365 – Motor de Veu Natural per Dispositius Mòbils

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Jordi Trujillo Sánchez
i dirigit per
Jordi Roig de Zárate
Bellaterra, 04 de Febrer de 2008

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignat, Jordi Roig de Zárate

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Jordi Trujillo Sánchez

I per tal que consti firma la present.

Signat: Jordi Roig de Zárate

Bellaterra, 04 de febrer de 2008

Agraïments:

Són moltes a les persones que directa o indirectament he d'agrair la realització d'aquest Projecte de Final de Carrera, tots ells han col·laborat de manera diferent a fer-ho possible. A Joan N. que des del primer moment s'ha interessat de com anava avançant tot el procés de realització, a Sònia P., a Martina G. i a Cristina L. que sempre m'han animat a seguir i m'han fet companyia en moments baixos de moral, i per últim, però no menys important, a Alberto A. que sempre ha estat al meu costat, recolzant-me, aguantant-me i donant-me grans consells que sense ells no ho hauria aconseguit, a tots, moltes gràcies.

Índex general

Capítol 1

Introducció

En aquesta introducció s'intenta presentar el projecte de final de carrera "Motor de veu natural per a dispositius mòbils" des d'un punt de vista acadèmic i tècnic, explicant conceptes sobre fonètica castellana. Es farà un petit estudi de viabilitat d'aquest projecte amb la possibilitat d'accessibilitat per a usuaris finals i s'explicarà l'estructura de la memòria.

1.1 Plantejament general

Les noves tecnologies ens envaeixen cada dia més, cada vegada tenim més aparells tecnològics a casa i al nostre voltant en general, encara que aquestes tecnologies intentin ajudar a les persones a facilitar la vida diària hi ha vegades que no es pensa en les persones amb discapacitats o aquestes tecnologies no estan fetes per a elles. Per això, aquí s'intenta presentar una petita ajuda a gent amb deficiències visuals que les pot ajudar una mica en una millor integració social i laboral.

Aquesta memòria reflecteix els objectius del projecte de final de carrera "Motor de veu natural per a dispositius mòbils", aquest motor de veu està basat en la fonètica espanyola, per tant, s'han de seguir les seves regles al cent per cent. El resultat pretén ser una reproducció fidel d'un text, frases o paraules el més a prop possible de la veu

real d'una persona i no sintètica. Aquest projecte és una continuació d'un projecte d'anys anteriors, que intenta millorar-se en el seu rendiment i velocitat d'execució.

El projecte consisteix en una aplicació que treballa sobre un dispositiu mòbil com una PDA o un telèfon amb un sistema operatiu Windows CE o Windows Mobile capaç de llegir una paraula i reproduir-la com a veu humana, per això s'han de fer servir sons ja creats anteriorment que formen part de la fonètica castellana. Aquests sons són els poden ser fonemes, que són els sons més petits i curts que es poden trobar (com "a", "e", "i", "o", "u", el so de qualsevol lletra, com "b", "k",...), els difonemes, que són la unió de dos fonemes ("pe", "ra",...), també hi ha els trifonemes i quatrifonemes, que s'explicaran més clarament el que són al llarg d'aquesta memòria. D'aquests sons, tenim a l'aplicació els fonemes i els difonemes, i amb la unió d'ells a través d'un algorisme de transcripció, que segueix les regles de la fonètica, podem aconseguir passar de text a veu i que es reprodueixi. Tot això fa que hi hagi una especial atenció en la càrrega de memòria de l'aplicació per a que no es col·lapsi o doni errors i en la velocitat de processament. Amb això, intentarem arribar a un sistema molt més optimitzar a l'anterior que faci aquesta conversió de text a veu humana per instaurar-ho en dispositius mòbils electrònics.

Per tant, el principal objectiu és millorar el projecte anterior amb la comunicació entre un dispositiu mòbil y les persones, ja que es podria aconseguir que una PDA o un mòbil llegís correctament un arxiu de text o qualsevol paraula escrita a la pantalla, inclús es podria aconseguir que un correu electrònic que arribi a una PDA es pugui transcriure i ser recitat per a l'usuari final, la qual cosa podria ajudar a persones amb discapacitats físiques a moure's per la pantalla i saber en tot moment per on estan ubicades.

Amb tot això, i degut a les limitacions que ens trobem, com la competència amb altres aplicacions que existeixen que també tradueixen a veu humana texts en dispositius mòbils però de forma sintètica, per tant, s'intentarà fer un model econòmic, capaç de reproduir veu natural, i que pugui arribar a ser una donació a una fundació com

la ONCE[1], i que pugui arribar a tothom que ho necessiti, tant físicament com econòmicament.

Per tant, aquest projecte intenta ser útil a nivell social per a la gent del voltant que ho necessiti i puguin fer servir dispositius electrònics amb el mínim de problemes per a ells, en especial està pensat per a persones amb discapacitats visuals, que respongui en temps real, el màxim d'econòmic possible, amb una qualitat de veu molt propera a la humana i amb la seva entonació correcta. Amb això aconseguiríem una interacció molt directa entre humà i dispositiu, saber recórrer la pantalla en tot moment i fer-lo servir de la manera més normal possible.

1.2 Estudi de viabilitat del projecte

Per realitzar el projecte, cal tenir en compte la possibilitat de que sigui o no viable tant a nivell realitzable, econòmic, temporal i acadèmic. Amb això s'intenta que a la implementació faci servir pocs recursos de memòria i rapidesa en l'execució.

Aquest sistema treballa sobre un sistema operatiu Windows CE o Windows Mobile per a dispositius mòbils com un telèfon mòbil o una Pocket PC que faci servir aquests sistemes operatius, ja que són els més utilitzats arreu, i s'intentarà que funcioni en un temps acceptable. Actualment existeixen varis sistemes que converteixen texts en veu però no són econòmics o no assoleixen les màximes que es proposen, ja que no són reproductors de veu natural i són amb veu sintètica, a més a més, cars i implementats per a ordinadors i no per a dispositius mòbils, per tant aquests no ens serveixen, ens interessen aplicacions que corrin sobre aquests dispositius, però la majoria són diccionaris i pensats per a que l'usuari aprengui bé la fonètica de l' idioma. Per tant, el que es proposa es un sistema que transcriu un text que rep des d'un fitxer o a la interfície de l'aplicació i retorna la lectura en llengua castellana en un temps relativament real i en una entonació correcta, respectant els signes de puntuació.

[1] ONCE: Organización Nacional de Ciegos Españoles

Per tot això ja tenim un estudi previ i una primera implementació, amb un estudi lingüístic totalment necessari per a la creació del projecte, seguint uns algorismes i regles fonètiques. Bàsicament, i a grans trets, els passos que farà servir el sistema són: recollir el text que vol ser reproduït, normalitzar-ho amb un transcriptor fonètic i separar en unitats més petites i anar a buscar els sons que tenim ja guardats (fonemes i difonemes, que amb la conjunció de diversos d'ells podem arribar a formar paraules amb sentit), tots aquests s'han de concatenar seguint les normes lingüístiques i reproduir-se.

Primer de tot s'ha hagut de recollir tota la informació del projecte anterior, entendre el seu funcionament i com poder millorar-ho. S'han fet les proves sobre aquest projecte, veure les mancances com alguns problemes de memòria, el temps d'execució, i llocs a la implementació que millorar. Al considerar que té una bona finalitat, he decidit centrar-me en aquest projecte i intentar aconseguir els objectius per millorar l'estat en el que es troba inicialment l'aplicació.

Aquest projecte estarà implementat amb Visual Studio 2003 en C#, es faran proves sobre el seu propi emulador amb el sistema operatiu Windows CE i també sobre una PDA, encara que ja donem per suposat que a la PDA trigarà més en executar-se degut a la quantitat d'accessos a disc. La raó de fer servir el Visual Studio en C# és perquè així es podria instaurar aquesta aplicació en moltíssims dispositius mòbils, ja que la majoria d'ells treballen amb Windows Mobile, a més a més, de que fent servir aquest, es guanyaria temps i recursos del dispositiu ja que podem aconseguir una aplicació interpretada i no compilada.

Per aconseguir els propòsits especificats, es treballarà amb aquests recursos:

- Ordinador portàtil ASUS, 0'5 Gb de RAM
- Windows XP Home Edition
- Visual Studio .NET 2003
- Connexió a Internet
- Manuals de programació en C#
- Manuals de funcionament de Visual Studio :NET

- Llibreria MSDN de Visual Studio :NET 2003[2]
- Pocket PC e750 WIFI Toshiba
- Microsoft ActiveSync
- Projectista d'enginyeria en informàtica
- Professionals que recolzin al projectista

1.3 Planificació temporal del treball

L'estructura del projecte consta de diferents punts que són també els de la planificació temporal:

Documentació: Estudiar tot el procedent del projecte anterior i comprendre el funcionament de l'aplicació.

Disseny: Hem de resoldre les mancances anteriors, per tant, hem de mirar per on hem d'atacar els problemes, aquí intervenen les reunions amb els professors.

Desenvolupament: Implementar les solucions que creiem convenients per a la millora del projecte, fer les proves pertinents, tot seguint pautes de disseny.

Avaluació del projecte: Jocs de proves que demostrin que s'han aconseguit les fites proposades.

Finalització del projecte: Documentació final sobre l'aplicació.

Aquests punts s'han de tenir enllestits dins del període acadèmic, totes les fases requereixen el seu temps específic per fer-les, les que poden portar més temps i cal dedicar més hores és al desenvolupament i a la documentació, ja que al ser la

[2] http://msdn2.microsoft.com

continuació d'un projecte anterior, hem de saber com funciona la seva implementació, les fites que ha arribat a complir provant l'aplicació, això entra dins de la part de documentació, llegint tots els annexos, les regles gramaticals i fonètiques en les que s'ha basat el projectista anterior, i tenir clares les mancances de l'aplicació.

No es pot considerar com una planificació lineal ja que per tal d'anar millorant s'han de tornar a passos anteriors, al fer jocs de proves que demostrin que no hem arribat al cas que buscàvem, s'ha de tornar a fer un nou disseny per atacar el problema, com per exemple, al trobar nous colls d'ampolla que s'han d'anar solventant poc a poc amb noves estratègies en el disseny.

Aquestes etapes es poden veure reflectides al següent esquema (Figura 1):

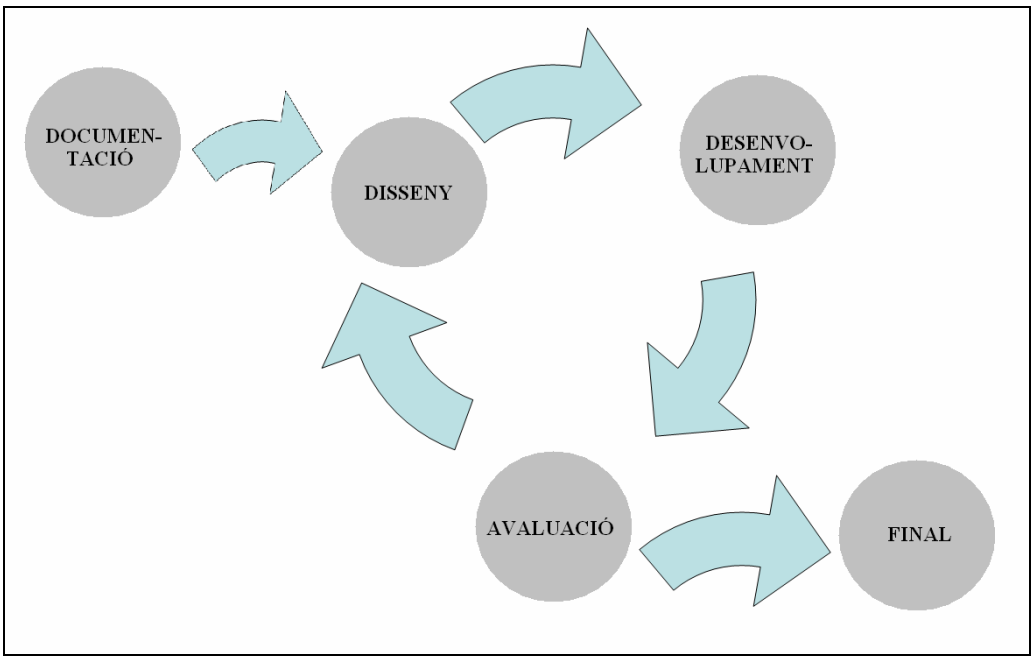

Figura 1: Etapes del procés del projecte

A continuació podem veure una taula de les hores que es podrien dedicar a cada una de les tasques, tenint en compte la coordinació del projectista amb la seva vida personal i laboral, vacances i festes (Taula 1):

Fase	Hores programades
Documentació	350 hores
Estudi memòria anterior	150 hores
Estudi annexos de la memòria	50 hores
Estudi regles fonètiques	50 hores
Jocs de proves	100 hores
Disseny	200 hores
Desenvolupament	600 hores
Avaluació del projecte (jocs de proves)	50 hores
Finalització del projecte (memòria)	250 hores
Total	*1450 hores*

Taula 1: Planificació temporal en hores de la realització del projecte

1.4 Estructura de la memòria

Aquesta memòria està dividida en diferents capítols que expliquen el funcionament i on s'ha aconseguit arribar al llarg de la realització del projecte de final de carrera "Motor de veu natural per a dispositius mòbils". Són un total de sis capítols, on ara mateix estem situats al primer, que és la introducció, on ens centrem una mica en el tema i veiem quin és el treball global realitzat, també veiem un estudi de viabilitat per saber si aquest projecte ho és, també consta una planificació temporal del treball realitzat durant el temps donat.

En el capítol segon trobem una explicació més clara dels conceptes que hem de tenir clars per saber el funcionament correcte de l'aplicació, allà s'explicarà els tipus de sons que es fan servir, la unió entre aquests per formar un so que estem esperant. També s'introdueix l'estat de l'art en el marc actual, s'expliquen alguns dels sistemes que converteixen texts en veu, les mancances que tenen respecte el nostre motor de veu que estem presentant.

En el tercer capítol es fa un estudi de tot el que tenim i el que hem necessitat per a la millora de l'aplicació inicial, s'expliquen els recursos hardware i software utilitzats, també es denoten els requeriments funcionals que ha de tenir l'aplicació, una aproximació a com funciona aquesta i les millores que s'han fet respecte a l'aplicació inicial.

El quart capítol s'explica la implementació i modificació de mòduls de l'aplicació, de manera interna, crides a funcions noves, les millores realitzades, així com els resultats d'aquests passos, i s'aniran comparant per veure la millora trobada en temps d'execució, sobretot inicial, que és la més important: el temps que triga a reproduir-se una paraula o grup de paraules des de que es llença la ordre de fer-se fins que s'escolta per l'altaveu. Per tant, també tindrem tots els jocs de proves que s'han fet durant la realització del projecte.

El cinquè capítol explica les conclusions a les que s'ha arribat durant el desenvolupament del projecte, així com les mancances que té l'aplicació i per on es pot atacar per arribar a ser útil.

Per últim, tenim el sisè capítol que mostra els annexes i la bibliografia que s'ha fet servir durant l'elaboració del projecte.

Capítol 2

Estat de l'art i primers conceptes

En aquest capítol es recull l'estat de l'art sobre l'aplicació del motor de veu natural per a dispositius mòbils i uns primers conceptes sobre com interactuen entre ells els sons fets servir. Primer passarem a explicar qué són els fonemes, difonemes, trifonemes i quatrifonemes, i després quins són els sitemes que converteixen el text en veu natural.

2.1 Tipus de sons que es fan servir

Durant l'execució del projecte es fan servir diferents sons, ells estan gravats en format wav dins d'una base de dades, van des de la unitat més petita de so que és el fonema, passant per la unió de dos fonemes, que formaran un difonema, i així successivament fins arribar als quatrifonemes.

Cal tenir em compte que es treballarà amb les ones que forma una paraula o so, d'aquesta manera es muntaran aquests sons. Per tant cal estudiar per on són més estables, la seva longitud, per tal de fer un bon ús d'aquests sons. Com per exemple, podem veure a la següent imatge que aquesta és la ona de la paraula "hola" (Figura 2):

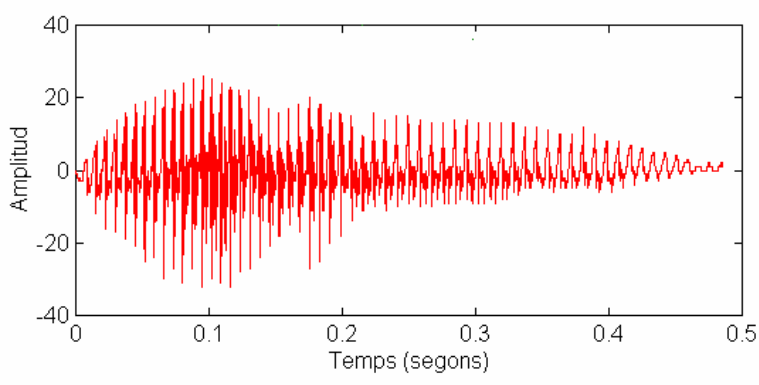

Figura 2: Ona corresponent a la paraula "hola"

2.1.1 Qué és un fonema?

És la unitat més petita de so que farem servir. En teoria, la unió entre ells farà un so més llarg i que pugui tenir sentit, dit d'una altra manera, són tots sons que composen la frase (en castellà tenim 63 fonemes), si agaféssim com exemple la paraula "mesa" veiem que faria servir quatre fonemes [m]-[e]-[s]-[a], però la unió d'ells com a fonema no formen la paraula, o més ben dit, so "mesa", ja que són sons separats. La unió d'aquests sons es fa amb una síntesi concatenativa que explicarem més endavant.

Podem veure en un exemple amb la paraula "hola" com es trobarien els fonemes, aquesta és l'ona de veu que forma aquesta paraula i per on es tallen aquests fonemes; podem observar l'ona tallada en [o]-[l]-[a] (Figura 3)

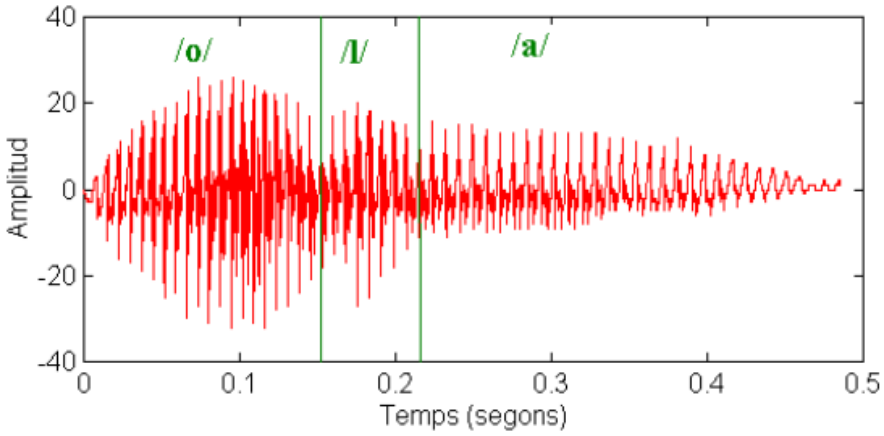

Figura 3: paraula "hola" separada en fonemes

2.1.2 Qué és un difonema?

Bàsicament és la unió de dos fonemes, però la cosa no és tan senzilla, ja que és aquesta unió entre tots dos fonemes però tenint en compte la ona de freqüència que crea un fonema, ja que s'han d'unir per la part més estable d'ells, és a dir, generalment seria per la meitat, i de la mateixa manera, s'haurien d'unir entre ells per la part més estable de la ona per crear el so amb sentit. Si tornem a agafar l'exemple "mesa" i el separem en difonemes, quedaria de la següent manera:

[silenci-m]-[me]-[es]-[sa]-[a-silenci]

De totes maneres, aquestes unions entre difonemes, a vegades no queden del tot netes i no es crea un so totalment real, ja que existeixen sons molt curts i no donen temps a tenir part d'ona molt estable i no es poden tallar correctament, aquests són els fonemes [b], [k],... Per millorar això ens calen també els trifonemes.

A la següent figura veiem el mateix exemple que en el cas anterior però fent els talls en difonemes, més o menys per la meitat del so on la ona és més forta i més estable, veiem que la paraula "hola", tallada en difonemes correspon a [silenci-o]-[ol]-[la]-[a-silenci] (Figura 4)

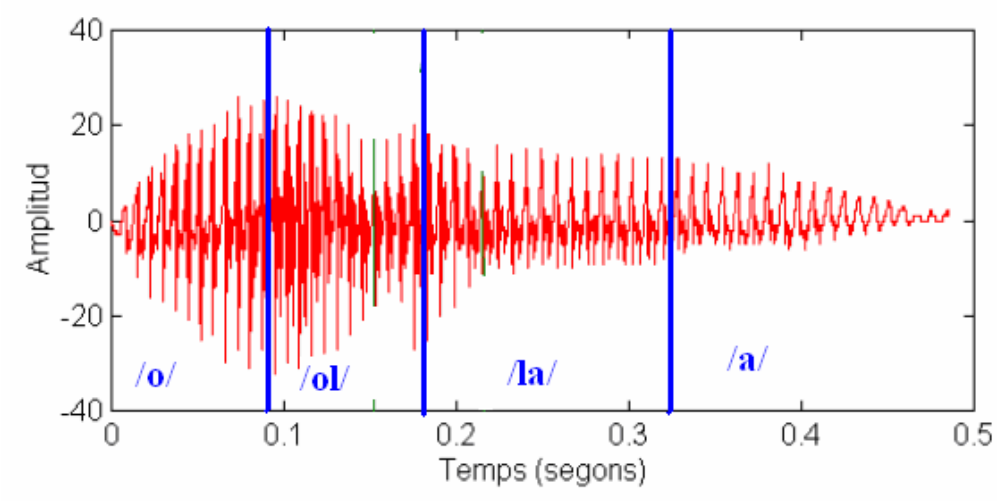

Figura 4: Paraula "hola" separada en difonemes

Tenint una base de dades sencera de tots els difonemes possibles en castellà, ens és possible representar qualsevol paraula d'aquesta llengua, a la següent taula podem observar tots els que hi ha (taula 2), podem observar sons en blanc o buits representats per "_", o podem veure la presència de números del 1 al 5, aquests representen cada una de les vocals però de forma tònica, per tant, seguint amb l'exemple de la paraula "hola", la dividiríem correctament en difonemes com:

[_4]-[4l]-[la]-[a_]

on "4" representa la "o" tònica.

	_	p	t	k	b	d	g	f	m	n	;	@	s	z	=	x	y	(	%	#	i	e	a	o	u	3	2	1	4	5
_		_p	_t	_k	_b	_d	_g	_f	_m	_n	_;		_s			_x	_y	_(	_%	_#	_i	_e	_a	_o	_u	_3	_2	_1	_4	_5
p	p_																				pi	pe	pa	po	pu	p3	p2	p1	p4	p5
t	t_																				ti	te	ta	to	tu	t3	t2	t1	t4	t5
k	k_																				ki	ke	ka	ko	ku	k3	k2	k1	k4	k5
b																					bi	be	ba	bo	bu	b3	b2	b1	b4	b5
d	d_																				di	de	da	do	du	d3	d2	d1	d4	d5
g																					gi	ge	ga	go	gu	g3	g2	g1	g4	g5
f	f_									fn									f%		fi	fe	fa	fo	fu	f3	f2	f1	f4	f5
m	m_	mp			mb				mm												mi	me	ma	mo	mu	m3	m2	m1	m4	m5
n	n_		nt			nd		nf		nn			ns		n=			n(	n%	n#	ni	ne	na	no	nu	n3	n2	n1	n4	n5
;																					;i	;e	;a	;o	;u	;3	;2	;1	;4	;5
@				@k			@g									@x														
s	s_	sp	st	sk				sf								sx		s(	s%		si	se	sa	so	su	s3	s2	s1	s4	s5
z									zm	zn							zy													
=																														
x	x_																				xi	xe	xa	xo	xu	x3	x2	x1	x4	x5
y																					yi	ye	ya	yo	yu	y3	y2	y1	y4	y5
(																					(i	(e	(a	(o	(u	(3	(2	(1	(4	(5
%	%_	%p	%t	%k					%m	%n											%i	%e	%a	%o	%u	%3	%2	%1	%4	%5
#	#_																				#i	#e	#a	#o	#u	#3	#2	#1	#4	#5
i	i_	ip	it	ik	ib	id	ig	if	im	in	i;		is	iz		ix	iy	i(	i%	i#	ii	ie	ia	io	iu	i3	i2	i1	i4	i5
e	e_	ep	et	ek	eb	ed	eg	ef	em	en	e;		es	ez		ex	ey	e(	e%	e#	ei	ee	ea	eo	eu	e3	e2	e1	e4	e5
a	a_	ap	at	ak	ab	ad	ag	af	am	an	a;		as	az		ax	ay	a(	a%	a#	ai	ae	aa	ao	au	a3	a2	a1	a4	a5
o	o_	op	ot	ok	ob	od	og	of	om	on	o;		os	oz		ox	oy	o(	o%	o#	oi	oe	oa	oo	ou	o3	o2	o1	o4	o5
u	u_	up	ut	uk	ub	ud	ug	uf	um	un	u;		us	uz		ux	uy	u(	u%	u#	ui	ue	ua	uo	uu	u3	u2	u1	u4	u5
3	3_	3p	3t	3k	3b	3d	3g	3f	3m	3n	3;		3s	3z		3x	3y	3(	3%	3#	3i	3e	3a	3o	3u					
2	2_	2p	2t	2k	2b	2d	2g	2f	2m	2n	2;		2s	2z		2x	2y	2(	2%	2#	2i	2e	2a	2o	2u					
1	1_	1p	1t	1k	1b	1d	1g	1f	1m	1n	1;		1s	1z		1x	1y	1(	1%	1#	1i	1e	1a	1o	1u					
4	4_	4p	4t	4k	4b	4d	4g	4f	4m	4n	4;		4s	4z		4x	4y	4(	4%	4#	4i	4e	4a	4o	4u					
5	5_	5p	5t	5k	5b	5d	5g	5f	5m	5n	5;		5s	5z		5x	5y	5(	5%	5#	5i	5e	5a	5o	5u					

Taula 2: Taula de difonemes posibles dins d'una paraula

Veiem que hi ha moltes cel·les buides, això és degut perquè hi ha sons que no es donen al castellà, com per exemple, la presència de dos vocals tòniques juntes en una paraula.

2.1.3 Qué és un trifonema?

Es podria dir que és la unió de tres fonemes però en realitat el definim millor com un so que està format per una ona acústica amb la transició d'un so a un altre, d'aquest a un sencer i d'aquest fins a l'últim so. Normalment són combinacions d'una vocal seguida de dos consonants. Veiem la paraula "alpe", si ho separem en difonemes i trifonemes, veiem que el formen tres difonemes i un trifonema, quedaria de la següent manera:

[silenci-a]-[alp]-[pe]-[e-silenci]

2.1.4 Qué és un quatrifonema?

Són dos difonemes que junts ocupen molt poc temps i si es tallés, la qualitat seria molt dolenta, agafem la paraula "algo", en principi la dividiríem en difonemes així:

[silenci-a]-[al]-[lg]-[go]-[o-silenci]

Però el difonema [lg] és tan curt que no el podem fer servir com a tal, i si ho ajuntéssim en trifonemes, perdria qualitat de so. Per tant, aquest so es podria dividir en dos fonemes i un quatrifonema:

[silenci-a]-[alGo]-[o-silenci]

2.2 La unió dels sons

Una vegada sabem que són els fonemes, difonemes, trifonemes i quatrifonemes, i sabem que s'han d'unir, cal explicar el tipus d'unió entre ells, tal com seguia el projecte anterior, ho fem amb una síntesi concatenativa, no ho fa seguint regles, sinó únicament unint sons que ja han sigut gravats i nosaltres els tenim en diferents bases de dades (la de cada tipus de fonema). Això vol dir, que tenint els sons possibles emmagatzemats, al unir-se no es poden fer de qualsevol manera, sinó que han de seguir unes regles, aquestes no les aplicarà la síntesi sinó que ho farà la mateixa aplicació seguint un algoritme de transcripció.

Una unió concatenativa de síl·labes tampoc seria correcta ja que hi haurien sons que no es tindrien en compte com difonemes entre les síl·labes. Per tant, per aconseguir una concatenació neta i correcta entre sons i el més real possible, s'haurien de fer servir tots quatre sons que hem comentat abans.

Però tots aquests sons que s'han de concatenar han de sortir d'algun lloc, i és a través d'un corpus, a partir d'uns sons gravats, on s'han estudiat prèviament tots els sons possibles, i s'han tallat, això requereix un estudi previ ben a fons i un temps molt elevat. A més, aquí es tindrà en compte l'entonació, per les comes, punts, interrogacions, exclamacions... Aquests sons formaran part de diverses bases de dades del projecte: fonemes, difonemes, trifonemes i quatrifonemes. En aquesta versió actual del projecte, només treballarem amb fonemes i difonemes. Els trifonemes i quatrifonemes seran part important de la millora següent per a futurs estudiants.

2.3 Sistemes que converteixen el text en veu natural

Avui en dia, al mercat, hi ha molts sistemes que converteixen text en veu, o també podem trobar pàgines web que ho fan i et deixen escollir diferents idiomes o veus (masculina o femenina), i depenent de l' idioma han de seguir unes regles establertes, pròpies de cada idioma. Però la majoria, són aplicacions que corren dins de l'ordinador, com per exemple els que treballen directament a la pantalla de l'ordinador (el punter del ratolí es posa a sobre d'una paraula i aquesta es llegida pel sistema i reproduïda) són els *lectors de pantalla*, és a dir, llegeixen a partir de l'exploració de la pantalla. També existeixen els *lectors de texts* que reprodueixen una narració d'un llibre, o un missatge sencer i amb sentit. La diferència que pot haver-hi entre aquests dos és a l'entonació, en el primer cas no necessitaria cap entonació especial, en canvi, en el segon sí, ja que es troben punts, comes, punts-i-comes, pauses en general i sons entre paraules seguides que són diferents a si es fan servir si les paraules es llegeixen o es produeixen per separat.

I podem trobar alguns exemples d'aquestes aplicacions com pot ser ATT (Advanced Text to Speech) que llegeix texts que es copies en el porta papers (Ctrl+C) i el programa mostra una finestra per si es vol cancel·lar la lectura i sinó s'iniciarà la lectura en segons i és capaç d'exportar aquest so a un arxiu WAV o MP3 (Figura 5).

Figura 5: Aplicació ATT – Advanced Text to Speech

Un altre programa és el "2nd Speech Center" (Figura 6) que també és capaç de reproduir des del porta papers i guardar-ho en format WAV o MP3, originalment té el seu motor de veu en anglès però es poden instal·lar en espanyol, francès o italià.

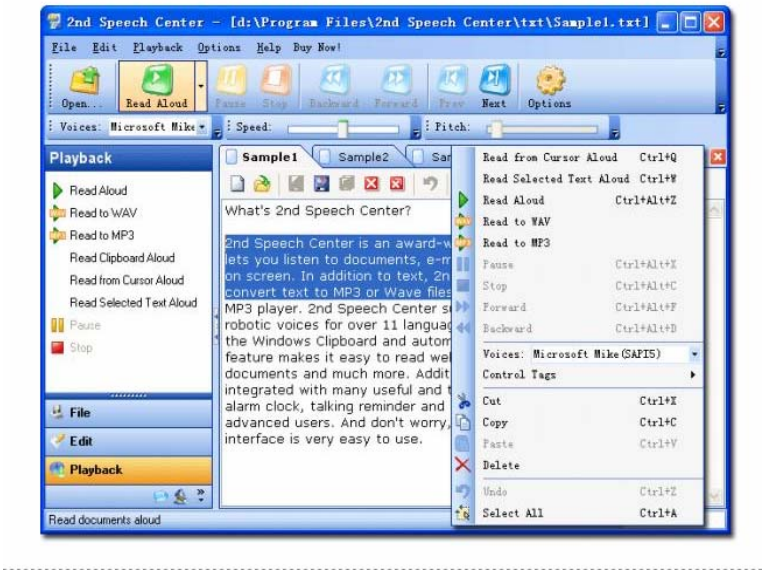

Figura 6: Aplicació "2nd Speech Center"

També tenim webs a Internet que ho fan[3], molts d'ells també donen l'opció de descarregar-se l'arxiu en .wav o .mp3, o escoltar-ho online (Figura 7).

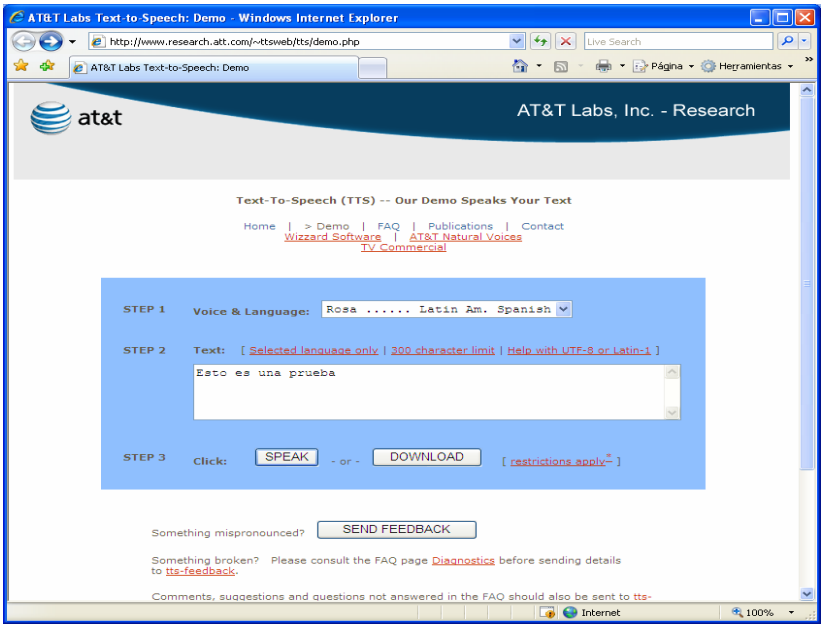

Figura 7: Plana web que transforma de text a veu – AT&R Labs, Inc.

[3] http://www.research.att.com/~ttsweb/tts/demo.php

Aquestes aplicacions mencionades són per executar sobre un ordinador personal, per tant no arriben a les nostres fites, en canvi, podem trobar algun programa per a dispositius mòbils com el "Flite" (Figura 8) que intenta emular la veu humana sintetitzant els caràcters presents al seu camp de text de la seva finestra principal però suporta freqüències molt baixes. Tampoc serveix, ja que volem aconseguir que en comptes de veu sintetitzada sigui amb veu natural.

Figura 8: Aplicació "Flite" per a Pocket PC

Els lectors de texts necessiten fer servir les regles gramaticals de la llengua que estan reproduint; cada llengua té les seves regles, sons que no es poden donar a la construcció d'una paraula, cadena de lletres que existeixen o no dins la llengua. Un exemple pot ser que en castellà no es doni el cas de trobar o de reproduir-se un so format per les lletres "mkpo", o "wwrr", no hi haurà cap regla que arribi a codificar això. Un altre exemple sobre les regles pot ser que en català es dóna el cas de les vocals febles, que en castellà no existeixen, per tant, si féssim l'aplicació en català s'hauria d'introduir aquests casos. Tot això ha suposat un estudi previ fonètic que un enginyer en informàtica no acostuma a conèixer i necessita ajuda de professionals d'altres doctrines. Podem veure els mòduls que formen un lector de texts en la següent figura, podem observar que cal fer un anàlisis lingüístic després de processar el text, que són les regles que han de fer servir per una correcta pronunciació (Figura 9):

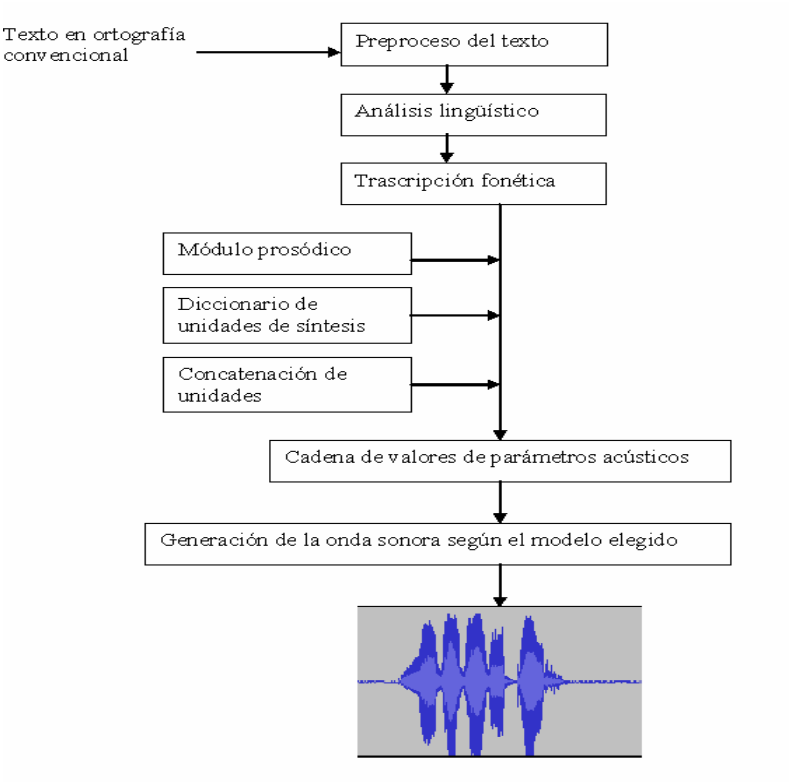

Figura 9: Mòduls que formen un lector de text

Per tant, veiem que hi ha diverses aplicacions que fan el que estem seguint, per una banda el "Flite", ja que s'executa sobre un dispositiu mòbil, o el ATT, però aquest s'executa sobre ordinador, per tant no totes les aplicacions existents compleixen els requisits que busquem en aquest projecte, com la veu natural, el temps de resposta i la màquina on s'ha d'executar. Alguns d'aquests programes es poden trobar a Internet, descarregar i instal·lar, la majoria són versions de proves, d'aquests en trobem pocs per a dispositius mòbils.

Capítol 3

Anàlisi de requeriments, recursos i funcionament de l'aplicació

En aquest capítol s'explicarà quins són els recursos hardware i software que s'han fet servir al llarg del projecte. També s'explica l'estat de l'aplicació anterior, el funcionament i les mancances o defectes que hi havia i que s'han intentat solventar al llarg de la realització de projecte.

3.1 Recursos

3.1.1 El Hardware

L'ordinador fet servir durant el desenvolupament del projecte ha sigut un ordinador portàtil amb les següents característiques:

- Ordinador portàtil ASUS, 0'5 Gb de RAM
- Windows XP Home Edition

A part del desenvolupament del codi en aquest ordinador, també s'han fet en ell les proves sobre emuladors de dispositius mòbils basats en el sistema operatiu Windows Mobile i Windows CE, aquests sistemes operatius, són els més extensos arreu per a dispositius mòbils.

Les altres proves que s'han anat fent, s'han fet fora de l'emulador, ja han sigut proves físiques i reals. Per això hem necessitat un dispositiu mòbil real, que ha sigut un Pocket PC e750 WIFI Toshiba (Figura 10), les característiques que més ens interessen per el desenvolupament del projecte són les següents:

- Processador Intel PXA 255, 400 MHz
- Pantalla tàctil
- 64 MB de RAM
- Flash ROM de 32 MB
- Sistema de so estèreo de 16-bits amb micròfon integrat
- Windows Mobile
- Mòdul de memòria Secure Digital (SD) de 128 MB
- Possibilitat de fer servir mòdul de memòria Compact Flash

Figura 10: Pocket PC e750 WIFI Toshiba

3.1.2 El software

El software que s'ha fet servir durant el temps que ha durat el desenvolupament del projecte ha sigut el Microsoft Visual Studio .NET 2003 Professional, els emuladors que ell incorpora tant per Pocket PC 2002, com en emulador de Windows CE .NET. La majoria de les proves s'han fet al voltant d'aquests emuladors però també s'ha fet servir el Pocket PC real que tenim, per fer-lo servir també hem necessitat la instal·lació del software Microsoft ActiveSync per la sincronització entre el Visual i el dispositiu físic.

3.1.2.1 Microsoft Visual Studio .NET 2003

L'aplicació ja estava basada en aquest llenguatge i com que vaig decidir continuar amb la seva estructura i la seva base, vaig continuar treballant amb el Microsoft Visual Studio que em va facilitar el departament, ja que no és un software lliure. Dins d'ell hem treballat amb el llenguatge C#, i a més a més, s'ha pogut incrustar algun mòdul en un altre llenguatge com XML i també s'ha intentat utilitzar en algun moment una integració d'un mòdul de JAVA, encara que es va descartar per motius que ja explicarem més endavant. Per tant, veiem que el Microsoft Visual Studio pot treballar amb diversos vegades en el mateix projecte, cosa que ens facilita el desenvolupament.

De totes maneres, el Microsoft Visual Studio (Figura 11) té una extensa llibreria d'ajuda com MSDN de Visual Studio .NET 2003, on també trobem una versió online i totalment actualitzada, i moltes ajudes a través de fòrums i webs que parlen sobre programació en .NET. A més, és capaç de deixar treballar amb emuladors de dispositius mòbils que és on volem fer servir l'aplicació final.

Figura 11: Microsoft Visual Studio .net Professional

Però amb tot això també és necessari l'entorn .NET Compact Framework[4] que ens ajuda a programar aplicacions per a dispositius mòbils, ja que tenen limitacions de recursos. És un conjunt de biblioteca de biblioteques de classes que s'han dissenyat expressament per a ell. Gràcies al Compact Framework es poden executar programes independents del hardware i del sistema operatiu, ha sigut el que ens ha deixat treballar perfectament amb XML, optimitza els recursos del sistema i intenta obtenir un rendiment òptim quan compilem en temps real.

3.1.2.2 Els emuladors

El Microsoft Visual Studio .NET ens ofereix la possibilitat de treballar primerament amb emuladors de dispositius mòbils, gràcies a això s'han pogut trobar molts errors debbugant l'aplicació durant la implementació. Com podem veure a la figura 12, al compilar el sistema ens deixa quatre opcions on executar el programa.

[4] http://www.microsoft.com/downloads/details.aspx?displaylang=es&FamilyID=262D25E3-F589-4842-8157-034D1E7CF3A3

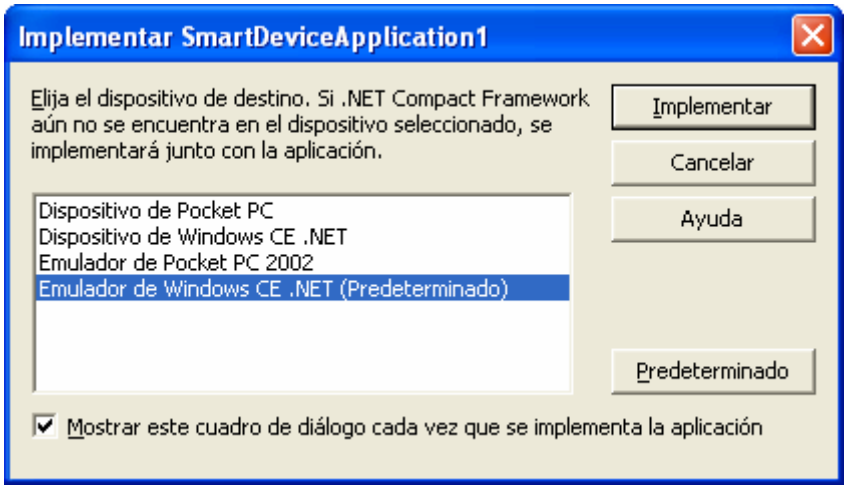

Figura 12: Opcions de Visual per escollir on implementar l'aplicació

3.1.2.2.1 Emulador de Windows CE .NET

És a on bàsicament s'han fet totes les proves, gràcies a la similitud de la interfície tant semblant al Windows al que estem acostumats, ha sigut una part important per a dur a terme l'execució del programa i les proves, ja que si fèiem servir logs que guardaven el temps d'execució durant les proves, sempre ha sigut més fàcil obrir-los i estudiar-los amb aquest, i al ser d'unes dimensions més grans, ens ha ajudat a fer servir més ràpidament l'aplicació(Figura 13).

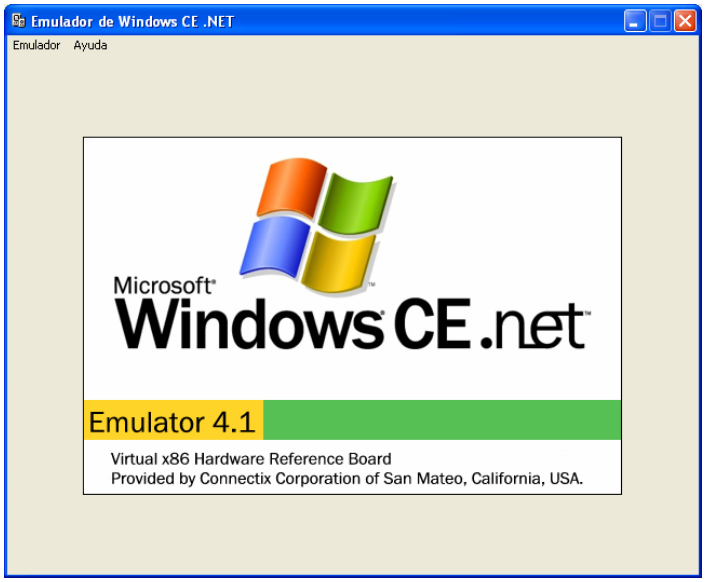

Figura 13: Emulador de Windows CE .NET

3.1.2.2.2 Emulador de Pocket PC 2002

Durant el desenvolupament del projecte, gairebé no s'ha fet servir, aquest emulador (Figura 14) sobretot s'ha fet servir per crear una interfície amb dimensions correctes per a un Pocket PC, això vol dir, reduir les dimensions de la pantalla principal de l'aplicació.

Figura 14: Emulador Pocket PC 2002

3.1.2.3 Microsoft ActiveSync

És el software necessari per a gestió de sincronització entre ordinador i PDA o dispositiu mòbil via USB. Amb ell es transfereixen les dades necessàries d'un lloc a un altre, com per exemple els correus electrònics d'un Oultook, o cites, entre altres coses.

Nosaltres el fem servir per a descarregar la nostra aplicació al dispositiu mòbil físic per comprovar el seu comportament real. Una vegada escollida la opció adequada a on implementar l'aplicació, per tant a un dispositiu de Pocket PC, ja podem treballar amb el Pocket PC de la mateixa manera que fèiem amb l'emulador.

Figura 15: Implementar aplicació sobre dispositiu físic

Figura 16: Microsoft ActiveSync

3.2 Requeriments funcionals

Tal com es va comentar a la introducció, un dels aspectes més importants a tenir en compte a l'hora de millorar l'aplicació anterior és la utilització de la memòria i el temps de procés, sobretot l'inicial. Sabem que ha de funcionar sobre dispositius mòbils que treballen sobre el sistema operatiu Windows Mobile, ja que és el més comú i així es podria arribar a més persones amb les necessitats que s'intenta cobrir aquesta aplicació.

Aquest motor de veu natural per a dispositius ha d'agafar un text, o bé des de la pantalla o bé des d'un arxiu de text (*.txt), per tant, aquesta serà el paràmetre d'entrada. Aquest es processarà i serà el que finalment es reprodueixi d'una manera el més fidel possible a la veu natural, per tant, necessitem una qualitat acceptable d'aquest so, que no soni tan sintètica i que arribi a tenir una entonació correcta, i tot això s'hauria d'aconseguir en un temps acceptable o gairebé immediat.

La manera en que funciona l'aplicació és a través de mòduls funcionals que segueixen l'esquema següent (Figura 17):

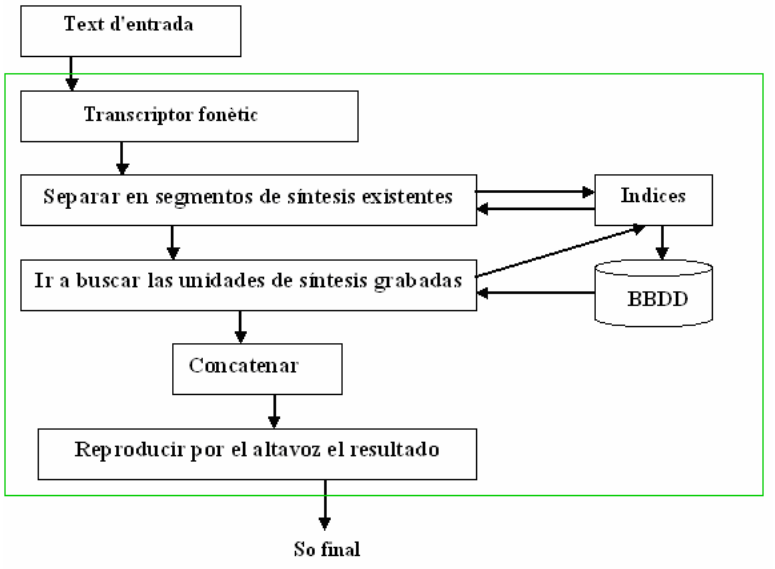

Figura 17: Mòduls de l'aplicació

Al seguir aquest esquema, veiem que hem de passar per una sèrie de mòduls, agafant el text d'entrada de la pantalla (lector de pantalla) o des d'un arxiu de text. Haurà de carregar totes les dades externes possibles al principi per tenir més velocitat a l'aplicació, com la interfície de l'aplicació, estructures XML que indiquen on s'han de buscar els sons a la base de dades (això s'explicarà en el següent capítol), aquesta base de dades seran sons gravats en format WAV, per ser un estàndard i tenir molta informació per a poder fer-lo servir, i en espanyol, aquesta base de dades es reparteix en fonemes, difonemes, trifonemes i quatrifonemes, com ja hem explicat, i s'uniran concatenant-se i les regles de transcripció fonètiques ho controla un mòdul específic. El reproductor que es fa servir, és un reproductor integrat dins de l'aplicació i no és un d'extern, cosa que ens ajuda amb el temps de resposta, i a més a més, treballa reproduint-se des de la memòria i no des d'un fitxer, per tant, no cal accedir a un fitxer físic per reproduir-se.

Aquest sistema ara ja treballa amb paral.lelització a l'hora de fer el processament, això ajuda a també a la velocitat del temps de resposta. En comptes d'esperar a processar tota una frase, es processen totes les paraules de la frase d'entrada de manera paral·lela i es reprodueixen per ordre. El problema que trobem aquí és que quan una persona parla, no deixa "espais blancs" entre cada paraula, només ho fa amb els signes de puntuació o per agafar aire i continua parlant, això està contemplat a l'aplicació, es llençarà la transcripció d'un grup de paraules fins que es trobi un signe de puntuació, o fins un número determinat de caràcters per tal de poder agafar aire, només caldria canviar un paràmetre dins de la codificació del projecte, això ara no és possible i es fa amb cada paraula ja que hi ha un coll d'ampolla molt important en el transcriptor fonètic que cal resoldre en els treballs posteriors.

Com es va parlar al capítol anterior, hi ha un quadre de tots els sons dels difonemes possibles dins d'una paraula, però no ens és prou al voler contemplar la opció de suprimir els "espais blancs" entre paraules. Recordem el quadre de la taula 2.

Podem agafar el cas de les dues esses juntes "ss", en castellà no hi ha cap paraula amb elles, per tant, observem a la taula que aquest so no es contempla, per tant, en un principi no s'hauria de fer servir, però en el cas que necessitem, podem trobar dos paraules, una que acabi amb essa i que la següent comenci amb ella, per tant, si

suprimim els “espais blancs” de la pronunciació, veiem que sí que es pot donar aquest so, per exemple amb les dues paraules “paredes sucias”. Llavors caldria contemplar tots els sons de la taula, tots els que no hi són, això ajudaria a millorar el so final, i la concatenació entre paraules seria més real i creïble.

La concatenació de sons es farà amb fonemes i difonemes, ja que amb ells es pot aconseguir un so que sigui ben clar i que s’entengui clarament. La concatenació és senzilla i ocupa menys espai de memòria. Fer servir trifonemes i quatrifonemes ens ajudaria a reduir una mica el temps de formació del so final.

S’ha de tenir clar quins són aquests sons gravats i com estan identificats dins de la base de dades per poder-los tractar. Cal estudiar bé els noms dels fitxer i que representa cada un. Per tal d’incloure’l bé dins de l’arxiu XML per poder fer una cerca correcta.

3.3 Funcionament de l’aplicació

Acabem de veure quins són els requeriments funcionals necessaris per l’execució de l’aplicació i amb un esquema general de com treballa el motor de veu per a dispositius mòbils. Però podem veure amb més detall el següent diagrama de flux, aquest diagrama és el que teníem de l’aplicació anterior i no hi ha grans canvis en el seu procés, per tant, per veure el funcionament i presentar el que tenim inicialment el fem servir (Figura 18)

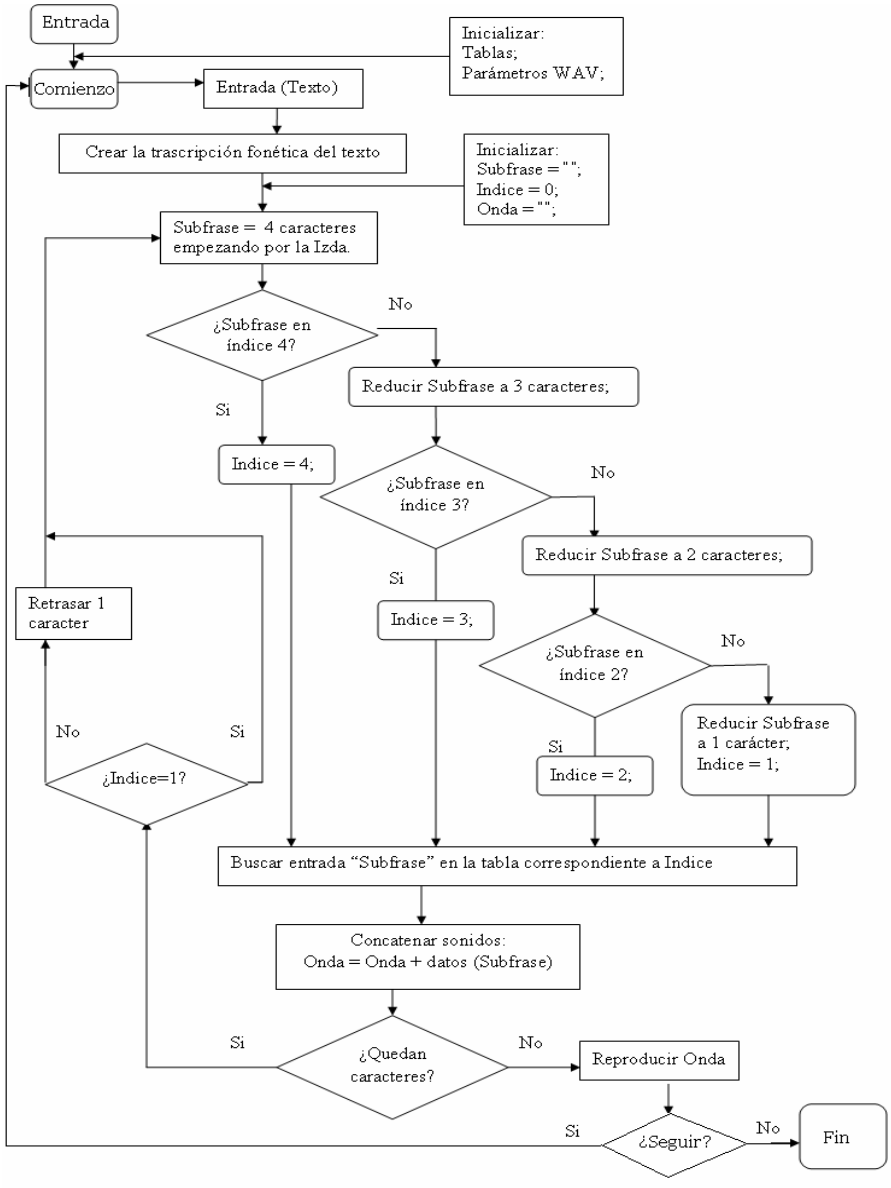

Figura 18: Diagrama de flux de l'aplicació

Tenim en compte que treballar amb quatrifonemes i trifonemes faria un so molt més net amb una bona qualitat final i ajudaria a la rapidesa de l'aplicació, ja que no hauria de passar per tants estats. Per tant, s'haurien d'agafar els primers quatre caràcters i processar-los i si no ho troba a la base de dades de quatrifonemes, s'agafarien tres i aniria a la base de dades de difonemes, i així successivament, fins arribar als fonemes, si és que no s'ha trobat el que s'ha anat cercant durant el procés. Una vegada s'aconsegueix trobar l'arxiu de so adequat per aquest segment a la seva base de dades es va concatenant per fer la ona de sortida i reproduir-la. Com que s'ha decidit treballar amb difonemes i fonemes, l'esquema anterior queda de la següent manera (Figura 19)

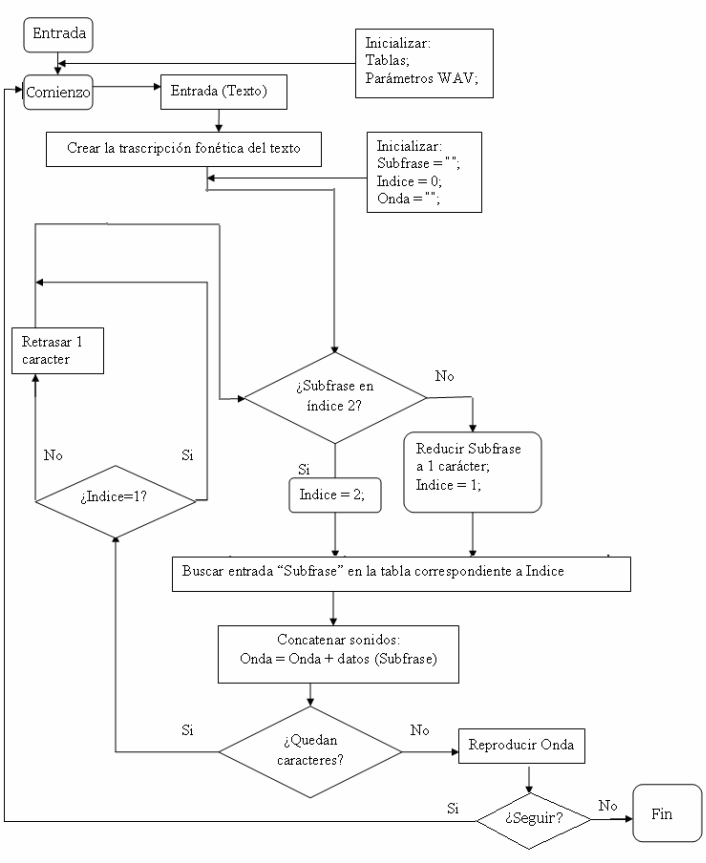

Figura 19: Diagrama de flux real de l'aplicació

En aquesta fase es treballarà seguint aquest esquema, no deixem de banda recordar que treballar amb quatrifonemes i trifonemes serà una millora en treballs futurs. Hi ha diversos aspectes que s'han millorat en aquesta versió, tal com s'ha indicat, ara es treballa fent servir una paral.lelització en el procés amb totes les paraules i abans s'havia de processar tota la fase sencera i esperar, i podem veure que hi havia un temps de procés elevat que era una de les coses més importants també a millorar, a la següent taula veiem els temps de resposta de l'aplicació inicial amb els exemples que hi havia (Taula 3):

PROVA	***PANTALLA***
Apéndice	4 seg
apéndice 2 de mis documentos	18 seg
mis documentos	7 seg
pantalla con mensajes	9 seg
Ventana	3 seg
2	2 seg
documento con mensajes	14 seg
mis mensajes	5 seg

Taula 3: Joc de proves de l'aplicació inicial

A continuació veiem la interfície que teníem inicialment que també s'ha millorat i s'han afegit funcionalitats com poder llegir arxius de text, ja que en aquest moment només podia llegir des de pantalla (Figura 20).

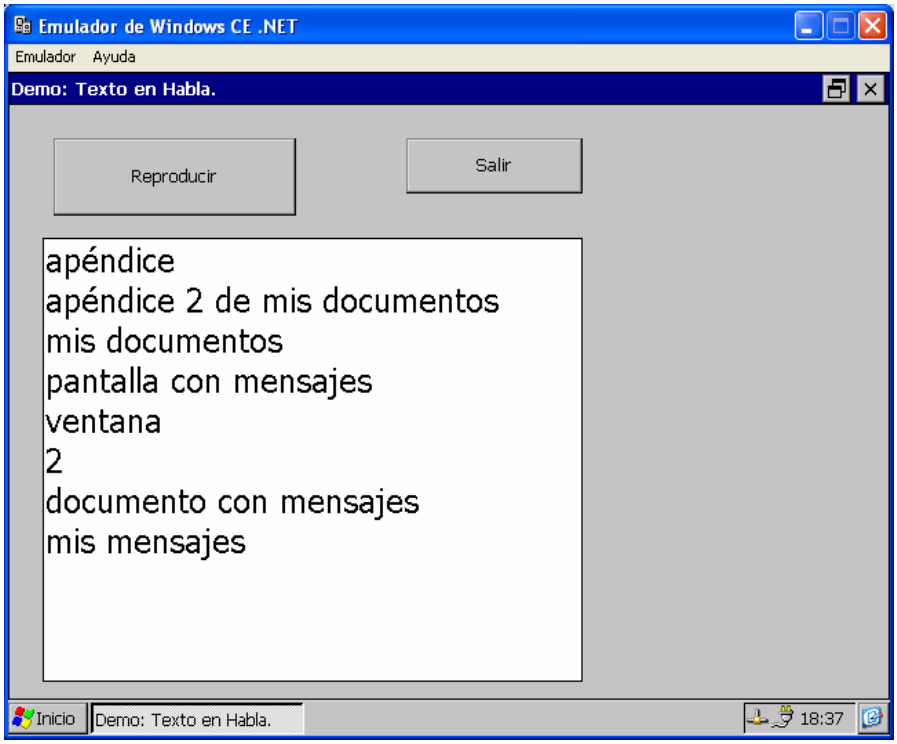

Figura 20: Interfície de l'aplicació original

Altres millores importants, que s'explicaran en el següent capítol, és la introducció de cerca en arxius XML que ajuden a trobar els arxius wav pregravats per formar els sons i així estalviar-nos la cerca tan complexa a través de taules que feia servir. També s'ha canviat el reproductor de l'aplicació, anava integrat un reproductor d'un arxiu "tmp.wav" que creava l'aplicació, això feia que trigués un temps addicional al crear aquest arxiu, s'omplís i es reproduís, en la versió actual s'ha aconseguit fer desaparèixer aquest arxiu i que el so es reprodueixi a partir de ser carregat en memòria. En algun moment, fins i tot, s'ha intentat treballar amb algun procés en JAVA, ja que el Microsoft Visual Studio deixa també treballar amb llibreries creades en JAVA, per tal de poder fer servir funcionalitats que ajudarien a la implementació, però es va trobar un problema, a l'emulador s'havia de carregar un fitxer que pogués interactuar tots els dos llenguatges, però era tan gran que el dispositiu mòbil es quedava sense espai, i es va decidir prescindir d'ell.

En resum, les millores aconseguides es poden reflectir en la següent llista i es comentaran en el següent capítol:

- Optimització del codi
- Treballar amb fils en paral·lel
- Modificació del reproductor wav que millora problemes de memòria
- Treballar fils amb vàries paraules a la vegada
- Utilització de cerques en XML
- Utilització de JAVA
- Reproducció del so final des de la memòria i no des d'un fitxer.

Capítol 4

Desenvolupament i resultats

En aquest capítol s'explica com ha millorat l'aplicació poc a poc, totes les modificacions realitzades, les proves i millores que s'han anat implementant al llarg del desenvolupament del projecte. Finalment, es fa una comparativa de tots els resultats junts i estadístiques d'elles, sobretot fixant-nos en el temps de resposta inicial que és un dels colls d'ampolla més importants.

4.1 Introducció

Per a la utilització i realització del present projecte, es va fer anys enrere un treball previ molt laboriós sobre les transcripcions fonètiques, un estudi sobre el parla humana, la creació dels sons, com es forma la seva ona acústica, els sons que es formen, com s'uneixen entre ells. Sabem que el sistema final el que farà serà recitar en veu natural el text que se li doni d'entrada, ja sigui a partir d'un arxiu de text com des de la mateixa aplicació a la pantalla, per aconseguir això cal tenir en compte que tenim una base de dades de sons, que seran els sons possibles en la llengua, i que s'uniran entre ells, la figura 17 ens recorda l'esquema bàsic a seguir per aconseguir-ho.

4.2 Mòduls implementats i les seves millores

4.2.1 Format WAV

Durant tota l'elaboració del projecte s'ha treballat amb arxius wav, i amb unes llibreries que varen ser implementades en el programa original, i per tant s'han de tenir en compte a l'hora de fer-los servir. Es va fer servir arxius wav ja que és un estàndard i relativament fàcils de fer servir per treballar amb ells, ja que normalment no fan servir cap mena de compressió i tenen bona qualitat de so.

Un arxiu WAV té varis camps que ocupen 44 bytes a la capçalera i la resta són les dades del so, aquests camps són necessaris per a que l'aplicació pugui treballar amb els sons de les base de dades per poder-los unir, a continuació veiem com es composa la trama d'un arxiu wav, i s'explica el significat dels camps, sense entrar en gaire detall (Figura 21):

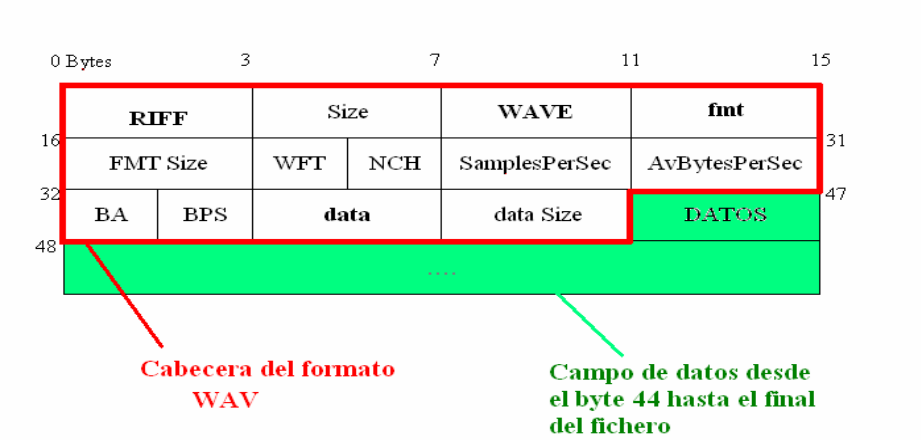

Figura 21: Camps d'un arxiu WAV

Nom del camp	Significat
RIFF	Inici de la capçalera (4 bytes)
Size	Tamany des del final del camp al final del wav (4 bytes)
WAVE	Indica el format de les dades (4 bytes)
Fmt	Especifica millor el format (4 bytes)
FMT Size	Indica el tamany dels sis camps següents (4 bytes)
WFT	Tipus de codificació soportat (PCM)
NCH	Número de canals (mono o estèreo)
SamplePerSec	Freqüència del mostreig del senyal
AvBytesPerSec	Mitja de bytes per segon
BPS	Bits per mostra
Data	Indica que a continuació apareixen les dades
DataSize	Tamany del camp de les dades a reproduir
Dades	Resta del wav amb les dades a reproduir

Taula 4: Significat dels camps d'un arxiu WAV

A la implementació inicial ja s'emmagatzemava a memòria principal la capçalera d'un arxiu wav de manera organitzada, gràcies a una estructura que es va dir WAVE_HEADER (veure codi font), juntament amb les funcions que es van crear d'una llibreria de funcions per a arxius wav per poder-los manipular, aquesta és la classe WavUtils (veure codi font) on podem trobar el WAVE_HEADER. En aquesta classe de WavUtils hi ha funcions que ajuden a llegir i transmetre les dades dels arxius guardats a les bases de dades cap al programa principal, es varen crear en el seu moment per solventar problemes a l'hora de manipular aquests arxius, ja que s'havia de fer a nivell de bits per poder reproduir-se correctament. Aquesta classe consta de funcions per a copiar fitxers wavs d'un lloc a altre de manera idèntica, moure's dins de l'arxiu a un paràmetre dins de la capçalera, carregar capçalera al WAVE_HEADER, concatenar fitxers wav, entre d'altres (veure codi font).

Gràcies a aquestes estructures ja creades ens ha sigut fàcil poder manipular els arxius de les bases de dades i com s'ha comprovat durant el procés de millora, no s'ha hagut de modificar cap d'aquestes classes, ja que no suposava cap cost addicional en el temps d'execució de l'aplicació.

4.2.2 El transcriptor fonètic

Aquest mòdul és el més llarg de tota l'aplicació, ja que ha de seguir moltíssimes regles fonètiques que s'han d'aplicar i per tant, s'han de fer moltes comprovacions. S'havia de tenir clar tots els conceptes provinents de la implementació inicial, com el nom dels fitxers wav, s'ha de saber clarament a quin so real pertany, una "rr" equival a "$" a la seva codificació, vocals tòniques "a", "e","i", "o", "u" (amb accent, "á", "é", "í", "ó", "ú") són "1", "2", "3", "4" i "5", entre d'altres, com es va explicar a la seva respectiva memòria.

A grans trets, el comportament del transcriptor és que al arribar una paraula del programa principal, l'analitza sintàcticament llegint els caràcters i en funció dels següents, s'apliquen unes determinades regles del castellà. Aquest retornarà la paraula transcrita que haurà de passar per la funció d'accentuació. El transcriptor és cridat des del programa inicial, al enviar l'ordre de reproducció, com s'explica més endavant, s'envia un fil per a cada paraula que la processi independentment i treballar de manera paral·lela, la crida al transcriptor és aquesta (Figura 22)

```
//iniciar estructures
//enviar dades al transcriptor
Thread th2 = new Thread(new ThreadStart(t.Transcribe));  //crear fil
th2.Start();  //llençar el fil pel transcriptor
```

Figura 22: Crida al transcriptor des del programa principal

Aquest transcriptor és capaç de llegir números de "0" a "9", però no més enllà, ja que estan codificats només aquests, si li entra "10", el reproductor dirà "uno cero", en comptes de "diez", només per a caràcters numèrics. És un procés difícil de tractar aquest

que ens ocuparia molt de temps, ja que caldria veure i estudiar la pronunciació dels números y com unir els sons veient quan es repeteixen.

El problema que presenta aquest mòdul és molt important, ja que al millorar-se diferents colls d'ampolla que hi havia inicialment a tota l'aplicació i tot haver-se optimitzat el codi de manera que desapareixessin els bucles i comparacions repetides, és el que gasta més temps de processament i fa que s'alenteixi tota l'aplicació al tractar les paraules, sobretot quan són llargues. Degut a la quantitat d'informació que es processa en el transcriptor, es fa difícil acotar aquest temps de resposta, s'ha comprovat que no hi hagi codi o comprovacions inservibles per a la transcripció per poder prescindir d'ells, però ha sigut impossible, ja que les comprovacions que hi ha són necessàries.

4.2.3 Funció accentuació

Aquesta funció és l'encarregada de fer la separació en síl·labes de la frase transcrita per poder aplicar-li les regles d'accentuació de les paraules en castellà, això vol dir assignar la síl·laba tònica corresponent a la paraula, tenint en compte l'accent ortogràfic i tònic. Es tenen en compte regles com per exemple els diftongs, o que les lletres per pr, pl, br, fr, fl, tr, dr, cl, cr, gr, gl seguides d'una vocal formen una síl·laba, o la consonant que es troba entre dos vocals forma síl·laba amb la segona vocal, diferenciació entre paraules planes, agudes o esdrúixoles, entre d'altres.

Bàsicament, aquesta funció el que fa és canviar la vocal de la síl·laba tònica pel seu identificador necessari, això vol dir que una paraula que tingui una "A" com a tònica dins de la síl·laba, la canviarà per un "1", la "E" per un "2" i així successivament, si reprenem l'exemple de la paraula "hola", al sortir pel transcriptor apareixeria com "_ola_" i ho recolliria la funció d'accentuació quedant com "_4la_".

Queda clar que el pas previ a aquesta funció és passar pel transcriptor, i la sortida, una vegada feta l'accentuació, ja és la frase o paraula completament ben transcrita i preparada per a que es pugui processar per buscar els sons adequats per a la correcta reproducció que estem esperant.

Aquesta part es va optimitzar part del codi, en els bucles es varen eliminar comparacions i iteracions que no eren necessàries, com per exemple transcrivint el codi generat amb “IF” per “SWITCH” que és més eficient, d’aquesta manera, el temps que triga en executar-se aquesta funció es pot considerar com despreciable, ja que gràcies a les comprovacions fetes amb ajuda de logs ens indicaven el un temps gairebé nul.

4.2.4 Paral.lelisme

Un dels principals coll d’ampolla trobat a l’inici de la implementació i que ens porta a buscar cada vegada més millores és el temps que triga l’aplicació des de que si envia l’ordre de que reprodueixi la frase que volem fins a que es reprodueix el primer so per l’altaveu del nostre dispositiu. A l’aplicació inicial agafava totes les paraules d’una frase i les processava de cop, deixant un espai en blanc entre aquestes paraules. Degut a que això era un cost important, es va pensar si seria possible crear fils que anessin processant totes les paraules a la vegada, es guardessin a un arxiu **tmpID.wav** (on ID és la posició que ocupa la paraula a la frase) i després es reproduís, això va comportar que hi hagués un problema afegit que calia solventar, es començaven a reproduir les paraules tanmateix es creaven els diferents arxius temporals finals wav, per tant, es va decidir crear un mecanisme d’ordenació de reproducció entre ells, consistia en que es creés un arxiu temporal buit, ID.tmp, a la mateixa vegada que el tmpID.wav i el reproductor comprovava l’existència d’aquest ID.tmp i reproduïa el seu corresponent tmpID.wav, però això no solucionava del tot el problema, ja que a vegades es donava el cas de que el reproductor intentés reproduir dos sons que arribaven a la vegada i es va intentar arreglar adormint els fils però no era la manera òptima, aquest era un problema de sincronia que s’ha arribat a resoldre gràcies a fer canvis en la manera de plantejar el reproductor, a l’apartat del reproductor s’explicarà més detalladament per les fases que ha sofert (Pàgina 57 – 4.2.5 El Reproductor)

Al fer servir els fils que treballen el paral·lel (Threads) s’ha de treballar amb les seves llibreries i treballar des de bon principi amb ells, per tant, al enviar l’ordre de

reproduir la frase o l'arxiu de text ja es llença una crida a un fil que fa que el reproductor es vagi executant, Reproduce és la classe que conté el reproductor:

```
Reproduce r= new Reproduce();  //inicialitzar estructures
Thread reproducir = new Thread(new ThreadStart(r.ReproduceWav));
reproducir.Start();
```

Figura 23: Crida al thread del reproductor

De la mateixa manera, es llençava un Thread per cada paraula que forma la frase que es vol reproduir, amb el seu número identificatiu per crear el tmpID.wav i ID.tmp, però en aquesta versió final, prescindim tant dels arxius tmpID.wav com dels ID.tmp, ja que gràcies a la utilització de fluxos de dades de memòria, s'ha aconseguit que el reproductor faci sonar des d'allà el resultat, en canvi sí que necessitem un identificador dels threads, que ens el dóna aquesta variable *numth (Figura 24)*

```
numth++;
Thread th2 = new Thread(new ThreadStart(t.Transcribe));
th2.Start();
```

Figura 24: Crida al thread transcriptor amb l'identificador adequat

Amb l'ús dels fluxos de memòria (streams) hem aconseguit part de la sincronització que necessitàvem, el reproductor estarà a l'espera des de que s'envia l'ordre d'execució, aquest conté un array d'streams (búffer), llavors es genera un fil per cada paraula que s'hagi de processar i col·loca el so transcrit a la seva posició dins d'aquest búffer. El reproductor final va comprovant de mode seqüencial el seu búffer durant tot el procés i quan s'omple la primera posició la reprodueix i passa a comprovar si està plena o no la següent posició, així successivament fins al final de la frase que es vol reproduir.

Per tant, d'aquesta manera estalviem temps gràcies a que ja no es creen arxius wav amb els resultats que posteriorment s'han d'obrir i tancar per a la seva reproducció, ni arxius que serveixin per a la sincronització, tot es fa a la memòria, fent servir els MemoryStream (Figura 25):

```
MemoryStream sonido=new MemoryStream();  //es crea la variable a
                        //memòria que contindrà el so final a reproduir
```

Figura 25: Ús de MemoryStream

El búffer de fluxos de memòria que es crea en el reproductor és el següent:

```
            public void Sound(MemoryStream stream)
             {
                   buf[identificador]=stream;
             }
```

Figura 26: L'stream deixa el seu contingut a la possició de búffer adequat

I el que fa el control de l'ordre de reproducció dins d'aquest búffer és el següent:

```
n=1;    //primera possició del búffer
if (buf[n]!=null)
{
      MemoryStream stream=buf[n];
      Llegim les dades del buffer
      Reproduir(buf[n])
      buf[n]=null;
      n++;
}
```

Figura 27: Reproducció del búffer

Però a més, es va decidir que fos en aquest punt on es controlés el problema que hi havia amb els espais en blanc entre paraules, és a dir, que per fer-lo més real, com ja hem comentat anteriorment, les persones al parlar, no deixen espais buits entre paraules i parlen seguit fins fer una pausa, per agafar aire, al trobar un punt, una coma o punt i coma. Es va decidir primer fer les proves unint paraules de la frase de dos en dos, y que aquestes dues paraules juntes fossin un Thread, com que es va observar que aquesta concatenació de paraules era viable es va decidir fer un control sobre un determinat número de caràcters, i per tant es va fer un control on el Thread eren deu caràcters, si es començava a llegir i abans del desè caràcter es trobava un signe de puntuació o espai en blanc, aquí es llençava el Thread, si en canvi la posició desena estava dins d'una paraula, es continuava fins a que acabés la paraula. Es va comprovar que aquesta via era viable, ja que donava un so final acceptable però deu caràcters continuaven sent pocs per a la parla humana, ja que volta sobre els cent caràcters seguits sense agafar aire,

però si fèiem servir aquesta mesura, el temps de resposta no era òptim ja que el coll d'ampolla passa a ser el transcriptor fonètic.

A continuació podem veure la inicialització de les estructures i el pas de paràmetres per crear els fils i el control que s'acaba de comentar:

```
...
Reproductor = New Thread Reproductor  //es crea un fil per al
                                      //reproductor
Reproductor.Iniciar()   //s'inicialitza el reproductor
Comptar els caràcters d'entrada

Per a cada paraula{
          Transcriptor = New Thread Transcriptor   //es llença un fil
                                                   //transcriptor
          S'envien les dades al transcriptor
          Comença la fase del transcriptor
         }
…
```

Figura 28: Inicialització d'estructures i pas de paràmetres per crear els fils

Com ja s'ha comentat, queda implementada la opció d'escollir un cert número de caràcters per llençar els fils per transcriure'ls. Aquest és un control senzill ja que només compta fins a un cert número de caràcters i si no ha trobat cap signe de puntuació, llença un transcriptor, i sinó, el llença abans, només caldrà canviar el paràmetre de control dins de l'aplicació en el seu moment oportú abans de la finalització total de l'aplicació, per tant, el pseudocodi d'aquesta part, realment queda de la següent manera:

```
...
Reproductor = New Thread Reproductor     //es crea un fil per al
                                         //reproductor
Reproductor.Iniciar()  //s'inicialitza el reproductor
Comptar els caràcters d'entrada

Per a cada caràcter{
          Transcriptor = New Thread Transcriptor   //es llença un fil
                                                   //transcriptor
          Si no encara no hem arribat al número de caràcters que volem
però hem trobat un signe
              de puntuació {
                   Reiniciem comptador
                   S'envien les dades al transcriptor
                   Comença la fase del transcriptor
           } sinó {
                   Continuem processant la frase
           }
…
```

Figura 29: Pseudocodi que llença el Reproductor i transcriptor

4.2.5 Utilització d'arxius de text com entrada

En aquest punt ha sigut senzill trobar una solució adequada; a la interfície s'ha creat un botó "Examinar" que va a buscar un arxiu del tipus txt, s'encarrega d'obrir-lo i tractar el seu contingut de la mateixa manera que els exemples que hi ha a la pantalla. Això fa que l'aplicació sigui també un lector de text, llavors s'ha de tenir molta cura amb l'entonació feta servir, els signes d'exclamació i d'interrogació, punts, comes... Un problema afegir a la lectura d'arxius de text és el temps de resposta, és capaç d'incrementar-se en dos segons ja que s'ha de manipular al anar a buscar-lo a disc, obrir-lo i llegir-lo.

Aquí veiem com tenim reduït a que l'arxiu d'entrada que s'agafi és un *.txt, una vegada feta aquesta comprovació, es procedeix a obrir l'arxiu i llegir-ho, i dins de la seva implementació treballa exactament igual que si s'agafessin els exemples inicials des de la pantalla, fent servir el paral·lelisme entre les paraules que formen el contingut de l'arxiu i les seves respectives inicialitzacions. A continuació podem veure el codi en C# d'aquesta part de la implementació:

```
// openFileDialog1
//
this.openFileDialog1.Filter = "Text Files(*.txt)|*.txt";
//

private void button1_Click(object sender, System.EventArgs e)
{
      if( this.openFileDialog1.ShowDialog() == DialogResult.OK )
      {
            System.IO.StreamReader sr = new
                  System.IO.StreamReader(openFileDialog1.FileName);

            try
                  {
                     … //igual que si ho fes des de la pantalla
                  }
                  catch(Exception es)
                  {
                        MessageBox.Show(es.ToString());
                  }
                        sr.Close();
                  }
                  else
                        MessageBox.Show("Se canceló la operación");

            }
```

Figura 30: Ús de txt com a entrada

4.2.6 Utilització del XML

En aquesta millora de l'aplicació, es crea un algoritme de cerca de recursos per trobar els sons adients basat en llenguatge XML, aquesta implementació és una millora a l'algoritme de cerca inicial que feia servir l'aplicació en anys anteriors, estava basat en un mètode que emmagatzemava les dades WAV en taules de dades relacionals a la memòria, s'havien creat quatre taules la de "Cuatrifonemas", "Trifonemas", "Difonemas" i "Fonemas". Aquestes taules contenien la informació necessària per a poder manipular els arxius wav, com la seva longitud, el nom... La seva creació i ampliació era fàcil, ja que feia servir el mètode **Add** per crear camps, i deixaven fer cerques directes o per patrons. Però aquestes taules s'havien d'omplir amb les dades pertinents dels arxius que es tenien guardats, indicant el nom, a la taula que han de pertànyer i el directori en el que es trobaven. Per fer això, hi havia un arxiu de text per a

cada taula que es volia crear, que s'havia de processar, obrir-lo, llegir-lo, agafar el nom dels sons que s'havien d'omplir i omplir la base de dades.

Però degut a la quantitat d'informació que s'ha de pujar a la memòria amb les taules ens trobàvem amb problemes, i l'emulador o el dispositiu ens demanava contínuament transformar en memòria per a programes, la memòria d'emmagatzemament. Es va decidir intentar implementar una manera més senzilla de fer aquesta cerca per millorar aquest problema i el rendiment en la cerca dels sons per poder-los concatenar-los. Per això, es va crear una estructura d'arbre amb tots els sons dels que disposem, i s'ha creat un per a cada taula, basat en el llenguatge XML, podem veure com s'implementaria l'arbre dels difonemes, on en ell s'indica l'identificador que ha d'anar a buscar l'aplicació per formar el so final, i el path on trobar-lo:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- tag que engloba los xml -->
<xml>
   <!-- tag que engloba a los difnonemas que comienzan por A -->
   <grupo id="A">
      <elemento id="a_" direccion="\Program
                           Files\SmartDeviceApplication1\a_.wav" />
      <elemento id="an" direccion="\Program
                           Files\SmartDeviceApplication1\an.wav" />
      <elemento id="ap" direccion="\Program
                           Files\SmartDeviceApplication1\ap.wav" />
      <elemento id="as" direccion="\Program
                           Files\SmartDeviceApplication1\as.wav" />
   </grupo>
   <grupo id="1">

      <elemento id="1#" direccion="\Program
                           Files\SmartDeviceApplication1\1#.wav" />
      <elemento id="1n" direccion="\Program
                           Files\SmartDeviceApplication1\1n.wav" />
      <elemento id="1x" direccion="\Program
                            Files\SmartDeviceApplication1\1x.wav" />
   </grupo>

   <!-- tag que engloba a los difnonemas que comienzan por B -->
   <grupo id="B">
      <elemento id="be" direccion="\Program
                           Files\SmartDeviceApplication1\be.wav" />
   </grupo>
...
  </grupo>


</xml>
```

Figura 31: arxiu XML que engloba els difonemes

De la mateixa manera es construeixen els demés arbres de cerca per a les altres bases de dades. Aquestes estructures es carreguen al iniciar l'aplicació, per tant queden en memòria la seva localització de manera fàcil i ràpid. Es carreguen tots els fitxers creats xml amb la següent rutina:

```
public XmlNodeList CargaXML(string nombreFichero)
{
      // Preparem un objecte del tipus XML

      XmlDocument xDoc = new XmlDocument();

      // Carreguem l'objecte amb la informació del XML

      xDoc.Load(@"\Program
                  Files\SmartDeviceApplication1\"+nombreFichero);

      // Recollim tots els elements en una llista

      XmlNodeList xml = xDoc.GetElementsByTagName("xml");

      lista = ((XmlElement)xml[0]).GetElementsByTagName("grupo");

      return lista;

}
```

Figura 32: Funció que carrega el contingut del XML

Per arribar a fer servir aquesta estructura, l'encarregat és el transcriptor, ja que fa la cerca de l'element que necessita, tal com es veia al gràfic del capítol anterior (Figura 19), al tallar la frase en subfrases de difonemes i fonemes, ha d'anar a buscar a la base de dades el so adequat, això ho fa amb la següent crida, si és un difonema:

```
direccion=LeeXML(lista_difo,""+x,segmento,"difonemas.xml");
```

Figura 33: Crida a la funció LeeXML

On crida la llista de difonemes que existeixen, amb el segment adequat, que és el so transcrit, en el seu XML corresponent, la funció LeeXML és la següent:

```
public string LeeXML(XmlNodeList lista, string difID, string elemID,
string nombreFichero)
{
      foreach (XmlElement nodo in lista) {

                  // del <difonema> recollim el seu ID

            string difonemaID = nodo.GetAttribute("id");

            // busquem el difonema amb l'ID que volem

            if (difonemaID==difID)
            {

            // llista amb els elements que tingui el TAG "elemento"

      XmlNodeList lista2 =
                ((XmlElement)nodo).GetElementsByTagName("elemento");

            // per a cada node de la llista, busquem el que volem

                  foreach (XmlElement subnodo in lista2)
                  {
                        //recollim el ID del <elemento>

                        string elementoID =
                                          subnodo.GetAttribute("id");

                        if (elementoID==elemID)
                        {
                        // recollim la direcció

                        string elementoDIR =
                                    subnodo.GetAttribute("direccion");

                        return elementoDIR;

                        }
                  }
      }
       }
      return null;
}
```

Figura 34: Funció LeeXML

D'aquesta manera hem aconseguit una cerca molt més simple que l'anterior, més fàcil d'estendre la base de dades, ja que només s'ha d'indicar la direcció necessària de l'arxiu wav i el seu id que volem posar dins de la base de dades, introduint-lo dins del

seu TAG corresponent al arxiu xml necessari. No cal recórrer tota l'estructura ja que va directe al TAG que necessita, això millora el rendiment de l'aplicació a l'hora de la cerca, a més, no col·lapsa la memòria. Amb l'ús del XML fem servir un llenguatge estàndard i el podem fer servir amb altres llenguatges, com el C# que és en el que corre l'aplicació, i a més a més, degut a la quantitat de dades que fa servir la base de dades, es poden manipular aquesta informació de forma ben estructurada.

4.2.7 Utilització de JAVA

Com ja s'ha comentat, treballar amb Visual Studio .NET i el Framework facilita la implementació de noves estructures, però a més a més s'ha vist que s'ha pogut incrustar llenguatge XML. Investigant, també s'ha trobat que dins de l'aplicació es podria fer servir parts en JAVA[5], com classes que ens ajudessin a problemes trobats al llarg del projecte, com podria ser la utilització de streams per a reproduir sons finals, que amb JAVA hi ha moltíssima informació i és un llenguatge que avui en dia cada vegada més gent el fa servir i aconsegueix dominar.

Després de molt investigar es va trobar la manera d'introduir JAVA en aplicacions que treballen en C#. Per fer això existeix el IKVM.NET que és una implementació de codi obert fet en JAVA per a aplicacions que treballen amb el Framework, consisteix en una màquina virtual de JAVA implementada en .NET, una implementació en .NET de classes de llibreries de JAVA i eines que fan que .NET i JAVA puguin cooperar.

Una vegada instal·lat aquest paquet en c:\ikvm i els executables i llibreries en C:\ikvm\bin, podem crear i compilar una classe en JAVA, una vegada fet això, on es podria dir aquesta classe JavaToNet.class, des de la línia de comandes cal fer el següent (Figura 35):

[5] http://www.codeproject.com/KB/cs/csharpikvm.aspx

Figura 35: Compilació de classes de Java per fer-les servir en .net

Això crearà una llibreria .dll (JavaToNet.dll) que es podria haver fet servir a la nostra aplicació. Però després de fer diverses proves, on una d'elles només calia que la classe fes una suma de 2+3 i ho mostrés a la pantalla de l'emulador, es va decidir que no es faria servir per limitacions del dispositiu, ja que al compilar l'aplicació, per a poder concordar aquests dos llenguatges, a l'emulador s'havia de pujar una llibreria dll, C:\ikvm\bin\IKVM.GNU.Classpath.dll, que ocupa 10MB, per tant, ens quedàvem sense espai físic al dispositiu, cosa que fa inviable aquesta opció i es va descartar directament. Per tant, les ajudes que oferien les APIs de JAVA i fòrums no ens poden ajudar en aquest aspecte.

4.2.8 El reproductor

El reproductor és una de les parts que més canvis ha sofert al llarg de la implementació degut a la seva evolució en el transcurs del projecte. De bon principi, es creava un únic arxiu **tmp.wav** que reproduïa tota la frase sencera de cop, el que feia que fos un gran problema degut al temps que trigava en començar a reproduir-se, sabem que aquest fitxer és la concatenació de diferents sons que han sigut gravats i que formen la frase que es vol reproduir, l'estructura **WaveHeader** ajudava a aquest fet de la següent manera, creava el wav i posava tota la informació a dins per reproduir-la de cop:

```
//Variables
uint tamanoFichero = tamano + 36;
string fichero_final = directorio + "tmp.wav";
FileStream ftmp = new FileStream(fichero_final, FileMode.Create);
BinaryWriter fBinW = new BinaryWriter(ftmp);

//Crear WAV
WUtils.WriteChars(fBinW, "RIFF");
fBinW.Write(tamanoFichero);
WUtils.WriteChars(fBinW,"WAVE");
WUtils.WriteChars(fBinW, "fmt ");
fBinW.Write(WaveHeader.FmtSize);
fBinW.Write(WaveHeader.wFormatTag);
fBinW.Write(WaveHeader.nChannels);
```

```
fBinW.Write(WaveHeader.nSamplesPerSec);
fBinW.Write(WaveHeader.nAvgBytesPerSec);
fBinW.Write(WaveHeader.nBlockAlign);
fBinW.Write(WaveHeader.wBitsPerSample);
WUtils.WriteChars(fBinW, "data");
fBinW.Write(tamano);
WUtils.WriteChars(fBinW, bufer);

//Cerrar
fBinW.Close();
ftmp.Close();
```

Figura 36: Estructura WAVE_HEADER

Aquí ja s'utilitzaven fluxos de dades en arxius per crear l'arxiu wav sencer, amb les seves capçaleres i dades finals, però veiem el problema comentat, ha de manipular-se el fitxer de tal manera que al final de posar les dades en ell es tingui que tancar, i no es pugui accedir a ell, ni reproduir-se durant la seva creació i tampoc es poden fer servir aquestes estructures mentre ho faci ell.

Quan ja s'aconseguia el fitxer a reproduir, el mateix programa principal reproduïa el **tmp.wav** de la següent manera:

```
try
{
      WaveOut wo = new WaveOut();
      wo.Play(fichero_final, 512*1024, 0xffff, 0xffff);
}
catch(Exception e)
{
      MessageBox.Show("Error al reproducir. " + e.ToString());
}
```

Figura 37: Reproducció inicial del fitxer tmp.wav

L'avantatja que trobem en aquest reproductor és que està integrat dins de l'aplicació i no cal cap programa extern que obri immediatament l'arxiu per reproduir-lo. Degut al caire que ha pres el projecte i de provar l'aplicació, es va decidir prescindir d'aquest reproductor, una de les principals raons era els problemes que hi havia amb la memòria, al fer cinc reproduccions seguides, tant l'emulador com el dispositiu es quedaven sense memòria i calia donar-li més o reiniciar-lo. Per tant, com que a la mateixa vegada es va decidir utilitzar threads per un treball en paral·lel de l'aplicació, i com s'ha explicat, queda llençat un fil reproductor esperant a que li arribi informació, es va fer una classe que només s'encarregués del reproductor, però això no arreglava els problemes que hi havia amb la memòria i ens col·lapsava tot el dispositiu.

Es va continuar fent proves sobre el reproductor i investigant maneres diferents d'aconseguir resoldre aquest problema, es va trobar ajuda a la web MSDN[6], on fent servir aquesta classe que proposen ells, usant els flags adequats, l'error de memòria quedava solventat, tot això fent servir llibreries pròpies del llenguatge. Tot i així, quedava present el problema del solapament de sons, no hi havia forma de controlar la finalització de la reproducció, per tant, hi havia sons que començaven a reproduir-se abans de que l'anterior parés, això ens va fer pensar en adormir els fils un cert temps abans de començar a reproduir-los però en moments en els que les paraules eren molt curtes quedava un gran espai buit entre elles, en aquest punt, encara s'utilitzava el control dels arxius **ID.tmp** per ordenar els diferents arxius **tmpID.wav** que es creaven.

Això va donar que pensar, en que s'hauria d'aconseguir d'alguna manera fer servir els fluxos de dades i intentar fer servir la memòria, ja que crear wav's finals i reproduir-los, consumia un cert temps i es considerava un dels colls d'ampolla de l'aplicació. La idea va ser carregar els arxius wav finals en memòria al crear-los per tal de reproduir-los des d'allà però no presentava cap millora important. Fins que es va decidir prescindir totalment dels arxius finals wav. Tal com s'ha explicat a l'apartat del paral.lelisme, es va aconseguir crear un array dins d'un MemoryStream i treballar directament d'allà, per tant, finalment, el reproductor no dóna problemes de memòria, es va aconseguir una millora tant en espai físic del disc, ja que no es graven arxius wav i es va aconseguir una bona millora de resposta de l'aplicació.

Les dades que rep aquest búffer de MemoryStream és enviat des de la classe del transcriptor al reproductor una vegada ha aconseguit transformar tot el text d'entrada en les dades de sortida pertinents, sempre treballant amb búffers interns que van acumulant aquestes dades mentre es va generant i finalment treballa com s'ha explicat anteriorment.

Aquests són els Flags i les llibreries que es fan servir a la classe del reproductor que ens ajuden a la bona utilització d'ell:

[6] http://msdn2.microsoft.com/es-es/library/ms229685(VS.80).aspx

```
private enum Flags
{
      SND_SYNC = 0x0000,  /* play synchronously (default) */
      SND_ASYNC = 0x0001,  /* play asynchronously */
      SND_NODEFAULT = 0x0002,  /* silence (!default) if sound not
found */
      SND_MEMORY = 0x0004,  /* pszSound points to a memory file */
      SND_LOOP = 0x0008,  /* loop the sound until next sndPlaySound */
      SND_NOSTOP = 0x0010,  /* don't stop any currently playing sound
      SND_NOWAIT = 0x00002000, /* don't wait if the driver is busy */
      SND_ALIAS = 0x00010000, /* name is a registry alias */
      SND_ALIAS_ID = 0x00110000, /* alias is a predefined ID */
      SND_FILENAME = 0x00020000, /* name is file name */
      SND_RESOURCE = 0x00040004  /* name is resource name or atom */

}
[DllImport("CoreDll.DLL", EntryPoint="PlaySound", SetLastError=true)]
private extern static int WCE_PlaySound(string szSound, IntPtr hMod,
int flags);

[DllImport("CoreDll.DLL", EntryPoint="PlaySound", SetLastError=true)]
private extern static int WCE_PlaySoundBytes (byte[] szSound, IntPtr
hMod, int flags);
```

Figura 38: Flags i llibreries que fa servir el nou reproductor

I la crida que fa que s'aconsegueixi la reproducció utilitza aquests flags ja que ajuda a que l'acció es faci de manera síncrona i des de la memòria, aquesta crida és la següent:

```
    WCE_PlaySoundBytes (m_soundBytes, IntPtr.Zero, (int)
                                (Flags.SND_SYNC |Flags.SND_MEMORY )
```

Figura 39: Crida a la reproducció de so a través de soundBytes i de manera síncrona

Amb tot això han quedat resolts problemes com el del col·lapse de la memòria, la utilització i creació d'arxius a disc, ara es pot reproduir des de la memòria i el reproductor ja no és un dels colls d'ampolla de l'aplicació.

Esquematitzant tot això, podem resumir el procés del reproductor de la següent manera:

Implementació inicial:

- *Utilització d'un fitxer temporal final wav per a la reproducció*: El transcriptor s'encarregava d'unir totes les partes en un únic fitxer que el reproductor s'encarregava de reproduir.
- *Generació de l'arxiu i reproducció seqüencial:* Fins que no s'acabava de crear completament el fitxer final, no es podia reproduir, això produeix un temps d'espera no desitjat abans de la reproducció.
- *Un arxiu per un so:* La utilització d'un arxiu com a font de so, implica que només pot reproduir una frase o paraula per arxiu, això perjudica sèriament la funcionalitat.
- *Problemes de memòria:* Mitjançant la classe WaveOut al reproduir, es consumia molta memòria després de realitzar vàries proves.

Les millores:

- *Generació paral·lela amb reproducció seqüencial:* Per a explotar al màxim els recursos dels que disposem, es va paral·lelitzar la generació dels sons, però a més a més, la reproducció havia de ser seqüencial i s'havia de sincronitzar, es va fer amb els identificadors, amb MemoryStream, búffers del reproductor.
- *Utilització de MemoryStreams* per a reproduir, ens permet anar omplint un stream mentre es reprodueix i treballar des de la memòria, així s'eviten els accessos a discs i la penalització en temps que comporta.
- *Búffer d'streams per a varis sons:* El reproductor es pot anar utilitzant fins que s'omplen tots els búffers, això implica que amb un únic búffer a memòria, es reprodueixin tots els sons que desitgem.
- *Es solventen els problemes de memòria:* Gràcies a la nova implementació de la classe reproductor recomanat per la llibreria msdn i la utilització de les banderes corresponents, queden solventats els problemes amb la memòria i ajuda a la sincronització dels sons.

4.3 Comparatives i estadístiques dels jocs de proves

Des del principi de la implementació del projecte, sempre s'han fet proves de la reproducció, sempre s'ha comprovat la qualitat del so, que el que es reproduïa fos correcte i concordés amb el que s'esperava. Una vegada això ja quedava clar, ens havíem de centrar en el temps de resposta de l'aplicació i veure com millorava cada vegada que s'implementava un dels passos anteriors. A continuació veurem diferents quadres on reflecteix aquest temps de resposta (des de que s'envia l'ordre de reproduir, fins que comença a fer-ho).

Les proves s'han fet sobre l'emulador de Windows CE i amb el dispositiu mòbil, amb la PDA Toshiba, aquestes proves han sigut amb diferents paraules que són les següents (Taula 5):

	PROVES
1	*Apéndice*
2	*apéndice 2 de mis documentos*
3	*mis documentos*
4	*Pantalla con mensajes*
5	*Ventana*
6	*2*
7	*documento con mensajes*
8	*mis mensajes*

Taula 5: Proves que es realitzen

Inicialment, com sabem, no treballàvem amb arxius de text per tant, només es provava l'aplicació directament amb els exemples que es tenien a pantalla:

- Emulador (Taula 6):

	PROVES	***PANTALLA***	***TXT***
1	*Apéndice*	6 seg	X
2	*apéndice 2 de mis documentos*	15 seg	X
3	*mis documentos*	8seg	X
4	*Pantalla con mensajes*	9 seg	X
5	*Ventana*	3 seg	X
6	*2*	2 seg	X
7	*documento con mensajes*	12 seg	X
8	*mis mensajes*	5 seg	X

Taula 6: Joc de proves inicial a l'emulador

- PDA (Taula 7):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice*	8 seg	X
2	*apéndice 2 de mis documentos*	18 seg	X
3	*mis documentos*	8 seg	X
4	*pantalla con mensajes*	10 seg	X
5	*Ventana*	3 seg	X
6	*2*	2 seg	X
7	*Documento con mensajes*	14 seg	X
8	*mis mensajes*	6 seg	X

Taula 7: Joc de proves inicial a la PDA

Una vegada fetes les comprovacions inicials, s'han fet diversos passos per anar traient els colls d'ampolla. En aquest pas, s'ha fet una optimització de codi original i s'han inserit threads els quals treballen per a cada paraula, un thread per paraula i crea un fitxer per cadascun d'ells, es crea la classe nova que reprodueix els nous fitxers ordenats i a més ja es pot reproduir a partir de txt. També hi ha un temps d'espera inicial de 4 segons com a delay per assegurar que s'omple el primer thread per a reproduir-se. Encara trobem varis problemes de memòria quan s'han de reproduir varis threads, ja que era a causa de l'emulador que ens l'omplia.

- Emulador (Taula 8):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice*	4 seg	4 seg
2	*apéndice 2 de mis documentos*	11 seg	11 seg
3	*mis documentos*	5seg	9 seg
4	*pantalla con mensajes*	7 seg	6 seg
5	*Ventana*	5 seg	6 seg
6	*2*	5 seg	5 seg
7	*documento con mensajes*	8 seg	8 seg
8	*mis mensajes*	5 seg	6 seg

Taula 8: Joc de proves amb txt i paral.lelisme a l'emulador

- PDA (Taula 9):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice*	Problema de memòria	Problema de memòria
2	*apéndice 2 de mis documentos*	Problema de memòria	Problema de memòria
3	*mis documentos*	7 seg	11 seg
4	*pantalla con mensajes*	8 seg	10 seg
5	*Ventana*	5 seg	10 seg
6	*2*	5 seg	5 seg
7	*documento con mensajes*	9 seg	11 seg
8	*mis mensajes*	6 seg	7 seg

Taula 9: Joc de proves amb txt i paral.lelisme a la PDA

El següent pas és el resultat de canviar el número de paraules que agafa el thread, es decideix agafar les paraules de 2 en 2, continua havent problemes de memòria i a vegades no es reprodueix bé. Ja que poden haver paraules llargues i que es sobreposin es va introduir un delay de 5 segons inicial.

- Emulador (Taula 10):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice*	5 seg	10 seg
2	*apéndice 2 de mis documentos*	7 seg	Problema de memòria
3	*mis documentos*	6 seg	9 seg
4	*Pantalla con mensajes*	7 seg	11 seg
5	*Ventana*	5 seg	10 seg
6	*2*	5 seg	7 seg
7	*Documento con mensajes*	8 seg	Problema de memòria
8	*mis mensajes*	5 seg	7 seg

Taula 10: Joc de proves agafant paraules de 2 en 2 a l'emulador

- PDA (Taula 11):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice*	8 seg	Problema de memòria
2	*apéndice 2 de mis documentos*	Problema de memòria	Problema de memòria
3	*mis documentos*	9 seg	11 seg
4	*Pantalla con mensajes*	10 seg	Problema de memòria
5	*Ventana*	7 seg	11 seg
6	*2*	5 seg	7 seg
7	*documento con mensajes*	Problema de memòria	Problema de memòria
8	*mis mensajes*	6 seg	8 seg

Taula 11: Joc de proves agafant paraules de 2 en 2 a la PDA

Degut a que els temps no milloren gaire, i sabent que una persona pot parlar sense fer pausa varis caràcters seguits sense parar a agafar aire, ara introduïm una petita modificació i fem la prova agafant threads de 10 caràcters, que és fàcil canviar-ho per un número superior que sigui més a prop a la que fan les persones. S'introdueix el control si troba una coma, punt i coma, punt o qualsevol altre signe de puntuació. El que indica el final de la prova i thread és el punt. També tenim el delay de 5 segons. Continuem tenim problemes amb la memòria.

- Emulador (Taula 12)

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice.*	5 seg	12 seg
2	*apéndice 2 de mis documentos.*	14 seg	Problema de memòria
3	*mis documentos.*	5seg	11 seg
4	*Pantalla con mensajes.*	5 seg	9 seg
5	*Ventana.*	5 seg	8 seg
6	*2.*	5 seg	8 seg
7	*documento con mensajes.*	7 seg	Problema de memòria
8	*mis mensajes.*	5 seg	7 seg

Taula 12: Joc de proves agafant threads de 10 caràcters

- PDA (Taula 13)

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice.*	7 seg	12 seg
2	*apéndice 2 de mis documentos.*	16 seg	Problema de memòria
3	*mis documentos.*	13 seg	18 seg
4	*Pantalla con mensajes.*	15 seg	19 seg
5	*Ventana.*	7 seg	11 seg
6	*2.*	7 seg	12 seg
7	*Documento con mensajes.*	15 seg	Problema de memòria
8	*mis mensajes.*	9 seg	14 seg

Taula 13: Joc de proves agafant threads de 10 caràcters a la PDA

En aquest cas continuem amb la mateixa filosofia, agafant threads de 10 caràcters, el canvi nou que s'ha fet ha sigut fer servir una implementació d'un reproductor més senzill i que ens ha fet estalviar-nos molt de codi. A més, ens estalvia tot el problema de memòria, encara que a vegades hi ha problemes de solapament de sons.

- Emulador (Taula 14):

	PROVA	***PANTALLA***	***TXT***
1	*apéndice.*	5 seg	5 seg
2	*apéndice 2 de mis documentos.*	9 seg	So sol.lapat
3	*mis documentos.*	8 seg	8 seg
4	*Pantalla con mensajes.*	8 seg	11 seg
5	*Ventana.*	5 seg	5 seg
6	*2.*	5 seg	5 seg
7	*documento con mensajes.*	9 seg	13 seg
8	*mis mensajes.*	5 seg	5 seg

Taula 14: Joc de proves a l'emulador amb un nou reproductor

- PDA (Taula 15):

	PROVA	***PANTALLA***	***TXT***
1	*apéndice.*	7 seg	5 seg
2	*apéndice 2 de mis documentos.*	So sol.lapat	So sol.lapat
3	*mis documentos.*	10 seg	13 seg
4	*Pantalla con mensajes.*	11 seg	13 seg
5	*Ventana.*	7 seg	7 seg
6	*2.*	7 seg	10 seg
7	*documento con mensajes.*	13 seg	15 seg
8	*mis mensajes.*	7 seg	9 seg

Taula 15: Joc de proves a la PDA amb un nou reproductor

Aquí hem decidit prescindir de les taules que es carreguen a memòria i fer la cerca a través d'arxius XML en comptes de recórrer tota la memòria, a més s'introdueixen fluxos de dades per reproduir, es continua mantenint l'arxiu wav final però es carrega en un stream de dades per a reproduir-lo.

- Emulador (Taula 16):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice.*	5 seg	7 seg
2	*apéndice 2 de mis documentos.*	8 seg	10 seg
3	*mis documentos.*	4 seg	5 seg
4	*Pantalla con mensajes.*	5 seg	8 seg
5	*Ventana.*	3 seg	4 seg
6	*2.*	3 seg	4 seg
7	*documento con mensajes.*	7 seg	9 seg
8	*mis mensajes.*	3 seg	5 seg

Taula 16: Joc de proves usant XML a l'emulador

- PDA (Taula 17):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice.*	7 seg	9 seg
2	*apéndice 2 de mis documentos.*	13 seg	16 seg
3	*mis documentos.*	7 seg	9 seg
4	*Pantalla con mensajes.*	12 seg	16 seg
5	*Ventana.*	5 seg	7 seg
6	*2.*	3 seg	4 seg
7	*documento con mensajes.*	9 seg	13 seg
8	*mis mensajes.*	5 seg	7 seg

Taula 17: Joc de proves usant XML a la PDA

Per últim, podem veure les proves finals i el que ha trigat cada un dels exemples anteriors. En aquest moment, es prescindeix dels arxius finals wavs, recordem que ara es treballa reproduint des d'un búffer de memòria (MemoryStream) que va reproduint els sons tan bon punt van arribant a ell i ho fa de manera ordenada, sense problemes de memòria i solapament de sons entre ells:

- Emulador (Taula 18):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice.*	3 seg	4 seg
2	*apéndice 2 de mis documentos.*	7 seg	9 seg
3	*mis documentos.*	2 seg	2 seg
4	*Pantalla con mensajes.*	4 seg	4 seg
5	*Ventana.*	2 seg	2 seg
6	*2.*	1 seg	1 seg
7	*documento con mensajes.*	4 seg	5 seg
8	*mis mensajes.*	1 seg	2 seg

Taula 18: Joc de proves final a l'emulador reproduint des de MemoryStreams

- PDA (Taula 19):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice.*	5 seg	7 seg
2	*apéndice 2 de mis documentos.*	14 seg	16 seg
3	*mis documentos.*	2 seg	2 seg
4	*Pantalla con mensajes.*	7 seg	8 seg
5	*Ventana.*	3 seg	3 seg
6	*2.*	2 seg	2 seg
7	*documento con mensajes.*	9 seg	11 seg
8	*mis mensajes.*	3 seg	3 seg

Taula 19: Joc de proves final a la PDA reproduint des de MemoryStreams

Per acabar amb aquest tema, a continuació podem veure les comparacions dels temps de resposta de l'aplicació entre la implementació inicial que calia millorar i l'actual, si veiem les comparacions entre els temps en l'emulador:

- Inicial (Taula 6):

	PROVES	***PANTALLA***	***TXT***
1	*Apéndice*	6 seg	X
2	*apéndice 2 de mis documentos*	15 seg	X
3	*mis documentos*	8seg	X
4	*pantalla con mensajes*	9 seg	X
5	*Ventana*	3 seg	X
6	*2*	2 seg	X
7	*documento con mensajes*	12 seg	X
8	*mis mensajes*	5 seg	X

Taula 6: Joc de proves inicial a l'emulador

- Final (Taula 18):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice.*	3 seg	4 seg
2	*apéndice 2 de mis documentos.*	7 seg	9 seg
3	*mis documentos.*	2 seg	2 seg
4	*Pantalla con mensajes.*	4 seg	4 seg
5	*Ventana.*	2 seg	2 seg
6	*2.*	1 seg	1 seg
7	*documento con mensajes.*	5 seg	5 seg
8	*mis mensajes.*	1 seg	2 seg

Taula 18: Joc de proves final a l'emulador reproduint des de MemoryStreams

I si mirem els temps en el dispositiu físic:

- Inicial (Taula 7):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice*	8 seg	X
2	*apéndice 2 de mis documentos*	18 seg	X
3	*mis documentos*	8 seg	X
4	*pantalla con mensajes*	10 seg	X
5	*Ventana*	3 seg	X
6	*2*	2 seg	X
7	*documento con mensajes*	14 seg	X
8	*mis mensajes*	6 seg	X

Taula 7: Joc de proves inicial a la PDA

- Final (Taula 19):

	PROVA	***PANTALLA***	***TXT***
1	*Apéndice.*	5 seg	7 seg
2	*apéndice 2 de mis documentos.*	14 seg	16 seg
3	*mis documentos.*	2 seg	2 seg
4	*Pantalla con mensajes.*	7 seg	8 seg
5	*Ventana.*	3 seg	3 seg
6	*2.*	2 seg	2 seg
7	*Documento con mensajes.*	9 seg	11 seg
8	*mis mensajes.*	3 seg	3 seg

Taula 19: Joc de proves final a la PDA reproduint des de MemoryStreams

Podem veure aquests temps de millora gràficament, podem veure la millora a l'emulador, on podem veure les barres en color blau com a les proves inicials i les vermelles les finals:

	PROVES
1	*Apéndice*
2	*apéndice 2 de mis documentos*
3	*mis documentos*
4	*Pantalla con mensajes*
5	*Ventana*
6	*2*
7	*documento con mensajes*
8	*mis mensajes*

Taula 5: Proves que es realitzen

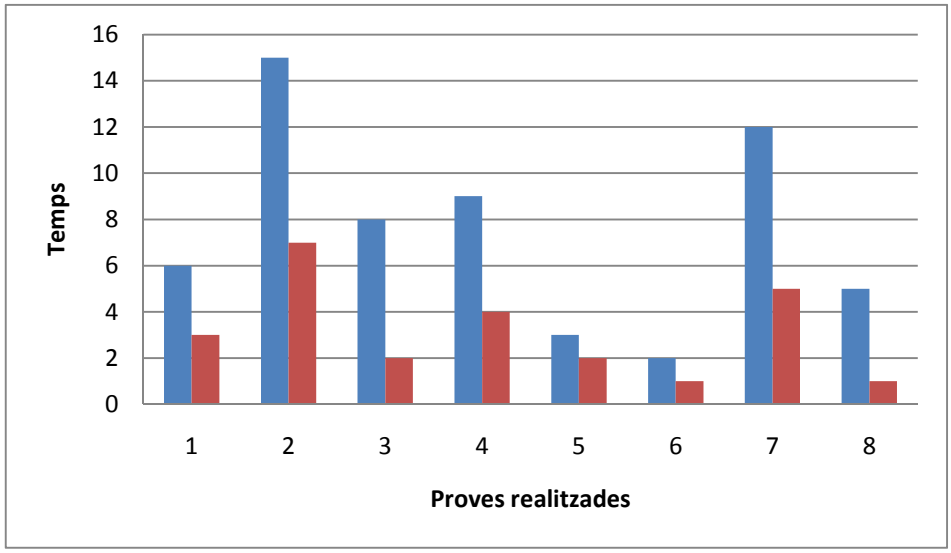

Figura 40: Comparació de les proves realitzades sobre l'emulador

De la mateixa manera veiem la comparativa de les proves a la PDA com:

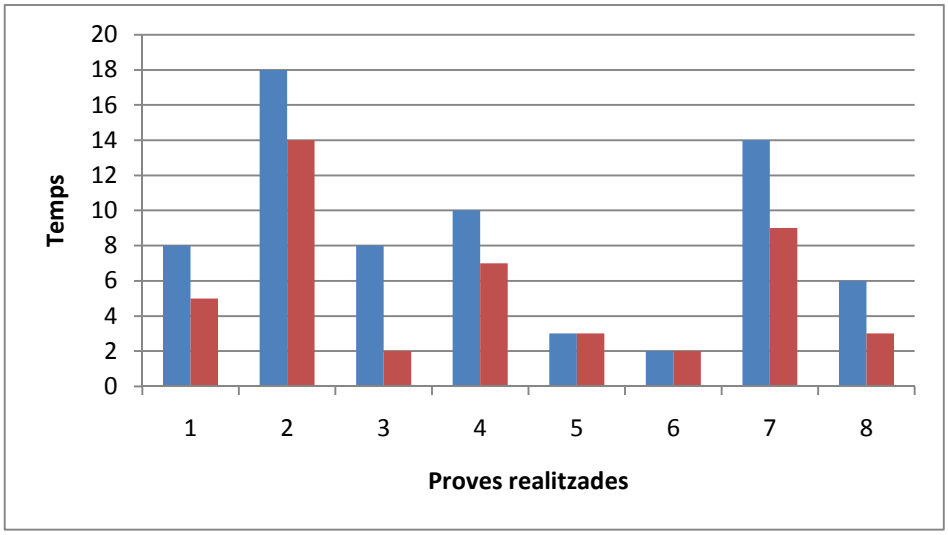

Figura 41: Comparació de les proves realitzades sobre la PDA

Podem veure el temps guanyat a cada una de les proves, a les gràfiques s'observa com a cadascuna de les proves hem guanyat temps, veiem a les següents taules quines són aquests guanys, restant-li als temps de l'aplicació inicial, els actuals (Taula 20):

	PROVES	***A L'EMULADOR***	***A LA PDA***
1	*Apéndice*	*3 seg*	*3 seg*
2	*apéndice 2 de mis documentos*	*8 seg*	*4 seg*
3	*mis documentos*	*6 seg*	*6 seg*
4	*Pantalla con mensajes*	*5 seg*	*3 seg*
5	*Ventana*	*1 seg*	*0 seg*
6	*2*	*1 seg*	*0 seg*
7	*documento con mensajes*	*7 seg*	*5 seg*
8	*mis mensajes*	*4 seg*	*3 seg*

Taula 20: Taula de guanys temporals en la realització del projecte

Amb aquestes dades podem recollir una mitja de temps de millora per l'emulador i per a la PDA per a aquestes proves, fent la suma d'ells i dividint-lo entre el número de proves. Per a l'emulador, tindríem un guany mig de 4.37 segons, i per al dispositiu mòbil de 3 segons.

Capítol 5

Treball futur i conclusions

En aquest últim capítol es reflexa les conclusions a les que s'ha arribat una vegada finalitzat el projecte i com millorar-lo en un treball futur.

5.1 Conclusions

S'ha aconseguit millorar un motor de veu natural per a dispositius mòbils que treballa sobre plataformes Windows Mobile i Windows CE i que tenia alguns problemes de memòria i de temps de resposta. Es va plantejar de manera que s'anessin resolvent el problema del temps de resposta, juntament amb els que posteriorment anessin sortint, com el solapament de sons o problemes de memòria, també han sigut solventat a mida que anaven sortint. Per tant, aquests colls d'ampolla s'han anat reduint, ja que gràcies a l'ajuda de les implementacions produïdes s'han anat minimitzant, fins arribar en alguns punts, com en la reproducció, a arribar a quantitats de temps gairebé despreciables. En total s'ha aconseguit un guany mig en segons, entre emulador i dispositiu físic, d'uns gairebé 4 segons, tal com es pot veure a la Taula 20. Per tant, els objectius que s'han anat seguint al llarg de la realització del projecte s'han complert.

Aquestes millores s'han aconseguit gràcies a les següents implementacions:

- Lector de pantalla i text: Ara és una aplicació que pot reproduir paraules que hi ha a la pantalla de l'aplicació com des d'arxius de text.

- Optimització del codi: Neteja del codi en bucles i comprovacions repetides o innecessaris.

- Treballar en paral·lel: Fer que l'aplicació treballi en fils a la vegada.

- Modificació reproductor: Es solventen els problemes de la memòria.

- Utilització de cerques en XML: S'utilitza un llenguatge estructurat per a gestionar les bases de dades.

- Reproducció a traves d'streams: Es reprodueix el so final a través de fluxos de memòria i no a través de fitxers.

Totes aquestes millores, tal com s'ha vist en els jocs de proves, han ajudat a resoldre els problemes de memòria, temps de resposta i de procés del sistema, ja que són punts crítics en aplicacions d'aquest tipus. A més, s'han solucionat situacions delicades com el solapament de sons entre ells i s'ha fet un sistema més simple de gestionar, sobretot a nivell de base de dades al treballar amb llenguatges estructurats com el XML. S'ha comprovat clarament que pot fer-se servir en un dispositiu mòbil com una PDA i per tant és exportable a qualsevol telèfon o emulador amb sistemes operatius de Windows.

Per tant, aquest sistema s'apropa encara més a la finalitat que esperem, que sigui un sistema capaç d'ajudar a gent invident a fer servir un dispositiu mòbil, ja que a través d'aquest motor de veu pot reproduir de manera ràpida i clara paraules i texts.

5.2 Treball futur

El projecte actual deixa obertes vàries tasques que s'haurien de realitzar per tal de millorar l'aplicació en un futur:

- Reduir el coll d'ampolla que representa el transcriptor fonètic, estudiar un algoritme més estructurat i més ràpid per a recollir la informació d'aquest.

- Una vegada reduït el coll d'ampolla del transcriptor, fer una reproducció fluida entre les paraules, unint-les.

- Lectura de números i acrònims del transcriptor.

- Estudiar la entonació a frases senceres per poder ser un bon lector de text.

- Millorar-lo com a lector de pantalla usant el cursor.

- Adaptar el transcriptor a altres llengües.

Capítol 6

Bibliografia i annexes

A continuació es poden veure annexes que han servit per a la realització del projecte com parts del codi font fet servit, també es pot consultar la biografia usada, tant la bàsica, com la consultada i pàgines webs.

Bibliografia

1. QUILIS, A. (1993) *Tratado de fonología y fonética españolas*. Madrid: Gredos (Biblioteca Románica Hispánica, Manuales, 74).

2. RÍOS, A. (1999) *La transcripción fonética automática del Diccionario Electrónico de Formas Simples Flexivas del español: un estudio fonológico en el léxico.* Estudios de Lingüística del Español 4. http://elies.rediris.es/elies4/

3. KLATT, D.H. (1987) *Review of Tex-to-Speech Conversion for English.* Journal of the Acoustical Society of America 82,3: 737-793; in ATAL, B.S.& MILLER, L.J.& KENT, R.D. (Eds.) (1991) Papers in Speech Communication: Speech Processing. New York: Acoustical Society of America. pp. 57-114.

4. DUTOIT, T. (1996) *An Introduction to Text-To-Speech Synthesis*. Kluwer Academic Publishers, 326 pp.

5. NAVARRO TOMÁS, T. (1918) *Manual de pronunciación española*. Madrid: Consejo Superior de Investigaciones Científicas, Instituto Miguel de Cervantes (Publicaciones de la Revista de Filología Española, III). 21ª edición, 1982. - Madrid Consejo Superior de Investigaciones Científicas (Textos Universitarios, 3), 25ª edición, 1991.

6. GIL, J. – LLISTERRI, J. (2004) *Fonética y fonología del español en España (1978 – 2003)*, Lingüística Española Actual 26, 2: 5-44. ISSN: 0210-6345 http://liceu.uab.es/~joaquim/publicacions/Gil_Llisterri_04_Fonetica_Espanol.pdf

Bibliografia web

1. Organización Nacional de Ciegos de España.
   http://www.once.es

2. Llibreries MSDN
   http://msdn.microsoft.com

3. Ús de XML en C#
   http://www.devjoker.com/contenidos/Articulos/29/Como-leer-XML-con-C.aspx

4. Ús de JAVA en C#
   http://www.codeproject.com/KB/cs/csharpikvm.aspx

5. Reproductor de sons
   http://msdn2.microsoft.com/es-es/library/ms229685(VS.80).aspx

6. Trancripción fonética automática
   http://elies.rediris.es/elies4/Cap2.htm

7. Lectors de pantalla
   http://es.wikipedia.org/wiki/Lector_de_pantalla

8. Aplicacions de proves, ATT&T Labs
   http://www.research.att.com/~ttsweb/tts/demo.php

9. Flite
   http://www.viksoe.dk/code/flite.htm

10. Sobre els fonemes
    http://es.wikipedia.org/wiki/Fonema

11. Webs de programació en C#
    http://msdn2.microsoft.com/es-es/library/67ef8sbd(VS.80).aspx
    http://www.programacion.com/tutorial.php?id=csharp
    http://www.koders.com/zeitgeist/csharp/

12. Fòrums de MSDN per C#
http://forums.microsoft.com/MSDN-ES/ShowForum.aspx?ForumID=298&SiteID=11

13. Altres fòrums de programació en C#
    http://www.es-asp.net/Foro/foro-c--f.aspx
    http://www.lawebdelprogramador.com/news/new.php?id=227&texto=C%20sharp

Annexe A

- WAVE_HEADER (de la classe WaveUtils):

```
      public struct WAVE_HEADER
      {
            public string RiffID; // max 4 chars
            public uint RiffSize;
            public string WaveId; // max 4 chars
            public string FmtID; // max 4 chars
            public uint FmtSize;  //
            public ushort wFormatTag;
            public ushort nChannels;
            public uint nSamplesPerSec;
            public uint nAvgBytesPerSec;
            public ushort nBlockAlign;
            public ushort wBitsPerSample;
            public string DataID; //max 4 chars
            public uint nDataBytes; //Longitud de los datos

      };
```

- RellenaWAVE_HEADER (de la classe WaveUtils)

```
#region esta función RellenaWAVE_HEADER coge la cabecera de un wav y
la mete en los campos de la estructura WAVE_HEADER
            public void RellenaWAVE_HEADER(FileStream fwav, ref
WAVE_HEADER waveHeader, ref BinaryReader fBinR) //SE LE PASA el primer
wav de todos
            {
                  // 1º cogemos el filesize del RIFF
                  this.SeekToFileSize(fwav);
                  waveHeader.RiffSize = fBinR.ReadUInt32();
                  // 2º cogemos el resto de los parametros del WAVEfmt
                  this.SeekToWavefmt(fwav);
                  waveHeader.FmtSize = fBinR.ReadUInt32();
                  waveHeader.wFormatTag = fBinR.ReadUInt16();
                  waveHeader.nChannels = fBinR.ReadUInt16();
                  waveHeader.nSamplesPerSec = fBinR.ReadUInt32();
                  waveHeader.nAvgBytesPerSec = fBinR.ReadUInt32();
                  waveHeader.nBlockAlign = fBinR.ReadUInt16();
                  waveHeader.wBitsPerSample = fBinR.ReadUInt16();
                  // 3º cogemos el data size
                  this.SeekToDataSize(fwav);
                  uint dataId = fBinR.ReadUInt32();
                  waveHeader.nDataBytes = fBinR.ReadUInt32();

                  //Ya tenemos en la estructura toda la cabecera
            }
```

- Classe Reproduce

```
public class Reproduce
{
      private byte[] m_soundBytes;
      private enum Flags
      {
            SND_SYNC = 0x0000,  /* play synchronously (default) */
            SND_ASYNC = 0x0001,  /* play asynchronously */
            SND_NODEFAULT = 0x0002,  /* silence (!default) if sound
                                        not found */
            SND_MEMORY = 0x0004,  /* pszSound points to a memory file
            SND_LOOP = 0x0008,  /* loop the sound until next
                                   sndPlaySound */
            SND_NOSTOP = 0x0010,  /* don't stop any currently playing
                                  sound */
            SND_NOWAIT = 0x00002000, /* don't wait if the driver is
                                        busy */
            SND_ALIAS = 0x00010000, /* name is a registry alias */
            SND_ALIAS_ID = 0x00110000, /* alias is a predefined ID */
            SND_FILENAME = 0x00020000, /* name is file name */
            SND_RESOURCE = 0x00040004  /* name is resource name or
                                          atom */

            }

      [DllImport("CoreDll.DLL", EntryPoint="PlaySound",
                                                SetLastError=true)]
      private extern static int WCE_PlaySound(string szSound,
                                          IntPtr hMod, int flags);

      [DllImport("CoreDll.DLL", EntryPoint="PlaySound",
                                                SetLastError=true)]
      private extern static int WCE_PlaySoundBytes (byte[]szSound,
                                         IntPtr hMod, int flags);


      public string directorio = @"\Program
                                    Files\SmartDeviceApplication1\";
            //public string fichero_final;
            public int contador=1;
            public FileStream ftmp;
            MemoryStream[] buf = new MemoryStream[20];

            public int identificador;


            /// <summary>
            /// Construct the Sound object to play sound data from the
            specified stream.
            /// </summary>
            public void Sound(MemoryStream stream)
            {
                  buf[identificador]=stream;
            }


            public Reproduce()
            {
```

```
                        }
            public void ReproduceWav()
            {
                  int n=1;
                  for (int cont2=0;cont2<1000;cont2++)
                  {


                  if (buf[n]!=null)
                  {
                        MemoryStream stream=buf[n];
                        // read the data from the stream
                        m_soundBytes = new byte
                                          [stream.GetBuffer().Length];
                        m_soundBytes=stream.GetBuffer();

                        WCE_PlaySoundBytes (m_soundBytes, IntPtr.Zero,
                          (int) (Flags.SND_SYNC | Flags.SND_MEMORY ));
                        buf[n]=null;
                        n++;

                  }

            }
      }
```

- Funció del programa principal que dóna l'ordre de transcriure i reproduir la frase i crea tots els fils.

```
private void transcribir_Click(object sender, System.EventArgs e)
{
      if(listaFrases.SelectedItem.ToString() != null)
      {
            try
            {
            String entrada = listaFrases.SelectedItem.ToString();

            char p;
            int chartotal=0;
            foreach (char c in entrada)
            {
                  p=c;
                  chartotal++;
            }
            int numchar=0;
            string cacumulada="";
            string f="";

            Reproduce r= new Reproduce();  //iniciar estructuras
            Thread reproducir = new Thread(new
                                    ThreadStart(r.ReproduceWav));
            reproducir.Start();
            numth=0;

            foreach (char c in entrada)
            {
```

```
                    numchar++;
                    TranscriptorCF t = new TranscriptorCF();  //iniciar
                    t.BdD=BdD;
                    t.directorio=directorio;
                    t.indice1=indice1;
                    t.indice2=indice2;
                    t.indice3=indice3;
                    t.indice4=indice4;
                    t.WaveHeader=WaveHeader;
                    t.lista_difo=lista_difo;
                    t.lista_fo=lista_fo;
                    t.reproductor=r;

                    if (numchar<1 && (c=='.' || c==';' || c==':'))
                    {
                          numchar=0;
                          cacumulada=f+c;
                          numth++;
                          t.identificador=0+numth;
                          t.frase=cacumulada;
                          t.reproductor=r;
                          Thread th2 = new Thread(new
                                        ThreadStart(t.Transcribe));
                          th2.Start();
                          f="";
                    }
                    else if (numchar>=1 && (c=='.' || c==' ' || c==';'
                                                          || c==':'))
                    {
                          numchar=0;
                          if (c=='.' || c==' ' || c==';' || c==':')
                            {cacumulada=f;}
                          else{cacumulada=f+c;}
                          numth++;
                          t.identificador=0+numth;
                          t.frase=cacumulada;
                          Thread th2 = new Thread(new
                                          ThreadStart(t.Transcribe));
                          th2.Start();
                          f="";
                    }
                    else
                    {
                          f=f+c;
                    }
              }
        }
        catch(Exception es)
        {
        MessageBox.Show(es.ToString());
        }
}
```

Resum

S’ha optimitzat un motor de veu natural per a dispositius mòbils com una PDA o un telèfon mòbil basat en un sistema operatiu Windows. La finalitat d’aquest treball és la d’ajudar a fer més fàcil la utilització d’aquests aparells a gent invident i que pugui acabar sent una donació a la ONCE.

Resumen

Se ha optimizado un motor de voz natural para dispositivos móviles como una PDA o un teléfono móvil basado en un sistema operativo Windows. La finalidad de este trabajo es la de ayudar a hacer más fácil la utilización de estos aparatos para gente invidente i que pueda acabar siendo una donación a la ONCE.

Summary

A natural voice motor has been optimized for mobile devices such as PDAs or mobile phones based on a Windows operating system. The purpose of this work is to help make these devices easy to use for blind people, therefore it may even end up being a donation to ONCE.