético-silábico consiste en que cada tecla representa una sílaba y la disposición de las teclas tiene una ordenación fonética. Está diseñado para aumentar la productividad a la hora de escribir, ya que mejora el tiempo que se tarda en encontrar la tecla deseada y con una pulsación se escriben dos o más letras.
4. **Pasa páginas:** Permite la visualización y lectura de libros a través de un teclado que funciona por escaneo automático y activación por conmutador. Puede también ser activado por un sonido captado por un micrófono.
5. **ViaVoice:** Con este programa se puede controlar el ordenador con la voz sin tocar ni el teclado ni el ratón con los dedos.
6. **Comunicador Morse:** Facilita la comunicación de personas con fuertes discapacidades motóricas. A través de este comunicador pueden escribir en la pantalla del ordenador, hablar, jugar y controlar algunos elementos del entorno (puertas, ventanas, televisión, cama, etc.). Como canal de comunicación utilizan el alfabeto morse.

### Discapacidad auditiva

Los discapacitados auditivos también se han visto beneficiados por la aparición de una serie de programas que les permiten tener acceso a través del ordenador a una serie de actividades encaminadas a poder comunicarse con los demás.

Ejemplos de hardware: digitalizador de voz, lector óptico de tarjetas, teclado de conceptos, sintetizador de voz y pantalla táctil.

Los principales programas que podemos utilizar y que son más accesibles son:

1. **Globus:** Se trata de un visualizador fonético. A través de la utilización del micrófono del ordenador, es posible visualizar el sonido de distintas maneras y hacer pequeñas actividades de imitación de los sonidos emitidos por el profesor (ritmo, intensidad...), o de competir con el ordenador, emitiendo en un sencillo juego de carreras. El programa ha sido creado por el Proyecto Fressa.
2. **Speech Viewer III - IBM :** se trata de una herramienta que transforma palabras o sonidos hablados en atractivos gráficos. Asimismo, incrementa la efectividad de la terapia de lenguaje y habla en las personas que presentan problemas de lenguaje, habla y audición. Está diseñado para ayudar a personas que tengan alguna discapacidad, principalmente de habla, lenguaje y auditiva, parálisis cerebral, retraso mental, daño cerebral. Es muy útil para el doctor o terapeuta de lenguaje así como educadores de personas sordas.
3. **BALDI:** se trata de un Tutor Virtual para Aprender a Hablar. Está dirigido a niños que presentan graves problemas de sordera y que desean aprender a hablar con normalidad. El programa y todo lo relacionado con su aplicación vienen en inglés.
4. **Sign Language Teacher:** se trata de un sencillo manual, en inglés, dirigido a los profesores que trabajan con discapacitados y que quieren aprender el lenguaje de la mímica.
5. **Tcomunica:** es un software destinado a los alumnos con necesidades educativas especiales, en concreto, las ocasionadas por una parálisis cerebral. Al no poder utilizar la voz se aprende a través de la utilización de símbolos, con colores, etc.

### Discapacidad visual

Son muchos los programas que en los últimos años han ido apareciendo en el mercado dirigidos a las personas con discapacidad visual que han permitido que muchas de estas personas tengan una calidad de vida próxima o igual a los videntes. La llegada de las nuevas tecnologías ha permitido que estas personas desarrollen trabajos con ordenadores como lo haría cualquiera.

Ejemplos de hardware: adaptaciones al teclado, teclado de conceptos; teclado, línea e impresora Braille; y sintetizador de voz.

Algunos de los programas que se pueden utilizar para ayudar a las personas con discapacidad visual son:

1. **ZoomText Xtra 7.1.** Se trata de un programa magnificador de pantalla compatible con el sistema operativo Windows XP. Está diseñado para personas con baja visión, ya que agranda el tamaño de los programas en Windows y simultáneamente reproduce en voz sintetizada los textos por la tarjeta de sonido de la PC. Demo gratuita.
2. **Home Page Reader 3.0:** Es un navegador de internet que aprovecha la capacidad de habla de Via Voice Outloud (Text to Speech) de IBM para poder sintetizar la voz junto con Windows Explorer

Home Page Reader permite a las personas ciegas o con debilidad visual utilizar el Internet sin dificultad. Por sus características convierte la información que aparece en el monitor en texto audible, facilitando la lectura de pantallas completas, párrafos, oraciones, palabras y letras.

3. **Open Book: Ruby Edition 4.0:** Permite el acceso hablado a internet para personas ciegas y con debilidad visual.
4. **JAWS para Windows 3.7:** lector de pantalla para un completo control del sistema y las aplicaciones de la computadora especialmente diseñado para personas ciegas y con debilidad visual.
5. **ConPalabras** es un plug-in que permite que las páginas web hablen. Permite sintetizar mensajes contenidos en la página HTML o ficheros VoiceXML remotos. En esta página se puede encontrar el programa, descargarlo y ver las posibilidades de trabajo con él.

### Discapacidad psíquica

Los discapacitados psíquicos son quizá los que por el momento cuentan con menos posibilidades en el mundo de la utilización de los medios informáticos, aunque son ya muchos los programas que les están ayudando a entrar en él. El hardware utilizado para trabajar con alumnos que presentan este tipo de discapacidad está muy relacionado con: adaptaciones del teclado, conmutadores e interruptores, control ambiental, digitalizador de voz, interface de conmutadores, lector óptico de tarjetas, teclado de conceptos, sintetizador de voz, pantalla táctil y reconocimiento de voz.

- **Soale:** (Sistema Orientado al Aprendizaje de la Lectura - Escritura) muy útil para los niños con Síndrome de Down. Se utiliza para aprender a leer y a escribir y como herramienta básica se utiliza un PC. También puede ser muy útil para niños sin esta minusvalía psíquica. El programa se puede descargar y es gratuito.

### Proyectos y páginas web que ofrecen recursos informáticos

Las posibilidades que el mundo de la informática está abriendo a los discapacitados queda reflejado en la ingente cantidad de páginas creadas en internet con el único objetivo de que éstos tengan una posibilidad más de integrarse en una sociedad competitiva cien por cien. En ellas se ofrecen todo tipo de programas, algunos de forma gratuita y otros no, pero con el aliciente de que algunos de ellos son fáciles de instalar y que no hay que ser muy expertos en la materia para su utilización tanto en casa como en el colegio.

### Proyecto Fressa 2000 y Fressa 2002

El Proyecto Fressa, creado y dirigido por Jordi Lagares, ofrece un variado y completo software dirigido a ayudar a personas que presentan diferentes discapacidades. Este Proyecto ha ido evolucionando con los años y es, sin duda, uno de los mejores modelos de trabajo que sobre Educación Especial podemos encontrar hoy en toda la red.

En la página encontramos un detallado análisis tanto del software a utilizar como el modo de hacerlo. Además, en ella podemos encontrar manuales para la utilización del Proyecto Fressa, así como ejemplos de cómo se utilizan los diferentes programas.

El software que utiliza es el siguiente: Para personas con discapacidad motórica controlados por escaneo o ratón: Plaphoons, Kanghooru, Teclado silábico, Juego, platillos voladores, controlador del mouse, controlador de un Teclado, pasa páginas para leer libros, o ser leídos por el programa. Para personas con discapacidad motórica controlados por voz: controlador del mouse. Para personas con deficiencias auditivas: Globus, reconocimiento de fonemas. Para personas con discapacidad visual o motora: Navegador



Web Hablado, El Xerraire. Para personas con discapacidad visual: Lectura de libros para invidentes. Para personas con discapacidad motórica: Lectura de textos, SDK: Motor de reconocimiento de sonidos, DLL, y programas de ejemplo como utilizarla.

#### **Gopsol**

La información contenida en esta página se encuentra relacionada con varios programas que trabajan determinadas deficiencias tanto de tipo motriz, visual, auditiva, como psíquica.

Los programas son: Melani (construcción de estructuras sintácticas simples); Micon (juego de construcciones); Multireader (reconocimiento óptico de caracteres); One Finger (simplifica las operaciones del teclado); Ordne die Geschichte (ordenación de imágenes); Pipeline (conectar tuberías de diferentes formas); Programa de estimulación lingüística (reeducación dificultades lecto/escritura); Procesador e textos icónico; Serie de colores; visualizador fonético; etc.



## **Parte II**

# **Estudio de las tecnologías de la interacción**



## Capítulo 4

# Síntesis de voz

La síntesis de voz es la producción artificial de habla humana. Un sistema usado con este propósito recibe el nombre de sintetizador de voz y puede implementarse en software o en hardware. La síntesis de voz se llama a menudo en inglés text-to-speech (TTS), en referencia a su capacidad de convertir texto en habla. Sin embargo, hay sistemas que en lugar de producir voz a partir de texto lo hacen a partir de representación lingüística simbólica.

### 4.1. Visión general de la tecnología de síntesis de voz

Un sistema texto a voz se compone de dos partes: un front-end y un back-end. A grandes rasgos, el front-end toma como entrada texto y produce una representación lingüística fonética. El back-end toma como entrada la representación lingüística simbólica y produce una forma de onda sintetizada.

El front-end desempeña dos tareas principales. Primero, toma el texto y convierte partes problemáticas como números y abreviaturas en palabras equivalentes. Este proceso se llama a menudo normalización de texto o preprocesado. Entonces asigna una transcripción fonética a cada palabra, y 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 texto a fonema (TTP en inglés) o grafema a fonema (GTP en inglés). La combinación de transcripciones fonéticas e información prosódica constituye la representación lingüística fonética.

La otra parte, el back-end, toma la representación lingüística simbólica y la convierte en sonido. El back-end se llama a menudo sintetizador.

### 4.2. Orígenes

Mucho antes del desarrollo del procesado de señal moderno, los investigadores de la voz intentaron crear máquinas que produjesen habla humana. El Papa Silvestre II (1003), Alberto Magno (1198-1280) y Roger Bacon (1214-1294) crearon ejemplos tempranos de 'cabezas parlantes'.

En 1779, el científico danés Christian Gottlieb Kratzenstein, que trabajaba en esa época en la Academia Rusa de las Ciencias, construyó modelos del tracto vocal que podría producir las cinco vocales largas (a, e, i, o y u). Wolfgang von Kempelen de Vienna, Austria, describió en su obra *Mechanismus der menschlichen Sprache nebst der Beschreibung seiner sprechenden Maschine* ("mecanismo del habla humana con descripción de su máquina parlante", J.B. Degen, Wien) una máquina accionada con un fuelle. Esta máquina tenía, además, modelos de la lengua y los labios, para producir consonantes,

así como vocales. En 1837 Charles Wheatstone produjo una 'máquina parlante' basada en el diseño de von Kempelen, y en 1857 M. Faber construyó la máquina 'Euphonia'. El diseño de Wheatstone fue resucitado en 1923 por Paget.

En los años 30, los laboratorios Bell Labs desarrollaron el VOCODER, un analizador y sintetizador del habla operado por teclado que era claramente intelegible. Homer Dudley refinó este dispositivo y creó VODER, que exhibió en la Exposición Universal de Nueva York de 1939.

Los primeros sintetizadores de voz sonaban muy robóticos y eran a menudo intelegibles a duras penas. Sin embargo, la calidad del habla sintetizada ha mejorado en gran medida, y el resultado de los sistemas de síntesis contemporáneos es, en ocasiones, indistinguible del habla humana real.

A pesar del éxito de los sintetizadores puramente electrónicos, sigue investigándose en sintetizadores mecánicos para su uso en robots humanoides. Incluso el mejor sintetizador electrónico está limitado por la calidad del transductor que produce el sonido, así que en un robot un sintetizador mecánico podría ser capaz de producir un sonido más natural que un altavoz pequeño.

El primer sistema de síntesis computerizado fue creado a final de la década de 1950 y el primer sistema completo texto a voz se finalizó en 1968. Desde entonces se han producido muchos avances en las tecnologías usadas para sintetizar voz.

### 4.3. Tecnologías de síntesis

Las dos características utilizadas para describir la calidad de un sintetizador de voz son la naturalidad e inteligibilidad. La naturalidad de un sintetizador de voz se refiere a cuánto suena como la voz de una persona real. La inteligibilidad de un sintetizador se refiere a la facilidad de la salida de poder ser entendida. El sintetizador ideal debe de ser a la vez natural e intelegible, y cada tecnología intenta conseguir el máximo de ambas. Algunas de las tecnologías son mejores en naturalidad o en inteligibilidad y las metas de la síntesis determinan a menudo qué aproximación debe seguirse. Hay dos tecnologías principales usadas para generar habla sintética: síntesis concatenativa y síntesis de formantes.

#### 4.3.1. 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, 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.

##### Síntesis por selección de unidades

La 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. Típicamente, la división en segmentos se realiza usando un reconocedor de voz modificado para forzar su alineamiento con un texto conocido. Después se corrige manualmente, usando representaciones como la forma de onda y el espectrograma. 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 (selección de unidades). Este proceso se logra típicamente usando un árbol de decisión especialmente ponderado.

La selección de unidades da la máxima naturalidad debido al hecho de que no aplica mucho procesamiento digital de la señal al habla grabada, lo que a menudo hace que el sonido grabado suene menos natural, aunque algunos sistemas usan un poco de procesamiento de señal en la concatenación para suavizar las formas de onda. 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 fue 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.

#### **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 español 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 superpone a estas unidades mínimas mediante procesamiento digital de la señal, como codificación lineal predictiva, PSOLA o MBROLA.

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 difonos adolece de los defectos de la síntesis concatenativa y suena robótica como la síntesis de formantes, y tiene pocas ventajas respecto a estas técnicas aparte 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 parlantes. La naturalidad de estos sistemas puede ser muy alta 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.

#### **4.3.2. 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. Este método se conoce también como síntesis basada en reglas pero algunos aducen que muchos sistemas concatenativos usan componentes basados en reglas para algunas partes de sus sistemas, como el front-end, así que el término no es suficientemente específico.

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 empujados, donde la memoria y la capacidad de proceso son a menudo exigüas. 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.

### 4.3.3. Otros métodos de síntesis

La síntesis articulatoria ha sido un método de interés puramente académico hasta hace poco. 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 es el sistema basado en NeXT, originalmente desarrollado y comercializado por Trillium Sound Research Inc, que pasó más tarde a tener una licencia GPL y se continuó como gnspeech, siendo un proyecto GNU. El software original de NeXT y versiones del software para Mac OS/X y Linux GNUstep están disponibles en junto a manuales y documentos relevantes a los fundamentos teóricos del trabajo. El sistema, que fue comercializado por primera vez en 1994, 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 Speech 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.

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

La Síntesis basada en HMM es un método de síntesis basado en Modelos ocultos de Markov (HMM en inglés). En este sistema, el habla espectro de frecuencias (tracto vocal), frecuencia fundamental (fuente vocal), y la duración (prosodia) se modelan simultáneamente por modelos ocultos de Markov. Las formas de onda se generan desde estos modelos ocultos de markov mediante el criterio de máxima verosimilitud.

### 4.3.4. Desafíos de la síntesis de voz

#### Desafíos de la normalización de texto

El proceso de normalizar texto es pocas veces simple. Los textos están llenos de homógrafos, números y abreviaturas que tienen que ser transformados en una representación fonética.

Por supuesto, en lenguas donde la correspondencia entre el texto escrito y su equivalente fonético es poca (inglés) o ninguna (mandarín), la creación de estos sistemas se complica.



Muchos sistemas de texto a voz no generan representaciones semánticas de los textos de entradas, pues los sistemas para hacerlo no son fiables o computacionalmente efectivos. Como resultado, se usan varias técnicas heurísticas para estimar la manera correcta de desambiguar homógrafos, como buscar palabras vecinas y usar estadísticas sobre la frecuencia de aparición de las palabras.

Decidir como convertir números en palabras es otro problema que tienen que solucionar los sintetizadores de voz. Es un desafío bastante simple programar un sistema que convierta números en palabras, como por ejemplo transformar 1325 en "mil trescientos veinticinco". Sin embargo, los números aparecen en diferentes contextos, y 1325 puede ser un ordinal, uno tres dos cinco" si son los últimos dígitos de un DNI o "trece veinticinco" si es un número de teléfono. A menudo un sistema de síntesis de voz puede inferir como expandir un número en base a las palabras o números vecinos y la puntuación, y algunos sistemas proporcionan un sistema de especificar el tipo de contexto si es ambiguo.

De la misma forma, abreviaturas como 'etc.' se pueden transformar fácilmente en 'etcétera', pero a menudo las abreviaturas puede ser ambiguas. Por ejemplo la abreviatura 'am' puede ser 'ante meridiem' en el ejemplo: 'El vuelo aterrizará a las 11 am' o puede ser 'modulación de amplitud' o simplemente 'a eme' en el ejemplo 'Nos puede encontrar en la sintonía 1425 am'. Los sistemas con front end inteligentes pueden hacer estimaciones adecuadas acerca de como tratar abreviaturas ambiguas, mientras que otros pueden hacer lo mismo en todos los casos, dando resultados en ocasiones cómicos.

#### **Desafíos de los sistemas texto a fonema**

Los sintetizadores de voz usan dos aproximaciones básicas al problema de determinar la pronunciación de una palabra basándose en su pronunciación, un proceso que a menudo recibe el nombre de conversión texto a fonema o grafema a fonema, dado que fonema es el término usado por los lingüistas para describir sonidos distintivos en una lengua.

La aproximación más simple a este problema es la basada en diccionario, donde se almacena en el programa un gran diccionario que contiene todas las palabras de la lengua y su correcta pronunciación. Determinar la pronunciación correcta de cada palabra consiste en buscar cada palabra en el diccionario y reemplazar el texto con la pronunciación especificada en el diccionario.

La otra aproximación para convertir texto en fonemas es la aproximación basada en reglas, donde dichas reglas para la pronunciación de las palabras se aplican a palabras para extraer sus pronunciaciones basadas en su forma escrita.

Cada aproximación tiene ventajas y desventajas. La técnica basada en diccionarios tiene como ventajas ser rápido y preciso, pero falla completamente si una palabra dada no aparece en el diccionario, y, a medida que crece el diccionario crecen los requerimientos de memoria del sistema de síntesis. Por otra parte, la técnica basada en reglas funciona con cualquier entrada, pero la complejidad de las reglas crece sustancialmente a medida que se van teniendo en cuenta ortografías y pronunciaciones irregulares. Como resultado, casi cualquier sintetizador de voz usa una combinación de las dos técnicas.

Algunos lenguajes como el español tiene un sistema de escritura muy regular y la predicción de la pronunciación de palabras basada en deletreos es prácticamente correcta. Los sistemas de síntesis de voz para este tipo de lenguajes generalmente usan un enfoque basado en reglas como el enfoque central para la conversión texto-fonema y auxiliándose de diccionarios pequeños para algunas palabras de origen extranjero cuya pronunciación no se deduce de la escritura. En lenguajes como el inglés, dado que se trata de sistemas muy irregulares en su escritura, el enfoque se basa principalmente en diccionarios y solo para palabras no usuales se basa en reglas.

## 4.4. Sintetizadores de voz disponibles libremente

- **Festival** es un sintetizador de voz disponible libremente basado en concatenación de difonos y selección de unidades. Está disponible para español, inglés británico y americano y galés.
- **Flite** (Festival-lite) es una alternativa más pequeña de Festival diseñado para sistemas empujados y servidores de gran volumen de trabajo.
- **FreeTTS** escrito enteramente en Java, basado en Flite.
- **MBROLA** es un sistema de concatenación de difonos para unos 25 lenguas .
- **Gnusppeech** es un paquete extensible de texto a voz basado en síntesis por reglas articulatoria en tiempo real.
- **Epos** es un sistema texto a voz controlado por reglas diseñado principalmente para investigación. Disponible para checo y eslovaco.

### 4.4.1. Edinburgh Speech Tools

Fue a finales de los 80 y principios de los 90 cuando los ordenadores principales fueron capaces de soportar herramientas de síntesis de voz. La primera barrera que se encontró en este area de investigación fue que debido a la alta complejidad de la síntesis, los investigadores debían pasar la mayor parte del tiempo construyendo una base sobre la que desarrollar.

En 1994 Paul Taylor, Richard Caley y Alan Black empezaron a trabajar en un proyecto para el centro de tecnologías del habla de la Universidad de Edimburgo. El proyecto desarrollado fue una solida y flexible base para que otros investigadores pudiesen trabajar sobre ella. Tras varios años de trabajo el proyecto ya tenia nombre: EST y además estaba casi terminado. Fue entonces cuando Alan W Black apareció con el conocido sintetizador de voz, Festival.

Festival y EST son una colección de programas, ambos de código abierto. El uso de Festival requiere que el usuario baje y compile ambos programas. Los programas han sido diseñados para trabajar juntos. Si un usuario quisiera añadir una base de datos con una nueva voz en Festival, necesitaría las herramientas de grabación de voz de EST. Es más, todas las bases de datos de lexemas y difonemas usadas por Festival han sido creadas con herramientas de EST.

La colección de programas de EST esta formada por programas para la manipulación de audio, manipulación del contorno de la frecuencia fundamental, manipulación del 'pitch', manipulación de la forma de la onda y otras herramientas mas complejas. Esta colección es prácticamente un estándar para la mayoría de los programas de síntesis de voz .

La herramienta de esta colección mas relevante para este proyecto son los árboles CART. Los árboles de CART son un método básico para construir modelos estadísticos de un conjunto de datos. Estos árboles son usados por las herramientas de EST para calcular el grado de impurezas entre 'samples'.

El programa de EST usado para construir estos árboles es conocido con Wagon. Es un programa muy flexible que puede ser usado para construir todo tipo de árboles de CART.

#### 4.4.2. Festival

*'Uno de los grandes problemas en el desarrollo de síntesis de voz, y en otras áreas de procesamiento de lenguajes y habla, es que hay una gran variedad de técnicas conocidas que pueden ayudarte. Pero para mejorar parte del sistema es necesario tener todo el sistema en el cual puedas probar y mejorar tu parte. Festival está pensado como un todo en el cual tu puedas simplemente trabajar en tu parte para mejorar el todo. Festival está diseñado para permitir añadir nuevos módulos, fácil y eficientemente, sin que el desarrollador tenga que recorrer camino ya andado'*

El proyecto de Festival comenzó en 1994 en el centro de tecnologías del habla de la universidad de Edimburgo. Con algunos de los autores de EST en este proyecto, la dependencia entre Festival y EST es grande. Festival es una colección de herramientas de síntesis de voz, en las que está incluido un sintetizador de voz muy completo. Desde su primera versión Festival ha progresado considerablemente, y ahora es considerado como un estándar internacional. Algunas partes de la mayoría de los sintetizadores de voz desarrollados a partir de 1998, incluyendo las comerciales, son de Festival.

Festival usa un lenguaje de programación llamado Scheme. Scheme es un lenguaje muy flexible que empezó a desarrollarse en el MIT en 1975 y desde entonces ha sido muy frecuente su uso cuando había que trabajar con expresiones LISP. Festival está construido como una colección de objetos, que son :phonesets, lexemas, utterances, análisis de texto, entonación, duración y renderización de la forma de la onda.

**Phonesets** es uno de los objetos del núcleo de Festival. Un phoneset es un conjunto de símbolos que pueden ser definidos en términos de características. En ejemplo de características usadas en phoneset podría ser el lugar de articulación de una constante o el tipo de una vocal. Cuando definimos un phoneset, la definición consiste en un único nombre para el phoneset, una lista de características y la definición del fonema. Cuando definimos características, el nombre de la característica estará seguido de sus posibles valores. Una característica de ejemplo, que podría ser utilizada para describir la longitud de una vocal, podría ser definida así: "Length short long diphthong schwa 0"

**Lexicon** es un subsistema que proporciona la pronunciación de las palabras. Tiene tres elementos, una pequeña lista de palabras añadidas a mayores, una lista recopilada de gran tamaño y un método para manejar palabras que no están en ninguna de las dos listas. Está estructurado en tres partes: una cabecera, el tipo de la palabra y la pronunciación de la palabra. La cabecera es como esta escrita la palabra en minúsculas en un texto plano, para que pueda ser identificada. El tipo nos dice simplemente, p.ej, si es una vocal o un nombre. La pronunciación de la palabra es más compleja que las dos primeras partes; separa la palabra en pequeñas unidades para que la palabra suene como tiene que ser.

El proceso léxico es generalmente simple, pero sin embargo se convierte en algo más interesante cuando intervienen homógrafos. Un homógrafo se da cuando más de una palabra se deletrea igual, pero tienen diferentes pronunciaciones y significados. Un ejemplo en inglés sería number que tiene dos significados. El sintetizador tiene que realizar dos análisis: post y pre léxico para decidir que pronunciación es la más adecuada.

**Utterances** es la unidad usada para representar un bloque de texto. Generalmente un utterance puede ser considerado como una sentencia, pero no siempre ha de ser así. En Festival cada utterance es procesada de una en una. Hay varios pasos para convertir una utterance en voz audible.

- Tokenización: la cadena introducida se convierte en un vector de tokens.
- Identificación de tokens: cada token en el array debe ser chequeado para ver de que tipo son.

- Convertir token a palabra:cada token es procesado con un manejador distinto para cada tipo. Los tokens de salida son texto plano. Un ejemplo seria la conversión de '01-Abr-04' en 'Uno de abril del dos mil cuatro'
- Prosodia de la frase(parte de la gramática que enseña la correcta pronunciación y acentuación):el utterance es separado en frases prosodicas. Esto dará como resultado pausas en la salida del habla. P.ej, una pequeña pausa para una coma.
- Búsqueda lexica:el subsistema léxico se usa para calcular la pronunciación del token dado.
- Acentos de entonación:se aplica a las sílabas en las que sea necesario.
- Asignación de duracion:la duración de cada fonema en el utterance necesita ser calculada y almacenada en el propio utterance.
- Generar contorno F0(frecuencia fundamental):el acento tonal de la voz necesita ser calculado para el utterance
- Interpretar la forma de la onda:este paso es el mas complicado y depende del tipo de síntesis utilizado. Esto incluye cambiar los resultados de los pasos anteriores para conseguir un habla audible.

**Análisis de texto** es el nombre dado a los pasos de tokenizacion. El análisis de texto convierte tokens que no son palabras en tokens que si lo son .Cuando creamos la lista inicial de tokens cada palabra debe ser seleccionada. Esto puede ser llevado a cabo extrayendo palabras que se sitúan entre espacios blancos, sin embargo pueden darse casos donde hayan palabras que incluyan símbolos(.,:;). Despues de que las palabras sean introducidas en el array de tokens, el tipo de cada palabra debe ser identificado. Esto se lleva a cabo comparando expresiones regulares con el token. Los tokens generalmente son alfabéticos; pero pueden ser numéricos, una fecha,una hora,un url, un email... Después de que las palabras individuales hayan sido identificados, el siguiente paso es la desambiguación de homógrafos. Esto implica detectar y resolver homógrafos.Este proceso esta diseñado para localizar un homógrafo en una sentencia y el contenido alrededor suyo.

La **entonación** puede ser definida como las subidas y caídas del nivel de la voz. La entonación añade un elemento de realismo a las voces sintéticas. También puede ser utilizada para modificar el acento de una voz. Hay varios tipos de entonaciones usadas por Festival. La entonación simple consiste en usar un árbol CART para predecir si una sílaba tiene acento o no. La entonación en árbol usa dos árboles CART, uno para predecir los acentos y el otro para el tono de los finales de la frase.

El **calculo de la duración** sirve para calcular como las frases han de ser moduladas desde un punto de vista temporal. Existe una variable global en Festival que puede ser modificada en un script Scheme para cambiar la duracion. Si aumentas el valor de la variable, la voz de la salida sera mas lenta. Festival también soporta la técnica usada para calcular la duración creada por Dennis Klatt.

Festival utiliza el sintetizador **UniSyn**. El proceso de síntesis implica seleccionar unidades o fonemas de una base de datos,generalmente ya existente . Las unidades se concatenan acorde a la tokenización del utterance. La onda de la concatenización inicial es combinada con el calculo de la duración y la entonación para producir lo que se conoce como 'rendered waveform'. La 'rendered waveform' es ya el habla sintetizada

#### 4.4.3. Flite

Flite(festival-lite)es un sintetizador pequeño y rápido desarrollado en la Universidad de Carnegie-Mellon y dirigido principalmente para dispositivos moviles.Flite fue diseñado como una alternativa a

Festival utilizando las mismas voces construidas con FestVox. Flite ofrece:

- Mejor portabilidad, tamaño y velocidad, al estar escrito completamente en C (no en C++ o Scheme)
- Reimplementación de partes del núcleo de la arquitectura de Festival manteniendo la compatibilidad entre voces construidas para cada sistema
- Soporte para utilizar voces desarrolladas en FestVox.

El código fuente del sintetizador Flite está organizado en carpetas, cada una para un área específica de la síntesis de voz, las carpetas son: salida de audio(audio), localización de memoria, conversión endian, manipulación de cadenas de caracteres(utils), expresiones regulares de Henry Spencer(regex), ondas y pistas(speech), interprete de CART(stats), items, relaciones y utterances(hrg), léxico(lexicon), funciones genéricas de síntesis(synth), síntesis fonemas y creación de la prosodia(wavesynth).

#### 4.4.4. FreeTTS

Flite es bueno para la síntesis ligera. Sin embargo como todos los programas escritos en C, el código debe ser compilado separadamente para cada plataforma. Esto puede hacer que Flite sea particularmente complicado cuando tratamos de compilarlo cruzadamente para una PDA (dado que las modernas PDA's tienen un amplio rango de procesadores incompatibles). El grupo de integración del habla de Sun desarrolló un API para síntesis y reconocimiento del habla. El API conocido como Java Speech API (JSAPI), empezó a ser usado comúnmente cuando los grandes vendedores de software de síntesis de voz que utilizaban clases de Java comprobaron que JSAPI era compatible con sus sintetizadores.

El grupo de integración del habla empezaron a desarrollar su sintetizador de voz gratuito y de código abierto llamado Free Text To Speech (FreeTTS).

FreeTTS fue diseñado como un sintetizador ligero que utilizaría menos recursos que Festival al estar escrito en Java y que estaría muy influenciado por Flite.

Dado que FreeTTS iba a ser escrito en Java y Flite estaba en C, el código tendría que ser reescrito. Esto sin embargo, permitía a los desarrolladores analizar y mejorar las ideas de Flite. La estructura básica de FreeTTS es casi idéntica a la de Flite. Sin embargo, FreeTTS está orientado a objetos, esto provocó varios cambios respecto a Flite.

La síntesis de voz es un proceso complejo, y Java es un lenguaje interpretado por lo que la velocidad es un problema. Java 2 SE versión 1.4 es muy diferente que las anteriores versiones de Java. Más que centrarse en incrementar la funcionalidad de los API's de Java, Sun se centró en hacer Java más rápido. La técnica que ellos usaron para conseguir este incremento en la velocidad fue añadir más clases, pero que estas clases fueran dependientes de la plataforma. Los cambios incluían un nuevo paquete de E/S, optimizaciones en la compilación... J2SE 1.4 es excelente para programas Java que necesitan trabajar rápido. El principal problema surge cuando los programas que trabajan en una plataforma no son totalmente compatibles con otras plataformas (FreeTTS por ejemplo necesita parches para trabajar en Linux). J2SE 1.4 fue la primera versión de Java que incluía soporte para expresiones regulares. Es por estas razones que FreeTTS presenta una gran dependencia con J2SE 1.4.

Dado que FreeTTS está basado en Flite, el cual está basado en Festival, hay muchas similitudes con la arquitectura previamente descrita de Festival. Los objetos principales que utiliza FreeTTS son: FreeTTSSpeakable, Voice, VoiceManager, Utterance, FeatureSet, Relation, Item and UtteranceProcessor.

FreeTTSSpeakable es una interfaz usada para representar cualquier ítem que pueda ser hablado. Hay una implementación incluida de esta interfaz para hablar una cadena de caracteres. Voice es el objeto principal, proviene de una implementación de la interfaz anterior y usa otros objetos para desarrollar la síntesis. El objeto VoiceManager puede ser usado para acceder a todas las bases de datos de voces disponibles. Las voces están contenidas en ficheros jar en el classpath. El objeto utterance es usado para representar utterances, que son procesadas de una en una. FeatureSets es un par nombre-valor que es usado para mantener variables globales de utterance, como el pitch o la duración. Item es un grupo de objetos FeatureSet. Relation es una lista ordenada de objetos Item. Los objetos UtteranceProcessor son usados para desarrollar una operación individual sobre un utterance.

El proceso de síntesis en FreeTTS es muy similar a Festival. El primer paso es la tokenización. Cada token es extraído de la entrada, y almacenada con su número de línea, posición del fichero, puntuación, pre-puntuación y espacios en blanco. Después de la inicial tokenización, el siguiente paso es convertir cada token a tokens alfabéticos. El siguiente paso calcula como el utterance debe ser hablado. Esto se realiza creando una 'relation' en el utterance. El siguiente paso es la fase de segmentación. Esto implica utilizar el subsistema lexicon para separar cada palabra en sílabas. La unidad "segment relations" se usa para grabar los fonemas de cada palabra. Después del proceso de segmentación la utterance tiene 'relations' para cada segmento. El 'PauseGenerator' añade pausas en las relaciones de los segmentos. Cada utterance siempre comienza con una pausa.

En este momento el texto está en una forma muy segmentada, y está listo para una reproducción de muy baja calidad. La 'entonación' se aplica a los segmentos para añadir realismo al habla. Añade acentos y tonos de fin de habla a los segmentos. Este proceso, previamente descrito en Festival, usa dos árboles CART para acentos y predicción de tono. PostLexical Analysis es una etapa que 'convierte' los segmentos en caso de error. La siguiente fase es la predicción de la duración, lo cual solo implica predecir el tiempo requerido para pronunciar cada segmento. Esta predicción se calcula usando un árbol CART que prediga la longitud de cada segmento. La longitud de cada segmento es almacenada con los segmentos antes de avanzar a la siguiente etapa, que es generación del contorno. Esta calcula la frecuencia fundamental para que la utterance sea procesada. Esta basada en trabajos de Alan Black. El objeto UnitSelector es entonces seleccionado para seleccionar las unidades de la voz. El objeto Pitch-MarkGenerator se usa para calcular las marcas del Pitch para la utterance. Finalmente, las unidades se concatenan. Esta concatenación es enviada al dispositivo de audio.

#### 4.4.5. MBROLA

El objetivo del proyecto MBROLA, iniciado por el laboratorio TCTS de la Facultad politécnica de Mons (Bélgica), es obtener un conjunto de sintetizadores de voz para la mayor cantidad de lenguas posibles y que puedan disponerse gratuitamente para aplicaciones no comerciales. La última meta es mejorar la investigación académica en síntesis de voz, particularmente en la generación de la prosodia, conocido por ser uno de los mayores retos de la síntesis de voz.

El núcleo del proyecto MBROLA es, valga la redundancia, MBROLA, un sintetizador de voz basado en la concatenación de difonos. El toma una lista de fonemas como entrada, junto a la información prosódica (duración de los fonemas y una pequeña descripción del pitch), y produce samples de voz de 16 bits. Este sintetizador está disponible libremente, excepto para aplicaciones comerciales o militares.

Las bases de datos de difonemas desarrolladas con el formato de Mbrola deberían ser compartidos. Desde el proyecto MBROLA se incita a ello. Los términos de esa política de compartición pueden ser resumidos como sigue:

Después de un acuerdo entre el autor de MBROLA y el propietario de la base de datos de difonemas, la base de datos es procesada por el autor y adaptada al formato MBROLA, gratuitamente.

MBROLA no es un conversor de texto a voz realmente ya que no es capaz de leer un texto sino que necesita los fonemas.

Algunos de los sistemas de TTS compatibles con MBROLA son Festival, Euler, eLite...

Desde la universidad de Valladolid se ha desarrollado una base de datos en español llamada es4 por Cesar González Ferreras, Valentín Cardeñoso-Payo y David Escudero Mancebo.

#### 4.4.6. Microsoft Speech SDK 5.1

El Microsoft Speech SDK 5.1 añade soporte a las características de la anterior version del Speech SDK. Puedes usar el Win32 Speech API(SAPI) para desarrollar aplicaciones con Visual Basic, Visual C++...

El SDK también incluye gratuitamente motores de TTS en Inglés americano y chino simplificado.

Requisitos del sistema:

- Sistemas operativos soportados:
  - Windows XP Windows XP Professional o Home Edition
  - Microsoft Windows 2000, todas las versiones
  - Microsoft Windows Millennium Edition
  - Microsoft Windows 98, todas las versiones
- Microsoft Internet Explorer ® 5.0 o posterior
- Microsoft Visual C++ ® 6.0
- Microsoft Visual Studio.NET

#### 4.4.7. Comparativas estadísticas de algunos sintetizadores

##### Flite vs Festival

Flite fue diseñado con la intención de ser mas ligero que Festival por lo que no es sorprendente ver que es mas pequeño y rápido

Prueba	Flite	Festival	Flite es mas eficiente en un
Memoria	5 Mb	40 Mb	12.50 %
Velocidad	19.1 s	97.s	19.69 %
Tiempo hasta la primera habla(Utterances de 20 palabras)	45 ms	1000 ms	4.50 %
Tiempo hasta la primera habla(Utterances de 40 palabras)	75 ms	2000 ms	3.75 %

**Cuadro 4.1: Flite vs Festival en un PIII 500 Mhz Linux**

Esta tabla muestra una comparación entre Flite y Festival. El valor memoria indica la memoria requerida para que el programa sea cargado y ejecutado realizando síntesis en tiempo real. La velocidad esta calculada con Linux de sistema operativo sobre una maquina Intel Pentium 3. Flite es casi 5 veces mas rápida que Festival. El tiempo que tarda en el primer habla es una media de cuanto tarda el sintetizador en inicializarse y procesar el primer utterance. Hay datos para varios tamaños de utterances.

#### FreeTTS vs Flite

Los sintetizadores fueron probados anotando el tiempo requerido para cargar y procesar dos diferentes archivos de texto. Uno de ellos fueron los dos primeros capítulos de Alicia en el pais de las maravillas de Lewis Carroll (unos 20 minutos de texto), y el otro contiene la totalidad del libro Viaje al centro de la tierra de Julio Verne (unas 8 horas de texto).

Prueba	Flite	FreeTTS
Tiempo de carga para 'Alicia'	0.0 sg	4.1 sg
Tiempo de procesamiento para 'Alicia'	43.7 sg	24.1 sg
Tiempo de carga para 'Journey'	0.0 sg	7.0 sg
Tiempo de procesamiento para 'Journey'	1019.2 sg	341 sg
Tiempo hasta el primer Sample (10 palabras por frase)	195 ms	41 ms

**Cuadro 4.2: Flite vs FreeTTS en un CPU 296 Mhz Sparc v9**

Se puede ver como en ambas pruebas Flite carga el texto mucho mas rápido que FreeTTS. Esto se debe a que el soporte de E/S en Java no permite que un fichero sea directamente cargado en un buffer de memoria, mientras que C si. Esto quiere decir, que FreeTTS tiene que cargar el fichero carácter a carácter.

### 4.5. Sintetizadores de voz disponibles comercialmente

- DSC Text To Speech Software Aplicaciones y demostraciones de text a voz.
- Loquendo TTS para español (varios acentos), catalán, holandés, inglés, portugués, italiano, francés, alemán, griego, sueco y chino. Hay disponibles demostraciones interactivas vía web.
- RealSpeak por Nuance para español, inglés, alemán y griego entre otros muchos.
- IBM Research TTS (ejemplos para inglés americano, árabe, chino, francés y alemán).
- NeoSpeech VoiceText
- Sakrament Text-to-Speech Engine para ruso y algunas otras lenguas.
- SVOX Especialista suizo en soluciones empujadas de voz para 18 lenguas.
- Verbio TTS - Applied Technologies on Language and Speech (ATLAS) para francés, español e inglés.
- Vocaloid sintetizador de voz de Yamaha.
- ASY es un sistema de síntesis de voz articulatoria desarrollado en los Laboratorios Haskins .
- iFlyTek InterPhonic es un sintetizador basado en corpus desarrollado por una compañía china.



- VoiceText es un sistema de síntesis de voz concatenativo realizado por Voiceware, Corea.
- Wizzard Software ofrece sistemas texto a voz.
- Acapela Group con muchos sintetizadores de voz que soportan muchas lenguas
- Voces de texto a voz de ATIP para el alemán (también con acentos francés y Turco) e Inglés.
- Voces naturales AT-T para español, inglés, alemán y francés.
- Cepstral para español, inglés, italiano, alemán y francés.

#### 4.5.1. Fonix Speech

Sus principales características son:

- Nueve voces modificables: cuatro voces de hombre, cuatro de mujer y una de niño. Voces claras e inteligibles, incluso en entornos ruidosos.
- Siete idiomas disponibles: italiano, inglés americano y británico, español castellano y latinoamericano, francés y alemán.
- Poca utilización de la memoria: 1 mega por lenguaje. El uso de memoria RAM esta entre 64 y 256 Kb dependiendo de la implementación.
- Compatible con múltiples plataformas y sistemas operativos: todas las variantes windows, linux, symbian, palm 5 y Mac OSX.
- Capacidad de enunciar cartas, palabras y frases( incluso ficheros zip).
- Control de rapidez del habla de 75 a 600 palabras por minuto.
- Soporta los SAPI de Microsoft.

#### 4.5.2. Acapela Mobility

Incluye TTS y ASR(multilingual speech recognition).

Las características del TTS son:

- 22 lenguajes disponibles
- Voces masculinas y femeninas.
- Control óptimo del sonido.
- Compatible con C, C++, VB, C#.
- Sistemas operativos: PPC2002 y Windows Mobile 2003.

#### 4.5.3. Cepstral

Cepstral proporciona voces de muy buena calidad que consumen pocos recursos de memoria y disco (20-90 MB cada voz). Es posible evaluar las voces de este fabricante antes de comprarlas. Tiene voces en Inglés-Americano, Inglés-Británico, Francés-Canadiense, Alemán, Español-Latinoamericano.

#### **4.5.4. AT&T Natural Voices**

Son voces de alta calidad con una entonación excelente. Primero es necesario instalar el motor con las voces de Mike y Crystal (Inglés-Americano), y después se pueden añadir voces en otras lenguas. Cada voz necesita al menos 600 MB de disco. Tiene voces en Inglés-Americano, Inglés-Británico, Francés-Parisino, Alemán, Español-Latinoamericano, Inglés-Acento de la India.

## Capítulo 5

# Uso de interfaces animadas y síntesis de voz

### 5.1. Necesidad de interfaces de usuario amigables

El crecimiento de los ecosistemas de la información y la computación, está relacionado con un cambio fundamental en la sociedad, caracterizado por la evolución de una industria hasta ahora tradicional de manejo de bienes, hacia la de una industria del conocimiento y de la economía de servicios de información. La producción, distribución y administración de información se han transformado ya en las actividades principales de las sociedades modernas basadas en el manejo del conocimiento.

La necesidad de interactuar con los ordenadores está penetrando en muchos aspectos de nuestra vida cotidiana. Ya sea en bancos, estaciones de tren, hoteles o aeropuertos, se encuentran innumerables puntos de información electrónicos en los que reside gran parte de la información necesaria, o al menos útil, para desplazarse, comerciar o informarse acerca de temas de ocio. Las personas que no están básicamente tecno-alfabetizadas se encuentran hoy en día con serias limitaciones en su acceso a la información, ya sea ésta trivial o fundamental.

Ya a finales de los 60 se decía que en el futuro, no se requeriría de personas orientadas al ordenador, sino de ordenadores orientados a las personas. El tiempo ha demostrado el acierto de esta predicción. En este sentido, la interfaz entre el ser humano y los sistemas informáticos debe ser mejorada continuamente para lograr que el acercamiento a la comunicación con las máquinas no resulte tan traumático para muchos.

La mejora en la comunicación hombre-máquina ha dependido del grado de desarrollo tanto del hardware como de los sistemas operativos. Puede ser de utilidad echar la vista atrás y recordar su evolución

### 5.2. Desarrollo de las Interfaces de Usuario (IU)

En los años 60, los sistemas 'más interactivos' utilizaban terminales teletipo, semejantes a una máquina de escribir (TTY), que necesitaban del papel como recurso de visualización. Luego, los diseñadores de interfaces basadas en tubos de rayos catódicos (CRT) basaron sus modelos de presentación gráfica y textual según el mismo modelo. Aquellos kilométricos listados de papel pasaban ahora de forma virtual a estar detrás de un cristal.

La metáfora del teletipo fue la base del conocido MS-DOS, con su línea de comandos basada en texto

que aún se padece y que tanto atemoriza a los no iniciados en ese arcaico C: .

En la década de los 60, algunos investigadores como Ivan Sutherland (inventor de la primera interfaz basada en ventanas) y Douglas Engelbart (inventor del ratón), estuvieron ya diseñando sistemas espaciales para pantallas de rayos catódicos que emulaban la complejidad gráfica de los documentos impresos utilizando la capacidad de ajuste dinámico de caracteres que ofrecía el ordenador superando las limitaciones del papel.

La base conceptual de la mayoría de las interfaces gráficas de usuario o GUI (Graphic User Interface) de hoy en día fue desarrollada durante los años 70 en los laboratorios de Xerox Palo Alto Research Center (PARC). Estos conceptos incluyen metáforas gráficas explícitas en pantalla para objetos tales como documentos y programas de ordenador; ventanas múltiples superpuestas para subdividir actividades en la pantalla; y manipulación directa de ventanas, iconos y otros objetos utilizando el ratón de Engelbart como elemento de señalización.

Los ordenadores de entonces, y aún los actuales, han requerido siempre de razonamiento abstracto. La tarea de los investigadores de PARC fue la de crear una interfaz que pudiese explotar además las habilidades manipulativas y visuales del usuario. El objetivo fue diseñar analogías gráficas de objetos familiares del mundo real en la pantalla, para crear la ilusión de que la información digital podía manipularse tan fácilmente en el ordenador, como se hacía con los documentos impresos en un escritorio.

El trabajo sobre interfaces realizado en Xerox PARC a mediados de los 70 ha establecido la mayoría de las convenciones funcionales y visuales de las GUI actuales, y han sido las antecesoras de las GUI de Apple Macintosh, de Microsoft Windows, y de otras como UNIX Motif, NextStep, o Open Look.

De las GUI anteriores, el estándar actual, no de facto pero sí de mercado, lo representa Windows y sus aplicaciones. Su interfaz se puede definir como aquella interfaz gráfica en la que el usuario navega a través de menús y ventanas de diálogo con la ayuda del ratón y del teclado.

El hecho de llegar a un cierto estándar ha facilitado a los usuarios habituados al uso de este tipo de interfaces el conocer ciertas partes comunes de todas las aplicaciones, correspondientes a las tareas más frecuentes, teniendo la seguridad de que funcionan igual en otros programas. Por lo tanto: pasado una primera etapa de acercamiento y adaptación a estas interfaces, la interacción del usuario con la máquina no es difícil, aunque tampoco cómoda (por no ser natural), ni poderosa (la interfaz de salida se reduce casi exclusivamente a la pantalla y como interfaces de entrada tan sólo se dispone del teclado y del ratón) y además puede resultar algo fría.

### 5.3. Interfaces de Usuario con personajes animados

El estudio de la interacción hombre-ordenador o HCI (Human Computer Interface) puede contribuir a generar mejores sistemas y servicios multimedia con interfaces de usuario realmente amigables.

En la actualidad los investigadores en HCI se están dedicando a explorar nuevas metáforas y arquitecturas. Sus investigaciones están descubriendo las numerosas ventajas que presenta la interacción con un sistema multimodal respecto a la interacción de sistemas unimodales.

Una definición de interfaz multimodal sería: es aquella interfaz que no presenta la limitación de entrada y salida de datos mediante un solo canal. Estos sistemas tratan de implicar varias de las posibles modalidades de comunicación que utilizan las personas. Estas modalidades comprenden habla, reconocimiento de gestos, reconocimiento de la posición de la mirada, del movimiento de los labios, de la

expresión facial, de la escritura, etc.

El propósito de utilizar diferentes modos es liberar a las personas del uso de una interfaz rígida, eliminando las barreras de comunicación presentes en la mayoría de los sistemas actuales. El principal objetivo es llegar a una interacción natural entre persona y máquina, esto es, una interacción que se asemeje en la medida de lo posible a la interacción persona-persona.

Además, los sistemas multimodales salvan la dificultad de expresar oralmente información espacial, también eliminan ambigüedades y reducen la tasa de error al poder mostrar la información en diversos modos al mismo tiempo. La información expresada por los distintos modos de comunicación no siempre es redundante sino que en muchas ocasiones cada uno de ellos proporciona información complementaria e incluso puede que imprescindible para la consecución de una tarea específica.

Fomentado por los numerosos beneficios potenciales de este tipo de comunicación multimodal, se ha iniciado recientemente una línea de trabajo basada en el desarrollo de Interfaces de Usuario que permiten la interacción hablada con un agente informático. Para ello, incorporan reconocimiento del habla, comprensión del lenguaje natural, gestión de conversación y apariencia de personaje animado para simular mejor una interacción persona a persona.

Algunas de las motivaciones que han conducido al desarrollo de estas interfaces son:

- Los agentes animados con interfaces conversacionales proporcionan un paradigma intuitivo de interacción ya que el usuario no necesita adquirir nuevos conocimientos.
- Estas interfaces presentan redundancia y complementariedad en los modos de entrada. Lo que aumenta la fiabilidad de la comunicación entre sistema y usuario.
- Los usuarios encuentran estos sistemas más amigables y cooperativos. Los agentes autónomos pueden utilizar esa ventaja para entablar una conversación con los usuarios de forma más natural. La creación de un agente animado engloba la investigación lingüística, tecnología del habla, ilustración gráfica, etc. Al tratarse de un campo que abarca numerosas disciplinas, el desarrollo de estos sistemas presenta abundantes problemas en la creación de cada componente y en la integración de los mismos.

Una de las principales dificultades consiste en reproducir de forma fidedigna un diálogo cara a cara. Estos diálogos presentan un comportamiento irregular con numerosas excepciones a las reglas. Se ha de prestar atención a cómo sincronizar los labios del agente con lo que dice, a las expresiones faciales utilizadas y a los gestos que realiza.

Los desarrolladores necesitaremos de herramientas que nos faciliten el trabajo de integración de los agentes. Una de esas herramientas es Microsoft Agent.

## **5.4. Elementos de conversación cara a cara con personajes animados**

Cuando se diseña interfaces con personajes, la motivación última es conseguir trabajar con computadoras sin teclados, las cuales acepten entradas naturales no entrenadas y respondan en consecuencia. En situaciones tales como esas, se necesitan personajes bien implementados para poder interactuar con ellos usando las múltiples facetas de la conversación natural.

Una conversación cara a cara se caracteriza principalmente por el empleo del lenguaje, pero se emplean,

además, otras técnicas o habilidades: los interlocutores emplean gestos con las manos para enfatizar o representar ideas, se dirigen miradas expresivas y utilizan variaciones en el tono o melodía de las palabras o frases articuladas.

El ejemplo de la conversación puede ser considerado como el paradigma de la interacción humana. Es por ello que cuando los diseñadores quieren construir interfaces hombre/máquina recurren a la metáfora de la conversación cara a cara como ideal de esta interacción.

Nickerson, en 1976, fue de los primeros que argumentó acerca de la utilidad de la aplicación de esta metáfora a la interacción hombre/máquina. Expuso una lista de las características que debería poseer una interfaz. Esas características son: poseer elementos de comunicación no verbal, ser capaz de tomar la iniciativa, dar sensación de presencia e incluir reglas de transferencia del control.

Aunque estas reglas se ganaron el reconocimiento por parte de los diseñadores, hasta la fecha no se había intentado llevarlas a la práctica seriamente. Es ahora cuando los diseñadores están trabajando para conseguir interfaces que puedan sostener una conversación. Estas interfaces transmiten emociones y se comportan en función de la demanda del diálogo, de su propia personalidad y de convencionalismos sociales. Además, poseen un cuerpo para usarlo como elemento expresivo y se conocen como agentes conversacionales dotados de cuerpo.

Un camino para reflexionar acerca del problema al que se enfrentan los diseñadores es imaginar que es posible construir un agente conversacional dotado de cuerpo que muestre características tan humanas que pueda sostener una conversación cara a cara con un humano y plantearle serias dudas acerca de su condición humana o artificial. Para que ese agente sea capaz de superar esa prueba (el mismo tipo de prueba a la que sometía Harrison Ford a los replicantes en la película *Blade Runner*), ¿de qué modelos de conversación humana deberíamos dotarle y qué comportamientos superficiales debería mostrar?

Las conductas que deben mostrar los agentes para conseguir desarrollar una conversación que resulte natural son muy extensas, pero se han de vertebrar alrededor de las siguientes pautas:

- Emoción: que el agente exprese emociones ayuda a reforzar su mensaje durante una conversación. El perfil emocional de un agente lo determina, en parte, la forma de llevar a cabo sus acciones: El personaje puede exhibir expresiones faciales que denoten emoción y gestos expresivos, por ejemplo, para aconsejar, animar o enfatizar algo al usuario. Aparte de generar respuestas emocionales en los agentes, sería muy útil el reconocimiento de las emociones del usuario. Esto se podría hacer a partir de observar características como su habla, sus gestos y sus expresiones faciales. Con las herramientas PC actuales, implementar algo parecido resulta complejo y excede de los objetivos de este proyecto.
- Personalidad: Los Agentes con una personalidad consistente resultan más atractivos y hacen que la información que transmiten sea más amena y fácil de recordar, además, son percibidos por el usuario como más inteligentes y útiles. La personalidad de un Agente ha de ser elegida teniendo en cuenta sus tareas específicas en un contexto dado, pues la personalidad presenta al personaje con sus propias parcelas de conocimiento y perfiles de interés. Para conseguir que el personaje sea un individuo distinguible con un carácter propio, ha de ser dotado de una serie de actitudes que sean consistentes a lo largo de toda la interacción. Estas actitudes se revelan a través de los movimientos, conversaciones e interacciones con el usuario y con otros personajes.
- Se deben considerar los objetivos de la conversación: La razón por la que el hablante está comunicando una cosa (es decir, el objetivo que tiene en mente el hablante) y la forma de esta comunicación están muy relacionadas. Las contribuciones a una conversación pueden ser de propuesta y de interacción :

- La información de propuesta corresponde al contenido de la conversación. Incluye procesos de habla con sentido y gestos utilizados como complemento al habla o basados en el contenido de la oración (como los gestos para indicar un tamaño).
- La información de interacción corresponde a las señales que regulan el proceso de la conversación. Incluye procesos de habla y comportamientos no verbales que no producen información (como afirmaciones con la cabeza para indicar que se sigue la conversación).

A partir de este análisis se pueden inferir ciertos modelos que son de aplicabilidad directa en los agentes. Así, teniendo en cuenta lo que un personaje quiere expresar, se puede acompañar su discurso con cierta expresión facial o gesto corporal. Como ejemplo, el personaje podría apretar el puño mientras realiza una amenaza, fruncir el ceño cuando no ha entendido algo y solicita una explicación o realizar afirmaciones con la cabeza para indicar que atiende al usuario.

- Pautas de conversación: Como puede ser la gestión del turno de palabra, las interrupciones y de la toma de iniciativa. Como los interlocutores no suelen hablar al mismo tiempo, se pueden determinar técnicas para establecer a quién pertenece el turno en cada momento. Para determinar esto, se pueden utilizar factores que incluyan la mirada o la entonación. Las interrupciones producidas por el oyente no se realizan únicamente mediante la voz, sino que también se pueden producir con un gesto para solicitar que se desea el turno.

Aún no se pueden construir agentes con técnicas conversacionales perfectas. Los modelos de emoción, de personalidad, de conversación son todavía rudimentarios. Y el número de conductas conversacionales que pueden ser realizadas en tiempo real usando cuerpos animados es todavía extremadamente limitado. Pero a la vez que se empieza a comprender las habilidades que subyacen en la conversación humana, y se aprecian las conductas que la conforman, se aproxima al día en el que una conversación cara a cara con un agente pueda llegar a ser imaginable.

## 5.5. Directrices para el diseño de la interacción con personajes

Para diseñar una buena interfaz basada en personajes, antes que nada, es necesario seguir las reglas de diseño de interfaces de usuario en general, y además, tener en cuenta las peculiaridades intrínsecas a este tipo de interfaces.

### 5.5.1. Emplear sonido

Uno de los elementos fundamentales para constituir un entorno de interacción con personajes es el sonido. Su presencia, no es en sí un requisito necesario, pero sí una parte esencial para la constitución de un entorno de máxima interacción. Como ya ha sido comentado, los humanos consideramos al habla como una forma habitual, espontánea y sencilla para comunicarnos entre nosotros.

Evidentemente, esta vía de comunicación puede ser sustituida por otras en aquellos contextos en que las circunstancias lo requieran. Porque, hablar no es siempre la mejor forma de entrada para solicitar una tarea. Por ejemplo, debido a la alternancia del turno de palabra en el lenguaje natural, esta forma de entrada puede ser en ocasiones más lenta que otras. Además, igual que el teclado, la entrada de voz es una pobre interfaz para apuntar en la pantalla a menos que se proporcione algún tipo de representación mnemotécnica. Por lo tanto, hay que evitar usar el habla como interfaz exclusiva. Se debe proporcionar otros caminos alternativos (ratón, teclado...) para acceder a cualquier funcionalidad básica.

Muchos de esos problemas se solventan si se implementa una interfaz multimodal, donde se complementa la entrada de habla con información visual que ayude a especificar el contexto y las opciones que

se admiten.

Para la construcción de un entorno de interacción con personajes en el que se utilice el sonido como interfaz de usuario puede ser necesaria la presencia de ciertos periféricos y sistemas software:

- Micrófonos, que se utilizan para reconocer la voz y obtener una representación, en forma de texto, de las palabras pronunciadas por el usuario.
- Gestores de diálogos, que pueden incluir:
  - Analizadores, parsers y gramáticas que identifican el texto reconocido dentro de un diálogo.
  - Gestores del flujo de diálogo que hacen posible establecer una conversación coherente con el entorno.
- Altavoces para comunicarse con el usuario, lo que puede implicar:
  - Generadores de sonido grabado.
  - Sintetizadores de voz que permitan generar una voz reconocible por el usuario a partir del texto que se quiere transmitir.

### 5.5.2. No ser exclusivo

Cuando se incluye un personaje interactivo en la interfaz de usuario de una aplicación no ha de hacerse con el ánimo de reemplazar la interfaz primitiva, sino con la intención de mejorarla. Obligar al usuario a tratar en exclusiva con el personaje puede acarrear un serio efecto negativo. Para evitarlo hay que permitir al usuario otras vías alternativas de interacción con la interfaz.

Mantener al usuario con el control es un principio de diseño que debe de cumplirse siempre. Este hecho implica que el usuario ha de poder decidir cuando quiere interactuar con el personaje y cuándo no.

### 5.5.3. Proporcionar la apropiada realimentación

Una interfaz de usuario que incorpore personajes interactivos puede proporcionar formas más naturales de realimentación. Además crea al usuario la expectativa de que la comunicación se va a realizar conforme a las normas de interacción social.

La localización del personaje en pantalla debe poder ser inferida por el usuario, según el contexto o estado, esto dará idea de una actuación racional del personaje. Así, por ejemplo, el personaje puede desaparecer siempre por la posición de pantalla en la que aparece.

Una realimentación es adecuada dependiendo del momento. Durante una interacción usuario/ personaje se pueden distinguir distintos estados:

- Cuando no hay interacción usuario/personaje: Durante este estado hay que evitar las distracciones. Esto puede hacerse moviendo al personaje a un punto de stand-by, donde no interfiera con el desarrollo de la acción, por ejemplo, mientras el usuario lee un texto. En esa situación es preferible dotar al personaje de una conducta ociosa (como respirando o mirando alrededor) para mantener la ilusión del contexto social, pero, reduciéndose al mínimo los efectos de sonido y las animaciones. Otra opción a valorar es la posibilidad de esconder al personaje. En este caso hay que dejar claro al usuario por qué se oculta el personaje y qué es lo que puede hacer para volver a verle.
- Cuando existe una interacción directa, es decir, el personaje participa en la tarea actual del usuario: En ese momento se debe situar al personaje en el foco de atención.



- Cuando se pretende capturar la atención del usuario: Para ello, particularmente si el personaje está fuera del foco de atención, se puede intentar la translación del personaje, la ejecución de una animación muy activa o de un movimiento o animación dirigiéndose al usuario. En la siguiente figura vemos un ejemplo. Se trata de un fotograma correspondiente a una animación del personaje Merlín en la que simula que golpea con los nudillos en la pantalla del ordenador.



Figura 5.1: Merlín atrayendo la atención del usuario

#### 5.5.4. Usar variaciones naturales

Una interfaz convencional, con menús y ventanas de diálogo, posee consistencia al ser predecible, causal y repetible.

Por contra, en una interfaz con personajes es necesario variar las respuestas de una forma natural para que el usuario no la considere aburrida, sin interés, ruda o poco inteligente. La comunicación humana raramente repite exactamente. Incluso repitiendo algo en una situación similar, se cambian las palabras, los gestos o las expresiones faciales.

#### 5.5.5. Interacción social

La comunicación humana es fundamentalmente social. Así, la parte más esencial de ésta, el lenguaje, pierde efectividad si no se ve reforzado por unos convencionalismos sociales que vamos aprendiendo por imitación a través de relacionarnos con los demás.

Estos refuerzos pueden ser tanto verbales como no verbales. Aspectos relativos a la voz podrían ser: la entonación o la ordenación de las palabras, y aspectos no verbales: la postura corporal, los ademanes o gesticulación (especialmente con las manos) y las expresiones faciales.

La efectividad de este tipo de comunicación es debida a que añadiendo estos refuerzos se define mejor lo que el mensaje quiere transmitir. El comunicador refleja su actitud ante el interlocutor y ante el mensaje, muestra su identidad y expresa emociones. De este modo añade el aspecto subjetivo que personaliza y humaniza la comunicación.

Es una necesidad el emplear refuerzos sustitutivos en los canales de comunicación que por sus limitaciones no ofrecen la posibilidad de incluir los de la comunicación bis a bis. Un ejemplo de esto es el uso generalizado de los llamados smileys o emoticones en los correos electrónicos.

Las interfaces software hasta la fecha han dejado al margen la componente social de la comunicación. Pero ahora se está demostrando que las personas reaccionamos con naturalidad ante estímulos sociales presentados en contextos interactivos. Esta respuesta automática se refuerza ante un personaje animado con ojos y boca. La consciencia del usuario de estar tratando con un personaje artificial no rebaja sus expectativas de percibir en el personaje una conducta socialmente apropiada. Por lo que este aspecto debe ser cuidado al máximo.

### 5.5.6. Uso de gestos

El objetivo es que los personajes utilicen los mismos recursos de comunicación que las personas. Entre ellos se encuentran los gestos y las expresiones faciales.

Su uso sirve de gran ayuda porque existen ciertas informaciones que sólo se pueden expresar mediante gestos. Además los gestos sirven como complemento a acciones habladas o producen información redundante que ayuda en la mejora del entendimiento. Por lo tanto el uso y análisis de gestos constituyen un factor importante en el desarrollo de agentes animados.

Por ejemplo, los personajes pueden elevar las cejas para indicar que se ha entendido lo que el usuario ha pronunciado, jugar con la mirada mientras habla y utilizar diferentes recursos corporales para incrementar la sensación de naturalidad en la interacción.

Uno de los gestos más empleados por agentes animados es el de apuntar. Se puede combinar este gesto con el desplazamiento del personaje por la pantalla señalando la localización de cierto ítem. Como vemos en la siguiente figura, Genio señala hacia un botón para informar al usuario que debe pulsarlo.



Figura 5.2: Empleo de Gestos: Genio apuntando a un botón

### 5.5.7. Crear una personalidad

Dotar de personalidad a un personaje es importante para que transmita sensación de realismo.

Conviene que la personalidad de cada personaje esté bien definida y sea distintiva. No importa tanto que el personaje sea muy educado o simpático, como que su personalidad resulte atractiva y en general una personalidad débil o ambigua no gusta tanto.

La personalidad que debe ser elegida para un personaje dependerá del rol que haya de interpretar:

- Si es dirigir al usuario hacia metas específicas, conviene una personalidad dominante.

- Si el propósito del personaje es responder a las peticiones del usuario se debe usar una personalidad más sumisa.

Por otra parte, existen estudios que demuestran que los usuarios prefieren interactuar con personajes cuya personalidad se asemeja a la suya propia. Por lo tanto, otra solución es adaptar la personalidad a la del usuario. Para ello existen dos posibilidades, la primera es permitir que el usuario escoja entre una colección de personajes cada uno con una personalidad distinta, y la segunda es observar el estilo de interacción del usuario y modificar dinámicamente la personalidad del personaje. Aunque esto último es difícil, porque los humanos nos mostramos flexibles en las relaciones y solemos variar nuestra actitud dependiendo del interlocutor.

El programa Microsoft Office ofrece una galería de personajes para que el usuario elija el que mejor le parezca. En la siguiente figura podemos ver una selección a modo de ejemplo.



Figura 5.3: Galería de personajes del programa Microsoft Office

### ¿De qué herramientas se dispone para dotar de personalidad a un personaje?

La primera impresión que causa es importante. La postura, gestos, apariencia, elección de las palabras y estilo son rasgos sociales a través de los cuales se prejuzga una personalidad. El nombre de un personaje, como se presenta a sí mismo, como habla, como se mueve, y como responde a la acción del usuario contribuye a establecer una personalidad básica.

Pero, para crear una personalidad, no es necesario dotar al personaje de inteligencia artificial o crear animaciones de alta calidad. Los profesionales de la animación lo saben y llevan años usando los más sencillos rasgos de la comunicación social para dotar de ricas personalidades a objetos inanimados. Por ejemplo, los dibujantes de la Disney fueron capaces desde sus comienzos de dibujar simples sacos de harina que expresaban emociones.

Así pues, la estrategia a seguir es incidir sobre el aspecto psicológico. Si el personaje hace aseveraciones, se muestra seguro y desencadena acciones por su cuenta, habremos creado una personalidad dominante. Mientras que, una personalidad más sumisa puede caracterizarse mediante la articulación de frases interrogativas y por preferir la sugerencia a la orden. Además, las personalidades dominantes llevan el peso o la iniciativa en la interacción con el usuario.

El engreimiento suele ser mirado con escepticismo y antipatía por parte del usuario. Por lo tanto, se debe evitar la autoalabanza, a menos que sea una parte humorística de la personalidad que queremos que el personaje proyecte. Si hemos de resaltar la pericia del personaje, una buena idea es hacerlo de

manera indirecta desde otra fuente, como por ejemplo usando otro personaje o mediante una explicación descriptiva.

### 5.5.8. Aspecto físico

Como se ha comentado, el aspecto físico es importante porque contribuye a definir la personalidad del personaje.

Las formas de los personajes dependen del tipo de uso al que vayan a estar destinados en cada proyecto: Si se piensa que el uso de personajes con forma humanoide puede incrementar las expectativas de los usuarios más allá de lo que el agente puede ofrecer, es mejor usar personajes con forma de animal, objeto animado, etc. Por otra parte, si se usan personajes humanoides, se puede conseguir que los usuarios sitúen las expectativas al nivel que lo harían con los humanos y que por lo tanto disminuya su dificultad para interactuar con el ordenador, que siempre resulta un interlocutor extraño.

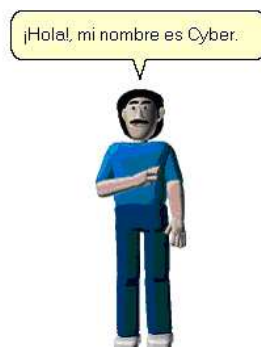


Figura 5.4: Cyber: personaje humanoide creado por Randy Casson

### 5.5.9. Observar el protocolo apropiado

Los humanos aprendemos normas sociales de protocolo, y las usamos para relacionarnos.

Nuestro personaje también ha de someterse a esas normas, en cuanto que, debemos simular una comunicación lo más humanizada posible. El usuario espera una reciprocidad en su interacción y si esto no ocurriera, podría juzgar la conducta del personaje como incompetente u ofensiva.

En situaciones formales o novedosas, la cortesía es esperada. Sólo después de haber alcanzado una cierta familiaridad en la relación se permite una relación no tan formal. Por lo tanto, ante la duda, la regla es ser cortés.

Por ejemplo, consideremos el protocolo apropiado para empezar y terminar una interacción social. Se puede mostrar la disponibilidad para entablar una conversación mirando al interlocutor potencial o aproximándose a la persona. El inicio de la misma suele corresponder con un intercambio verbal estereotipado. Para concluir la conversación se puede producir una frase de agradecimiento seguida de señales visuales.

Este modelo es de aplicación directa en una interfaz con personajes. Así, siempre debe evitarse que el personaje aparezca en pantalla o desaparezca sin la oportuna explicación. El personaje puede saludar

al iniciar una conversación y llegado el momento de despedirse, avisar de la intención de partir antes de hacerlo sin más. Las siguientes figuras nos muestran un ejemplo práctico. En la primera viñeta, podemos ver como Peedy da la bienvenida al usuario al comienzo, por ejemplo, de una visita guiada, y en las otras viñetas vemos como advierte al usuario de su partida y se despide.



Figura 5.5: Peedy dando la bienvenida

Durante una conversación, los integrantes de la misma se sitúan enfrente uno del otro mientras hablan. Darse la vuelta en ese momento, sin ningún motivo, indica desinterés y suele ser considerado como descortés.

La cortesía, o la ausencia de ella, se demuestra bien en la forma que tenemos de responder a una pregunta. En efecto, no basta sólo con encontrar la respuesta oportuna, sino que debemos cuidar la entonación, la expresión facial y las demás conductas no verbales para no resultar ofensivos.

Otra conducta que puede ser percibida como descortés es la falta de modestia. Por lo tanto, a menos que estemos tratando de implementar un estilo de personalidad descortés deberemos tratar de evitar que nuestro personaje presuma de sus prestaciones.

#### 5.5.10. Usar la alabanza

Debido a que los personajes crean un contexto social, hay que prestar un cuidado especial con el empleo de la crítica y la alabanza.

A la gente le suele agradar una actitud adulatoria. Pero, aunque los humanos respondemos bien ante las alabanzas (incluso siendo innecesarias), los límites de su administración en interfaces software no están bien definidos.

La alabanza es especialmente efectiva en situaciones donde los usuarios no confían demasiado en su preparación para la realización de una tarea. Pero, la mayoría de las interfaces software se esfuerzan en evitar evaluar a los usuarios, aunque está demostrado que la alabanza es más efectiva que la ausencia de evaluación.

¿Por qué es interesante usar la alabanza? Porque, tradicionalmente, muchas interfaces han sido diseñadas tan neutras que los usuarios las perciben como críticas: raramente reciben realimentación positiva cuando las cosas están operando normalmente, pero sí mensajes de error cuando van mal. Un ejemplo que puede ser visto en la siguiente figura, es el típico mensaje de error de aplicación de una interfaz Windows.

Por lo tanto, de vez en cuando resulta beneficioso lanzar mensajes de ánimo, como puede verse en la siguiente figura, donde el personaje resalta los buenos resultados obtenidos por el usuario.



Figura 5.6: Mensaje de alabanza de Peedy: realimentación positiva

En relación a la crítica, ésta debería ser usada con mucha moderación. Incluso aunque se crea que la crítica es oportuna, hay que tener cuidado, pues la condición humana del usuario es reacia a admitir los errores y las críticas suelen ser redirigidas hacia la fuente de donde parten.

#### 5.5.11. El personaje y el usuario como miembros del mismo equipo

Un factor importante a la hora de diseñar una IU con personajes es decidir qué tipo de relación usuario/personaje se quiere implementar. Como esta relación es de tipo social se debe estudiar las dinámicas sociales que pueden presentarse en la interacción.

Así, puede ser de utilidad presentar al personaje como miembro del mismo equipo que el usuario. Ya que, cuando un equipo es creado, la dinámica del grupo tiene un poderoso efecto sobre sus miembros. Para empezar, la gente en un grupo o equipo tiene una tendencia mayor a identificarse con sus compañeros que con gente de afuera. Y además, los miembros de un equipo tienen más voluntad para cooperar y modificar sus actitudes y conductas.

Crear un sentimiento de equipo se consigue mediante dos factores: identificación e interdependencia

Se puede crear identificación mediante la utilización de un nombre de equipo, color, símbolo o cualquier otro identificador que usuario y personaje compartan. Por ejemplo, dar la posibilidad al usuario de elegir un nombre de equipo o emplear un icono que podría aparecer con el personaje. La identificación con el personaje puede también conseguirse mediante comentarios del personaje. Por ejemplo, el personaje podría presentarse a sí mismo como un compañero del usuario y comentar que forman un equipo.

Lograr interdependencia puede resultar más duro y requerir más tiempo para ser implementado. Aún así, es importante considerarla pues la interdependencia parece tener un impacto social más fuerte que la identidad de equipo. Crear un sentimiento de interdependencia involucra el demostrar continuamente la utilidad y la fiabilidad del personaje para el usuario. Para ello, sería importante poder responder afirmativamente a dos preguntas en relación al personaje: ¿proporciona valor añadido al usuario? y, ¿opera de manera fiable y predecible?.

Para generar el sentimiento de equipo del que hablábamos anteriormente, es necesario presentar al per-

sonaje como un compañero del usuario, pero, en determinados escenarios, puede ser de mayor utilidad presentar al personaje como un sirviente o como un experto. La solución para alcanzar ambos objetivos (interdependencia y dinámica de grupo), puede ser un término medio: crear un sentimiento de igualdad entre usuario y personaje donde el usuario sea dependiente del personaje pero sin un sentido de inferioridad.

Esto podría ser tan simple como que el personaje se catalogara a sí mismo como un compañero de equipo o un colega, mejor que como un mago o experto. O utilizando expresiones apropiadas cuando se solicita información al usuario. Por ejemplo, el personaje podría decir "Trabajemos juntos para resolver esta cuestión...".

#### 5.5.12. Considerar efectos de género

Las respuestas sociales se ven afectadas por el género. Así, existe un estereotipo que considera que los hombres tienen una mayor facilidad o inclinación para tratar con temas técnicos mientras que las mujeres tienen una mayor habilidad para las relaciones interpersonales. El anterior ejemplo no va encaminado a fomentar o perpetuar los estereotipos, sólo a advertir de su existencia, para así poder evaluar su efecto en la interacción con el personaje.

Las connotaciones de género de un personaje no sólo son propiciadas por su nombre o su apariencia. Incluso un personaje de género neutro, como por ejemplo Clipo (el ayudante de Microsoft Office al que podemos ver en la figura de abajo), puede percibirse como masculino o femenino dependiendo de las características de su voz o sus movimientos.

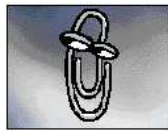


Figura 5.7: Clipo, el ayudante de Office no tiene un género definido

## 5.6. Sonido como Interfaz de Usuario

Uno de los principales atractivos de las aplicaciones que incluyen personajes es la posibilidad de que estos hablen. Este atractivo es aún mayor si se consigue centrar la capacidad de interactuar con el usuario en el empleo del sonido.

Una interfaz basada en Agentes Animados se puede catalogar dentro del apartado de las Interfaces habladas de usuario o SUI (Spoken User Interfaces), que consisten en interfaces de usuario con las que se interactúa mediante comandos fijos de voz o mediante diálogos y de las que se obtiene una contrapartida también de forma audible.

Este tipo de interfaces fueron concebidas como sustituto de las Interfaces gráficas de usuario (GUI), pero poco a poco se fueron adaptando a situaciones específicas donde ofrecían ventajas concretas sobre los entornos visuales. Estas interfaces son de gran utilidad en entornos telefónicos, sistemas de navegación por Internet, sistemas en los que las manos y los ojos están ocupados (por ejemplo asistentes para la conducción) o como en el caso de los Agentes Animados para la creación de entornos con interacción de modo natural.

### 5.6.1. Reconocimiento del habla

Las interfaces de usuario controladas mediante la voz necesitan obtener una representación en texto de las palabras pronunciadas por el usuario, esto es, necesitan incluir reconocimiento del habla.

Microsoft Agent facilita la posibilidad de trabajar con un motor de reconocimiento de voz pues proporciona la interfaz de programación para controlarlo.

Sin embargo, la entrada de habla presenta muchas dificultades. Los motores de habla actuales operan sin partes sustanciales del repertorio de la comunicación hablada humana, tales como gestos, entonación, y expresiones faciales. Como el habla natural es por lo general ilimitada, es fácil para el hablante exceder el vocabulario real, o la gramática, del motor de reconocimiento.

Se puede tratar de restringir las posibilidades de error si el sistema se centra en reconocer comandos de voz, como el motor de reconocimiento que Microsoft Agent soporta.

Aún así no se estará libre de equivocaciones, porque el reconocimiento de habla debe a menudo tratar con grandes variaciones en el entorno del hablante. Por ejemplo, el ruido de fondo, la calidad del micrófono o su localización pueden afectar a la calidad de la entrada. Igualmente, las diferentes pronunciaciones de los hablantes o incluso variaciones en el mismo hablante, como por ejemplo cuando está resfriado, hacen que sea difícil convertir los datos acústicos en representaciones comprensibles. Finalmente, las máquinas de habla deben tratar con palabras o frases de similar sonido dentro de un lenguaje.

El éxito en el uso de la entrada de voz no sólo es debido a la calidad de la tecnología. Incluso el reconocimiento humano, el cual excede cualquier tecnología de reconocimiento, falla a algunas veces. Para tratar de paliarlo, en la comunicación humana se usan estrategias que mejoran la probabilidad de éxito y que proporcionan recuperación frente a errores cuando algo va mal. El ejemplo de la comunicación humana llevado al ámbito del reconocimiento de voz por sistemas artificiales, demuestra que la efectividad de la entrada de voz también depende de la calidad de la interfaz de usuario que la presenta.

Así, para conseguir que la interacción resulte lo más natural y adecuada posible la interfaz ha de manejar correctamente los errores producidos por los reconocedores del habla:

- Los errores de borrado (cuando el reconocedor no puede determinar qué se ha pronunciado) se pueden solventar reconduciendo al usuario en la conversación de modo que repita la frase con otras palabras o intente hablar de forma más clara para el reconocedor. "Los errores de sustitución (en los que se reconoce una expresión distinta a la pronunciada) pueden producir fallos inesperados en el comportamiento del sistema. Este tipo de errores hace que pueda resultar necesario asegurarse de que se ha comprendido correctamente lo dicho por el usuario. Como la continua comprobación de este tipo de errores supone una considerable carga dentro de la conversación, lo mejor es que se realice solamente en momentos críticos o en los casos en los que se piense que la entrada no se corresponde con lo esperado. Los Agentes Microsoft facilitan al desarrollador la mejor y las dos siguientes mejores alternativas devueltas por la máquina de reconocimiento de



habla. Además, indican el porcentaje de confianza de todas las posibilidades. Esta información puede usarse para tratar de inferir qué es lo que se ha dicho. Por ejemplo, si las puntuaciones de confianza de la mejor y de la primera alternativa son cercanas, esto indicaría que el motor de habla tiene dificultad en discernir entre una de ellas. En tal caso, se puede pedir al usuario que repita para intentar mejorar el reconocimiento. Sin embargo, si la mejor y la siguiente alternativa devuelven el mismo comando, eso reforzaría la indicación de un reconocimiento correcto.

- Los errores de inserción (cuando el ruido produce una frase) se pueden tratar con algunos de los métodos anteriores, o incluso desconectando el micrófono o el reconocedor en determinados momentos de la conversación. Muchos sistemas optan por una interfaz click-to-speak (en los que el micrófono o el reconocedor no están operativos hasta que el usuario los activa explícitamente) en contraposición con una interfaz de estilo open-microphone (donde el micrófono y el reconocedor funcionan continuamente). Con el método click-to-speak se consigue descartar hasta un 12,4 El motor de reconocimiento de Microsoft Agent es de tipo click-to-speak.

## 5.7. Adaptación a las Interfaces de usuario con habla

El vocabulario usado en sistemas con interfaces de usuario tradicionales (por ejemplo GUIs) no se puede transferir correctamente a las Interfaces de usuario con habla(SUIs). Para estos nuevos casos resulta más conveniente utilizar frases que cumplan las convenciones establecidas dentro de las conversaciones. Sin embargo, para conseguir una alta capacidad de reconocimiento el sistema debe intentar conducir al usuario a respuestas que se encuentren dentro de sus capacidades. Esto implica que se deban utilizar oraciones para el sistema concreto que se está desarrollando.

Los escenarios de entrada de voz a través de personajes que definen claramente el contexto tienen más éxito. El contexto ayuda a clarificar y dirigir las elecciones del usuario (que aprende el rango de la gramática en cada momento) y favorece el reconocimiento de habla al limitarse la gramática activa en cada contexto.

Microsoft Agent incluye construcciones que incrementan el éxito de la entrada de voz mediante la clarificación de la gramática el sistema. Por ejemplo, la ventana de comandos mostrada cuando el usuario dice Abrir Ventana de Comandos de Voz sirve como guía visual de la gramática activa de la máquina de habla.

Puede permitirse que el usuario solicite el contexto mediante un comando, como podría ser Ayuda o ¿Dónde estoy?, a lo cual la interfaz debería responder clarificando el contexto actual, como, por ejemplo, la última acción que la aplicación llevó a cabo. El modo de organización y presentación de la información puede presentar complicaciones dentro de un entorno conversacional. El flujo de información en una SUI puede resultar confuso para el usuario. Si el diálogo implica subdiálogos puede ser difícil determinar en qué punto se encuentra la conversación, si ya se ha retornado a un estado anterior o si las respuestas corresponden al diálogo en que se suponía estar.

Dado que el habla es un método no persistente de transmisión de la información (lo que puede provocar una carencia de retención por parte del usuario) las oraciones que transmite el sistema han de ser breves, concisas e informativas. Se ha de evitar utilizar palabras que no sean necesarias o que resulten extrañas o repetitivas. Aplicando el principio de que el sistema debe proporcionar la menor cantidad de información posible, aunque nunca menos de la necesaria.

Microsoft Agent trata de paliar el efecto de la no persistencia mediante la duplicación de la información hablada en el globo de texto, pero éste también es un modo no persistente.

## 5.8. Simular por parte del sistema una conversación natural

Uno de los principales desafíos de este tipo de aplicaciones consiste en poder simular el papel de hablante o de oyente de forma suficientemente convincente como para producir una comunicación satisfactoria con el usuario.

En ese sentido, la comunicación es algo más que el reconocimiento de palabras. El proceso de diálogo implica mostrar señales que indiquen a quién corresponde el turno de palabra o que se ha comprendido el mensaje del interlocutor. Los personajes pueden mejorar las interfaces conversacionales mediante señales como inclinaciones de cabeza, asentimientos, o movimientos que indican que el motor de habla está en estado de escucha y que algo está siendo reconocido. Por ejemplo, Microsoft Agent tiene animaciones ligadas al estado de Escuchando cuando el usuario presiona la tecla que indica que quiere decirle algo al personaje.

Otro elemento importante en toda conversación es la entonación. Actualmente se está trabajando en conseguir herramientas de síntesis con unos niveles de entonación similares a los humanos, en los que la voz no suene metálica y sea lo más natural posible.

Las pausas que se realizan en las conversaciones constituyen otro de los factores que se deben tener en cuenta dentro de los diálogos. Se ha de procurar que las pausas generadas por los retrasos en el proceso de reconocimiento sean lo suficientemente cortas para que sean percibidas como naturales. Los silencios se han de evitar siempre que su presencia no corresponda a la situación esperada dentro de una conversación.

Por último, no todos los sistemas permiten interrumpir al sintetizador mientras está pronunciando una oración, lo que puede repercutir en una falta de naturalidad en el diálogo. Microsoft Agent interrumpe la salida de audio cuando detecta entrada de voz por el micrófono, aunque presenta el globo de texto. El sintetizador también debería poder aumentar o disminuir la velocidad de su discurso para simular, en mayor medida, el habla humana. Agent puede ajustar programáticamente la velocidad o el pitch de la voz del sintetizador.

## **Parte III**

# **Desarrollo de la aplicación**



## Capítulo 6

# Análisis del sistema. Definición del problema.

El análisis se dedica a la comprensión y modelado de la aplicación y del dominio en el cual funciona. La entrada inicial de la fase de análisis es una descripción del problema que hay que resolver y proporciona una visión general conceptual del sistema propuesto. Otras entradas adicionales del análisis son un diálogo con el cliente y un conocimiento de fondo del mundo real. La salida del análisis es un modelo formal que captura los tres aspectos esenciales del sistema: los objetos y sus relaciones, el flujo dinámico de control y la transformación funcional de datos que están sometidos a restricciones.

### 6.1. Consideraciones iniciales

El objetivo de esta aplicación es que los niños se acercan al mundo multimedia de una manera amena y educativa.

Pretende ser una herramienta intuitiva y sencilla en la que conceptos como hipervínculo o puntero de ratón, queden fijados para el usuario de un modo inconsciente.

De todo esto se infiere que la aplicación ha de ser vistosa, con colores e imágenes, y con una interfaz poco complicada.

### 6.2. Metodología empleada

El método a seguir deberá ser apropiado para la orientación a objetos, se partirá de la realidad y se irá construyendo una serie de modelos cada vez más detallados hasta llegar a un modelo implementable o solución. Para ello se usa el Lenguaje de modelado unificado (UML).

Se intentará que el método de trabajo sea continuo e incremental, procurando eliminar fronteras entre las diversas fases del desarrollo, realizando modificaciones progresivas si es preciso hasta llegar a la implementación final.

El método además será una evolución natural en la que determinadas ideas pueden ser reconsideradas lo que implica modificar o eliminar elementos a medida que se avanza en el desarrollo, siguiendo un proceso iterativo.

Se buscará realizar un esfuerzo en la calidad de los componentes software de la solución final (clases).

De igual forma se intentará que las clases tengan independencia, apostando así por la reutilización y facilidad de mantenimiento.

Este proceso así descrito corresponde con 'El proceso Unificado', metodología de desarrollo software dirigida por casos de uso. Se centra en la funcionalidad del sistema orientada a satisfacer las necesidades del usuario final que interactúa con la aplicación.

Para la implementación se usará el lenguaje C#, utilizando Visual C# .NET apropiado para programación orientada a objetos.

### 6.3. Resultados de las entrevistas.

La captura de requisitos no es tan sencilla en este caso, debido a las dificultades que tienen a la hora de comunicarse las personas a las que va dedicado este proyecto, por eso, nos basamos en otras experiencias, como pueden ser las personas más cercanas (familiares, tutores, profesores ...)

La fuente principal de información han sido los estudiantes de la facultad de Magisterio. De sus entrevistas llegue a la conclusión de que la interfaz tendría que ser sencilla y que el uso de un motor de voz sin más resultaría algo confuso. Es decir, los niños tendrían que ver quien está hablando. Eso fue lo que me dio el salto para utilizar Ms Agent.

Otras entrevistas fueron realizadas con monitores de ocio y tiempo libre que están más familiarizados con el aspecto de educación no formal y de juegos. De estos encuentros saqué las ideas para los juegos específicos.

### 6.4. Definición de actores

Los actores son personas o entidades externas a la aplicación que interactúan con ella, realizando un intercambio de información. Éste podrá ser tanto de entrada como de salida. En sistemas más complejos, un actor puede representar varios papeles. En estos casos, se definen distintos actores para representarlos.

<b>ACT-001</b>	<b>Alumno</b>
<b>Descripción</b>	Este actor representa al niño que utilizara la aplicación

Cuadro 6.1: Descripción del actor Alumno de los casos de uso.

<b>ACT-002</b>	<b>Profesor</b>
<b>Descripción</b>	Este actor representa a la persona que realiza tareas que no pueden ser realizadas por el niño

Cuadro 6.2: Descripción del actor Profesor de los casos de uso.

## 6.5. Diagramas de caso de uso

Se emplean para visualizar el comportamiento del sistema, una parte de él o de una sola clase. De forma que se pueda conocer como responde esa parte del sistema. El diagrama de casos de uso es muy útil para definir como debería ser el comportamiento de una parte del sistema, ya que solo especifica como deben comportarse y no como deben estar implementadas las partes que define. Un caso de uso especifica un requerimiento funcional, es decir esta parte debe hacer esto cuando pase esto.

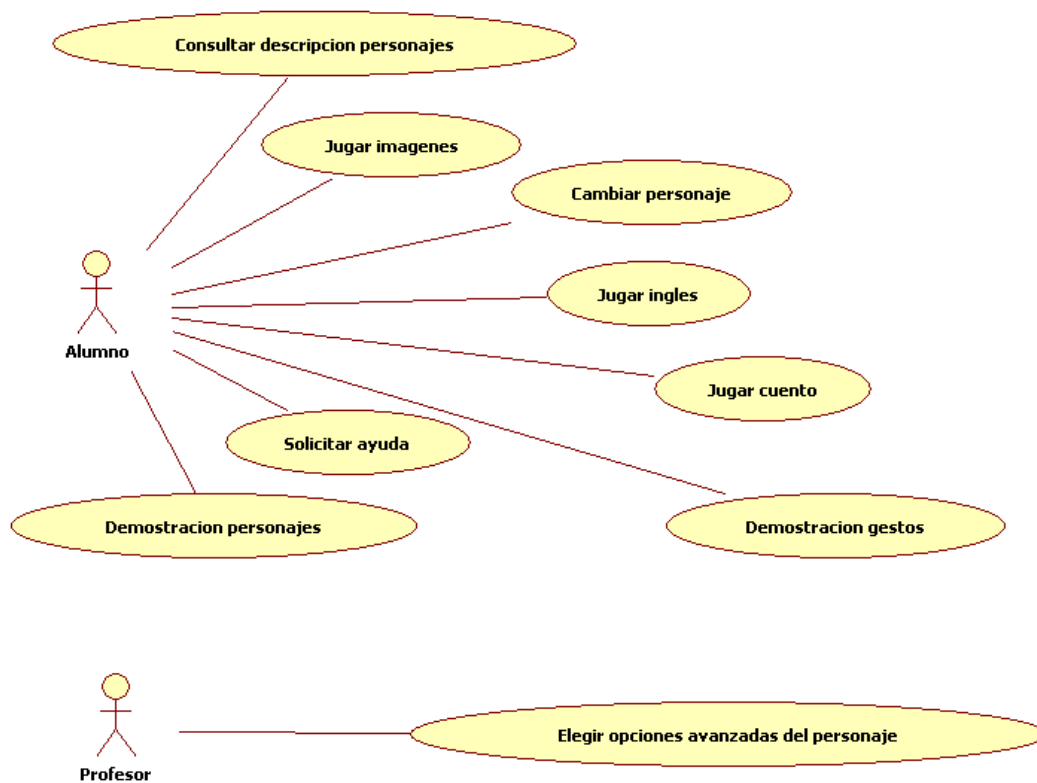


Figura 6.1: Diagrama de los casos de uso

## 6.6. Modelo de clases

El modelo de objetos muestra la estructura estática de datos correspondientes al sistema del mundo real, y la organiza en segmentos manejables describiendo clases de objetos del mundo real, y sus relaciones entre sí. Lo más importante es la organización de más alto nivel del sistema, en clases conectadas mediante asociaciones.

### 6.6.1. Diagrama inicial de clases

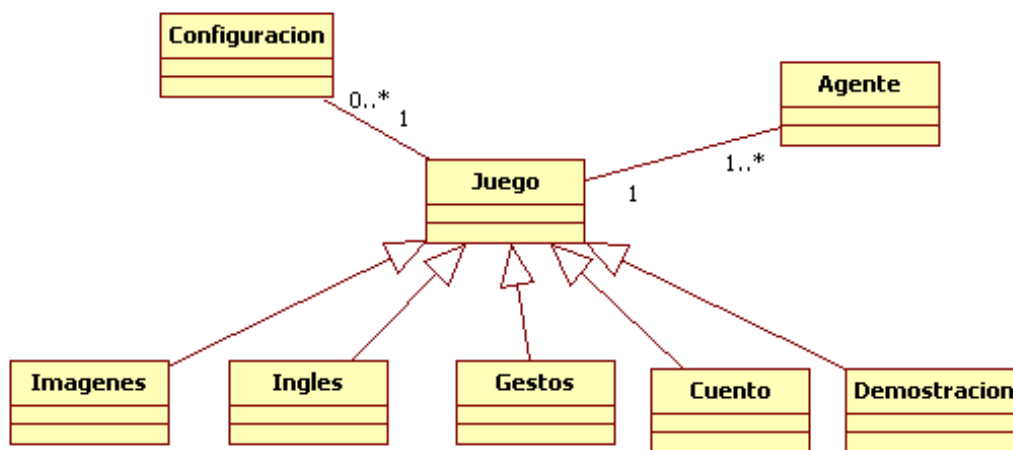


Figura 6.2: Diagrama inicial de clases

## 6.7. Descripción de los casos de uso

El comportamiento de un caso de uso se puede especificar describiendo un flujo de eventos de forma textual, lo suficientemente claro para que una persona ajena al sistema lo entienda fácilmente. A la hora de escribir este flujo de eventos, se debe incluir cómo y cuándo empieza y termina el caso de uso en cuestión, cuándo interactúa con los actores y qué objetos intercambian, el flujo básico y los alternativos de comportamiento.

A medida que se obtiene una mejora en la comprensión de los requisitos del sistema, estos flujos de eventos se especifican gráficamente mediante los diagramas de interacción. Normalmente se utiliza un diagrama de secuencia para especificar el flujo principal de un caso de uso.



### 6.7.1. Jugar imágenes

Caso de uso 1	Jugar imágenes	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) elige el juego de las imágenes
	2	El sistema carga al personaje por defecto
	3	El sistema muestra los diferentes grupos de imágenes
	4	El actor Alumno (ACT-0001) elige un grupo
	5	El actor Alumno (ACT-001) introduce una palabra
	6	El sistema actua segun la conveniencia de la palabra

Cuadro 6.3: Descripción del caso de uso Jugar imágenes

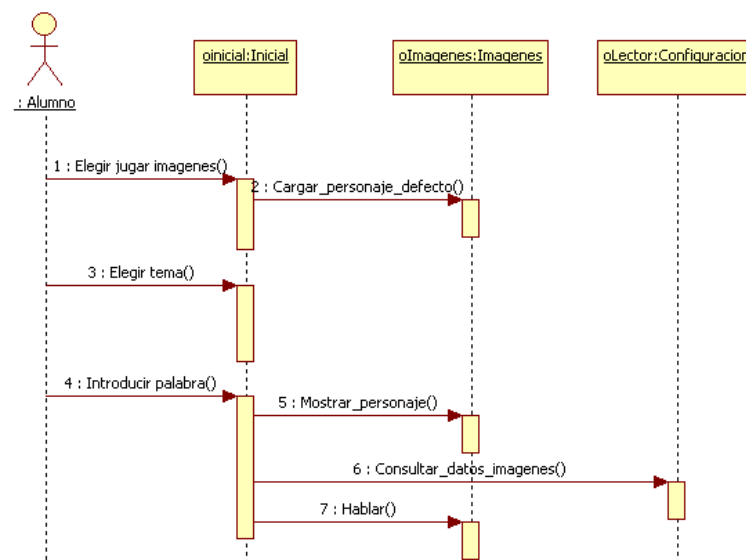


Figura 6.3: Diagrama del caso de uso Jugar imágenes

6.7.2. Jugar inglés

Caso de uso 2	Jugar inglés	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) elige el juego de inglés
	2	El sistema carga el personaje por defecto
	3	El actor Alumno (ACT-0001) elige jugar
	4	El sistema muestra la imagen y las letras descolocadas
	5	El actor Alumno (ACT-001) introduce una palabra
	6	El sistema actua segun la conveniencia de la palabra

Cuadro 6.4: Descripcion del caso de uso Jugar ingles

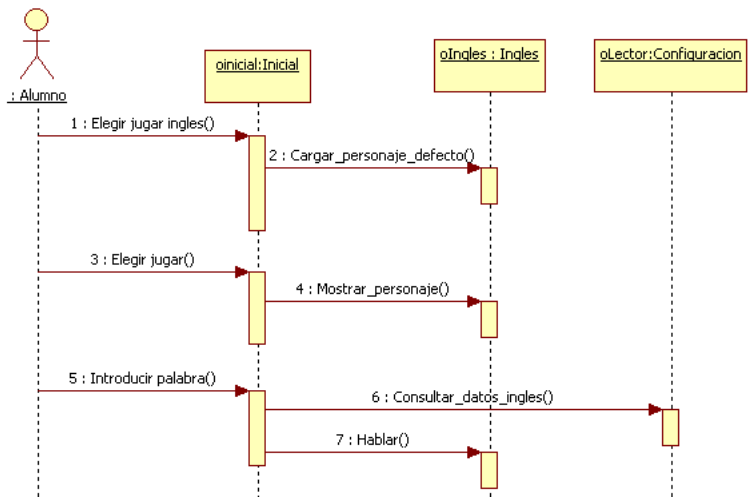


Figura 6.4: Diagrama del caso de uso Jugar inglés

### 6.7.3. Jugar cuento

Caso de uso 3	Jugar cuento	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) elige el juego del cuento
	2	El sistema carga al personaje por defecto
	3	El sistema muestra una lista con los cuentos disponibles
	4	El actor Alumno (ACT-0001) elige un cuento de la lista
	5	El sistema reproduce por medio del personaje el cuento correspondiente

Cuadro 6.5: Descripción del caso de uso Jugar cuento

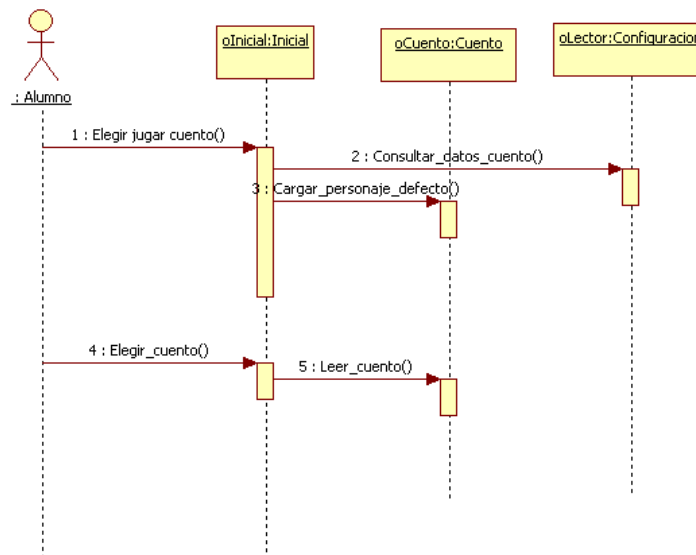


Figura 6.5: Diagrama del caso de uso Jugar cuento

### 6.7.4. Consultar descripción de los personajes

Caso de uso 4	Consultar descripción de personajes	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) solicita la descripción de los personajes
	2	El sistema muestra el menu de información

Cuadro 6.6: Descripción del caso de uso Consultar descripción de los personajes

### 6.7.5. Demostración de gestos

Caso de uso 5	Demostración de gestos	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) solicita la demostración de gestos
	2	El sistema carga el personaje por defecto
	3	El sistema muestra los posibles gestos
	4	El actor Alumno (ACT-0001) elige el gesto que desea ver
	5	El sistema muestra el gesto por medio del personaje elegido

Cuadro 6.7: Descripción del caso de uso Demostración de gestos

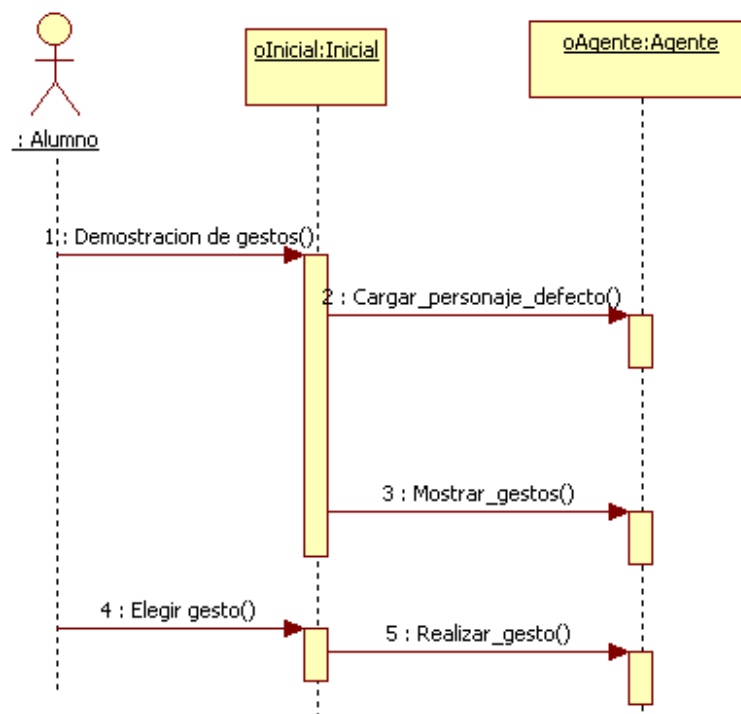


Figura 6.6: Diagrama del caso de uso Demostración de gestos

### 6.7.6. Demostración de personajes

Caso de uso 6	Demostración de personajes	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) elige demostración de personajes
	2	El sistema muestra la demostración

Cuadro 6.8: Descripción del caso de uso Demostración de personajes

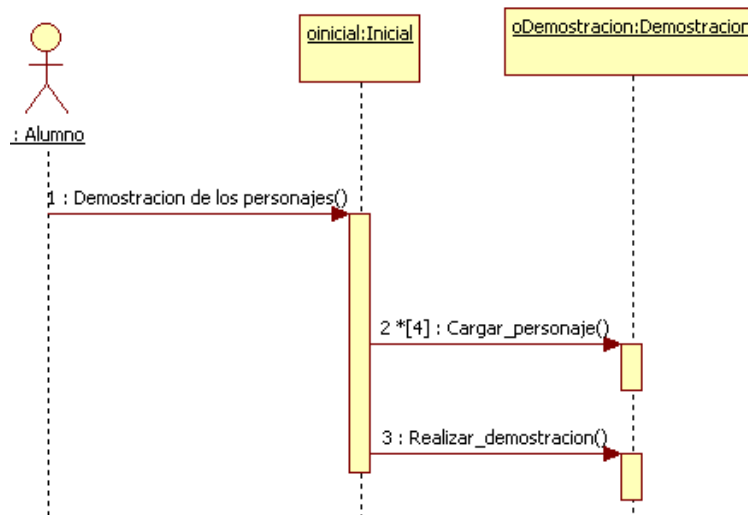


Figura 6.7: Diagrama del caso de uso Demostración de personajes

### 6.7.7. Solicitar ayuda

Caso de uso 7	Solicitar ayuda	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) solicita ayuda al sistema
	2	El sistema le ofrece la ayuda referente a todos los temas de los que dispone
	3	El actor Alumno (ACT-0001) selecciona de entre todos los temas uno
	4	El sistema le proporciona la ayuda referente a ese tema en concreto

Cuadro 6.9: Descripción del caso de uso Solicitar ayuda

### 6.7.8. Cambiar personaje

Caso de uso 8	Cambiar personaje	
Secuencia normal	Paso	Acción
	1	El sistema muestra un fichero de dialogo con los personajes disponibles
	2	El actor Alumno (ACT-0001) elige el personaje
	3	El sistema carga el personaje

Cuadro 6.10: Descripción del caso de uso Cambiar personaje

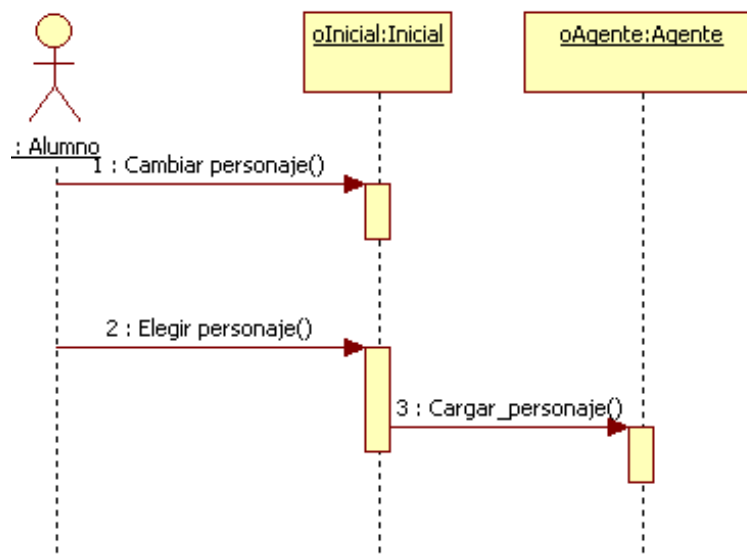


Figura 6.8: Diagrama del caso de uso Cambiar personaje

### 6.7.9. Elegir opciones avanzadas del sistema

Caso de uso 9	Elegir opciones avanzadas del sistema	
Secuencia normal	Paso	Acción
	1	El actor Profesor (ACT-0002) elige las opciones avanzadas del personaje
	2	El sistema muestra un menu con las posibilidades
	3	El actor Profesor (ACT-0002) elige los cambios que quiere
	4	El sistema actualiza los cambios

Cuadro 6.11: Descripción del caso de uso Elegir opciones avanzadas del sistema

## Capítulo 7

# Diseño

Una vez completada la fase de análisis, y antes de pasar a la implementación y programación, es necesaria una actividad de nivel superior.

La fase de diseño es una fase intermedia entre la vista abstracta de un sistema y la implementación real de éste. Los productos de esta fase pueden ser más cercanos a una visión abstracta o a una visión implementable.

En esta parte del diseño se desarrollan los casos de uso que habían sido descritos anteriormente. También serán desarrollados los diagramas de secuencia mostrando el orden de las llamadas en el sistema. Se utilizará un diagrama de secuencia para cada caso de uso a representar y para las excepciones que puedan presentar estos casos. Es imposible representar en un solo diagrama de secuencia todas las secuencias posibles del sistema, por ello se escoge un punto de partida.

El diagrama se realiza con los objetos que forman parte de la secuencia, estos se sitúan en la parte superior. De estos objetos sale una línea que indica su vida en el sistema. Esta línea simple se convierte en una línea gruesa cuando representa que el objeto tiene el foco del sistema, es decir cuando está activo.

Los diagramas de secuencia mostrarán los mensajes que los objetos participantes intercambian entre ellos a lo largo del tiempo.

## 7.1. Descripción de los Casos de uso

### 7.1.1. Solicitar ayuda

Caso de uso 1	Solicitar ayuda	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) selecciona con el ratón la opción Solicitar ayuda
	2	El sistema despliega todos los temas referentes a la aplicación que disponen de ayuda
	3	El actor Alumno (ACT-0001) elige que opción desea consultar
	4	El sistema muestra la información referente a ese tema

Cuadro 7.1: Descripción del caso de uso Solicitar ayuda

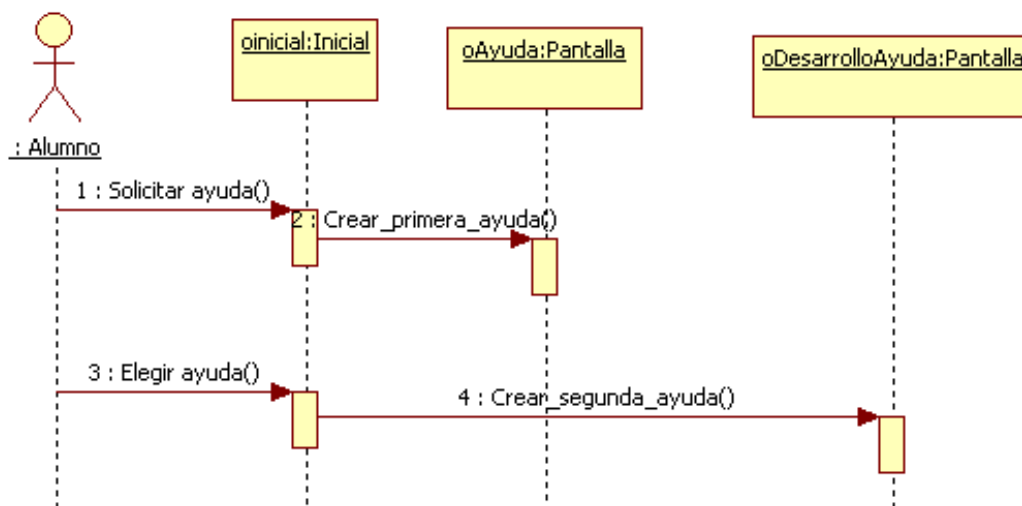


Figura 7.1: Diagrama del caso de uso Solicitar ayuda



## 7.1.2. Cambiar personaje

Caso de uso 2	Cambiar personaje	
Secuencia normal	Paso	Acción
	1	El sistema muestra un fichero de dialogo con los personajes disponibles
	2	El actor Alumno (ACT-0001) elige el personaje
	3	El sistema carga un personaje con las características que estén en el fichero de configuración

Cuadro 7.2: Descripción del caso de uso Cambiar personaje

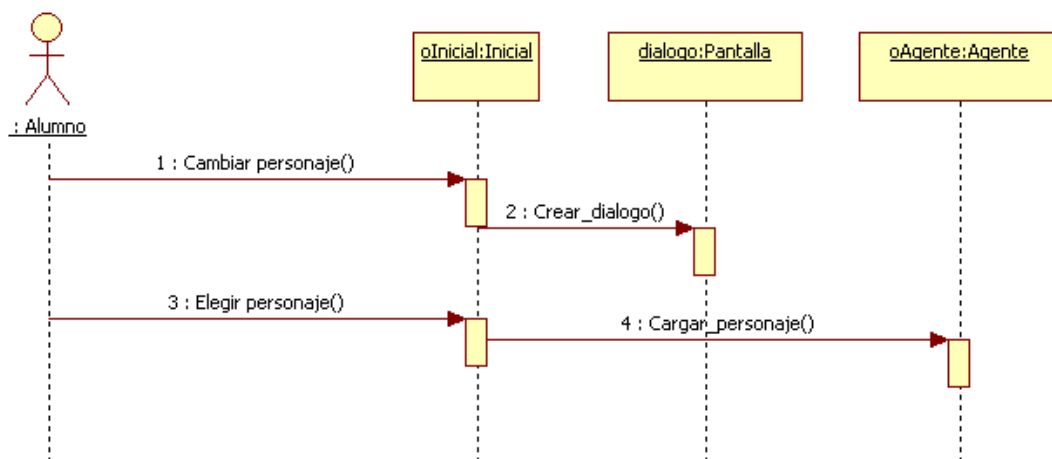


Figura 7.2: Diagrama del caso de uso Cambiar personaje

### 7.1.3. Jugar imágenes

Caso de uso 4	Jugar imágenes	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) elige el juego de las imágenes
	2	El sistema carga al personaje por defecto
	3	El sistema muestra los diferentes grupos de imágenes
	4	El actor Alumno (ACT-0001) elige un grupo
	5	El sistema muestra las imágenes con sus respectivas cajas de texto
	6	El actor Alumno (ACT-0001) introduce el texto correspondiente a la imagen
	7	El sistema a través del agente lee el texto según su aceptación

Cuadro 7.3: Descripción del caso de uso Jugar imágenes

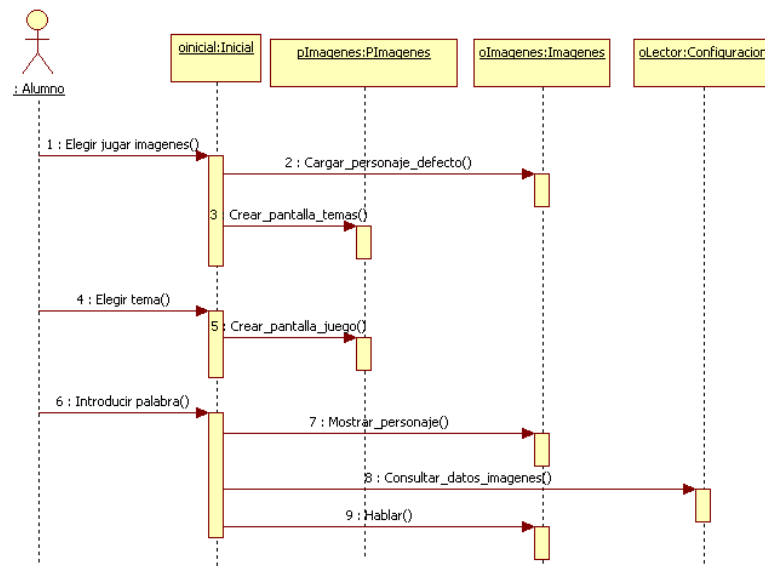


Figura 7.3: Diagrama del caso de uso Jugar imágenes

7.1.4. Jugar inglés

Caso de uso 5	Jugar inglés	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) elige el juego de inglés
	2	El sistema carga el personaje por defecto y lo muestra
	3	El actor Alumno (ACT-0001) elige jugar
	4	El sistema muestra la imagen y las letras descolocadas
	5	El actor Alumno (ACT-0001) introduce el texto correspondiente a la imagen
	6	El sistema a través del agente lee el texto según su aceptación

Cuadro 7.4: Descripción del caso de uso Jugar inglés

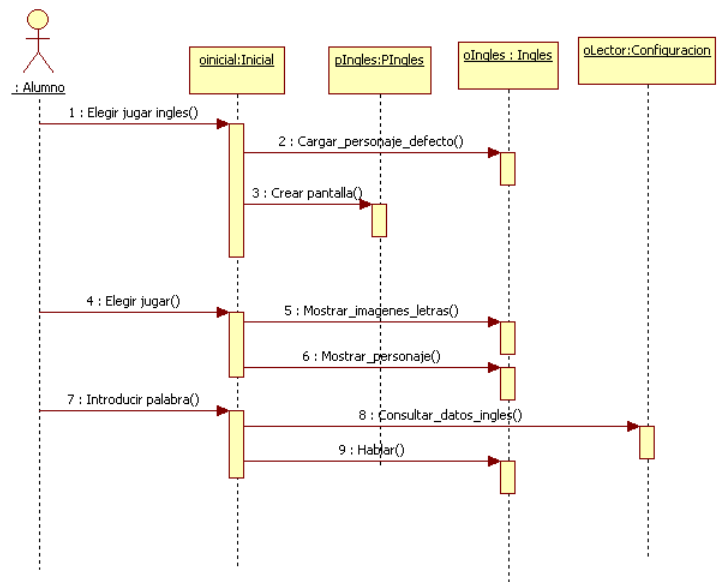


Figura 7.4: Diagrama del caso de uso Jugar inglés

### 7.1.5. Jugar cuento

Caso de uso 6	Jugar cuento	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) elige el juego del cuento
	2	El sistema carga al personaje por defecto
	3	El sistema muestra una lista con los cuentos disponibles
	4	El actor Alumno (ACT-0001) elige un cuento de la lista
	5	El sistema muestra una imagen correspondiente con el cuento elegido
	6	El sistema reproduce por medio del personaje el cuento correspondiente

Cuadro 7.5: Descripción del caso de uso Jugar cuento

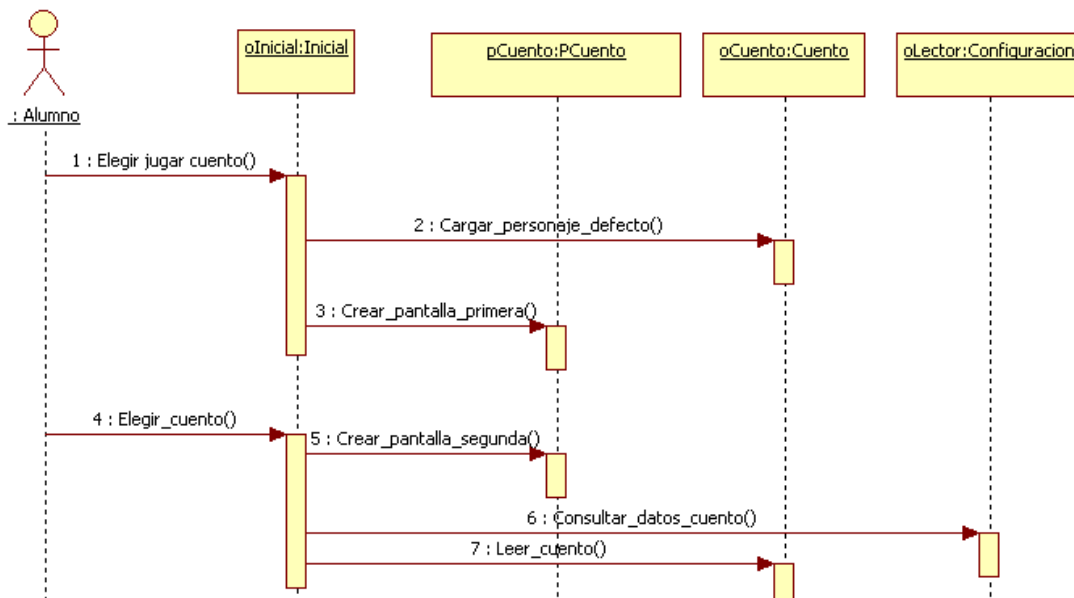


Figura 7.5: Diagrama del caso de uso Jugar cuento

**7.1.6. Consultar descripción de los personajes**

Caso de uso 7	Consultar descripción de personajes	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) solicita la descripción de los personajes
	2	El sistema muestra el menu de información

Cuadro 7.6: Descripción del caso de uso Consultar descripción de los personajes

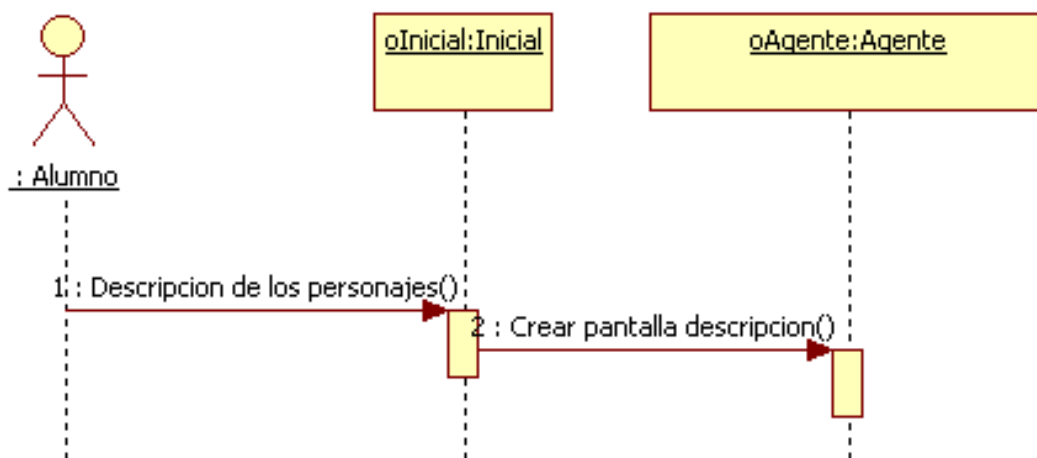


Figura 7.6: Diagrama del caso de uso Consultar descripción de los personajes

### 7.1.7. Demostración de gestos

Caso de uso 8	Demostración de gestos	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) solicita la demostración de gestos
	2	El sistema carga el personaje por defecto
	3	El sistema muestra los posibles gestos
	4	El actor Alumno (ACT-0001) elige el gesto que desea ver
	5	El sistema muestra el gesto por medio del personaje elegido

Cuadro 7.7: Descripción del caso de uso Demostración de gestos

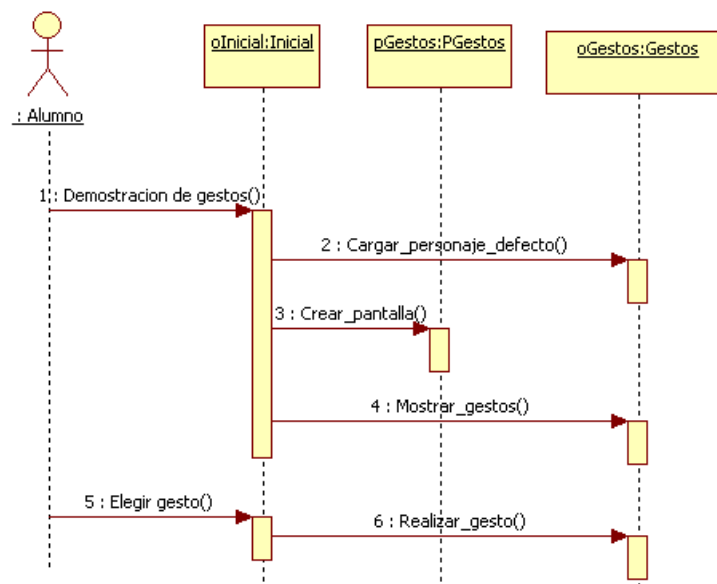


Figura 7.7: Diagrama del caso de uso Demostración de gestos

### 7.1.8. Demostración de personajes

Caso de uso 9	Demostración de personajes	
Secuencia normal	Paso	Acción
	1	El actor Alumno (ACT-0001) elige demostración de personajes
	2	El sistema muestra los personajes
	3	El sistema a través de los personajes reproducen unos textos y hacen unos gestos

Cuadro 7.8: Descripción del caso de uso Demostración de personajes

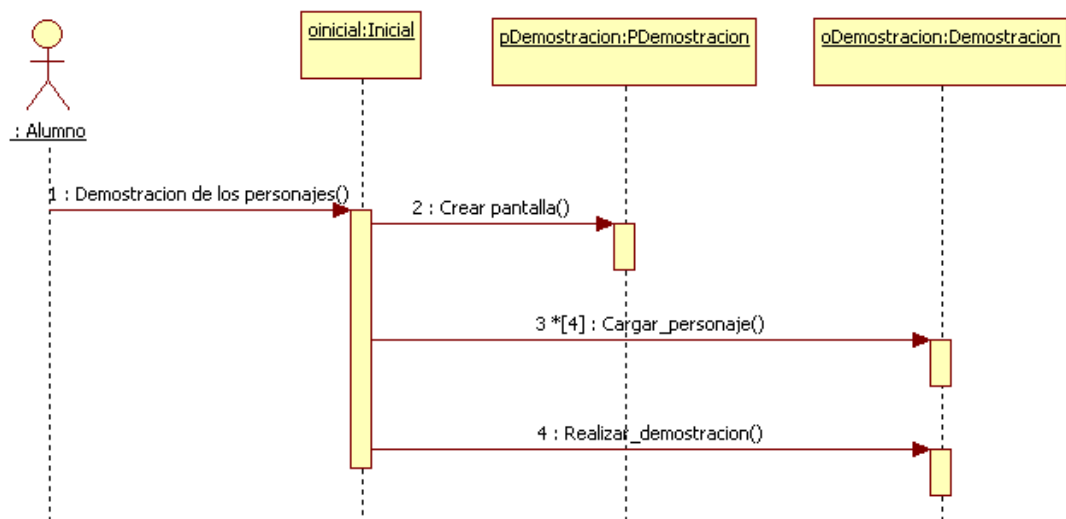


Figura 7.8: Diagrama del caso de uso Demostración de personajes

### 7.1.9. Elegir opciones avanzadas del sistema

Caso de uso 11	Elegir opciones avanzadas del sistema	
Secuencia normal	Paso	Acción
	1	El actor Profesor (ACT-0002) elige las opciones avanzadas del personaje
	2	El sistema muestra un menu con las posibilidades
	3	El actor Profesor (ACT-0002) elige los cambios que quiera
	4	El sistema actualiza los cambios

Cuadro 7.9: Descripción del caso de uso Elegir opciones avanzadas del sistema

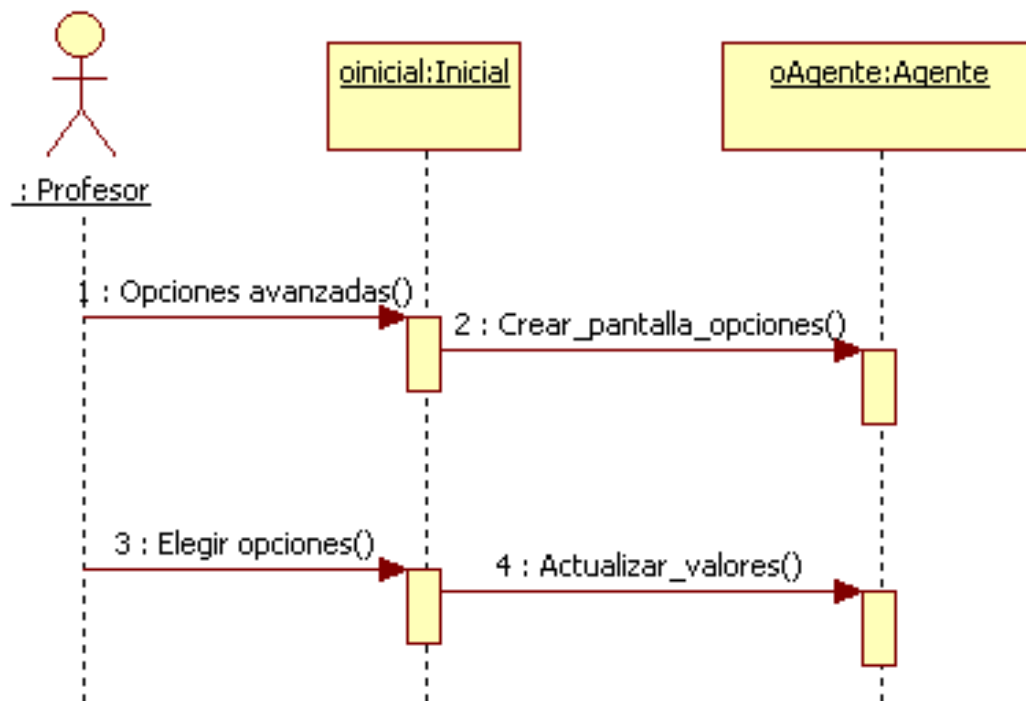


Figura 7.9: Diagrama del caso de uso Elegir opciones avanzadas del sistema

## 7.2. Diagrama de clases final

El primer paso en la elaboración de un modelo de objetos es encontrar las clases necesarias para la aplicación. Las clases suelen corresponderse con sustantivos o entidades físicas. Aunque hay que evitar estructuras propias de la computadora.

A continuación se describen las clases encontradas con una breve descripción:

- **Imágenes:** es la clase encargada del funcionamiento del juego Imágenes
- **PImágenes:** es la clase encargada de crear los controles de las pantallas del juego Imágenes
- **Cuento:** es la clase encargada del funcionamiento del juego Cuento
- **PCuento:** es la clase encargada de crear los controles de las pantallas del juego Cuento
- **Inglés:** es la clase encargada del funcionamiento del juego Inglés
- **PInglés:** es la clase encargada de crear los controles de las pantallas del juego Inglés
- **Gestos:** es la clase encargada del funcionamiento de la demostración de gestos
- **PGestos:** es la clase encargada de crear los controles de la pantalla de Gestos
- **Demostración:** es la clase encargada del funcionamiento de la demostración de la presentación
- **PDemostracion:** es la clase encargada de crear los controles de la pantalla de la presentación



- **Juego**: es la clase abstracta de todos los posibles juegos de la aplicación
- **Configuración**: es la clase que almacena textos y permite su lectura
- **Agente**: ofrece las mismas funcionalidades que la clase IAgentCtlCharacter del MsAgent
- **Pantalla**: de esta clase derivan todas las demás pantallas e incluye métodos para pantallas especiales

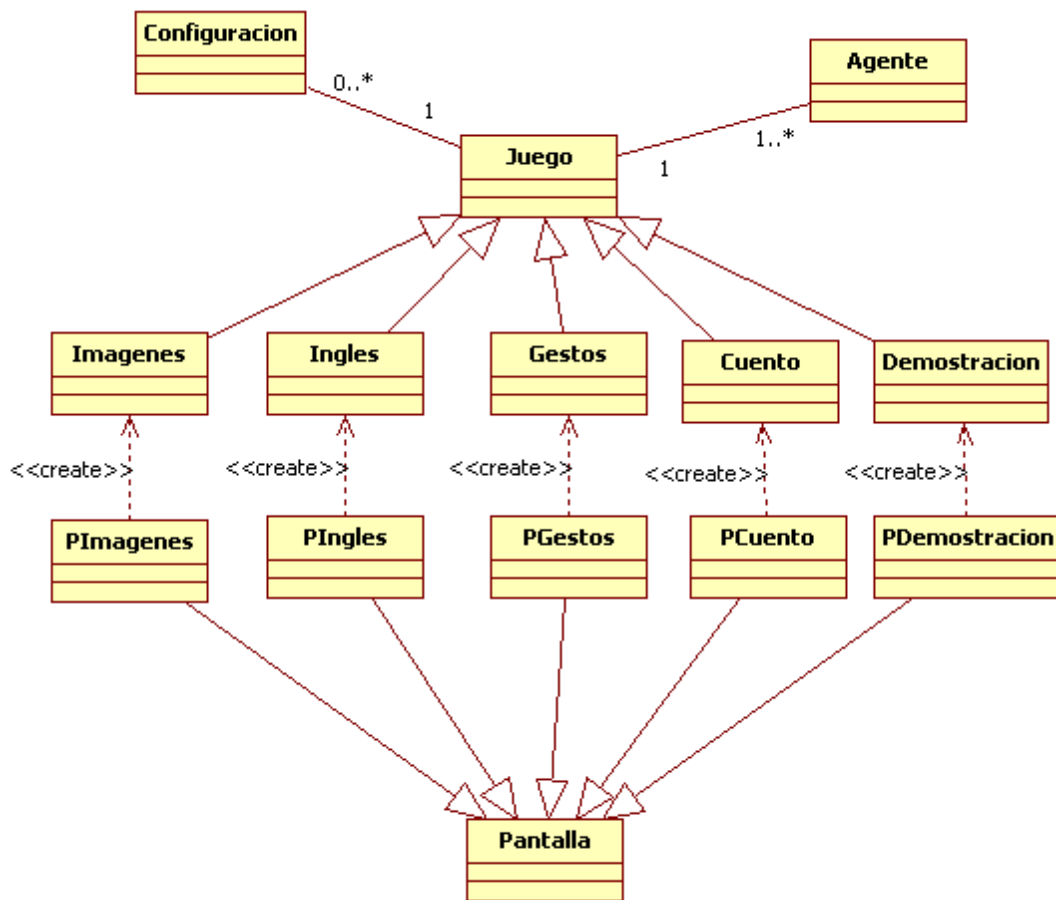


Figura 7.10: Diagrama de clases final

### 7.3. Especificación de clases

**Clase Imágenes** Es la clase encargada del funcionamiento del juego Imágenes

**Atributos:**

- AxAgentObjects.AxAgent AxAgent: punto de entrada a las funcionalidades del MsAgent;
- IAgentCtlCharacterEx Character: agente propiamente dicho;

- string [] nombres:contiene los nombres de las imágenes para ser cotejadas con el texto introducido;
- string [] definiciones:contiene los títulos de las definiciones según el nivel del usuario;
- XmlTextReader Reader :en este fichero se encontraran los datos de configuración y las definiciones.

**Métodos:**

- IAgentCtlCharacterEx cargar\_personaje\_defecto (AxAgentObjects.AxAgent AxAgent, IAgentCtlCharacterEx Character):devuelve el agente por defecto
- void mostrar\_personaje(IAgentCtlCharacterEx Character):muestra al personaje de determinada manera
- void hablar(IAgentCtlCharacterEx Character,string s,int i):el personaje lee la definición correspondiente según el texto introducido y el nivel del usuario.

**Clase PImagenes** Es la clase encargada de crear los controles de las pantallas del juego Imágenes.

**Metodos:**

- Crear\_pantalla\_temas():crea la primera pantalla con todos los controles necesarios para seleccionar el tema de imagenes.
- Crear\_pantalla\_juego():crea la segundapantalla con todos los controles necesarios para utilizar el juego propiamente dicho.

**Clase Cuento** Es la clase encargada del funcionamiento del juego Cuento.

**Atributos:**

- AxAgentObjects.AxAgent AxAgent: punto de entrada a las funcionalidades del MsAgent;
- IAgentCtlCharacterEx Character: agente propiamente dicho;
- string [] nombres:contiene los nombres de los cuentos con fin de acceder al cuento elegido;
- XmlTextReader Reader :en este fichero se encontraran los datos de configuración y los cuentos.

**Métodos:**

- IAgentCtlCharacterEx cargar\_personaje\_defecto(AxAgentObjects.AxAgent AxAgent, IAgentCtlCharacterEx Character):devuelve el agente por defecto
- public void leer\_cuento(CheckedListBox checkedListBox1,IAgentCtlCharacterEx Character, PictureBox pictureBox1) :el personaje lee el cuento seleccionado y se muestra una imagen relacionada.

**Clase PCuento** Es la clase encargada de crear los controles de las pantallas del juego Cuento.

**Metodos:**

- Crear\_pantalla\_primera():crea la pantalla con todos los controles necesarios para que se pueda seleccionar
- Crear\_pantalla\_segunda():crea la pantalla que muestre la imagen del cuento y sea escenario de la lectura.

**Clase Inglés** Es la clase encargada del funcionamiento del juego Inglés.

**Atributos:**

- AxAgentObjects.AxAgent AxAgent: punto de entrada a las funcionalidades del MsAgent;
- IAgentCtlCharacterEx Character: agente propiamente dicho;
- string [] nombres: contiene los nombres de las imágenes para ser cotejadas con el texto introducido;
- XmlTextReader Reader :en este fichero se encontraran los datos de configuración y las definiciones.

**Métodos:**

- IAgentCtlCharacterEx cargar\_personaje\_defecto(AxAgentObjects.AxAgent AxAgent, IAgentCtlCharacterEx Character): devuelve el agente por defecto
- void mostrar\_personaje(IAgentCtlCharacterEx Character): muestra al personaje de determinada manera
- void hablar(IAgentCtlCharacterEx Character, string s,): el personaje lee la definición correspondiente según el texto introducido.
- public void mostrar\_imagenes\_letras(PictureBox pictureBox1, Label label1): muestra ciertas imágenes y letras desordenadas.

**Clase PIngles** Es la clase encargada de crear los controles de las pantallas del juego Inglés.

**Metodos:**

- Crear\_pantalla(): crea la pantalla con todos los controles necesarios para que se pueda jugar a Inglés.

**Clase Gestos** Es la clase encargada del funcionamiento de la demostración de gestos.

**Atributos:**

- AxAgentObjects.AxAgent AxAgent: punto de entrada a las funcionalidades del MsAgent;
- IAgentCtlCharacterEx Character: agente propiamente dicho;
- XmlTextReader Reader :en este fichero se encontraran los datos de configuración.

**Métodos:**

- IAgentCtlCharacterEx cargar\_personaje\_defecto(AxAgentObjects.AxAgent AxAgent, IAgentCtlCharacterEx Character): devuelve el agente por defecto
- void mostrar\_gestos(ListBox List1, IAgentCtlCharacterEx Character): muestra una lista de los gestos disponibles para ese personaje.
- void mostrar\_personaje(IAgentCtlCharacterEx Character): muestra al personaje de determinada manera
- realizar\_gesto(IAgentCtlCharacterEx Character): según el gesto seleccionado el personaje lo realiza.

**Clase PGestos** Es la clase encargada de crear los controles de la pantalla de Gestos.

**Metodos:**

- Crear\_pantalla():crea la pantalla con todos los controles necesarios para que se puedan ver todos los gestos que se quiera.

**Clase Demostración** Es la clase encargada del funcionamiento de la demostración de la presentación:

**Atributos:**

- AxAgentObjects.AxAgent AxAgent: punto de entrada a las funcionalidades del MsAgent;
- IAgentCtlCharacterEx Character: agente propiamente dicho.

**Metodos:**

- void realizar\_demostracion(IAgentCtlCharacterEx peedy, IAgentCtlCharacterEx merlin, IAgentCtlCharacterEx Robby, IAgentCtlCharacterEx Genie, AgentObjects. IAgent CtlRequest peedyRequest, AgentObjects. IAgentCtl Request merlinRequest, AgentObjects. IAgentCtl Request RobbyRequest, AgentObjects. IAgentCtl Request GenieRequest):es el metodo encargado de encadenar la sucesion de gestos,movimientos y demas acciones de los personajes.
- IAgentCtlCharacterEx cargar\_personaje(AxAgentObjects.AxAgent AxAgent,string name):carga el personaje que se le diga en el string.

**Clase PDemostracion** Es la clase encargada de crear los controles de la pantalla de la presentación

**Metodos:**

- Crear\_pantalla():crea la pantalla con todos los controles necesarios para que se pueda realizar la demostracion de personajes.

**Clase Configuración** Es la clase que almacena textos y permite su lectura y escritura:

**Atributos:**

- XmlTextReader Reader:es el encargado de leer los datos que se encuentran en los ficheros XML.

**Metodos:**

- Consultar\_datos\_imagenes():devuelve las definiciones de las imagenes.
- Consultar\_datos\_cuento():devuelve los cuentos.
- Consultar\_datos\_ingles():devuelve las frases de los instrumentos.

**Clase Agente** Ofrece las mismas funcionalidades que la clase IAgentCtlCharacter del MsAgent.

**Metodos:**

- Crear\_pantalla\_opciones():este metodo encapsula una funcion del API del MsAgent que te muestra el panel de control de los personajes
- Actualizar\_valores():este metodo actualiza los valores de las opciones avanzadas cambiadas
- Crear\_pantalla\_descripcion():este metodo encapsula una funcion del API del MsAgent que te muestra una pantalla con las descripciones disponibles.

**Clase Pantalla** De esta clase derivan todas las demás pantallas e incluye métodos para pantallas especiales.

**Metodos:**

- Crear\_primera\_ayuda():muestra la pantalla de ayuda en la que se dispone de todas las opciones disponibles de ayuda.
- Crear\_segunda\_ayuda():muestra la pantalla especifica de ayuda elegida.
- Crear\_dialogo():crea un open\_file\_dialog que sera utilizado para cargar un personaje nuevo.

**Clase Juego** Es la clase abstracta de todos los posibles juegos de la aplicación.



## Capítulo 8

# Implementación

### 8.1. Software utilizado

Microsoft Visual Studio.Net,C# y Microsoft Agent 2.0 están convenientemente explicados en los apéndices aunque en este capítulo se haga referencias a ellos para demostrar como fueron utilizados en la implementación.

El entorno de desarrollo utilizado fue Microsoft Visual Studio.Net y el lenguaje C# con clara vocación de orientación a objetos.

Este lenguaje agrupa las clases en regiones,por lo que su acceso en el código es sencillo.Basta una sentencia de inclusion al inicio para utilizar todas las funcionalidades de dicha región.

Por ejemplo:la region principal es *System* y de ella heredan otras regiones como *System.Collections* o *System.Drawing*.El uso de Microsoft Visual Studio. Net facilita el acceso a clases de estas regiones ya que cuenta con un asistente intuitivo que nos informa de que clases consta una de estas regiones.

Si fuese una clase nos mostraría sus atributos y métodos con información sobre sus parámetros, acciones, precondiciones...

En todo el aspecto de creación de controles,el acceso a código no suele ser necesario porque el Visual Studio.Net cuenta con un fácil entorno de diseño.

Un ejemplo del código de la aplicación desarrollada sería:

```
OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.AddExtension = true;

openFileDialog.Filter = "Microsoft Agent Characters (*.acs)—*.acs";

openFileDialog.FilterIndex = 1 ;

openFileDialog.RestoreDirectory = true ;

if(openFileDialog.ShowDialog() != DialogResult.OK)
```

```

return ;

try { AxAgent.Characters.Unload('CharacterID'); }

catch { }

AxAgent.Characters.Load('CharacterID', (object)openFileDialog.FileName);

Character = AxAgent.Characters['CharacterID '];

```

A continuación mencionamos las aplicaciones y paquetes que hemos usado:

- Para el desarrollo de la aplicacion:
  - Microsoft Windows XP Edición Profesional.
  - Microsoft Visual Studio.NET 2003
  - Microsoft Agent 2.0
- Para la memoria:
  - Adobe Photoshop 7.0
  - GIMP 2.0
  - Microsoft PowerPoint
  - Microsoft Word 2000
  - REM 1.2.2
  - StarUML
  - MikTeX
  - WinEdt

### 8.1.1. Microsoft Agent 2.0

Este conjunto de servicios ha sido la base de la aplicación. Para acceder a ella en Visual Studio.Net hace falta disponer de la librería *AxInterop.AgentObjects.dll*, fácilmente instalable. Para acceder a ella debemos definir una referencia a la clase *AxAgent* encontrándose esa clase en dicha librería.

Desde esta referencia podemos acceder a todos los servicios del MsAgent, como el servicio de recolector de eventos, reconocimiento de voz o el acceso a las características del agente utilizado.

Un ejemplo sería:

```
AxAgent.ShowDefaultCharacterProperties(200,200);
```

Esta referencia es la que carga las funcionalidades del asistente que le digamos:

```

AxAgent.Characters.Load("merlin.acs", null);

Character=AxAgent.Characters["merlin.acs"];

```



La referencia *Character* es de tipo *IAgentCtlCharacterEx* también disponible en la librería citada anteriormente.

Un ejemplo de acceso a las funcionalidades de *Character* sería:

```
Character.Show(null);
```

```
Character.MoveTo(50,50,null);
```

```
Character.Speak("Hola buenas tardes", null );
```

```
Character.Hide(5);
```

Para elegir un motor de voz específico tendremos que cambiar el atributo *TTSMoDeID* de *Character*:

```
Character.TTSMoDeID = "2CE326E0-A935-11d1-B17B-0020AFED142E";
```

Y para el idioma el atributo *LanguajeID*:

```
Character.LanguajeID = 1033;
```

## 8.2. Hardware empleado

Equipo de desarrollo:

- S.O. Microsoft Windows XP Edición Profesional Service Pack 2
- Pentium IV 2,4 Mhz
- 256 Mb Ram
- HD 60Gb



## Capítulo 9

# Pruebas

En este capítulo se muestran las pruebas realizadas a la aplicación para comprobar su correcto funcionamiento.

Se ha realizado también como parte de estas pruebas un análisis de casos límites para ver la respuesta del programa

Se realizaron todo tipo de pruebas pero aquí solo se incluirán las pruebas de integración.

### 9.1. Pruebas realizadas

Descripción	Comprobación de la ayuda
Acción realizada	Selección del tema de ayuda deseado
Resultado esperado	Mostrar ayuda
Resultado obtenido	Ayuda mostrada
Observaciones	Correcto

**Cuadro 9.1: Comprobación de la ayuda**

Descripción	Comprobación de la opción salir del menu
Acción realizada	Seleccionar la opción salir del menu
Resultado esperado	Cerrar la aplicación
Resultado obtenido	Aplicación cerrada
Observaciones	Correcto

**Cuadro 9.2: Comprobación de la opción salir del menu**

Descripción	Comprobación de la opción Menu principal del menu
Acción realizada	Seleccionar la opción Menu principal del menu
Resultado esperado	Volver a la pantalla del menu principal
Resultado obtenido	Pantalla del menu principal mostrada
Observaciones	Correcto

**Cuadro 9.3: Comprobación de la opción Menu principal del menu**

Descripción	Comprobación del botón Cambiar personaje
Acción realizada	Pulsar el botón Cargar
Resultado esperado	Cargar el personaje seleccionado y cerrar el abierto
Resultado obtenido	Personaje cargado y cerrado
Observaciones	Correcto

**Cuadro 9.4: Comprobación del botón Cambiar personaje**

Descripción	Comprobación de la elección de tema
Acción realizada	Elegir un tema de la lista de posibles
Resultado esperado	Mostrar la pantalla correspondiente al tema elegido
Resultado obtenido	Pantalla correspondiente mostrada
Observaciones	Correcto

**Cuadro 9.5: Comprobación de la elección de tema**

Descripción	Comprobación de Introducir palabra
Acción realizada	Introducir una o varias palabras en la caja de texto
Resultado esperado	El personaje lee un texto según el valor introducido
Resultado obtenido	Texto leído correcto
Observaciones	Correcto

**Cuadro 9.6: Comprobación de Introducir palabra**

Descripción	Comprobación de consultar descripción
Acción realizada	Seleccionar Descripción del menu
Resultado esperado	Mostrar la pantalla de descripción
Resultado obtenido	Pantalla de descripción mostrada
Observaciones	Correcto

**Cuadro 9.7: Comprobación de consultar descripción**

Descripción	Comprobación de Opciones Avanzadas
Acción realizada	Seleccionar Opciones Avanzadas del menu
Resultado esperado	Mostrar la pantalla de Opciones Avanzadas
Resultado obtenido	Pantalla de Opciones Avanzadas mostrada
Observaciones	Correcto

**Cuadro 9.8: Comprobación de consultar Opciones avanzadas**

Descripción	Comprobación del botón Jugar
Acción realizada	Pulsar el botón Jugar
Resultado esperado	El personaje se presenta y aparecen una imagen y unas letras desordenadas
Resultado obtenido	Presentado el personaje y aparecidas la imagen y las letras desordenadas
Observaciones	Correcto

**Cuadro 9.9: Comprobación del botón Jugar**

Descripción	Comprobación del botón Mostrar
Acción realizada	Pulsar el botón Mostrar
Resultado esperado	El personaje realiza el gesto que hemos seleccionado en la lista
Resultado obtenido	Gesto realizado
Observaciones	Correcto

**Cuadro 9.10: Comprobación del botón Mostrar**

Descripción	Comprobación del botón Detener
Acción realizada	Pulsar el botón Detener
Resultado esperado	El personaje termina el gesto que estaba realizando
Resultado obtenido	Gesto abortado
Observaciones	Correcto

**Cuadro 9.11: Comprobación del botón Detener**

Descripción	Comprobación del botón Presentación
Acción realizada	Pulsar el botón Presentación
Resultado esperado	Los personajes realizan una presentación ordenada de sus habilidades
Resultado obtenido	Presentación ordenada
Observaciones	Correcto

**Cuadro 9.12: Comprobación del botón Presentación**

Descripción	Comprobación de Mostrar salida de voz en un globo de texto
Acción realizada	Seleccionar la posibilidad de que el audio sea acompañado de un globo de texto
Resultado esperado	Aparece un globo de texto
Resultado obtenido	Globo mostrado
Observaciones	Correcto

**Cuadro 9.13: Comprobación de Mostrar salida de voz en un globo de texto**

Descripción	Comprobación de Reproducir audio con voz
Acción realizada	Seleccionar la posibilidad de que el texto leído sea dicho con una voz sintetizada
Resultado esperado	Escuchar la voz del personaje
Resultado obtenido	Voz escuchada
Observaciones	Correcto

**Cuadro 9.14: Comprobación de Reproducir audio con voz**

Descripción	Comprobación de Reproducir efectos de sonido del personaje
Acción realizada	Seleccionar la posibilidad de que los movimientos del personaje sean acompañados de sonidos
Resultado esperado	Escuchar efectos de sonido con cada gesto
Resultado obtenido	Efectos escuchados
Observaciones	Correcto

**Cuadro 9.15: Comprobación de Reproducir efectos de sonido del personaje**

Descripción	Comprobación de la velocidad del habla
Acción realizada	Cambiar la velocidad del habla del personaje
Resultado esperado	Diferentes velocidades en el habla
Resultado obtenido	Se aprecia la diferencia de velocidad
Observaciones	Correcto

**Cuadro 9.16: Comprobación de la velocidad del habla**

**Parte IV**

**Manual de usuario**





## Capítulo 10

# Manual de usuario

### 10.1. Descripción de la aplicación

Esta aplicación se compone de un conjunto de juegos para la educación primaria. Estos juegos tendrán en el reconocimiento de objetos de la realidad su base pedagógica, además de en la interacción con los personajes animados.

A parte de los juegos tendremos presentaciones de los personajes y de sus gestos para demostrar al usuario sus habilidades.

Este programa le permitirá llevar a cabo las siguientes opciones:

1. Jugar Imágenes
2. Jugar Inglés
3. Jugar Cuento
4. Demostración Personajes
5. Demostración Gestos

En las siguientes páginas se detalla el uso y funcionamiento de la aplicación para un eficaz manejo.

### 10.2. Guía de instalación

A continuación se detalla cómo instalar el software necesario para el funcionamiento del Ms Agent y la instalación de la aplicación elaborada para el proyecto.

Todo el software necesario para la instalación de la aplicación en un PC de sobremesa se proporciona en el CD-ROM que se adjunta con la memoria en el subdirectorio Objetos dentro del directorio Fuentes, no obstante puede obtenerse también en <http://msagentring.org/setup.aspx>.

A continuación detallamos los pasos requeridos para la instalación de los requisitos software previos y de la aplicación:

1. Instalación del Ms Agent
2. Instalación de la aplicación Veo y escucho

### 10.2.1. Instalación del Ms Agent

Esta instalación comporta varios pasos:

#### Instalación de los componentes del núcleo

Los usuarios de Windows XP/2000/Me pueden saltarse este paso, ya que este núcleo ya viene instalado en estos sistemas operativos.

En otro caso ejecute MSAgent.exe y el programa de setup hara el resto.

#### Instalación de los motores de TTS

Podemos instalar todos los motores de TTS(Text-to-Speech) que queramos. Estos serán los encargados de proveer las capacidades de salida de audio de los personajes.

Para el idioma inglés utilizaremos el Lernout & Hauspie TruVoice American English TTS Engine, fácilmente instalable por el programa de setup tv\_enua.exe.

Para el idioma español hemos de instalar el motor de voz y el diccionario(en inglés no fue necesario pues ya venia instalado). El primer programa de instalación lo encontraremos en lhttspe.exe y el segundo en agtx0C0A.exe.

#### Instalación del panel de control

Este panel nos permitirá cambiar las características de nuestros personajes en tiempo de ejecución. El programa de instalación viene en SpchCpl.exe.

#### Instalar nuevos personajes

El núcleo del MS Agent solo trae cuatro personajes instalados: Peedy the parrot, Genie, Merlin the Wizard, Robby the Robot.

Si se quieren añadir personajes lo único que habrá que hacer es bajarse personajes con la extensión .acs de por ejemplo <http://msagentring.org/chars.htm> y copiarlos en ...

En esta aplicación tendremos mas personajes disponibles de los predefinidos.

### 10.2.2. Instalación de la aplicación Veo y escucho

En el CDRom adyacente a esta memoria encontraremos una carpeta llamada Veo\_y\_escucho en la que encontraremos un subdirectorio llamado Debug. Dentro de el tendremos un setup.exe. Al ejecutar dicho fichero tendremos un fácil asistente que instalará la aplicación.

## 10.3. Manual de usuario de la aplicación

### 10.3.1. Bienvenida

Es la pantalla inicial y se muestra nada más arrancar la aplicación. Esta pantalla contiene el título de la aplicación y el nombre del autor.

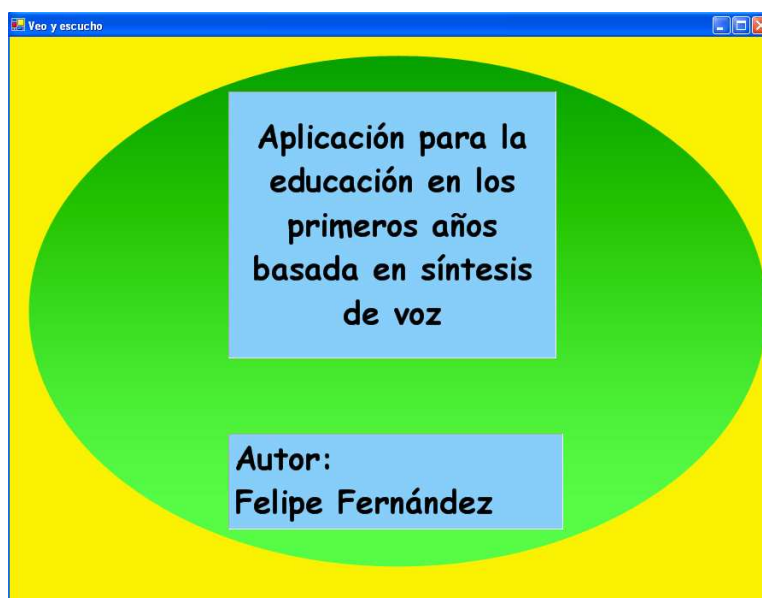


Figura 10.1: Pantalla inicial

Al pulsar sobre el título aparece la pantalla del menu.

Podemos hacer uso de sus opciones en cualquier pantalla, y se accede a ella pulsando el botón menu que se encuentra en el menu superior.



Figura 10.2: Pantalla del menu

A continuación se describe cada una de las opciones que ofrece esta pantalla.

### 10.3.2. Imágenes

Nada mas seleccionar esta opción un personaje por defecto será cargado. Si deseáramos cambiarlo a lo largo del juego tendríamos una opción en el menu superior llamada Cambiar personaje.

La primera pantalla que nos aparecerá tendrá una lista con los temas que podemos elegir:

- Animales
- Deportes
- Cosas de casa



Figura 10.3: Pantalla para elegir temática

Lo único que cambia entre esas tres opciones serán las imágenes. El fundamento del juego será el mismo.

La idea es que el usuario introduzca en la caja de texto correspondiente al nombre de la imagen de arriba. Para validar la entrada puede pulsar enter en el teclado o un botón de aceptación que estará situado en la parte inferior-derecha.

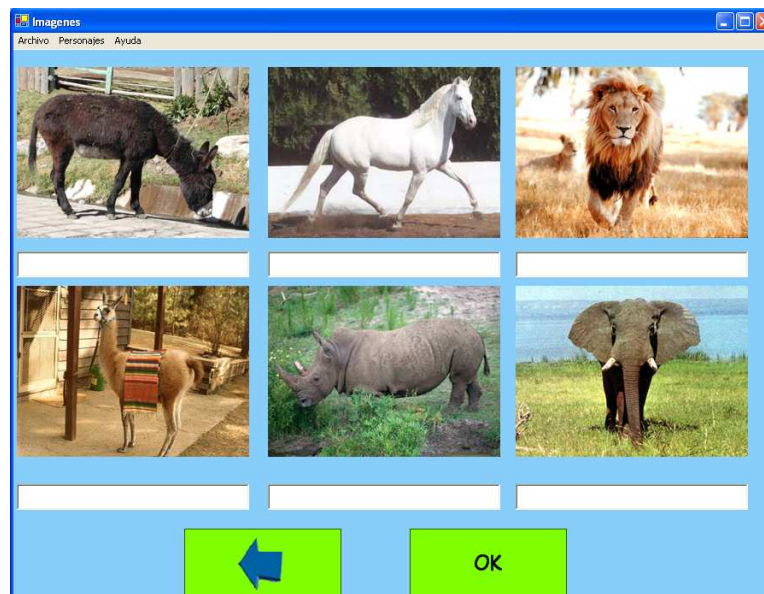


Figura 10.4: Pantalla del juego Imágenes

Si el nombre es correcto el personaje animado leerá una definición sobre esa imagen. Si no es correcto el personaje comunicará el error y animará al usuario a que vuelva a intentarlo.

En ambas pantallas dispondremos de un botón que nos permita volver al menú y de un menú superior que nos ofrece todas las opciones de la aplicación.

### 10.3.3. Inglés

Al igual que en el juego de imágenes al seleccionar esta opción el personaje será cargado por defecto. En este caso antes de empezar a jugar ya se mostrará informándonos en inglés del objetivo del juego: Ordenar las letras que nos muestran.

Sólo se mostrará una imagen y unas letras desordenadas de cada vez así que cada vez que queramos volver a jugar pulsaremos el botón Jugar.

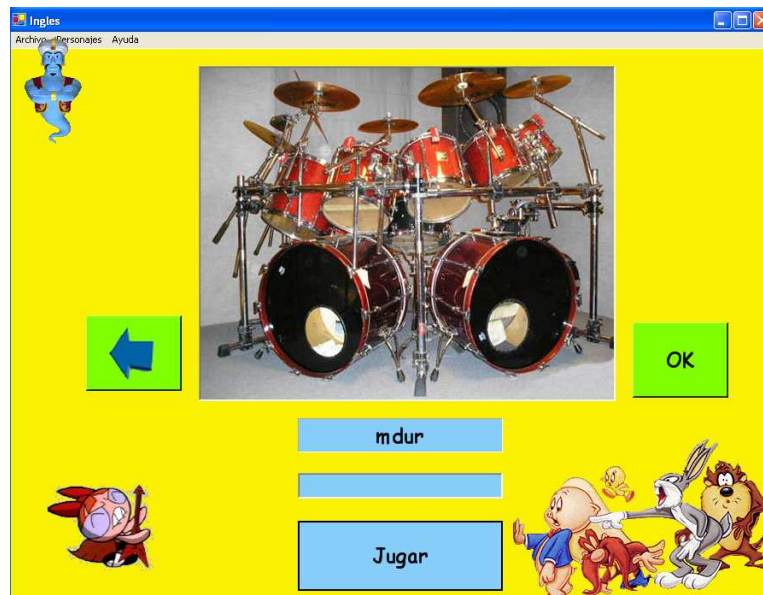


Figura 10.5: Pantalla del juego Inglés

Este juego aparte de fomentar el aprendizaje del idioma inglés, cuida la parte musical pues todas las imágenes serán de instrumento.

El fundamento del juego es el mismo que en el de imágenes: introducir el nombre de la imagen correspondiente en la caja de texto y validarla ya sea con la tecla enter o el botón puesto ex profeso.

En este juego se introducen las letras desordenadas ya que el nivel de inglés de un niño es mucho más bajo que el de su lengua materna.

El personaje animado leerá un mensaje de aceptación o de error según lo introducido.

Dispondremos de un botón que nos permita volver al menu y de un menu superior que nos ofrece todas las opciones de la aplicación.

#### 10.3.4. Cuento

Nada más seleccionar esta opción un personaje por defecto sera cargado. Si deseáramos cambiarlo a lo largo del juego tendríamos una opción en el menu superior llamada Cambiar personaje.

La primera pantalla sera una lista con todos los cuentos disponibles. El niño solo tendrá que seleccionar el que quiera y pulsar el botón Escuchar. En el caso de que no seleccionara ninguno se leería el cuento por defecto Los tres cerditos.

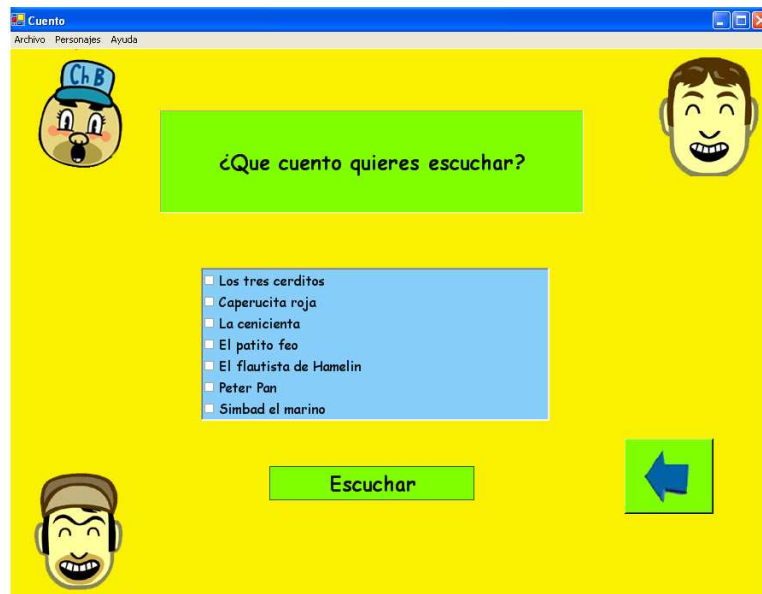


Figura 10.6: Pantalla de elección del cuento

El personaje animado leerá el cuento en una nueva pantalla. Se mostrara una imagen relacionada con el cuento.



Figura 10.7: Lectura del cuento Los tres cerditos

Dispondremos de un botón que nos permita volver al menu y de un menu superior que nos ofrece todas las opciones de la aplicación.

### 10.3.5. Presentación

Esta opción tiene una intención mas lúdica que pedagógica ya que mostrara una demostración de las habilidades de los personajes.

Los personajes que aparecen ya están seleccionados e irán realizando la mayoría de sus funcionalidades de manera ordenada para el deleite del usuario.



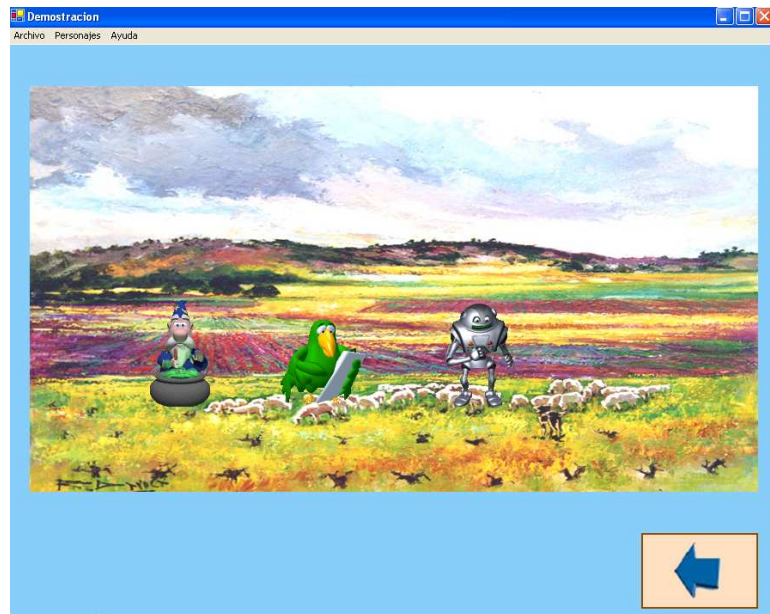


Figura 10.8: Pantalla donde se realizara la presentación

Dispondremos de un botón que nos permita volver al menu y de un menu superior que nos ofrece todas las opciones de la aplicación.

### 10.3.6. Gestos

En este apartado el usuario conocerá en profundidad las posibilidades de todos los personajes que no aparecen aquí por lo limitado de la aplicación.

Al igual que en los otros juegos el personaje inicial que se cargara será por defecto aunque el botón Cambiar personaje, que en esta ocasión estará situado también en la pantalla por ser mucho mas habitual su uso en esta sección, permitirá elegir cualquiera de los otros personajes.

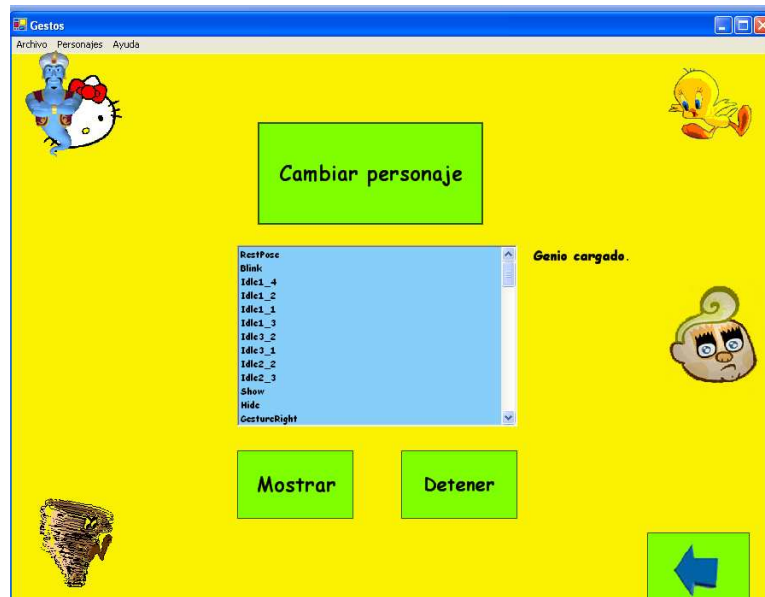


Figura 10.9: Pantalla donde se realizaran los gestos

Para cada personaje surgirá una lista de todos los gestos disponibles. Una vez elegido pulsando el botón Mostrar el personaje realizará dicho gesto. En todo momento podremos abortarlo con el botón Detener.

Adyacente a la lista encontraremos una etiqueta informativa que nos comunicará que el personaje está cargado.

Dispondremos de un botón que nos permita volver al menú y de un menú superior que nos ofrece todas las opciones de la aplicación.

### 10.3.7. Menu superior

Ese menú estará en la parte superior de la aplicación en todas las ventanas excepto en las de ayuda.

Desde ella se puede acceder al menú, al juego imágenes, inglés y cuento, a la presentación y a los gestos. También cambiar personaje, descripción de los personajes, opciones avanzadas del personaje, a la ayuda y al acerca de.

A continuación explicamos las pantallas que no han sido expuestas previamente.

### 10.3.8. Cambiar personaje

Los juegos y la sección de gestos parten con un personaje predeterminado, pero si queremos podemos elegir otros caracteres que realicen las labores de asistente.

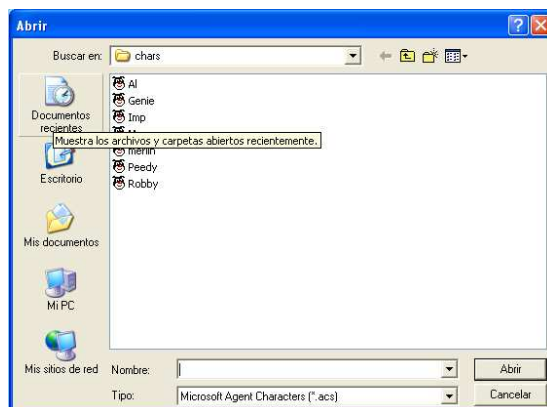


Figura 10.10: El dialogo se abre sobre la pantalla desde la que lo hayamos seleccionado

En el menu superior tendremos la opción de Cambiar personaje. Al pulsarla nos aparecerá un dialogo en el directorio donde se encuentren los .acs.

### 10.3.9. Descripción

La descripción que podamos encontrar sólo vendrá dada para los caracteres que al haber sido desarrollados hayan incluido esta opción.

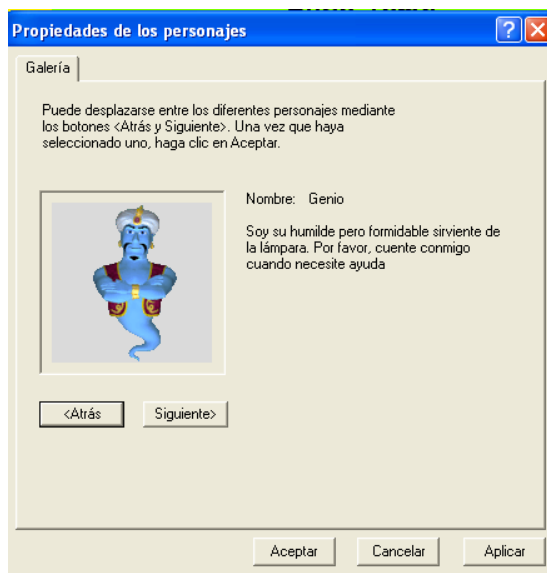


Figura 10.11: Pantalla con la descripción del personaje

Junto a la breve descripción nos obsequiaran con un saludo.

### 10.3.10. Opciones avanzadas de los personajes

En este desplegable podremos seleccionar los valores mas importantes de nuestro personaje.

Algunos de ellos son:globo de texto y la fuente de letra,velocidad del habla y efectos de sonido.

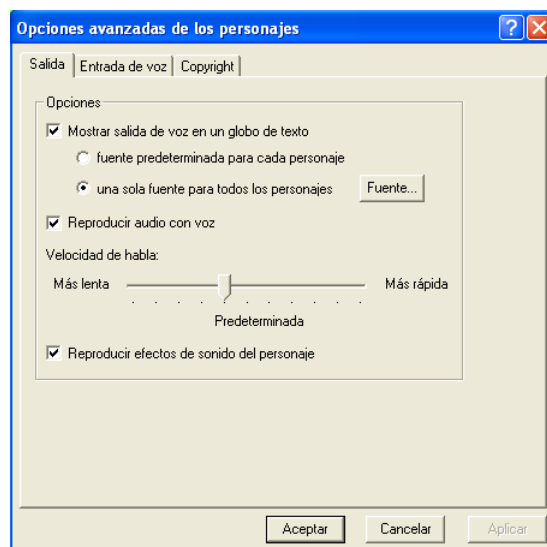


Figura 10.12: Pantalla con las opciones avanzadas del personaje

También encontraremos opciones de reconocimiento de habla pero al no ser utilizadas en esta aplicación no son operativas.

Estas opciones no se pierden,hasta que vuelvan a ser cambiadas permanecerán.

### 10.3.11. Consultar ayuda

El primer menu nos dará la opción de elegir que tipo de ayuda queremos:general,imágenes,inglés y cuento.



Figura 10.13: Pantalla con posibilidades de ayuda

Una vez seleccionada nos mostrará una pantalla con la ayuda deseada.

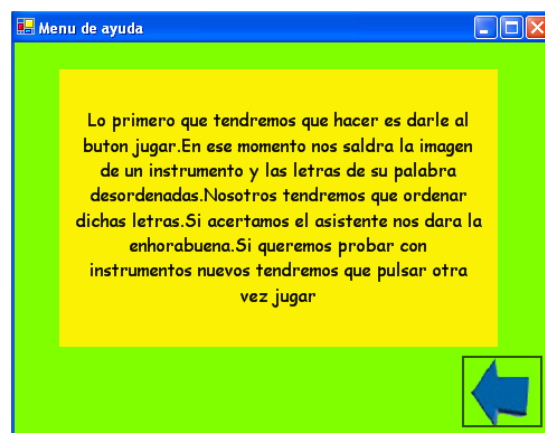


Figura 10.14: Pantalla con la ayuda seleccionada

### 10.3.12. Acerca de

Esta pantalla nos informa de quién y dónde ha sido desarrollada esta aplicación.



Figura 10.15: Pantalla de Acerca de

# **Parte V**

## **Conclusiones**





## Capítulo 11

# Conclusiones

### 11.1. Dificultadas encontradas

La síntesis de voz esta un una etapa inicial y ni que decir tiene los dispositivos móviles. Aunar sintetizador de voz, libre, en español, para Pocket PC y que trabaje bajo Windows fue imposible.

Desde aquí dar las gracias a Jordi Lagares y a David Escudero por dejarme sus motores de voz.

En el caso de Jordi, promotor del Proyecto Fressa, me proporciono su librería en español y catalán. El problema fue el tamaño de ese motor. Las características de memoria de un Pocket PC hacían imposible su uso.

El motor de David tuvo el gran problema de estar desarrollado en Linux. El mismo me dijo que el trabajo de portabilidad de Linux a Windows sería un proyecto en si. Mi idea de realizar una aplicación con carácter social seguía en mi cabeza, y no quise apartarme, para centrarme en el problema de la portabilidad, nada trivial por cierto.

Con Flite, mi gran esperanza a priori, sucedió lo mismo. Fue desarrollado bajo Linux. La mayor difusión de este sintetizador hizo que encontrara en la Red guías de como realizar la portabilidad. Pero tampoco me fue posible.

En este proceso di con el MsAgent. El problema del tamaño seguía siendo un obstáculo, pero la aparición de las interfaces animadas hizo que me replanteara la necesidad de una aplicación en dispositivo portátil.

El trabajo con Visual Studio.Net ha sido placentero e intuitivo. El único pero que le he visto fueron las dificultades para añadir librerías que no fueran de Microsoft.

### 11.2. Objetivos alcanzados

- El objetivo del estudio de las alternativas de síntesis de voz ha sido cumplido con creces. He visto que el trabajo en software libre con iniciativas como Festival y Flite, es ilusionante y dado su carácter abierto, muy participativo. Desde las universidades se esta trabajando mucho en desarrollar mejores sintetizadores.

- La aplicación cumple el valor pedagógico intrínseco de la adquisición de definiciones y reconocimiento de items del entorno.
- Mas importante aun creo ha sido el logro de la interacción fluida del asistente con el usuario, lo que logra un acercamiento al mundo multimedia ameno y eficaz.
- Este proyecto puede ser una via de entrada para otros en el estudio del Ms Agent y sus posibilidades mas allá de esta aplicación.
- El conocimiento del entorno de desarrollo Visual Studio . Net y del lenguaje C# ha sido amplio.

### 11.3. Posibles mejoras

En este capitulo mencionaremos posibles mejoras que podrían realizarse utilizando otras partes del Ms Agent para nuestra aplicación y otras posibles vias de estudio para aplicaciones posteriores. Proponemos a continuación alguna de ellas:

#### 11.3.1. Reconocimiento del habla

Este proyecto ha estado basado en la síntesis de voz pero el reconocimiento de voz puede ser un poderoso instrumento pedagógico.

Los motores de reconocimiento tienen las mismas dificultades para trabajar en dispositivos móviles, que los de síntesis. También la gente de Flite, de la universidad de Carnegie Mellon, han desarrollado motores de reconocimiento como el Sphinx.

Ms Agent cuenta entre sus posibilidades la de incorporar reconocimiento de voz utilizando los motores de Microsoft. En el manual del apéndice viene como utilizar adecuadamente esta funcionalidad

Las posibles mejoras en la aplicación podrían ser desde la sustitución de la entrada de texto por entrada de voz, o desarrollar juegos en los que el reconocimiento fuera la piedra angular.

#### 11.3.2. Desarrollo en aplicaciones web

Ms Agent puede ser utilizado en aplicaciones web. Es mas muchas webs comerciales empiezan a utilizar estos asistentes como reclamo de marketing.

Siguiendo con el paradigma de la Web 2.0 que pretende que todas nuestras aplicaciones se encuentren directamente en el servidor, podría realizarse esta aplicación en web. O pensar en un portal infantil que utilice estos agentes.

#### 11.3.3. Permitir la introducción de nuevos elementos

Seria interesante permitir la posibilidad de introducir nuevos cuentos, nuevas imágenes, o nuevos instrumentos.

No obstante la posibilidad de incluir mas personajes, que es algo bastante importante, viene convenientemente explicada en el manual de usuario.

# Bibliografía

## Bibliografía

1. Grady Booch,James Rumbaugh,Ivan Jacobson, ' El lenguaje unificado de desarrollo software ', Ed. Addison Wesley Iberoamericana,1999.
2. C.Larman,'UML y Patrones' Ed Pearson Educación, 2002.
3. John Sharp, Jon Jagger 'Microsoft Visual C#.NET Aprenda ya', Ed McGraw Hill,2002.
4. Charles Wright , 'Superutilidades para C#' Ed McGraw Hill,2003.

## Fuentes WEB

1. <http://www.cidse.itcr.ac.cr/revistamate/HERRAmInternet/Latex/wmlatexrevista/index.html>. Útil manual sobre Latex,imprescindible para el desarrollo de esta memoria.
2. <http://www.elguille.info/>.Web sobre programación en general en la que encontrar mucha información interesante sobre Visual Studio.Net y C#.
3. <http://msagentring.org/>.Anillo de páginas que trabajan con Ms Agent.Aquí puedes encontrar desde consejos,scripts o personajes desarrollados por ellos mismos.
4. <http://msdn.microsoft.com>. Completo página de ayuda de Microsoft donde encontrar información sobre el Ms Agent.
5. <http://www.perfectxml.com/>.Buena página de XML en la que encontrarás aspectos fundamentales sobre XML.
6. <http://www.speech.cs.cmu.edu/hephaestus.html>.Portal de la CMU donde se encuentran sus motores de síntesis de voz y de reconocimiento del habla.
7. <http://www.todopocketpc.com/>.Web muy útil para todo lo relacionado con los dispositivos móviles.Tiene un foro bastante activo.
8. <http://www.xtec.net/jlagares/f2kesp.htm>.Página del proyecto Fressa en la que muestra las aplicaciones realizadas para todo tipo de discapacidades.



## Apéndice A

# Microsoft Agent

### A.1. Potencial de Microsoft Agent 2.0

Si se busca un sistema avanzado de ayuda basado en personajes, Microsoft Agent, es una opción a tener en cuenta porque es una poderosa herramienta para animar y humanizar interfaces de usuario.

Microsoft Agent versión 2.0 proporciona un juego de servicios software programables que pueden usarse para complementar una interfaz de usuario clásica al facilitar la integración de personajes animados interactivos. Gestiona la animación de los mismos e incluye soporte para tecnologías complementarias como entrada y salida de habla, entre otras. Esto favorece y ensancha la comunicación que fluye entre un usuario y sus aplicaciones y permite desarrollar una interfaz de usuario conversacional con un ancho de banda contenido.

Esta tecnología proporciona valor añadido al ser incluida como un componente de una página web o de una aplicación convencional diseñada para plataformas Windows.

Algunas de las características de Microsoft Agent son:

- Se integra como un componente extra manteniendo una estructura orientada a objetos.
- El acceso a esta tecnología se realiza desde una interfaz de programación que puede ser codificada desde cualquier lenguaje que soporte COM, tal como C++ o Visual Basic. También incluye un control ActiveX que facilita la programación desde lenguajes tipo script como VBScript o JScript.
- Gracias al conversor texto a voz que permite incorporar, no es necesario transmitir la voz natural a través de la línea de comunicación, y se puede disponer de aplicaciones que hablen con un gran ahorro de ancho de banda. Aunque también es posible reproducir un fichero de audio grabado o sacar los mensajes por pantalla en un bocado de texto.
- Integra reconocimiento de habla, que puede ser usado para que la interfaz responda a comandos o para facilitar la entrada de datos.
- Está disponible en la red para descargarlo gratuitamente y puede ser empleado por los desarrolladores en sus aplicaciones sin coste de derechos de autor.

### **A.1.1. ¿Respecto a qué referencias o antecedentes se puede comparar a Microsoft Agent?**

Por un lado puede recordar a Microsoft Bob, el cual fue un rotundo fracaso de mercado, pero ambos se diferencian en tres importantes cuestiones:

- Microsoft Bob era una aplicación dirigida a nuevos usuarios de PC's. Consistía en una guía interactiva para una serie de programas. Por contra, Microsoft Agent no es una aplicación, sino una tecnología que se integra con las aplicaciones.
- Con Bob los personajes aparecían en una interfaz que enmascaraba la interfaz Windows. Microsoft Agent, sin embargo, se integra sin costuras en las aplicaciones ya existentes y en la interfaz del sistema operativo. Los personajes de Agent aparecen en sus propias ventanas no rectangulares, cuya forma es la silueta de la animación que se esté ejecutando, y pueden aparecer en cualquier lugar de la interfaz convencional.
- Además, al contrario que en Bob, los personajes de Agent pueden ser ocultos en cualquier momento a voluntad del usuario, puesto que no ha de ser el sustituto de la interfaz sino un complemento que intenta mejorarla.
- Por otro lado, al hablar de Microsoft Agent, se puede pensar que es algo similar al Asistente de Microsoft Office (Clipo por ejemplo). Pero, en realidad, Microsoft Agent es la herramienta para crear ese tipo de aplicaciones.

### **A.1.2. ¿Para qué pueden ser usados los personajes proporcionados por Microsoft Agent?**

No están pensados para ningún tipo específico de aplicación. Los personajes son como actores a los que los desarrolladores les escriben un guión convirtiéndolos en una herramienta determinada. Por ejemplo, los siguientes son sólo unos pocos de los muchos roles que los desarrolladores pueden escribir para un personaje:

- Puede ser un anfitrión que saluda a los usuarios la primera vez que encienden el ordenador o instalan una aplicación.
- Puede ser un guía turístico y hacer una breve explicación acerca de lo que se encuentra disponible en un sitio web o de las capacidades de una aplicación.
- Puede ser un tutor que muestra cómo puede llevarse a cabo una tarea paso a paso.
- Puede actuar como un agente de ventas, o dirigir al usuario a través de una serie de preguntas hasta la toma de una decisión. Puede entregar mensajes como recordatorios, avisos, alertas, o cosas similares.
- Puede actuar como asistente personal que busque información en Internet y la lea en voz alta.

Puede ser divertido, manteniendo la atención de los niños para enseñarles o entretenerles, por ejemplo, leyéndoles cuentos. En la siguiente figura podemos ver algunos de los personajes suministrados por Microsoft.

De izquierda a derecha podemos ver a Robby el Robot, Peedy el Loro, Genie y Merlin.



Figura A.1: Galería de personajes de Microsoft

Si no se quieren usar estos personajes, se puede crear un personaje propio y compilar sus animaciones usando el Editor de Personajes de Microsoft Agent

## A.2. Instalación

Para poder acceder a la interfaz del servidor de Microsoft Agent, éste debe estar previamente instalado sobre el sistema. Si se quiere utilizar Microsoft Agent como parte de otras aplicaciones, se debe adquirir una licencia de distribución. El acuerdo de licencia está disponible en la página web: <http://www.microsoft.com/sitebuilder/workshop/imedia/Agent/licensing.asp>

El motor de Microsoft Agent, los personajes de Microsoft, los motores de habla, alguna herramienta complementaria, como el editor de personajes, la documentación completa del API, y código de ejemplo pueden ser encontrados en el sitio web de Microsoft Agent en: <http://www.microsoft.com/msagent/>.

Cuando el fichero de instalación se ejecuta, Agent se instala automáticamente en el sistema. Por lo que, este fichero ejecutable, puede ser incluido como parte del programa de instalación de la aplicación que incorpore Agent como componente.

El fichero de instalación de Microsoft Agent no se instalará sobre Microsoft Windows 2000, porque esa versión del sistema operativo ya incluye su propia versión de Agent.

Para instalar correctamente Agent sobre un sistema, se debe asegurar que el sistema tiene una versión reciente de la librería de tiempo de ejecución de Microsoft Visual C++ (Msvcrt.dll), la herramienta de registro de Microsoft (Regsvr.dll), y las DLLs COM. La manera más fácil de asegurar que los componentes necesarios están en un sistema es requerir que Microsoft Internet Explorer versión 3.02 o posterior esté instalado. De forma alternativa, se pueden instalar los primeros dos componentes al instalar Microsoft Visual C++ y el componente que resta, las DLLs COM, puede ser instalado como parte de la actualización de Microsoft DCOM, disponible en la página web de Microsoft.

Igualmente, los personajes de Microsoft en formato ACS, están disponibles para distribución en la página web de Microsoft Agent.

Como los componentes de Microsoft Agent están diseñados como componentes de sistema operativo, Agent no puede ser desinstalado. Igualmente, cuando Agent está ya instalado como parte del sistema operativo, el fichero de auto instalación no instalará nada.

### A.3. Interfaz de Usuario de Microsoft Agent

Microsoft Agent proporciona varios componentes de interfaz para facilitar a los usuarios acceder e interactuar con los personajes, conocer su estado y cambiar parámetros globales que afectan a todos los personajes. Este apartado describe los elementos básicos de la interfaz de usuario de Microsoft Agent.

#### A.3.1. Ventana del personaje

Microsoft Agent muestra los personajes animados en sus propias ventanas, las cuales siempre aparecen superpuestas a las demás ventanas. La translación de los personajes en pantalla lo realizan las aplicaciones mediante el método MoveTo del API de Agent. Pero, además, un usuario puede mover la ventana de un personaje arrastrándolo mientras presiona el botón izquierdo del ratón. Las imágenes del personaje se mueven siguiendo al puntero del ratón.

#### A.3.2. Menú contextual de comandos

Cuando el usuario clicla con el botón derecho sobre un personaje, aparece un Menú Contextual emergente como el de la siguiente figura:



Figura A.2: Menú Contextual de comandos

El comando Abrir—Cerrar ventana Comandos de voz controla la aparición de la ventana de comandos de voz del personaje que esté activo en ese momento. Si los servicios de reconocimiento de habla están deshabilitados, este comando está deshabilitado. Si los servicios de reconocimiento de habla no están instalados, este comando no aparece.

El comando Ocultar ejecuta la animación asignada al estado Hiding del personaje y esconde al personaje.

Los comandos propios, OtrosComandosPropiosDeLaAplicación, que la aplicación quiera incluir en este menú, se sitúan detrás del comando Ocultar separados por una línea de división. Por ejemplo, puede verse en la figura anterior como la aplicación correspondiente ha incluido el comando Advanced Character Properties como una de las entradas.



### A.3.3. Icono del personaje en la barra de herramientas

Si un personaje ha sido creado para incluir un icono, el icono aparecerá en el área de notificación de la barra de herramientas cuando el servidor de Microsoft Agent está en ejecución y el personaje es cargado.



Figura A.3: Iconos de Merlín y Genio en la barra de herramientas

Pasando el puntero del ratón sobre el icono se muestra un pequeño cuadrado con el nombre del personaje (en el lenguaje del sistema). Así lo muestra la siguiente figura:



Figura A.4: Ventana con el nombre del personaje

Haciendo click sobre el icono se muestra al personaje. La acción asociada con el doble click sobre el icono depende de la aplicación que esté controlando al personaje en ese momento.

Al hacer click con el botón derecho sobre el icono se muestra el Menú Contextual del personaje. Cuando el personaje es visible, este menú muestra los mismos comandos que aquéllos mostrados al hacer click con el botón derecho sobre el personaje. Si el personaje está escondido, sólo son mostrados los comandos Abrir—Cerrar Ventana de Comandos de Voz y Mostrar.

### A.3.4. Ventana de Comandos de Voz

Si un motor de habla compatible está instalado, Microsoft Agent proporciona una ventana especial llamada Ventana de Comandos de Voz que muestra los comandos que están disponibles para reconocimiento de voz. La Ventana de Comandos de Voz sirve como recordatorio visual de lo que puede ser

dicho como entrada.

La ventana aparece cuando un usuario selecciona el comando Abrir Ventana de Comandos de Voz, bien por decir el comando o haciendo click con el botón derecho sobre el personaje y eligiendo ese comando desde el menú contextual del personaje. Sin embargo, si el usuario deshabilita la entrada de voz, la Ventana de Comandos de Voz no estará accesible.

La Ventana de Comandos de Voz muestra los posibles comandos en forma de árbol. Si la aplicación proporciona comandos de voz, estos aparecen expandidos al comienzo de la ventana. La ventana también incluye los comandos de voz globales proporcionados por Microsoft Agent. Si la aplicación no tiene comandos de voz, sólo aparecen los comandos de voz globales. El usuario puede trasladar la Ventana de Comandos de Voz. Microsoft Agent recuerda la última posición de la ventana y la vuelve a mostrar en esa posición si el usuario reabre la ventana.

### A.3.5. Globo de Texto

Para completar la salida de audio hablada, la interfaz de Microsoft Agent proporciona una salida a través de un Globo de Texto, similar al típico bocadillo de texto de los cómics. Las palabras aparecen en el bocadillo mientras van siendo pronunciadas por el personaje. El bocadillo desaparece cuando la salida de habla termina. La Ventana de Opciones Avanzadas de los personajes proporciona opciones para deshabitar este tipo de salida así como para controlar los atributos relativos a su aspecto. El Globo de Texto tiene el aspecto que muestra la siguiente figura:



Figura A.5: Globo de Texto

### A.3.6. Ventana informativa del estado Listening

Si el reconocimiento de habla está habilitado, el usuario puede presionar la tecla click-to-speak (presionar para hablar) para forzar a que el personaje esté atento a la entrada de voz (estado Listening) y así poder comenzar la entrada de voz. En ese momento, aparece una ventana especial que muestra información contextual relativa al estado actual de la entrada de voz. Si un motor de reconocimiento compatible no ha sido instalado o no ha sido habilitado, esta ventana no aparecerá

### A.3.7. Ventana de Opciones Avanzadas de Personajes

Microsoft Agent mantiene ciertas opciones globales que regulan la interacción con todos los personajes. La Ventana de Opciones Avanzadas de los personajes muestra estas opciones y sus configuracio-

nes actuales. Aunque es mostrada por la aplicación a través del API del Agente, sus opciones sólo son configurables por el usuario

### A.3.8. Página de salida

Esta página incluye las propiedades que controlan la salida de voz del personaje. Por ejemplo, el usuario puede determinar si quiere mostrar salida a través del Globo de Texto y cómo ha de ser esa salida, si se quiere reproducir salida de voz o si se prefiere mantenerla inhibida, puede indicar si se han de reproducir los efectos de sonido de las animaciones del personaje o puede ajustar la velocidad a la que hablan los personajes.

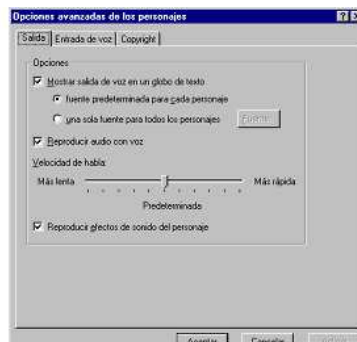


Figura A.6: Página de propiedades de salida

### A.3.9. Página de la entrada de voz

El usuario puede ajustar las opciones de la entrada de habla en esta página de propiedades. El usuario puede deshabilitar la entrada de habla, fijar la tecla de comienzo del estado de escucha, elegir si quiere mostrar la ventana de información del estado de escucha, y elegir si se ha de ejecutar una señal MIDI indicando que la entrada de habla está disponible

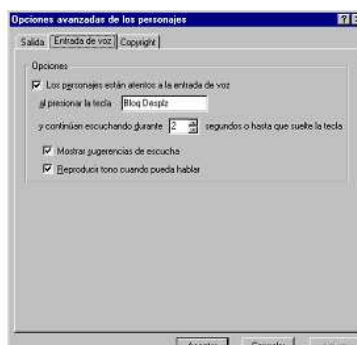


Figura A.7: Página de propiedades de la entrada de voz

### A.3.10. Página del Copyright

Esta página muestra el copyright e información de la versión de Microsoft Agent.



Figura A.8: Página del Copyright

#### A.3.11. Ventana de Propiedades del Personaje por Defecto

Las aplicaciones pueden cargar un personaje concreto o no especificar personaje y así cargar el Personaje por Defecto. La figura del personaje por defecto ha sido ideada por Microsoft para convertirla en el asistente centralizado de Windows, que será compartido por las diferentes aplicaciones. Este personaje es accesible desde cualquier aplicación, pero es únicamente seleccionable por el usuario a través de su Ventana de Propiedades. De este modo, las aplicaciones pueden conocer el personaje favorito del usuario. El acceso a esta ventana se realiza a través del API del Agente.



Figura A.9: Ventana de Propiedades del Personaje por Defecto

### A.4. Interfaz de programación de Microsoft Agent

La interfaz de programación de aplicación (API) de Microsoft Agent proporciona servicios de programación entre los que destacan: la habilidad de cargar un personaje, ejecutar una animación, hablar usando un sintetizador de voz, o un fichero de audio (sincronizando la salida automáticamente con el movimiento de los labios del personaje) y aceptar entrada de voz del usuario.

Microsoft Agent ha sido creado como un Servidor de Automatización OLE. La Automatización OLE consiste en que un programa, el cliente, puede invocar métodos almacenados en otro programa, el servidor, que se ejecuta de forma invisible, de tal modo que el usuario no es consciente de ninguna interacción. La Automatización OLE es una implementación específica de la arquitectura Modelo de Objetos Componentes (COM). La convención COM facilita la integración de componentes, pues regula

las interfaces y la forma de interactuar de los componentes, en este caso, cliente y servidor.

De esta manera, Microsoft Agent posibilita que múltiples aplicaciones, llamadas clientes o aplicaciones de cliente, hospeden y accedan a sus animaciones y a sus servicios de entrada y salida a un mismo tiempo. Un cliente puede ser cualquier aplicación que conecte con las interfaces COM del servidor de Microsoft Agent, AgentSvr.exe.

Como cualquier servidor COM, AgentSvr.exe sólo arranca cuando una aplicación cliente usa las interfaces COM y pide conectarse a él. Continúa ejecutándose hasta que todos los clientes cierran sus conexiones. Cuando no queda ningún cliente, el servidor se cierra automáticamente.

#### **A.4.1. Acceso a los servicios de programación**

Como Microsoft Agent ha sido creado como Servidor de Automatización OLE, dispone de una interfaz dual para que se acceda a sus servicios:

Por un lado, se puede integrar en una aplicación como un control ActiveX (OLE) y usar un controlador de automatización para acceder a los servicios.

Por otro lado, se puede llamar directamente a la interfaz COM de Microsoft Agent.

#### **A.4.2. Control ActiveX**

Usar Microsoft Agent como un control ActiveX se puede hacer desde cualquier lenguaje de programación que sea capaz de implementar un controlador de distribución, como por ejemplo C++, Visual Basic, o lenguajes tipo script como VBScript. Como los controles ActiveX están pensados para su uso en Internet, Microsoft Agent puede ser controlado a través de scripts que se incluyan en páginas Web si localmente está instalado la tecnología Agent y si el navegador soporta la interfaz ActiveX.

#### **A.4.3. Acceso directo a la Interfaz COM**

Hacer una llamada a un método o un acceso a una propiedad a través de una interfaz de distribución es un proceso que supone gran cantidad de sobrecarga, lo cual repercute en aumento del tiempo de ejecución respecto a la misma funcionalidad obtenida a través de una interfaz de tabla de funciones virtuales, vtable. Por esta razón, la Automatización OLE también permite a un cliente realizar vinculación mediante una vtable y llamar a las funciones de la interfaz de manera eficiente.

#### **A.4.4. Modelo de Objetos de Microsoft Agent**

El Modelo de Objetos Microsoft Agent consiste en los objetos de la siguiente lista:

- Request (Solicitud de Petición)
- Agent (Agente)
- Characters (Colección de Personajes)
- Character (Personaje)
- Commands (Colección de comandos)
- Command (Comando)

- Balloon (Globo de Texto)
- AnimationNames(Colección de Animaciones)
- SpeechInput (Entrada de voz)
- AudioOutput(Salida de audio)
- CommandsWindow (Ventana de Comandos)
- PropertySheet (Ventana de Opciones Avanzadas de Personajes)

Estos objetos están organizados según la jerarquía que muestra la siguiente figura

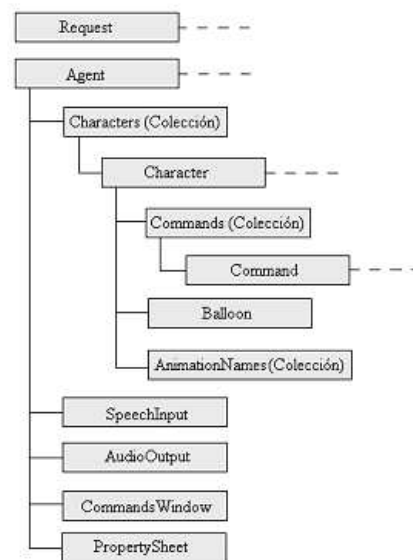


Figura A.10: Jerarquía de objetos del Objeto Agent

La línea de puntos indica que pueden existir múltiples objetos de ese tipo instanciados al mismo tiempo.

En la figura se puede apreciar como el objeto principal es el objeto Agent y como el resto de objetos dependen de él, salvo el objeto Request.

El objeto Request no está implementado dentro del objeto Agent, pero es parte integrante del servidor AgentSrv.exe. En concreto se encarga de lanzar los eventos que produce el servidor ante una acción del usuario o para sincronizar las animaciones. Más adelante el proceso de captación de eventos será explicado en detalle.

También es de destacar que alguno de esos objetos sean colecciones. El término colección define en general a un grupo de objetos donde, a diferencia de un array, no existen posiciones ordenadas. Las Colecciones de Automatización OLE son colecciones de objetos que cumplen el estándar de interfaz OLE. Son implementadas por pseudo objetos que controlan el acceso a los objetos que agrupan.

Microsoft Agent, siguiendo la norma COM, define unas interfaces que permiten a las aplicaciones acceder a los servicios que implementan los diferentes objetos que lo constituyen. A través de estas interfaces una aplicación puede controlar las animaciones de un personaje, los eventos de entrada de un usuario o especificar la salida de audio.

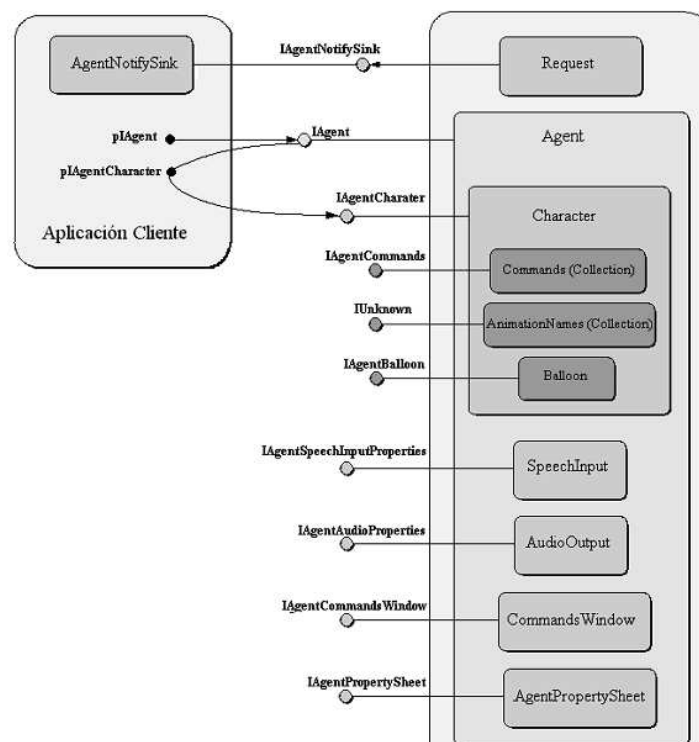


Figura A.11: Jerarquía de Objetos, con sus respectivas Interfaces de programación

El diagrama de la anterior figura muestra la relación entre el Modelo de Objetos y las interfaces de programación. En él se puede observar como la casi totalidad de los objetos del servidor son programables a través de las interfaces que tienden a las aplicaciones clientes.

La excepción la constituye el objeto `Request` que, como ya ha sido dicho, gestiona el mecanismo de generación de eventos y por lo tanto necesita de una implementación especial, así como de la colaboración de otro objeto, `AgentNotifySink`, que se instancia en el cliente y sirve para recoger y manejar los eventos.

Por simplicidad en el diagrama, se ha prescindido de las interfaces extendidas (por ejemplo se incluye la interfaz básica `IAgent` y no la interfaz extendida `IAgentEx`).

En los siguientes apartados se explica en detalle los mecanismos de acceso a las interfaces para programar los objetos así como la gestión de eventos.

#### **A.4.5. Puesta en marcha del Servidor AgentSvr.exe**

Cuando un cliente desea usar los servicios del objeto Agent, debe pedir a COM que localice el servidor de la clase del objeto, AgentSvr.exe. En ese momento, COM pide al servidor que cree el objeto y devuelve al cliente un puntero a la interfaz inicial del objeto. A través del puntero, el cliente puede llamar a los métodos del objeto o puede solicitar punteros a interfaces adicionales mediante la función miembro IUnknown.QueryInterface.

#### **A.4.6. Cargando el personaje y los datos de animación**

Una vez que se tiene el puntero a la interfaz IAgentEx, se puede usar el método Load para cargar un personaje y acceder a su interfaz IAgentCharacterEx. Hay tres diferentes posibilidades para especificar la ruta de directorios de un personaje concreto al método Load. La primera es compatible con Microsoft Agent 1.5, donde la ruta especificada es la ruta completa más el nombre del fichero del personaje. La segunda posibilidad es especificar únicamente el nombre de fichero, en cuyo caso, Agent mira en su directorio de personajes. La última posibilidad es pasarle a la función un parámetro vacío, para que se cargue el personaje por defecto.

Cuando los servicios de Microsoft Agent no se necesiten, como por ejemplo cuando la aplicación que los usa se va a cerrar, hay que liberar sus interfaces. La liberación de la interfaz del personaje no descarga el personaje. Hay que llamar al método Unload para descargarlo antes de liberar la interfaz IAgentEx.

#### **A.4.7. Crear un recolector de eventos (Notification Sink)**

El servidor AgentSvr.exe puede estar atento a la ocurrencia de ciertos acontecimientos, para cuando sucedan generar notificaciones de eventos al cliente. Para acceder a las notificaciones de los eventos que ocurran por Microsoft Agent, se debe crear un recolector de eventos (Notification Sink).

Este recolector consiste en una interfaz de distribución, IAgentNotifySink, a la que el servidor llama cuando lanza un evento. La interfaz proporciona funciones que recogen los eventos pero no los tratan, al ser funciones vacías. Por lo tanto, la aplicación cliente tiene que implementar el código de las funciones correspondientes a los eventos que quiera manejar.

Para poner en marcha el recolector se ha de crear un objeto AgentNotifySink y registrar la interfaz en el servidor, para que éste tenga un punto de entrada (el puntero a la interfaz, pIAgentNotifySink).

Cada vez que el Servidor detecta una acción del usuario merecedora de evento, o ocurre algún acontecimiento en la cola de animaciones de personajes, se crea un objeto Request, el cual generará el evento apropiado. Este objeto conoce cuál es la función de la interfaz IAgentNotifySink que recoge la información sobre ese evento, por lo que llama a la función IAgentNotifySink.Invoke() con el identificador de la función recolectora, ID REQUEST COMPLETE por ejemplo, y los parámetros apropiados (en el caso de RequestComplete, el identificador de petición de ejecución de una animación y un parámetro de estado)

La interfaz de distribución habrá implementado la función que maneje ese evento (IAgentNotifySink.RequestComplete) y gestionará la información aportada por el servidor de la forma que el programador de la aplicación cliente considere oportuna.



#### **A.4.8. Servicios de Animación**

Los servicios de animación de Microsoft Agent manejan la animación y la translación de la imagen de un personaje dentro de su propia ventana sobre la pantalla.

Cuando un cliente lo solicita, el servidor carga y ejecuta el fichero de animación del personaje. Una animación se define como una secuencia de fotogramas, compuesta de una o más imágenes y datos de efectos de sonido. El servidor dibuja la imagen, en una ventana cuyo marco se ajusta a la silueta de la imagen. De esta manera se permite que el personaje aparezca en cualquier lugar de la pantalla independientemente de la ventana de la aplicación cliente. Esta técnica de animación, destaca no sólo por su flexibilidad para mostrar al personaje, sino también porque crea la ilusión de que el personaje está operando dentro del entorno del usuario.

Se pueden definir personajes propios para usar con Microsoft Agent. Para crear las imágenes de las animaciones, cualquier herramienta de edición gráfica que pueda salvar en formato de mapa de bits de Windows (.bmp) puede ser empleada. Después, para ensamblar y compilar en forma de animación las imágenes de un personaje, se debe emplear el Editor de Personajes de Microsoft Agent. Esta herramienta permite definir las animaciones y las propiedades por defecto para el personaje, salvando el resultado en formato compatible con Microsoft Agent.

#### **A.4.9. Cargar un personaje**

Para poder animar un personaje, primero ha de ser cargado. Para ello se llama al método Load. El formato de fichero único (.acs) se usa cuando los datos del personaje y sus animaciones están almacenados localmente. Otros formatos de fichero (.acf, .aca) se deben usar para descargar animaciones individuales desde un servidor HTTP.

Una aplicación de un cliente puede cargar únicamente una instancia del mismo personaje. Cualquier intento de cargar un mismo personaje más de una vez fallará. Sin embargo, una aplicación puede tener múltiples instancias del mismo personaje, si el personaje es cargado en conexiones separadas a Microsoft Agent.

#### **A.4.10. Cargar el Personaje por Defecto**

Como ya ha sido comentado en el apartado de interfaz de usuario, el personaje por defecto puede ser elegido por el usuario a través de la hoja de propiedades, que incluye Microsoft Agent, conocida como la Ventana de Propiedades del Personaje por Defecto. Se puede llamar a esta ventana desde una aplicación mediante el método ShowDefaultCharacterProperties, pero sólo el usuario, mediante interacción directa, es capaz de cambiar su configuración.

La selección de un personaje por defecto, está limitada a aquellos que incluyan un juego mínimo de animaciones estándar. El objetivo de esta restricción es tener un nivel de consistencia entre personajes.

Como el personaje por defecto está pensado para propósito general y su uso puede ser compartido por otras aplicaciones al mismo tiempo, es conveniente evitar cargarlo cuando se quiera un personaje exclusivo para la aplicación.

Para cargar el personaje por defecto, se llama al método Load sin especificar un nombre de fichero o ruta de acceso.

Si el usuario no ha seleccionado un personaje como personaje por defecto, Microsoft Agent selecciona el primer personaje que soporte el juego de animaciones estándar. Si no hay ninguno disponible, el método fallará, señalando la causa en la variable de retorno.

El servidor notifica a los clientes que han cargado el personaje por defecto cuando un usuario lo cambia, pasándole el GUID del nuevo personaje. El servidor automáticamente descarga el primero y recarga el nuevo personaje. Las colas de todo cliente que tuviera cargado el personaje por defecto son paradas y purgadas. Sin embargo, las colas de los clientes que hubieran cargado el personaje explícitamente usando el nombre de fichero del personaje no se ven afectadas.

#### A.4.11. Animando un personaje

Una vez que el personaje es cargado, se pueden usar los métodos de animación de personajes. Típicamente, el primero en usarse es el método Show. El cual hace visible el marco del personaje y ejecuta la animación del estado Showing asignada al personaje, si el primer parámetro es FALSE.

Para ejecutar una animación, una vez que el marco del personaje es visible, se puede usar el método Play especificándole el nombre de una animación

Los nombres de las animaciones son específicos para cada personaje y son asignados durante la definición de éste. Cuando una animación se ejecuta, la forma de la ventana cambia para igualar la imagen en el marco. Esto resulta en una imagen gráfica trasladable, o sprite, mostrada superpuesta a todas las demás ventanas.

El método Speak permite programar al personaje para que hable, sincronizando la salida con el movimiento de los labios. Se darán detalles al respecto en la sección de servicios de salida de este capítulo.

Con MoveTo se traslada al personaje a una nueva posición. Cuando se llama a este método, Microsoft Agent automáticamente ejecuta la animación apropiada según la localización relativa del personaje en la pantalla respecto a la posición donde ha de ir.

La siguiente figura corresponde con un fotograma de la animación resultante de la ejecución del método MoveTo:



Figura A.12: Merlín ejecutando una animación de transición cuando se le pide con el método MoveTo que ocupe una nueva posición en la pantalla.

Igualmente, cuando se llama a `GestureAt`, el personaje señala hacia la posición de pantalla que le pasa como parámetro basándose en la localización del personaje.

Para ocultar el personaje, hay que llamar al método `Hide`. Antes de ocultarse el personaje, pasa automáticamente, como transición, por el estado `Hiding` si el primer parámetro de la función está fijado a `FALSE`.

Microsoft Agent procesa todas las llamadas a animaciones, o peticiones, de forma asíncrona. Esto permite al código de la aplicación continuar manejando otros eventos mientras la petición va siendo procesada. Las nuevas peticiones se colocan al final de una cola, tras las peticiones previas que restan por ejecutarse. Llamadas a posteriores funciones pueden ejecutarse antes de que termine una animación. Por ejemplo una función que vaya tras una llamada a los métodos `Play` o `MoveTo` se ejecutará antes de que la animación termine.

Para sincronizar las animaciones con la ejecución del código, se puede crear un objeto referencia a la petición de animación, y así se puede saber cuándo comienza o finaliza la animación, monitorizando los eventos `Request` que el servidor usa para avisar a sus clientes.

Si, por ejemplo se quiere que aparezca una ventana cuando el personaje termina una animación, esa ventana podría ser lanzada desde una subrutina que espera por el evento `RequestComplete` que corresponda con el identificador de la petición en cuestión.

Cuando un personaje está escondido, el servidor no ejecuta las animaciones que se le solicitan; sin embargo, las encola y procesa las peticiones de animación (como si fueran ejecutadas las animaciones) y devuelve el estado de la petición. Estando escondido, el personaje nunca tiene la entrada activa. Sin embargo, si el usuario dice el nombre del personaje (cuando la entrada de voz está habilitada) el servidor muestra el personaje de forma automática.

Cuando existe un único personaje, cada petición de animación se ejecuta al terminar de ejecutarse la petición previa; Es decir, se ejecutan secuencialmente. Cuando la aplicación cliente carga múltiples personajes al mismo tiempo, los servicios de animación de Microsoft Agent permiten animar los personajes independientemente o usar los métodos `Wait`, `Interrupt`, o `Stop` para sincronizar las animaciones de los personajes

Por ejemplo, si se tienen dos personajes, y se quiere que la petición de animación de un personaje espere hasta que la animación del otro personaje termine, hay que llamar al método `Wait` con el identificador de la petición de la animación del personaje.

Microsoft Agent ejecuta algunas animaciones de forma automática. Por ejemplo si el estado de un personaje no ha cambiado durante varios segundos, el Agente comienza a ejecutar las animaciones de estado ocioso (`Idling`) asignadas al personaje. O cuando está habilitada la entrada de voz, el Agente ejecuta la animación `Listening` y luego, cuando se detecta nivel de audio en el micrófono, el estado `Hearing`. Estas animaciones manejadas por el servidor son llamadas estados, y son definidos cuando un personaje es creado.

Por ejemplo, en la figura podemos ver la animación del estado `Hearing`.



Figura A.13: Estado Hearing.

#### A.4.12. Servicios de Entrada

La aplicación cliente proporciona la primera capa de la interfaz que interacciona con un personaje. Así, se puede programar un personaje para que responda a cualquier forma de entrada, desde apretar un botón o introducir algo por el teclado. Pero además, Microsoft Agent proporciona eventos, para que se pueda programar lo que debe ocurrir, cuando el usuario, hace click o doble click o arrastra al personaje.

#### A.4.13. Cliente con entrada activa

Como muchas aplicaciones cliente pueden compartir el mismo personaje y como múltiples clientes pueden usar diferentes personajes al mismo tiempo, el servidor designa un cliente como el de entrada activa y proporciona entrada de ratón y de voz sólo a la aplicación de ese cliente. Así, se mantiene el correcto manejo de la entrada de usuario, para que el cliente apropiado responda a la entrada.

Cuando el servidor procesa el método Show de un personaje, el cliente de ese personaje pasa a tener la entrada activa. A partir de entonces, es la interacción del usuario (hacer click sobre el personaje, decir el nombre de un personaje...) lo que determina cuál es la aplicación cliente que pasa a tener la entrada activa.

Cuando un personaje es ocultado, su cliente deja de ser el de entrada activa. El servidor automáticamente pasa a considerar como cliente activo a alguno de los restantes personajes con entrada activa. Cuando todos los personajes son ocultados, ningún cliente tiene entrada activa. Sin embargo, si en esta situación, la tecla Listening es presionada el reconocimiento sigue funcionando para el último personaje que fue cliente activo.

Se puede fijar que un cliente esté activo o no activo usando el método Activate. Se puede programar de tal manera que sólo al activar la ventana de aplicación, se le dé entrada activa al personaje mediante el método Activate, para evitar interferir con otros clientes del personaje. Por ejemplo, si el usuario hace click en tu ventana de aplicación, activando tu aplicación, puedes llamar al método Activate para recibir y procesar entrada de voz y ratón dirigido al personaje.

#### A.4.14. Soporte para Menú Contextual

Microsoft Agent incluye un Menú Contextual emergente para cada personaje. El servidor añade automáticamente al menú comandos para abrir la Ventana de Comandos de Voz y para esconder al personaje, así como los nombres de otros comandos de otros clientes del personaje para permitir a los usuarios conmutar entre clientes. El servidor automáticamente añade un separador en el menú entre sus propias entradas y aquellas definidas por el cliente.

El servidor muestra este menú automáticamente cuando un usuario clicca el botón derecho de ratón sobre el personaje. A parte de los comandos que incluye el Servidor por defecto, se pueden añadir comandos al menú para una determinada aplicación cliente mediante la definición de una colección de comandos (Commands).



Figura A.14: Menú Contextual de Comandos

El servidor manda un evento `AgentNotifySink.Command` con el identificador de comando (almacenado en la variable `lComentarTablaID`) cuando el usuario selecciona ese comando del menú mediante el ratón, el teclado o la ventana de comandos de voz. Es entonces en el recolector de eventos (en `Notify.cpp`) dónde se debe enlazar el comando con la funcionalidad que se le quiera dar.

Para quitar comandos del menú, se usa el método `Remove`. Los cambios en el menú sólo entran en funcionamiento cuando el menú se vuelve a dibujar, no mientras se está mostrando.

Si se prefiere proporcionar el servicio de menú emergente más personalizado, se puede usar la propiedad `AutoPopupMenu` para deshabilitar la aparición automática del menú gestionada por el servidor al clicar con el botón derecho.

Se puede usar la notificación del evento `Click` para crear una conducta propia de manejo de menú.

Cuando el usuario selecciona un comando del Menú Emergente o de la Ventana de Comandos de Voz, el servidor lanza el evento `Command` del cliente asociado y devuelve los parámetros de la entrada usando el objeto `UserInput`.

El servidor también proporciona el mismo menú contextual desde el icono del personaje en la barra de tareas. Cuando el personaje es visible y se hace click sobre él con el botón derecho, se muestra el mismo menú, que si se clicca sobre el personaje con el botón derecho del ratón. Si está oculto, sólo apa-

recen en el menú los comandos suministrados por el servidor.

#### **A.4.15. Soporte para entrada de habla**

Para complementar la entrada mediante ratón y teclado, Microsoft Agent incluye soporte directo para entrada de voz.

Aunque los servicios de Microsoft Agent incluyen el soporte para entrada de habla, un motor de reconocimiento de habla aislada, tipo Comando y Control (Command and Control), debe ser instalado en el sistema para poder acceder a los servicios de entrada de habla de Agent y conseguir que los personajes soportados por Agent sean capaces de recibir comandos de voz.

El motor de reconocimiento funciona mediante el método click-to-speak (presionar para hablar). El usuario puede comenzar la entrada de habla tras presionar y mantener sostenida la tecla Listening (ajustable desde las Ventana de Opciones Avanzadas de Personajes). En este modo Listening, si el motor de habla recibe el comienzo de la entrada de habla, mantiene el canal de audio abierto hasta que detecta el fin de la articulación de la voz. Sin embargo, cuando no recibe entrada no se bloquea la salida de audio. Esto posibilita que el usuario lance diversos comandos de voz mientras presiona la tecla, y el personaje puede responder cuando el usuario no esté hablando.

El modo Listening expira una vez que usuario deja de presionar la tecla Listening. El usuario puede ajustar el tiempo de expiración para este modo usando las Opciones Avanzadas de Personajes. Desde el código de la aplicación de cliente no se puede fijar este tiempo de expiración.

Si el personaje intenta hablar mientras el usuario lo está haciendo, la salida audible del personaje fallará, aunque el texto puede ser mostrado en el globo de texto. Si el personaje está usando el canal de audio mientras la tecla Listening es presionada, el servidor automáticamente devuelve control al usuario después de procesar el texto en el método Speak. Una señal MIDI opcional es lanzada para indicar al usuario que puede comenzar a hablar. Esto facilita al usuario el conocer cuándo es su turno de entrada, incluso si la aplicación que maneja el personaje no proporciona correctamente pausas lógicas en su salida.

Se puede usar también el método Listen para iniciar la entrada de habla. Llamando a este método comienza a funcionar el reconocimiento de habla por un periodo de tiempo predefinido. Si no hay entrada durante ese intervalo, Microsoft Agent desactiva el reconocimiento automáticamente y libera el canal de audio. Esto evita bloquear la entrada o la salida del dispositivo de audio y minimiza la carga que supone al procesador tener el motor de reconocimiento en funcionamiento. También se puede usar el método Listen para apagar la entrada de habla. Sin embargo, hay que tener cuidado porque al no funcionar la máquina de reconocimiento de habla de manera asíncrona, el efecto puede no ser inmediato. Como resultado es posible recibir un evento Command incluso después de que el código llamó a Listen para desactivar la entrada de voz.

Para soportar entrada de voz hay que definir una gramática, el conjunto de palabras que se quieren usar para que el motor de reconocimiento equipare lo que ha capturado con lo que tiene definido para cada comando (Command) de la colección de comandos.

Tanto si un usuario presiona la tecla Listening o si la aplicación cliente llama al método Listen para iniciar la entrada de voz, el motor de reconocimiento intenta casar lo que ha capturado con la gramática de los comandos que han sido definidos, y devuelve la información al servidor. Entonces el servidor

notifica a la aplicación de cliente usando el evento `Command` (`IAgentNotifySink::Command`), que indica el identificador del comando con mayor probabilidad. Mediante esta función se obtiene el objeto `UserInput` que incluye el identificador del comando que más probabilidad tiene de haber sido el que el usuario ha pronunciado y la de las dos alternativas siguientes (si las hay), una puntuación de confianza y el texto del vocablo entendido y el de cada una de las alternativas.

Cuando se entra en el modo `Listening`, el servidor automáticamente ejecuta la animación que el personaje tiene asignada para este estado. Luego, cuando detecta señal en el micrófono, ejecuta la animación del estado de escucha (estado `Hearing`). El servidor mantendrá al personaje en estado de atención hasta que detecte el fin de la entrada de micrófono. Esto proporciona la realimentación social apropiada indicándole al usuario que está siendo escuchado.

Si el usuario deshabilita la entrada de voz en la ventana de Opciones Avanzadas de Personajes, la tecla del estado de escucha permanecerá deshabilitada también. Igualmente llamar al método `Listen` cuando la entrada de voz está deshabilitada provoca que el método falle.

#### A.4.16. Elección del motor de habla

El fijar el identificador (ID) de lenguaje para un personaje determina su lenguaje por defecto para entrada de habla. Microsoft Agent requiere que para ese lenguaje se tenga un motor de habla instalado que cumpla con el estándar SAPI. Si la aplicación de un cliente no especifica una preferencia de lenguaje, el Agente intentará encontrar un motor de reconocimiento de habla que iguale el identificador del lenguaje por defecto del usuario. Si no hubiera motor disponible que igualara este lenguaje, el habla queda deshabilitada para ese personaje.

Se puede requerir también un motor de reconocimiento de habla especificando su identificador de modo (usando la propiedad `SRModeID` del personaje). Sin embargo, si el identificador de lenguaje para ese identificador de modo no es igual a la configuración del lenguaje del cliente, la llamada fallará. El motor de reconocimiento del habla permanecerá como el último motor satisfactoriamente fijado por el cliente, o si no fue ninguno, el motor que iguale al actual identificador del lenguaje del sistema. Si esto último tampoco es satisfactorio, la entrada de habla no estará disponible para ese cliente.

Microsoft Agent carga automáticamente un motor de reconocimiento del habla cuando la entrada de habla es iniciada al presionar la tecla `Listening` o cuando el cliente con entrada activa llama al método `Listen`. Sin embargo, un motor puede ser cargado también cuando se fija o se pregunta por el identificador de modo, fijando o preguntando por las propiedades de la Ventana de Comandos de Voz, preguntado por su estado (`SRStatus`), o cuando el habla está habilitada y el usuario muestra la página de Entrada de Habla de las Opciones Avanzadas de Personajes. Sin embargo, sólo se mantienen cargados los motores de habla que estén siendo usados por los clientes.

#### A.4.17. Eventos de entrada de habla

Además de la notificación de los eventos de comando, el agente también notifica al cliente con entrada activa cuando el servidor enciende o apaga el modo de escucha (estado `Listening`), usando los métodos `ListenStart` y `ListenComplete` (`IAgentNotifySinkEx.ListeningState`). Sin embargo, si el usuario presiona la tecla del modo de escucha y no hay motor reconocimiento de habla disponible para el personaje cliente con entrada activa, el servidor no genera un evento `ListenStart`.

#### **A.4.18. Ventana de Comandos de Voz**

La Ventana de Comandos de Voz muestra los comandos de voz activos disponibles para el personaje. La ventana aparece cuando el comando Abrir Ventana de Comandos es elegido en el Menú Emergente o la propiedad visible del objeto ComandsWindow es fijada a True.

Si el motor de habla no ha sido cargado todavía, preguntar por esta propiedad provocará que Microsoft Agent intente inicializar el motor. Si el usuario deshabilita el habla, la ventana aún se muestra; sin embargo, incluirá un mensaje de texto que informe al usuario que el habla está actualmente deshabilitada.

#### **A.4.19. Ventana de Opciones Avanzadas de Personajes**

La ventana de Opciones Avanzadas de Personajes se emplea para que los usuarios configuren las características comunes para la interacción con todos los personajes. Por ejemplo, los usuarios pueden deshabilitar la entrada de voz o cambiar los parámetros de entrada. Pueden también configurar la salida a través del globo de texto. Estos cambios están por encima de cualquier cambio fijado por una aplicación cliente o fijados como parte de la definición de un personaje. Una aplicación no puede cambiar o deshabilitar estas opciones, porque se aplican a las preferencias de usuario generales para la operación con todos los personajes. Sin embargo, el servidor notificará a cada aplicación activa, mediante el evento AgentNotifySink.DefaultCharacterChange, los cambios realizados por el usuario.

#### **A.4.20. Servicios de Salida**

Además de soportar las animaciones de un personaje, Microsoft Agent soporta salida de audio para el personaje. Esto incluye salida de voz y efectos de sonido. Para la salida de voz, el servidor automáticamente sincroniza las imágenes del movimiento de la boca con la salida. Se puede elegir entre sintetizar texto a voz (TTS), utilizar un sonido grabado o solamente salida mediante texto escrito en el globo de texto.

#### **A.4.21. Soporte para síntesis de voz**

Igualmente, si se quiere usar los servicios de habla de Microsoft Agent para que un personaje tenga salida de voz sintética, se debe instalar un motor texto a voz (TTS) compatible. Como los servicios de habla de Microsoft Agent están basados en el API de habla de Microsoft (Microsoft Speech API, SAPI), se deben usar motores que sean compatibles con dichas interfaces de habla.

Si usa síntesis de voz, el personaje tiene la habilidad de decir casi cualquier cosa, lo cual ofrece una mayor flexibilidad. Por otra parte, con audio grabado se puede dar al personaje una voz específica que lo haga único. Para especificar la salida, hay que pasar el texto como parámetro al método Speak.

Para habilitar la salida de texto a voz, hay que usar el método Speak. Microsoft Agent intentará automáticamente cargar el primer motor que iguale el identificador del lenguaje del personaje. Si la definición del personaje incluye un identificador de modo de un motor TTS específico, el motor está disponible, y además, iguala el identificador del lenguaje del personaje, Agent cargará el motor para ese personaje. Si no, Agent cargará el primer motor TTS que la interfaz de habla reconozca como válido para ese idioma. Se puede usar IAgentCharacterEx.SetTTSModeID para cargar un motor específico.

#### **A.4.22. Soporte para salida de audio grabado**

Microsoft Agent permite usar ficheros de audio simulando la salida de voz de un personaje. Para ello, se deben grabar previamente los ficheros de audio para poder reproducirlos durante el desarrollo



de la aplicación mediante el método `Speak`.

Los servicios de animación de Microsoft Agent soportan automáticamente sincronización entre el sonido y el movimiento de labios. Si queremos una sincronización mejorada, que incluya información sobre fonemas y pausas entre palabras, hay que usar un formato de fichero de sonido especial que Microsoft Agent es capaz de entender. Este formato se puede generar con una herramienta de Edición de Sonido con Información Lingüística de Microsoft.

#### A.4.23. Soporte para Globo de Texto

Cuando usamos el método `Speak`, la voz de salida puede aparecer también como salida de texto dentro de un globo, similar al típico bocadillo de los cómics. Esto puede ser empleado como algo que complementa o como una alternativa a la salida de voz si ésta no está disponible.

El método `Think` no produce salida de voz y el globo que encierra el texto tiene misma forma que se emplea en los cómics cuando el personaje está pensando.



Figura A.15: Método `Think` aplicado al personaje Merlín

Los Globos de Texto soportan tan sólo comunicaciones desde el personaje, no entradas del usuario. Por lo tanto el Globo de Texto no soporta controles de entrada. Sin embargo, se puede proporcionar entrada para el usuario a través de interfaces desde tu propio lenguaje de programación o desde otros servicios de entrada proporcionados por Microsoft Agent, tales como el menú emergente.

Cuando se define un personaje, se puede especificar si se quiere incluir soporte para Globo de Texto. Sin embargo, si se usa un personaje que incluya soporte para Globo de Texto, no se puede deshabilitar el soporte.

Se puede acceder a la interfaz del Globo de Texto para modificar sus características de presentación y personalizarlas.

#### A.4.24. Efectos de Sonido de las Animaciones

Microsoft Agent permite incluir efectos de sonido como parte de la animación de un personaje. Usando el Editor de Personajes de Microsoft Agent, se puede especificar el nombre de fichero de un

fichero de sonido estándar de Windows (.wav) que haya que incluir sobre un fotograma de una determinada animación.

Los efectos de sonido y la voz no pueden superponerse, así, la salida de voz no comienza hasta que el efecto de sonido haya terminado. Por lo tanto hay que evitar sonidos largos, o en bucle, como parte de una animación.

## Apéndice B

# Panoramica General .NET

### B.1. Plataforma.NET

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el software en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados. Ésta es la llamada plataforma .NET, y a los servicios antes comentados se les denomina servicios Web.

Para crear aplicaciones para la plataforma .NET, tanto servicios Web como aplicaciones tradicionales (aplicaciones de consola, aplicaciones de ventanas, servicios de Windows NT, etc.), Microsoft ha publicado el denominado kit de desarrollo de software conocido como .NET Framework SDK, que incluye las herramientas necesarias tanto para su desarrollo como para su distribución y ejecución, y Visual Studio.NET, que permite hacer todo lo anterior desde una interfaz visual basada en ventanas.

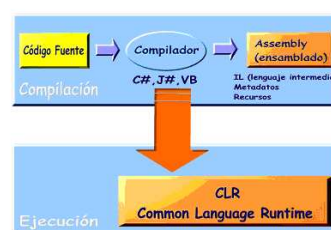


Figura B.1: NET Framework gráficamente

Permite ejecutar software en cualquier lenguaje sobre cualquier dispositivo. Internet puede hacer los negocios más eficientes y proporcionar servicios a los consumidores mediante el concepto y modelo de programación Servicio Web XML.

Propone nuevas formas de interactuar con Pcs (uso de la voz, reconocimiento de escritura...)

Lanzando una nueva plataforma software con Internet como sustrato esencial, con acceso a la información desde cualquier sitio y con gran variedad de dispositivos.

- Nuevo nivel: .NET Framework para compartir características y niveles de abstracción ya que se incluyen cambios de formato de los ejecutables, cambios de compiladores y su filosofía de trabajo, y cambio de la biblioteca de clases básicas.
- Nuevos dispositivos: teléfonos, PDAs, Tablet Pcs
- Entorno de hospedaje de Servicios Web personales: Servicios Web básicos como autenticación y almacenamiento de datos. Suscripción a software como Servicio a través de .NET myServices.

Frente a la situación actual:

- Lenguajes de programación y compiladores frente a archivos fuente binarios
- Ejecutables y enlace dinámico en DLLs.
- Modelos de componentes y encapsulamiento con tipos, interfaces, clases y objetos
- Aplicaciones distribuidas, Arquitecturas cliente/servidor en tres niveles
- Internet: Páginas activas, Middleware, seguridad, transacciones, atributos, Máquinas virtuales, interpretación y librerías de abstracción.

Aunque con algunos problemas como:

- IDLs y Librerías de Tipos complejos, se separa el interfaz de la implementación (ejecutable)
- Cada entorno de desarrollo debe implementar costosos mecanismos de infraestructura con factoría de clases y de interfaces.
- Control explícito del flujo de ejecución

En cuanto a Windows:

- Visual C++...(Visual Studio), punteros y API Win32
- Algunas incompatibilidades de tipos entre String y char array.
- Reutilización de implementación: herencia en un solo nivel.
- Distintas versiones de las DLLs (Infierno de las DLLs).
- Complejidad de instalación.
- Construir seguridad implícitamente sobre el sistema.

Soluciones:

- Compartir características y niveles de abstracción usando .NET Framework
- Múltiples lenguajes compilados

- Servicios en el desarrollo y ejecución de código, usando una nueva máquina virtual multilenguaje (Common Language Runtime CLR)
- Lenguaje Intermedio (MSIL)
- Espacios de nombres (Namespaces), librerías de clases base y tipos unificados
- Metadatos y ensamblados (assemblies)
- Modelos de programación ASP.NET para formularios Web y servicios Web XML
- En resumen simplificar y unificar.



Figura B.2: Desarrollo de software con .NET

El .Net Framework es un componente integral de Windows que proporciona apoyo para construir y ejecutar la nueva generación de aplicaciones y servicios Web XML. Está diseñado para cumplir los siguientes objetivos:

- Proveer de un contexto consistente de programación orientada a objetos tanto si el objeto se almacena y ejecuta localmente, se ejecuta localmente pero se distribuye vía Internet o se ejecuta remotamente.
- Proveer un contexto de ejecución de código que minimiza la implementación de software y los conflictos entre versiones.
- Proveer de un contexto de ejecución de código seguro.
- Para hacer la experiencia del desarrollador coherente variando los tipos de aplicaciones entre aplicaciones basadas en Windows y aplicaciones basadas en Web.
- Construir toda comunicación dentro de normas estándar para asegurar que el código basado en .NET Framework puede integrarse con cualquier otro código.

Net Framework tiene dos componentes principales: el Common Language Runtime (CLR) y .Net Framework Class Library o biblioteca de clase base (BCL).

El CLR maneja código en el tiempo de ejecución, gestión de memoria, de hilo, y remoting, al también implementar seguridad estricta de tipo, promueve seguridad y robustez. De hecho, el concepto de

gestión de código es un principio básico del CLR. El código que apunta al CLR es conocido como código administrado, mientras código que no le apunta es conocido como código no administrado. La biblioteca de clase, el otro componente principal de .NET Framework, es una colección orientada a objetos de tipos reusables para desarrollar aplicaciones tanto aplicaciones de interfaz de usuario de línea de comando o gráficas (la Interfaz Gráfica del Usuario GUI) para las aplicaciones se basa en las últimas innovaciones proporcionadas por ASP.NET, como Web Forms y los servicios de Web XML.

## B.2. El lenguaje C#

C# (leído en inglés 'C Sharp' y en español 'C Almohadilla') es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg, éste último también conocido por haber sido el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi.

Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el lenguaje nativo de .NET

La sintaxis y estructuración de C# es muy parecida a la de C++ o Java, puesto que la intención de Microsoft es facilitar la migración de códigos escritos en estos lenguajes a C# y facilitar su aprendizaje a los desarrolladores habituados a ellos. Sin embargo, su sencillez y el alto nivel de productividad son comparables con los de Visual Basic. En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la BCL usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el .NET Framework SDK

A continuación se recoge de manera resumida las principales características de C# Algunas de las características aquí señaladas no son exactamente propias del lenguaje sino de la plataforma .NET en general, y si aquí se comentan es porque tienen una repercusión directa en el lenguaje:

- Sencillez: C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:

- El código escrito en C# es autocontenido, lo que significa que no necesita de ficheros adicionales al propio fuente tales como ficheros de cabecera o ficheros IDL

- El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile lo que facilita la portabilidad del código.

- No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (.) acceder a miembros de espacios de nombres.

- Modernidad: C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción foreach que

permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico string para representar cadenas o la distinción de un tipo bool específico para representar valores lógicos.

-Orientación a objetos: Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo.

En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores public, private y protected, C# añade un cuarto modificador llamado internal, que puede combinarse con protected e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.

Respecto a la herencia C# sólo admite herencia simple de clases ya que la múltiple provoca más quebraderos de cabeza que facilidades y en la mayoría de los casos su utilidad puede ser simulada con facilidad mediante herencia múltiple de interfaces. De todos modos, esto vuelve a ser más bien una característica propia del CTS que de C#.

Por otro lado y a diferencia de Java, en C# se ha optado por hacer que todos los métodos sean por defecto sellados y que los redefinibles hayan de marcarse con el modificador virtual, lo que permite evitar errores derivados de redefiniciones accidentales. Además, un efecto secundario de esto es que las llamadas a los métodos serán más eficientes por defecto al no tenerse que buscar en la tabla de funciones virtuales la implementación de los mismos a la que se ha de llamar. Otro efecto secundario es que permite que las llamadas a los métodos virtuales se puedan hacer más eficientemente al contribuir a que el tamaño de dicha tabla se reduzca.

-Orientación a componentes: La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros)

-Gestión automática de memoria: Todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active -ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente-, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción using.

-Seguridad de tipos: C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:

-Sólo se admiten conversiones entre tipos compatibles. Esto es, entre un tipo y antecesores suyos,

entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacenase una referencia del segundo (downcasting) Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución el CLR y no el compilador, por lo que en realidad el CLR y el compilador colaboran para asegurar la corrección de las conversiones.

-No se pueden usar variables no inicializadas. El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del flujo de control del fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.

-Se comprueba que todo acceso a los elementos de una tabla se realice con índices que se encuentren dentro del rango de la misma.

-Se puede controlar la producción de desbordamientos en operaciones aritméticas, informándose de ello con una excepción cuando ocurra.

-C# incluye delegados, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se comprueba que los métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.

-Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C# siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

-Instrucciones seguras: Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=); y todo caso de un switch ha de terminar en un break o goto que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.

-Sistema de tipos unificado: En C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán "objetos")

El hecho de que todos los tipos del lenguaje deriven de una clase común facilita enormemente el diseño de colecciones genéricas que puedan almacenar objetos de cualquier tipo.

-Extensibilidad de tipos básicos: C# permite definir, a través de estructuras, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en pila (luego su creación, destrucción y acceso serán más rápidos) y se asignen por valor y no por referencia. Para conseguir que lo último no tenga efectos negativos al pasar estructuras como parámetros de métodos, se da la posibilidad de pasar referencias a pila a través del modificador de parámetro ref.

-Extensibilidad de operadores: Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, C# permite redefinir el significado de la mayoría de los operadores -incluidos los de conversión, tanto para conversiones implícitas como explícitas- cuando se apliquen a diferentes tipos de objetos.

Las redefiniciones de operadores se hacen de manera inteligente, de modo que a partir de una única



definición de los operadores ++ y – el compilador puede deducir automáticamente como ejecutarlos de manera prefijas y postifja; y definiendo operadores simples (como +), el compilador deduce cómo aplicar su versión de asignación compuesta (+=) Además, para asegurar la consistencia, el compilador vigila que los operadores con opuesto siempre se redefinan por parejas (por ejemplo, si se redefine ==, también hay que redefinir !=).

También se da la posibilidad, a través del concepto de indizador, de redefinir el significado del operador [] para los tipos de dato definidos por el usuario, con lo que se consigue que se pueda acceder al mismo como si fuese una tabla. Esto es muy útil para trabajar con tipos que actúen como colecciones de objetos.

-Extensibilidad de modificadores: C# ofrece, a través del concepto de atributos, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente información adicional a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la librería de reflexión de .NET . Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores.

-Versionable: C# incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.

Si una clase introduce un nuevo método cuyas redefiniciones deban seguir la regla de llamar a la versión de su padre en algún punto de su código, difícilmente seguirían esta regla miembros de su misma signatura definidos en clases hijas previamente a la definición del mismo en la clase padre; o si introduce un nuevo campo con el mismo nombre que algún método de una clase hija, la clase hija dejará de funcionar. Para evitar que esto ocurra, en C# se toman dos medidas:

-Se obliga a que toda redefinición deba incluir el modificador override, con lo que la versión de la clase hija nunca sería considerada como una redefinición de la versión de miembro en la clase padre ya que no incluiría override. Para evitar que por accidente un programador incluya este modificador, sólo se permite incluirlo en miembros que tengan la misma signatura que miembros marcados como redefinibles mediante el modificador virtual. Así además se evita el error tan frecuente en Java de creerse haber redefinido un miembro, pues si el miembro con override no existe en la clase padre se producirá un error de compilación.

-Si no se considera redefinición, entonces se considera que lo que se desea es ocultar el método de la clase padre, de modo que para la clase hija sea como si nunca hubiese existido. El compilador avisará de esta decisión a través de un mensaje de aviso que puede suprimirse incluyendo el modificador new en la definición del miembro en la clase hija para así indicarle explícitamente la intención de ocultación.

-Eficiente: En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. En C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes.

-Compatible: Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados Platform Invocation Services (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32. Nótese que la capacidad de usar punteros en código inseguro permite que se pueda acceder con facilidad a este tipo de funciones, ya que éstas muchas veces

esperan recibir o devuelven punteros.

También es posible acceder desde código escrito en C# a objetos COM. Para facilitar esto, el .NET Framework SDK incluye una herramientas llamadas *tlbimp* y *regasm* mediante las que es posible generar automáticamente clases proxy que permitan, respectivamente, usar objetos COM desde .NET como si de objetos .NET se tratase y registrar objetos .NET para su uso desde COM.

### B.3. Visual Studio.NET 2003

Visual Studio.Net 2003 es una completa herramienta para crear aplicaciones basadas en Microsoft.NET para Microsoft Windows, Web y dispositivos con compatibilidad nativa con Microsoft.NET Compact Framework y compatibilidad inherente con servicios Web XML.

Sus principales características son:

- Variedad de lenguajes

Eficaces e interoperables como Visual Basic.NET, Visual C++.NET, Visual C#.NET y Visual J#.NET

- Software profesional para Windows, Web y dispositivos

El diseño visual de formularios agiliza la creación de aplicaciones de escritorio completas para Windows, aplicaciones Web y aplicaciones para gran variedad de dispositivos.

- Rápido desarrollo para los niveles de servidor y datos

El diseñador de componentes y el explorador de servidores trabajan unidos para permitir la composición visual de componentes lógicos empresariales de nivel medio.

ADO.NET y Visual Database Tools permiten crear software profesional controlados por datos.

- Implementación y mantenimiento de aplicaciones simplificados

La implementación "sin tocar" permite distribuir aplicaciones basadas en Windows con la facilidad de las aplicaciones Web, mientras que la implementación de aplicaciones en paralelo reduce los problemas de versiones de las DLL. La compatibilidad integrada con la tecnología Windows Installer proporciona opciones avanzadas para crear paquetes de implementación para Windows y Web.

- Confiabilidad y seguridad

Basado en la plataforma probada de .NET Framework, Visual Studio.NET utiliza una directiva de seguridad detallada para modelos de seguridad de acceso del código, basados en funciones y en usuarios.

- Compatibilidad con versiones existentes

La actualización sin problemas de Visual Studio.NET 2002, la interoperabilidad con software existente basado en COM y la tecnología mejorada de actualización de Visual Basic garantizan el aprovechamiento de versiones existentes de proyectos.

- Desarrollo para dispositivos inteligentes

La compatibilidad nativa con .NET Compact Framework permite desarrollar, depurar e implementar automáticamente aplicaciones en dispositivos inteligentes, incluidos dispositivos que utilicen Microsoft Windows CE .NET y Pocket PC. Un sólido emulador garantiza el desarrollo rápido y preciso de aplicaciones para dispositivos móviles sin la necesidad de tener el dispositivo.

- Desarrollo de aplicaciones Web para dispositivos móviles

La compatibilidad con dispositivos inalámbricos permite ampliar fácilmente aplicaciones Web nuevas o existentes. Los controles de ASP.NET para dispositivos móviles representan de manera inteligente una amplia gama de dispositivos, liberando así a los programadores de preocupaciones sobre las funciones exclusivas de cada dispositivo.

El proyecto que nos ocupa ha sido desarrollado con Visual Studio.NET 2003 y Microsoft .NET Framework SDK 1.1 aunque actualmente ya existen en el mercado nuevas versiones publicadas por Microsoft Visual Studio 2005 Beta 2 y .NET Framework 2.0 Beta 2.

Visual C#.NET forma parte de Visual Studio.NET 2003, es una herramienta y un lenguaje moderno e innovador que permite generar software conectado a .NET para Microsoft Windows, Web y una amplia gama de servicios. A pesar de que el paquete ofrece una amplia gama de posibilidades y lenguajes, únicamente nos centraremos en proporcionar una breve explicación de las utilidades que hemos usado para la realización de la aplicación.

No pretende ser un manual exhaustivo sino una breve explicación de las herramientas mas comunes que hemos utilizado para la realización de la aplicación.

- Empezar a trabajar con Visual Studio

Una pantalla similar a la mostrada en la figura se presenta al arrancar Visual Studio:

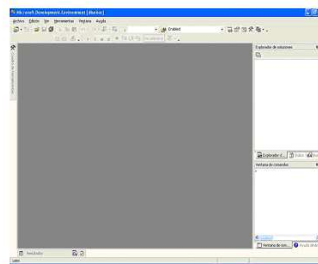


Figura B.3: Pantalla Inicial de Visual Studio.NET

Elijiendo la opción de menú Archivo Nuevo Proyecto se presenta la pantalla siguiente:

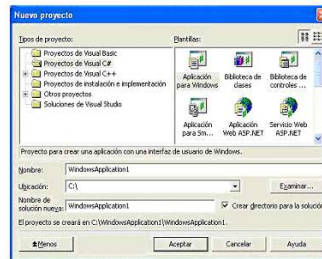


Figura B.4: Pantalla Nuevo Proyecto

En esta pantalla de las posibles opciones elegiremos Aplicación para Windows, destacar que podemos elegir nombre y ubicación para nuestro proyecto, al pulsar en Aceptar se mostrará la pantalla siguiente, con la que podemos empezar a trabajar:

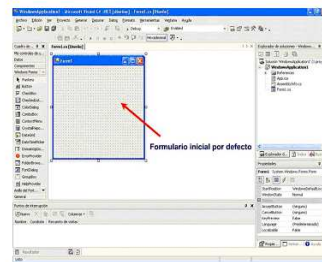


Figura B.5: Pantalla de Aplicación Windows Form

### ¿Qué son los formularios de Windows?

Los formularios de Windows constituyen un marco de trabajo para la creación de aplicaciones cliente de Windows que utilizan Common Language Runtime. Las aplicaciones de los formularios de Windows se pueden escribir en cualquiera de los idiomas compatibles con Common Language Runtime. Algunas de las ventajas de utilizar los formularios de Windows son las siguientes:

- Simplicidad y potencia: Los formularios de Windows constituyen un modelo de programación para el desarrollo de aplicaciones de Windows que combinan la simplicidad del modelo de programación de Visual Basic 6.0 con la potencia y la flexibilidad de Common Language Runtime.
- Menor coste total de propiedad: Los formularios de Windows se aprovechan de las características de implementación y de versión de Common Language Runtime con el fin de ofrecer costes de imple-

mentación reducidos y mayor solidez de las aplicaciones. Esto reduce significativamente los costes de mantenimiento (TCO) de aquellas aplicaciones escritas en los formularios de Windows.

- **Arquitectura de los controles:** Los formularios de Windows ofrecen una arquitectura para los controles y los contenedores de controles de acuerdo con una implementación específica de las clases contenedoras y de los controles. Esto reduce significativamente las cuestiones de interoperabilidad entre los contenedores y los controles.

- **Seguridad:** Los formularios de Windows se aprovechan por completo de las características de seguridad de Common Language Runtime. Esto significa que se pueden utilizar los formularios de Windows para implementar todo desde un control que no es de confianza, que se ejecuta en el explorador, a una aplicación de plena confianza instalada en el disco duro del usuario.

- **Compatibilidad con los servicios Web de XML:** Los formularios de Windows ofrecen una completa compatibilidad para establecer conexiones con servicios Web de XML de forma rápida y sencilla.

- **Gráficos con formato enriquecido:** Los formularios de Windows constituyen el primer vehículo de transmisión de GDI+, una nueva versión de la interfaz de dispositivo gráfico (GDI) de Windows que es compatible con la mezcla alfa, los pinceles de textura, las transformaciones avanzadas y el soporte de texto enriquecido, entre otros.

- **Controles flexibles:** Los formularios de Windows ofrecen un conjunto de controles enriquecido que engloba a todos los controles ofrecidos por Windows. Estos controles también ofrecen nuevas características, como estilos de "aspecto liso" para botones, botones de radio y casillas de verificación.

- **Conocimiento de los datos:** Los formularios de Windows ofrecen plena compatibilidad con el modelo de datos de ADO.NET.

- **Compatibilidad con controles ActiveX:** Los formularios de Windows ofrecen plena compatibilidad con los controles ActiveX. Es posible albergar fácilmente a controles ActiveX en una aplicación de formularios de Windows. También es posible albergar a controles de formularios de Windows como si se tratara de controles ActiveX.

- **Licencia:** Los formularios de Windows se aprovechan del modelo de licencia mejorado de Common Language Runtime.

- **Impresión:** Los formularios de Windows ofrecen un marco de trabajo de impresión que facilita que las aplicaciones proporcionen informes completos.

- **Accesibilidad:** Los controles de los formularios de Windows implementan interfaces definidas por Microsoft Active Accessibility (MSAA), lo que simplifica la creación de aplicaciones que sean compatibles con las ayudas de accesibilidad, como lectores de pantalla.

- **Compatibilidad en tiempo de diseño:** Los formularios de Windows se aprovechan plenamente de las características de los modelos de componentes y metadatos ofrecidas por Common Language Runtime, con el fin de proporcionar compatibilidad mediante el tiempo de diseño tanto a usuarios de controles como a implementadores de controles.

- **Eventos**

El modelo de programación de los formularios de Windows está basado en eventos. Cuando un control cambia de estado, así como cuando un usuario hace clic en un botón, se provoca un evento. Para

controlar un evento, la aplicación en cuestión registra un método de control de eventos para ese evento.

Se llama a un método de control de eventos sólo cuando tiene lugar un evento específico de un control determinado. Esto permite evitar tener un solo método en el formulario que controle todos los eventos de todos los controles. Esta característica también hace que el código sea más fácil de entender y mantener. Además, debido a que la arquitectura de eventos de los formularios de Windows está basada en delegados, los métodos de control de eventos tienen seguridad de tipos y se pueden declarar como privados. Esta capacidad hace posible que el compilador pueda detectar la falta de coincidencia en la firma del método durante la compilación. También mantiene la interfaz pública de la clase Form despejada de métodos de control de eventos públicos. Clases de eventos Cada evento tiene dos clases compatibles:

- Clase de delegado EventHandler utilizada para registrar el método de control de eventos. Firma de EventHandler que dicta la firma del método de control de eventos.

- Clase EventArgs que contiene datos acerca del evento provocado.

La firma de un EventHandler consiste en que el primer argumento contiene una referencia al objeto que provocó el evento (emisor) y en que el segundo argumento contiene datos acerca del evento (instancia de EventArgs). Por ejemplo, el evento Click de un Button utiliza el siguiente controlador de eventos.

Existen varios eventos que utilizan clases EventHandler e EventArgs genéricas. Sin embargo, algunos eventos necesitan información adicional que es específica del tipo del evento provocado. Por ejemplo, los eventos relacionados con el movimiento del mouse (ratón) incluyen información acerca de la posición del puntero del ratón y de los botones del ratón. Estos eventos definen sus propias clases que se deben heredar de las clases EventHandler y EventArgs. Por ejemplo, el eventoMouseDown utiliza las clases MouseEventArgs y MouseEventArgs.

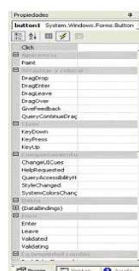


Figura B.6: Ventana Propiedades del Control

- Duración determinada y eliminación

El modelo de clase de .NET Framework proporciona el método Dispose de la clase Component. Se llama al método Dispose cuando un componente determinado deja de ser necesario. Por ejemplo, los formularios de Windows llaman al método Dispose que se encuentra en un formulario y a todos los controles que se encuentran en ese formulario, cuando se cierra el formulario en cuestión. Normalmente, se

utiliza Dispose para liberar grandes recursos de manera puntual y para quitar referencias a otros objetos, de manera que se puedan recuperar gracias al recolector de elementos no utilizados. También se le suele llamar para detener cualquier lógica de programa en ejecución asociada al formulario. Debería mantenerse el código del método Dispose tan sencillo y estable como fuera posible. Si se produce un error en el método Dispose, es probable que no se puedan liberar de la memoria los recursos más grandes.

-Formularios de Windows y gráficos

Common Language Runtime aprovecha la versión avanzada de la interfaz de dispositivo gráfico (GDI) de Windows denominada GDI+. GDI+ está diseñada para proporcionar un alto rendimiento y facilidad de uso. Admite gráficos en 2-D, tipografía e imágenes. Las clases de GDI+ residen en los espacios de nombres System.Drawing, System.Drawing.Drawing2D, System.Drawing.Imaging y System.Drawing.Text. Los espacios de nombres se encuentran en el System.Drawing.DLL de ensamblado. La clase Graphics representa una superficie de dibujo de GDI+. Para utilizar GDI+, primero es necesaria una referencia a un objeto Graphics. Normalmente, se obtiene una referencia a un objeto Graphics en el evento Paint de un control o formulario, o en el evento PrintPage de un PrintDocument.

## B.4. XML

XML (Lenguaje de marcado extensible), es un lenguaje utilizado para escribir datos. El objetivo de XML consiste en proporcionar un formato estándar que puedan leer, procesar y escribir diferentes aplicaciones que se ejecutan en hardware diferente. XML ha adquirido importancia creciente como formato estándar para el intercambio de datos.

**Objetivos de XML** En 1996, el Word Wide Web Consortium (conocido como W3C) emprendió la tarea de diseñar un formato de datos estándar. Perseguía varios objetivos. Uno de ellos era que este formato debía ser capaz de representar cualquier forma de datos estructurados. Esto significa que, además de especificar los propios datos, XML debía indicar también cómo se organizan de forma clara y sin ambigüedades. Otro de los objetivos consistía en garantizar que el formato fuera portable y utilizable en Internet. Esto era importante debido a que podrían existir multitud de aplicaciones con la necesidad de procesar datos, ubicadas en cualquier parte. Además el formato debía hacer que los desarrolladores les resultara tan sencillo como fuera posible escribir programas que consumieran y produjeran XML, ya que si fuera complejo nadie desearía usarlo.

**Estructura de XML** Para cumplir los requisitos de portabilidad y apertura, XML utiliza texto normal para representar datos incrustados en etiquetas que describen la estructura de los datos. Las etiquetas XML son los elementos encerrados entre paréntesis angulares (<, >). El primer elemento (<?xml version="1.0" que aparecerá en nuestro fichero XML indica la versión de XML a la que es conforme el documento, el resto del documento contiene los datos.

Un documento XML válido y bien formado debe ajustarse a una serie de criterios. Sólo debe existir un único elemento raíz actuando de contenedor para todos los datos, el cual contiene los elementos individuales como subelementos anidados. Para cada elemento "inicio«identificador», debe existir el correspondiente elemento fin denotado como </identificador> indica el final de los datos correspondientes a ese elemento. El sangrado y los saltos de línea no son significativos en XML pero se pueden utilizar para leer el documento con mayor facilidad.

**Esquemas XML** Un esquema describe la estructura de un documento XML y puede incluir información adicional como reglas y comprobaciones de validación. Por ejemplo una regla que prohíba valores negativos en un determinado ítem. Para que dos aplicaciones puedan utilizar el mismo archivo XML debe utilizarse el mismo esquema. Muchas organizaciones están desarrollando esquemas estándar para describir requisitos de datos comunes para diversas industrias, permitiendo la libre interoperatividad de las aplicaciones que se basan en dichos esquemas.

**API de XML y .NET Framework** Anteriormente se ha indicado que uno de los objetivos del W3C (Word Wide Web Consortium) consistía en diseñar un formato XML, que fuera fácil de programar y utilizar. Se han desarrollado diversas API para manejar XML; las dos más comunes son el Modelo de objetos de documentos (DOM, Document Object Model) y la API simple para XML (SAX, Simple API for XML). DOM es el resultado del trabajo del W3C (SAX no lo es); ha sufrido varias revisiones y sigue actualizándose a medida que la tecnología y los requisitos evolucionan. Microsoft ha adoptado muchas de las características de programación del DOM y las ha expuesto a través de la clase XmlDocument del espacio de nombres System.XML en la biblioteca de clases .NET Framework, también ha agregado algunas extensiones a la funcionalidad de la implementación.

**XML y Visual Studio** La filosofía es similar a la creación de formularios Windows, si ya hemos creado un proyecto como ya hemos mencionado, debemos agregar un nuevo elemento o uno ya existente en su caso en la opción de menú Proyecto, como muestra la figura

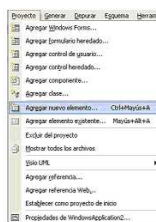


Figura B.7: Agregar un nuevo elemento

Se presenta una pantalla como la mostrada en la figura elegiremos Esquema XML y el nombre que deseemos darle a dicho esquema





Figura B.8: Agregar esquema XML

Al abrir el fichero creado podremos empezar a añadir elementos al esquema como se muestra en la figura siguiente arrastrándolo del cuadro de herramientas el elemento que queramos incluir en el esquema.

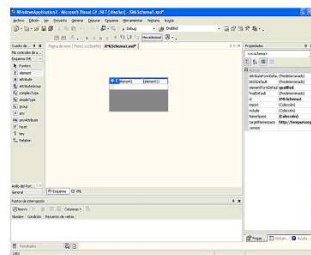


Figura B.9: Agregar elementos al esquema XML

De forma análoga se agrega al proyecto un archivo XML repitiendo los pasos anteriormente descritos pero para el elemento archivo XML.