Nela
Un entrenador de Braille para niños

TRABAJO DE FIN DE CARRERA

Enrique Matías Sánchez

Dpto. de Ingeniería Electrónica y de Comunicaciones
Escuela Universitaria Politécnica de Teruel
Universidad de Zaragoza

Mayo 2013

NELA
Un entrenador de Braille para niños

Memoria que presenta para optar al título de
Ingeniero Técnico en Informática de Gestión

Enrique Matías Sánchez

Dirigida por la Doctora

Inmaculada Plaza García

TRIBUNAL:

Presidente: _____

Secretario: _____

Vocal: _____

CALIFICACIÓN: _____

Dpto. de Ingeniería Electrónica y de Comunicaciones
Escuela Universitaria Politécnica de Teruel
Universidad de Zaragoza

Mayo 2013

A la memoria de mi abuela,
que cantaba y me contaba fábulas.

Para Samuel,
a quien regalo canciones y cuentos.

Agradecimientos

A mi familia, por todo.

A Bencer, Carolina, David Charro, Exekias, Jordi, Jorge, Isaac, Luisja, Marga, Meskio, Miquel, Nanna, Queru, Seajob, Xavi... Tal vez no lo sepan, pero su inspiración y ejemplo me ha sido de gran ayuda para llegar hasta aquí.

A todos los amigos en Teruel y Zaragoza que me han apoyado durante estos años (¡son demasiados para nombrarlos a todos!).

A los organizadores del VI Concurso Universitario de Software Libre, que fue un gran incentivo para mí.

A Bruno y Lorena, por su extraordinario trabajo al frente de la Oficina de Software Libre de la Universidad de Zaragoza.

A toda la comunidad del Software Libre: Debian, la FSF, KDE, hacklabs y hackmeetings... Nela no existiría sin ella.

A Inés Benedicto y Nuria Tregón, por su asesoría desinteresada sobre las necesidades de los niños con discapacidad visual.

A Uxue y los docentes que han probado Nela y han enviado comentarios y sugerencias. Saber que hay profesionales del campo que lo encuentran útil me llena de satisfacción.

A Inma, por su dedicación y paciencia.

Resumen

Es preciso que adquieras un don precioso...
es preciso que aprendas a leer.

Benito Pérez Galdós, *Marianela*

Este proyecto ha consistido en el desarrollo de un programa informático («Nela») que ayudase en el aprendizaje de la escritura usando el código Braille. Está dirigido a niños con discapacidad visual (visión reducida o ceguera) que hayan alcanzado la madurez necesaria para abordar la lectoescritora.

El programa muestra una imagen y solicita al niño que escriba una palabra, simulando una máquina Perkins con el teclado del ordenador. A continuación informa de si la respuesta ha sido correcta, y muestra la palabra escrita en Braille. Toda comunicación se realiza tanto de forma visual como sonora, sin requerir el uso del ratón y empleando letras de gran tamaño y diseñadas para una legibilidad óptima.

Se hace uso desde el primer momento de un vocabulario con significado afectivo y vivencial, introduciendo paulatinamente nuevas palabras con nuevas sílabas y nuevas letras, con dificultad progresiva. El programa se adapta al ritmo de aprendizaje de cada usuario, no introduciendo nuevas palabras hasta que no se hayan asimilado las anteriores, y reforzando las que le resulten más complicadas al niño.

Programado en C++ usando las bibliotecas Qt, es multiplataforma (funciona tanto en Microsoft Windows como en GNU/Linux) y está internacionalizado (preparado para traducirse a otros idiomas).

Este trabajo pretende ser una muestra más de la adecuación y potencial del software libre en la educación. Ha sido valorado positivamente por varios educadores, y obtuvo el primer premio en la fase estatal del VI Concurso Universitario de Software Libre, así como el premio especial de accesibilidad.

Palabras clave

Braille, lectoescritura, aprendizaje, ceguera, discapacidad visual, aplicación educativa, enseñanza, escuela.

Índice

Índice de figuras

Capítulo 1

Introducción

Resumen: Este capítulo presenta una breve introducción a Nela. El lector podrá hacerse una idea de qué es y para qué sirve. También se encuentra aquí una descripción del resto de capítulos de la memoria.

1.1. Introducción

El propósito de este proyecto es el desarrollo de un programa informático («Nela») que ayude en el aprendizaje de la escritura usando el código Braille, dirigido a niños con discapacidad visual[1] (visión reducida o ceguera) que hayan alcanzado la madurez necesaria para abordar la lectoescritora.

No abunda en el mercado el software dirigido a los discapacitados visuales, y debido a su limitado interés comercial, suele ser caro. El escaso software educativo disponible proviene en su mayoría de organizaciones de ciegos o de instituciones educativas. En particular, el destinado al aprendizaje de la lectoescritura es prácticamente inexistente.

En el año 2007, dos profesionales que trabajaban con niños con discapacidad visual en la provincia de Teruel (Lucía Azara Arruebo e Inés Azucena Benedicto Moya) expresaron sus necesidades a los miembros del grupo de investigación EduQTech[2] de la Universidad de Zaragoza, planteando el desarrollo de recursos que facilitasen la enseñanza y aprendizaje de los caracteres del código Braille y cubrir la laguna que en esos momentos existía.

De esta forma surgió el proyecto «TerBraille», que se definió pensando en dos etapas:

1. Una primera etapa en la que los usuarios pudieran conocer el aprendizaje de la escritura del código Braille a través de un dis-

[1] Discapacidad visual: término que engloba cualquier tipo de patología visual grave, ceguera total y deficiencia visual, en sus distintos grados de pérdida visual.

[2] `http://www.unizar.es/eduqtech/`

positivo electrónico. En concreto, se centró en la enseñanza de los caracteres alfanuméricos.

2. Una segunda etapa, en la que se introduciría a los usuarios en el manejo del ordenador, reforzando el aprendizaje de los caracteres alfanuméricos a la vez que extendiendo el aprendizaje a sílabas y palabras. Para ello se definiría un programa informático basado en el uso de la máquina Perkins, que posteriormente les permitiría trabajar con programas comerciales existentes actualmente en el mercado.

Los recursos generados en las dos etapas deberían poder utilizarse de forma independiente y complementaria, apoyando a los profesionales en aquellos aspectos más necesarios para los usuarios. Como requisito inicial se decidió que habrían de desarrollarse siguiendo una filosofía de software y hardware libre, y desde el momento de su concepción se pretendió que pudieran ser un referente como ejemplo de investigación y transferencia de resultados desarrollados en la provincia de Teruel. Los resultados obtenidos se están dando a conocer en Teruel, así como en foros especializados a nivel autonómico, nacional e internacional.

Los estudiantes Óscar Díaz Sáez [46] y David Murciano Ibáñez [117], a través de sendos Trabajos Fin de Carrera dirigidos conjuntamente por los doctores Carlos Medrano e Inmaculada Plaza, crearon la primera versión del dispositivo electrónico, basado en la plataforma Arduino. El dispositivo fue probado y evaluado inicialmente en un colegio de Teruel durante un año, y posteriormente por profesionales vinculados a la Once de Zaragoza, quienes lo utilizaron durante algunas semanas, dando una opinión global del dispositivo. Jorge Izquierdo Najes [81] ha realizado una segunda versión del dispositivo electrónico, basado esta vez en un microcontrolador Pic. En la Universidad Carnegie Mellon se ha desarrollado paralelamente *Automated Braille Writing Tutor*[3], otro dispositivo con similares objetivos y que se está probando en la India.

El presente trabajo constituye la segunda etapa del proyecto. Junto a estos dispositivos electrónicos mencionados, Nela forma un *kit* de enseñanza/aprendizaje de la lectoescritura usando Braille capaz de satisfacer las necesidades de usuarios y profesionales y así cubrir la laguna actualmente existente en educación infantil. Al formar parte de un proyecto global más amplio, se garantiza la continuidad del programa tras la presentación de este Trabajo de Fin de Carrera.

Si bien el programa va dirigido a niños ciegos y con dificultades visuales, su utilización se puede extrapolar a otros niños que no presentan estas discapacidades, pudiendo servir como herramienta para acercar el código Braille a las aulas. Así mismo, puede constituir una herramienta útil para adolescentes

[3] `http://www.techbridgeworld.org/brailletutor/`

y adultos que pierdan la visión durante sus primeras etapas de adaptación y aprendizaje, para docentes que deban trabajar con discapacitados visuales y para padres de niños de estas características.

El principal entorno de aplicación será la escuela, pero el programa está pensado para que pueda usarse de forma autónoma por el niño, por lo que también puede emplearse en el hogar sin apenas necesidad de atención parental.

El primer prototipo del software obtuvo:

- el primer premio en la fase final del VI Concurso Universitario de Software Libre[4], en el que compitieron a nivel estatal 99 proyectos de un total de 139 estudiantes.
- el premio especial de accesibilidad de dicho concurso.
- el primer premio en la fase local de Aragón, organizada por la Oficina de Software Libre de la Universidad de Zaragoza[5].

A raíz de estos reconocimientos el proyecto tuvo bastante repercusión en los medios locales, nacionales y extranjeros, incluyendo entrevistas en cadenas de radio y prensa escrita.

1.2. Estructura del trabajo

Esta memoria consta de los siguientes capítulos:

Introducción Capítulo en el que nos encontramos. Presenta brevemente el contexto de trabajo del proyecto, los objetivos de la aplicación a realizar y la estructura de la memoria.

El acceso a la lectoescritura En este capítulo se exploran los métodos que se utilizan hoy en la escuela con carácter general, y sus planteamientos psicopedagógicos.

El acceso de los discapacitados visuales a la lectoescritura En las memorias de los Trabajos de Fin de Carrera anteriormente mencionados se explora de forma bastante amplia la discapacidad visual, y las soluciones y dispositivos actuales. En esta memoria se profundizará en un aspecto concreto: la adquisición de la lectoescritura. En este capítulo se estudian las distintas soluciones que se han propuesto hasta llegar al estándar actual, el código Braille, del que se detallarán sus características y métodos de enseñanza.

[4] `http://www.concursosoftwarelibre.org/1112/`
[5] `http://osluz.unizar.es/cusl`

La filosofía del software libre Una de los puntos de partida de este proyecto es que todos los resultados debían ser libres. Para entender la importancia de este planteamiento, En este capítulo se estudia en detalle la filosofía del software libre: su origen, características, tipos de licencia, metodología de desarrollo, etc.

Apoyos tecnológicos: normativa que deben cumplir En este capítulo se pormenorizan los requisitos de accesibilidad que deben cumplir las aplicaciones informáticas para personas con discapacidad (norma UNE 139802:2009).

Descripción de los programas previos En este capítulo se expondrán los programas para la enseñanza-aprendizaje del Braille existentes con anterioridad, y sus características.

Definición de requisitos En este capítulo se recoge la especificación de requisitos de software para el entrenador de Braille para niños.

Tecnologías y herramientas empleadas En este capítulo se abordan las cuestiones referentes a las tecnologías usadas: lenguaje de programación elegido, entorno de desarrollo, herramientas utilizadas, etc.

Pruebas, propuestas de mejora y líneas de trabajo futuro En este capítulo se expone el resultado de las primeras pruebas piloto, y las ampliaciones que se van a realizar o que podría ser interesante implementar.

Conclusiones En este capítulo se reflexiona sobre las conclusiones generales del desarrollo del proyecto, se analizan los retos afrontados por el alumno y se termina con una pequeña conclusión a nivel personal.

Bibliografía Listado de las referencias usadas durante el desarrollo.

Anexo Se incluye como anexo el artículo escrito por invitación de la revista Novática, ya revisado y aprobado para su publicación en el próximo número.

Capítulo 2

El acceso a la lectoescritura

Resumen: En este capítulo se exploran los métodos que se utilizan hoy en la escuela con carácter general para la adquisición de la lectoescritura, y sus planteamientos psicopedagógicos.

El aprendizaje de la lectoescritura se puede dividir en tres etapas:

Apresto Es una etapa de preaprendizaje o preparación. En ella se hacen ejercicios de memoria visual y auditiva, pronunciación, percepción de detalles en láminas y dibujos, motricidad fina de la mano, etc. hasta alcanzar la madurez necesaria para pasar a la siguiente etapa.

Aprendizaje Ésta es la etapa de aprendizaje propiamente dicha, utilizando alguno de los múltiples métodos de enseñanza y materiales disponibles.

Perfeccionamiento El desarrollo de la lectura tiene lugar durante todo el periodo de escolaridad. Sus objetivos son ampliar los intereses del alumno, perfeccionar su modo de leer y capacitarlo para manejar cualquier tipo de texto.

Hay tres clases de métodos de enseñanza de la lectoescritura: sintéticos, analíticos y eclécticos.

2.1. Métodos sintéticos

Los métodos sintéticos parten del conocimiento de las letras aisladas, luego la formación de las sílabas y poco a poco la lectura y escritura de palabras, frases y oraciones, empleando las letras y combinaciones silábicas que ya conocen. Al conocer más letras y combinaciones silábicas, se va aumentando la capacidad de lectura y escritura de los niños, hasta que llegan a conocer todo el alfabeto y pueden leer y escribir todo.

Siguiendo este método, cuando se enseñan nuevas palabras, éstas deben contener letras y combinaciones silábicas que ya han sido aprendidas con anterioridad.

2.1.1. Método alfabético o deletreo

Entre las diversas modalidades de los métodos sintéticos está el alfabético, así llamado por seguir el orden del alfabeto durante el aprendizaje, y que se ha usado desde la antigüedad.

En la antigüedad la instrucción iba siempre de lo simple a lo complejo, de lo elemental a lo compuesto. Cualquier otro procedimiento habría parecido absurdo, como sostenían todavía San Ambrosio y San Agustín [96]. Por tanto, era preciso aprender primero las letras, después las sílabas, las palabras aisladas, las frases y, por fin, lo textos corridos. Nunca se iniciaba una nueva etapa sin haber agotado antes todas las dificultades de la precedente, y esto no se lograba sin emplear mucho tiempo en cada etapa. No había ninguna preocupación por despertar el interés del niño ni por su psicología.

Dionisio de Halicarnaso, en su libro Περὶ συνθέσεως ὀνομάτων (*Sobre la composición literaria*) dice:

> Cuando nos enseñan a leer, primero aprendemos los nombres de las letras, después sus formas y valores, y a su debido tiempo las sílabas y sus modificaciones, y finalmente las palabras y sus propiedades.

καὶ πᾶσαν αὐτῶν διακόψαι τὴν φλυαρίαν. τί δ' ἐστὶ τοῦτο; τὰ γράμματα ὅταν παιδευώμεθα, πρῶτον μὲν τὰ ὀνόματα αὐτῶν ἐκμανθάνομεν, ἔπειτα τοὺς τύπους καὶ τὰς δυνάμεις, εἶθ' οὕτω τὰς συλλαβὰς καὶ τὰ ἐν ταύταις πάθη, καὶ μετὰ τοῦτο ἤδη τὰς λέξεις καὶ τὰ συμβεβηκότα αὐταῖς, ἐκτάσεις τε λέγω καὶ συστολὰς καὶ προσῳδίας καὶ τὰ παραπλήσια τούτοις· ὅταν δὲ τὴν τούτων ἐπιστήμην λάβωμεν, τότε ἀρχόμεθα γράφειν τε καὶ ἀναγινώσκειν, κατὰ συλλαβὴν <μὲν> καὶ βραδέως τὸ πρῶτον· ἐπειδὰν δὲ ὁ χρόνος ἀξιόλογος προσελθὼν τύπους ἰσχυροὺς αὐτῶν ἐν ταῖς ψυχαῖς ἡμῶν ἐμποιήσῃ, τότε ἀπὸ τοῦ ῥᾴστου δρῶμεν αὐτὰ καὶ πᾶν ὅ τι ἂν ἐπιδῷ τις βιβλίον ἀπταίστως διερχόμεθα ἕξει τε καὶ τάχει ἀπίστῳ. τοιοῦτο δὴ καὶ περὶ τὴν σύνθεσιν τῶν

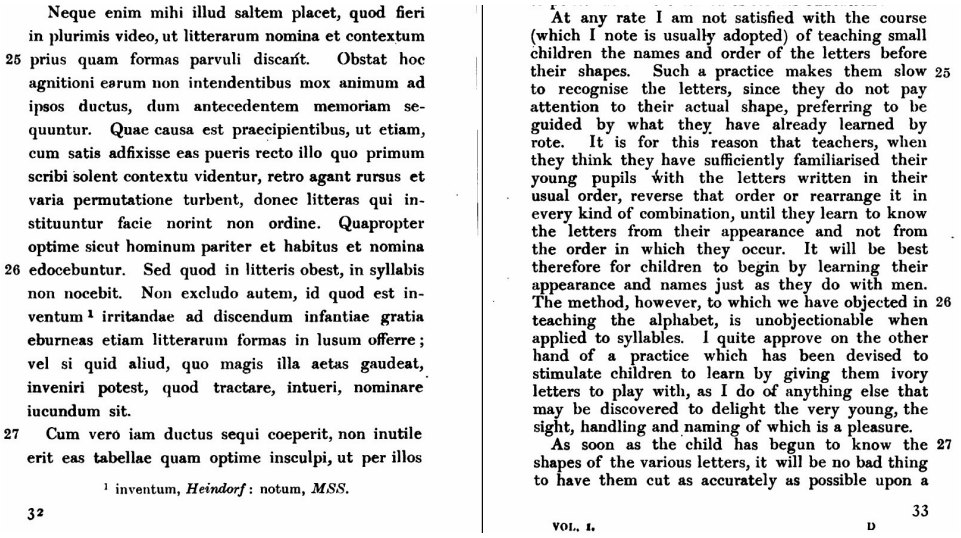
Why pursue the subject? A fact familiar to all of us is enough to silence these quibblers. What may this be? When we are taught to read, first we learn off the names of the letters, then their forms and their values, then in due course syllables and their modifications, and finally words and their properties, viz. lengthenings and shortenings, accents, and the like. After acquiring the knowledge of these things, we begin to write and read, syllable by syllable and slowly at first. And when the lapse of a considerable time has implanted the forms of words firmly in our minds, then we deal with them without the least difficulty, and whenever any book is placed in our hands we go through it without stumbling, and with incredible facility and speed. We must suppose that something of

Figura 2.1: Detalle de una edición bilingüe del libro de Dionysius Halicarnassensis [44]

Se comenzaba, pues, por el alfabeto: el niño aprendía, por orden, todas las letras, llamándolas por su nombre (alfa, beta, gamma...) si tener inicialmente a la vista sus formas. Logrado el primer objetivo, se hacía recitar el alfabeto en orden inverso. Cumplida esta primera etapa, les tocaba el turno a las sílabas: con igual rigor sistemático se hacía aprender, por orden, la seria silábica completa, sin que pudiera pasarse a los vocablos antes de haber agotado todas las combinaciones (primero las más simples, y después los grupos más complejos). Concluido finalmente el aprendizaje de las sílabas, podría

ahora pasarse al estudio de la palabra, primero monosilábicas, y después de dos o más sílabas. También aquí nos vemos en el extremo opuesto de la pedagogía actual: lejos de facilitar las cosas al niño por medio de una selección de voces simples, se le ponía de improviso en presencia de la máxima dificultad (términos médicos, trabalenguas...), por estimarse que, superados estos obstáculos, todo el resto marcharía solo. Por último se llegaba a la lectura de textos breves y fragmentos poéticos [17].

Será preciso aguardar hasta la época romana para que aparezcan algunos esfuerzos destinados a facilitar el aprendizaje. Marco Fabio Quintiliano (35–95 dC) pensaba que la aptitud para aprender era natural en el ser humano, y que se debía reforzar con premios en vez de con castigos. Recomendaba aprender los nombres de las letras y su aspecto a la vez, dejar jugar a los niños con letras talladas en marfil, y crear juegos competitivos entre grupos de niños, así como ejercicios preparatorios mediante un estilete que debía pasarse por las letras ahuecadas en una tablita, para que se adquiriese soltura de mano. Mantenía sin embargo el aprendizaje de las sílabas de la forma tradicional, y el uso de palabras difíciles aisladas.



Neque enim mihi illud saltem placet, quod fieri in plurimis video, ut litterarum nomina et contextum
25 prius quam formas parvuli discant. Obstat hoc agnitioni earum non intendentibus mox animum ad ipsos ductus, dum antecedentem memoriam sequuntur. Quae causa est praecipientibus, ut etiam, cum satis adfixisse eas pueris recto illo quo primum scribi solent contextu videntur, retro agant rursus et varia permutatione turbent, donec litteras qui instituuntur facie norint non ordine. Quapropter optime sicut hominum pariter et habitus et nomina
26 edocebuntur. Sed quod in litteris obest, in syllabis non nocebit. Non excludo autem, id quod est inventum [1] irritandae ad discendum infantiae gratia eburneas etiam litterarum formas in lusum offerre; vel si quid aliud, quo magis illa aetas gaudeat, inveniri potest, quod tractare, intueri, nominare iucundum sit.
27 Cum vero iam ductus sequi coeperit, non inutile erit eas tabellae quam optime insculpi, ut per illos

[1] inventum, *Heindorf*: notum, *MSS.*

32

At any rate I am not satisfied with the course (which I note is usually adopted) of teaching small children the names and order of the letters before their shapes. Such a practice makes them slow 25
to recognise the letters, since they do not pay attention to their actual shape, preferring to be guided by what they have already learned by rote. It is for this reason that teachers, when they think they have sufficiently familiarised their young pupils with the letters written in their usual order, reverse that order or rearrange it in every kind of combination, until they learn to know the letters from their appearance and not from the order in which they occur. It will be best therefore for children to begin by learning their appearance and names just as they do with men. The method, however, to which we have objected in 26
teaching the alphabet, is unobjectionable when applied to syllables. I quite approve on the other hand of a practice which has been devised to stimulate children to learn by giving them ivory letters to play with, as I do of anything else that may be discovered to delight the very young, the sight, handling and naming of which is a pleasure.
As soon as the child has begun to know the 27
shapes of the various letters, it will be no bad thing to have them cut as accurately as possible upon a

33

VOL. I. D

Figura 2.2: Detalle de una edición bilingüe de *Institutio oratoria* de Quintiliano [139]

En este método, como en la mayoría de los demás, las sílabas se estudian en un orden de complejidad creciente :

1. sílabas directas (consonante-vocal: ca, ma, te, ra, li, bo...).
2. sílabas inversas (vocal-consonante: as, el, en, ad, un...).
3. sílabas mixtas o directo-inversas (consonante-vocal-consonante: pal, car, bur, tes...).

4. sílabas trabadas abiertas, directas dobles o sinfones (con una «r» o «l» en mitad de la sílaba: pla, bre, fra, pri...).

5. sílabas trabadas cerradas o mixtas dobles (trabadas que terminan en consonante: pren, gran, tron, blan, flan...).

Generalmente se evita introducir los diptongos y triptongos hasta que el proceso esté bastante avanzado.

Este método de enseñanza de la lectoescritura no posee ninguna ventaja. Es un método memorístico, que rompe con el proceso normal de aprendizaje de la mentalidad infantil. Da mucha importancia a la decodificación (forma, nombre y sonido de las letras), desatendiendo lo principal, que es comprender lo leído. El niño que aprende a leer con este método, se acostumbra a deletrear, por lo que el aprendizaje y comprensión de la lectura es lento. Actualmente no tiene vigencia pedagógica.

2.1.2. Método fonético o fónico

El método fonético (*phonics* en inglés) no inicia la enseñanza con el nombre de las letras (por ejemplo, «efe»), sino con su sonido («fff»). Con ello se facilita la unión silábica tanto en sílaba directa o libre, como inversa o trabada. Encierra la dificultad de pronunciar el sonido (puro, sin vocal) de las consonantes oclusivas (d, b, p...). Dentro de este grupo hay variedades como el onomatopéyico y el gestual o kinestésico.

Figura 2.3: Página de *Orbis Pictus* [9]

Generalmente usando ilustraciones de un objeto que comience por esa letra o que produzca ese sonido. En 1527, Valentin Ickelsamer, el padre de la gramática alemana, fue el primero en colocar en columnas paralelas la imagen de un animal, su nombre impreso y la letra cuyo sonido fuera más similar a la voz o grito del animal [73, 142, 102]. Johann Amos Comenius publicó *Orbis sensualium pictus* en 1657, que fue el primer libro de texto ilustrado y el más popular durante más de un siglo, siendo traducido a 20 idiomas. En él cada letra iba acompañada de un dibujo representativo.

Este método evita el deletreo, pero puede resultar monótono y mecánico, descuidando el significado.

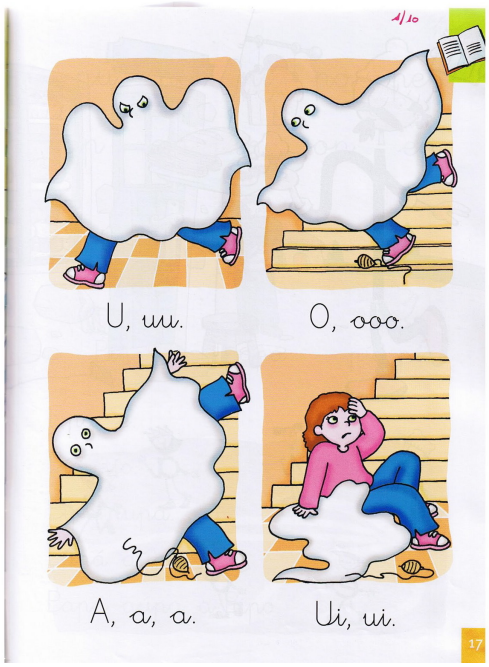


Figura 2.4: Página de *Letrilandia*, un método moderno y muy popular [179]

2.1.3. Método silábico

Fue introducido por el pedagogo aragonés Vicente Naharro (1750–1823), quien deja dicho que los nombres de las letras se han de enseñar a los niños acompañadas de una e, en caso de enseñárselas, y no sueltas como antes se enseñaban en los alfabetos ordinarios, porque aquella enseñanza era muy perjudicial a los adelantamientos de los niños [120].

Empieza por el aprendizaje de las vocales, y se continúa introduciendo las letras directamente combinadas formando sílabas (primero las directas primero, y posteriormente las inversas, trabadas y diptongadas). Cuando ya se cuenta con varias sílabas se unen para formar palabras y luego se

construyen oraciones.

Como apoyo a estos métodos nacieron los silabarios: listas de palabras sin sentido que podían leerse de izquierda a derecha o de arriba a abajo y en las que sólo cambiaba una vocal. En Latinoamérica tuvo bastante éxito el *Método de lectura en quince cuadros* de Juan Manuel Bonifaz [25].

Figura 2.5: Página de la cartilla *Nuevas Letras* de Wenceslao Ezquerra [50]

Omite el deletreo del método alfabético y la pronunciación de los sonidos de las letras por separado. Sigue un orden lógico en su enseñanza. Sin embargo, es poco motivante para el niño.

2.2. Métodos analíticos o globales

Los métodos analíticos o globales parten de la lectura de palabras o frases que luego se dividen en sílabas y palabras, respectivamente, para su aprendizaje. Fueron introducidos por Jean-Ovide Decroly (1871–1932), aunque tuvo precursores como Nicolas Adam (1716–1792), Friedrich Gedike (1754–1803) y Joseph Jacotot (1770–1840) [128]. Algunas variantes son el método de oraciones completas, el ideovisual y el método natural de Célestin Freinet.

Se fundan en que los niños perciben los objetos y palabras globalmente, y luego pasan a los detalles. Por esto se parte de la lectura de frases o palabras y se llega poco a poco al reconocimiento de las sílabas y letras que las constituyen. Sus partidarios defienden que el método global analítico es el que mejor contempla las características del pensamiento del niño.

Figura 2.6: Páginas de *Poquito a poco* [92]

Los métodos analíticos o globales se caracterizan porque desde el primer momento se le presentan al niño unidades con un significado completo. El método global consiste en aplicar a la enseñanza de la lectura y escritura el mismo proceso que sigue en los niños para enseñarles a hablar. El niño, gracias a su memoria visual, reconoce frases y oraciones y en ellas las palabras. También de manera espontánea establece relaciones y reconoce las elementos idénticos en la imagen de dos palabras diferentes. La palabra escrita es el dibujo de una imagen que evoca cada idea.

Los signos dentro de las palabras tienen un sentido, y de su presentación escrita son transformados en sonidos hablando, y el hecho de comprender enteras las palabras y la oración permite una lectura inteligente y fluida desde el principio.

Es un método agradable, sin mecanizaciones que lo hagan aburrido. La enseñanza es activa y como el niño lee desde el principio, le da la impresión que desde el principio sabe leer. Permite la lectura y la escritura, y propicia la adquisición de una ortografía correcta. Facilita el aprendizaje de la lectura con rapidez y comprensión, sin el tanteo y el titubeo de los métodos sintéticos.

Se usa con éxito con niños con síndrome de Down o con trastorno del espectro autista (por ejemplo, el método Leo con Lula [118]).

2.3. Métodos eclécticos o mixtos

Entre las metodologías sintéticas y analíticas han surgido una serie de metodologías eclécticas que intentan vencer las limitaciones de los métodos especializados anteriores tomando los aspectos más valiosos y significativos de ellos, con el propósito de facilitar el aprendizaje de la lectoescritura.

(a) Página de *Erstes Lesebuch* de Klauwell [88]

(b) Página de *Método práctico de alfabetización fundamentado en palabras generadoras* [114]

Figura 2.7: Métodos de palabras normales.

El primer método ecléctico fue el de las palabras normales, atribuido a Johann Karl Christoph Vogel (1795–1862), con mejoras de Adolf Klauwell (1818-1879). Otros lo atribuyen a Johann Baptist Graser (1766–1841).

Este método consiste en partir de una palabra normal, con sentido, denominada también generadora o generatriz, asociada a una imagen. Se hace copiar, leer, e identificar entre otras palabras. Posteriormente se descompone en sílabas y después en letras. Un segundo paso consiste en combinar esos elementos para formar nuevas palabras.

Este método es analítico-sintético por partir de la palabra a la sílaba y de esta a la letra; y sintético porque también va de la letra a la sílaba y de esta a la palabra. Para la enseñanza de cada letra nueva, se usa una palabra normal nueva. La palabra normal constará de una consonante nueva, si acaso lleva otras serán ya conocidas por los educandos.

Este método sigue el proceso natural del aprendizaje, y fomenta desde el principio del aprendizaje la compresión de la lectura. Permite que los alumnos tengan la oportunidad de ver diariamente el avance del proceso de aprendizaje y de apreciar su propio progreso en la lectura y escritura.

El maestro debe conocer el proceso antes de aplicarlo. No se deben usar palabras que no respondan a los intereses infantiles.

Otros métodos muy conocidos son el de Antonio Paláu, basado en el silábico, y el método Micho, basado en el onomatopéyico.

Figura 2.8: Detalle del *Método fotosilábico Paláu* [129]

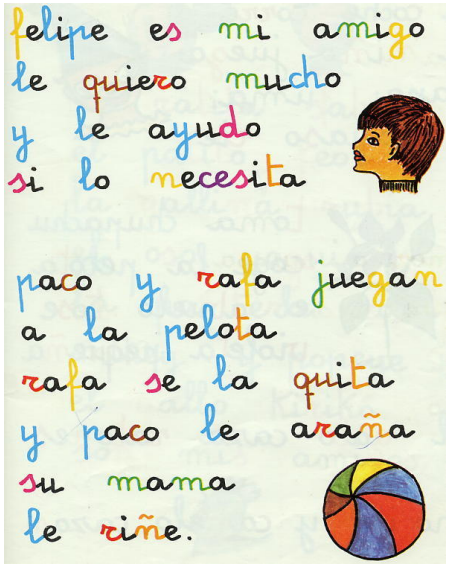


Figura 2.9: Página de *Micho* [23]

2.4. Enfoque constructivista

En la actualidad se están desarrollando métodos de enfoque constructivista que parten de la premisa de que se pueden escribir y leer textos aún antes de dominar el código alfabético, vinculando el lenguaje escrito desde el inicio a su función fundamental: comunicar. A la vez que las características del sistema alfabético, se aprenden las características del lenguaje escrito y sus distintas finalidades.

Se basa en los planteamientos de Lev Semionovich Vygotsky (1896–1934), Jean Piaget (1896–1980) y otros. Uno de sus propulsores más conocidos es el brasileño Paulo Reglus Neves Freire (1921–1997).

Emilia Ferreiro y Ana Teberosky [55] observaron que sus alumnos elaboraban hipótesis y adquirían conocimientos sobre la escritura y la lectura al margen de lo que se les enseñaba dentro del sistema escolar.

Estas hipótesis se elaboran durante un periodo de tiempo que suele comenzar a los dos años o dos y medio y finaliza hacia los seis. A largo de este este tiempo los niños van pasando por una serie de etapas en el proceso de construcción de la escritura y van realizando diversas hipótesis y suposiciones sobre la lectura: presilábica, silábica, silábico-alfabética, alfabética y finalmente ortográfica [103]. Todos los niños pasan por estas etapas, si bien a distinto ritmo.

Aunque son los niños quien construyen conocimientos sobre la lectoescritura sería muy difícil que llegasen a aprender a escribir y leer por sí mismos y sin una intervención educativa externa.

En este enfoque se trabaja con textos auténticos, completos y con significado. El primer texto que se escribe y reconoce es el nombre propio.

Se van aprendiendo las letras en un orden que no tiene nada que ver con el convencional establecido en la mayoría de los métodos de alfabetización. Primero se diferencia entre letras, dibujos y números, se adquiere conocimiento de convenciones como la linealidad, orientación izquierda-derecha, distribución del texto en el espacio...

Al principio el niño escribe una letra por sílaba, luego observa que cada sílaba puede tener más de una letra, aunque todavía no se escriban todas sino las más conocidas, o las iniciales y finales de sílaba, para terminar escribiendo todas las letras, incorporando poco a poco reglas que surgen de la propia reflexión: separación entre palabras, uso de mayúsculas y minúsculas...

Capítulo 3

El acceso de los discapacitados visuales a la lectoescritura

Resumen: En este capítulo se exponen las distintas soluciones que se han propuesto a lo largo del tiempo. Esta información se encontraba muy desperdigada, incompleta, poco documentada y frecuentemente con errores. A continuación se expone el sistema Braille: su historia, estructura, didáctica, materiales e instrumentos.

3.1. Perspectiva histórica

Desde la antigüedad hubo diversos inventos para facilitar la lectura y escritura a las personas sin visión (letras de madera, letras en relieve, regletas y pautas, nudos de distinto grosor en una cuerda, etc.), pero su utilización fue poco extendida por las deficiencias intrínsecas de estos códigos o los materiales empleados. La ceguera se convertía, por tanto, en un obstáculo para el acceso a la comunicación escrita.

En el año 1825, el joven Louis Braille, ciego desde los 5 años, creó el código que lleva su nombre, que hoy es el sistema de comunicación para leer y escribir más universalmente conocido por los discapacitados visuales. Su aceptación y difusión no fue fácil ni rápida, y no recibió reconocimiento oficial hasta 1854, cuando es adoptado por la Escuela Francesa para Niños Ciegos, dos años después del fallecimiento de su creador.

Antes, durante y después del tiempo en que Louis Braille desarrolló su código, también se desarrollaron muchos otros medios de lectura y escritura táctil para personas ciegas, que se exponen brevemente a continuación.

3.1.1. Antecedentes

Dídimo el Ciego (313-398), de Alejandría, padre de la Iglesia y ciego desde los cuatro o cinco años de edad, aprendió el alfabeto por medio de un sistema de piezas de marfil o madera, que representaban letras en relieve.

Zain-Din Al Amidi, profesor de la universidad Al-Mustansiriya de Bagdad, ideó un procedimiento para identificar las obras de su valiosa biblioteca y marcar cierta información, sobre todo el precio de los libros. Ajustaba trozos de fino papel para formar las letras o los números, que pegaba en el interior de la tapa, dentro de un pequeño cuadro del mismo grosor para evitar el aplanamiento.

Pero Mexía (1497-1551), erudito español hoy olvidado, publica en Sevilla, en 1540, una compilación titulada *Silva de varia lecion*, de la que tendrá varias ediciones y se traducirá en toda Europa hasta mediados del siglo XVII. En ella cita así *De recta pronunciatione* de Erasmus de Rotterdam (figura 3.1):

> Erasmo dize que aprendieron algunos hombres, ciegos del todo, a escribir perfectamente; que no dexa también de ser provechoso para las que tienen vista. Y es que se hizo una tabla de marfil u otro huesso o metal y en ella se cavaron y labraron todas las letras del abc. E poníasele un punçoncico en la mano, cuya punta fuera delgada, que pudiese correr liberalmente por las cavaduras de las letras de la tablita. Y, trayéndolo otro la mano muchas vezes assí, él sentir en el tiento de la mano la forma y hechura de la letra. E, haziendo esto muchas veces muy de espacio y con gran atención, puso en la memoria la ymagen de la letra. Y la mano, ya usada y diestra, vino a hazer las mismas letras fuera de la tabla, errando algunas vezes; y, emendándole, acertó finalmente a que con una péndola escrevía cualquiera cosa que quería

Un sistema que ya recomendaba Quintiliano en la antigua Roma para enseñar a escribir a los niños videntes [139], y que no posibilita la lectura.

El médico y matemático italiano Girolano Cardano (1501-1576) señaló en *De subtilitate* (Nuremberg, 1550) un procedimiento para enseñar a los ciegos a leer y escribir por el sentido del tacto. Tenían que seguir con un punzón de acero o con el lápiz el contorno de cada una de las letras del alfabeto, grabadas en metal o madera, hasta que pudieran distinguir las letras por el tacto y reproducirlas en papel. Cardano, sin embargo, no apunta cómo escribir en una línea recta con uniformidad de espacio entre las líneas. Además en la dificultad y la lentitud de un aprendizaje de estas características, del que, por otra parte, pone en duda su verdadera utilidad: «la cosa es admirable pero poco útil».

Francisco Lucas también trata en su *Arte de escrevir* (Madrid, 1580) el uso de letras en tablillas de madera. Al no estar las letras en relieve sino

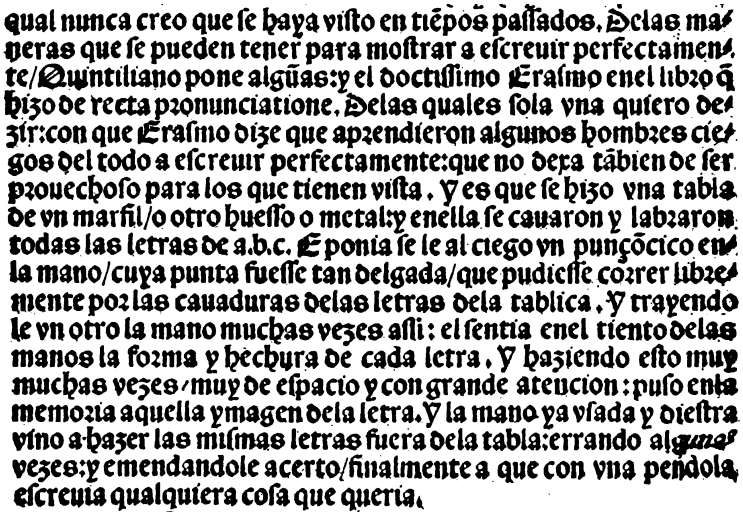

qual nunca creo que ſe haya viſto en tiẽpos paſſados. Delas maneras que ſe pueden tener para moſtrar a eſcreuir perfectamente/Quintiliano pone algũas:y el doctiſſimo Eraſmo enel libro q̃ hizo de recta pronunciatione. Delas quales ſola vna quiero dezir:con que Eraſmo dize que aprendieron algunos hombres ciegos del todo a eſcreuir perfectamente:que no dexa tãbien de ſer prouechoſo para los que tienen viſta. Y es que ſe hizo vna tabla de vn marfil/o otro hueſſo o metal:y enella ſe cauaron y labraron todas las letras de a.b.c. E ponia ſe le al ciego vn punçõcico en la mano/cuya punta fueſſe tan delgada/que pudieſſe correr libremente por las cauaduras delas letras dela tablica. Y trayendo le vn otro la mano muchas vezes aſſi: el ſentia enel tiento delas manos la forma y hechura de cada letra. Y haziendo eſto muy muchas vezes/muy de eſpacio y con grande atencion: puſo enla memoria aquella ymagen dela letra. Y la mano ya vſada y dieſtra vino a hazer las miſmas letras fuera dela tabla:errando algunas vezes:y emendandole acerto/finalmente a que con vna pendola eſcreuia qualquiera coſa que queria.

Figura 3.1: Cita de Erasmo [112].

grabadas en madera, los dedos tenían dificultad en discernir su configuración, y al usarse para imprimir, las letras se quedaban en blanco, mientras que el resto del papel quedaba ennegrecido [5]. En Roma, en el año 1575, el impresor Francesco Rampazzeto publicó ejemplares de letras grabadas en madera para instruir a los ciegos. En 1640, Pierre Moreau, un maestro escribano de París, hizo fundir letras móviles para su empleo por los ciegos, pero por falta de medios no pudo dar continuidad a su iniciativa [26]. En su obra *Deliciæ mathematicæ et physicæ*, publicada en Nuremberg en 1651, George Harsdörffer describe como los ciegos pueden reconocer, y ser enseñados a nombrar e imitar, letras grabadas en cera [86].

Todas estas experiencias, a pesar de su limitado éxito, demuestran como a lo largo de la historia ha surgido repetidamente un interés en lograr que los invidentes logren leer y escribir. A continuación se reseñarán los sistemas posteriores, que lograron un mayor predicamento.

3.1.2. Sistema Lana

Sistema de escritura táctil para uso de los ciegos, ideado por el científico y jesuita italiano Francesco Lana de Terzi (1631-1687). Se trata de uno de los primeros procedimientos, claro antecedente del sistema Barbier, inventado para facilitar la comunicación a los ciegos, y el que por primera vez, no acude al alfabeto latino, pues emplea puntos y líneas (figura 3.2).

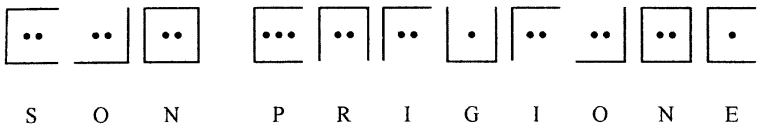
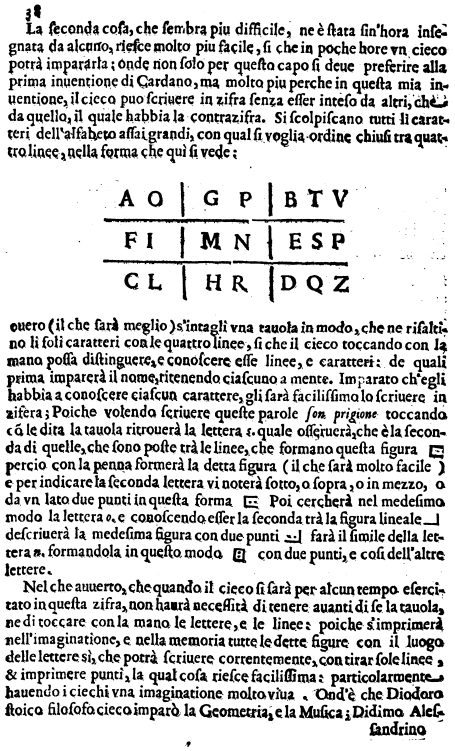


Figura 3.2: Ejemplo: codificación de «estoy prisionero».

El sistema se basa en una tabla, dividida en cuadrículas, que contiene las distintas letras del alfabeto y cuya ubicación exacta debe aprender la persona ciega. A la hora de escribir debe formar cada letra trazando líneas que reproducen la forma de la cuadrícula en la que se sitúa la letra elegida y marcando en su interior uno, dos, o tres puntos, según el orden que ocupe dicha letra (figura 3.3).



La ſeconda coſa, che ſembra piu difficile, ne è ſtata ſin'hora inſegnata da alcuno, rieſce molto piu facile, ſi che in poche hore vn cieco potrà imparárla; onde non ſolo per queſto capo ſi deue preferire alla prima inuentione di Cardano, ma molto piu perche in queſta mia inuentione, il cieco puo ſcriuere in zifra ſenza eſſer inteſo da altri, che da quello, il quale habbia la contrazifra. Si ſcolpiſcano tutti li caratteri dell'alfabeto aſſai grandi, con qual ſi voglia ordine chiuſi tra quattro linee, nella forma che quì ſi vede:

A O	G P	B T V
F I	M N	E S P
C L	H R	D Q Z

ouero (il che ſarà meglio) s'intagli vna tauola in modo, che ne riſaltino li ſoli caratteri con le quattro linee, ſi che il cieco toccando con la mano poſſa diſtinguere, e conoſcere eſſe linee, e caratteri: de quali prima imparerà il nome, ritenendo ciaſcuno a mente. Imparato ch'egli habbia a conoſcere ciaſcun carattere, gli ſarà faciliſſimo lo ſcriuere in zifera; Poiche volendo ſcriuere queſte parole *ſon prigione* toccando cõ le dita la tauola ritrouerà la lettera *s*. quale oſſeruerà, che è la ſeconda di quelle, che ſono poſte trà le linee, che formano queſta figura ⊏ percio con la penna formerà la detta figura (il che ſarà molto facile) e per indicare la ſeconda lettera vi noterà ſotto, o ſopra, o in mezzo, o da vn lato due punti in queſta forma ⊏ Poi cercherà nel medeſimo modo la lettera *o*, e conoſcendo eſſer la ſeconda trà la figura lineale ⌊ deſcriuerà la medeſima figura con due punti ⌊ farà il ſimile della lettera *n*. formandola in queſto modo ⊓ con due punti, e coſi dell'altre lettere.

Nel che auuerto, che quando il cieco ſi ſarà per alcun tempo eſercitato in queſta zifra, non haurà neceſſità di tenere auanti di ſe la tauola, ne di toccare con la mano le lettere, e le linee: poiche s'imprimerà nell'imaginatione, e nella memoria tutte le dette figure con il luogo delle lettere sì, che potrà ſcriuere correntemente, con tirar ſole linee, & imprimere punti, la qual coſa rieſce faciliſſima: particolarmente hauendo i ciechi vna imaginatione molto viua. Ond'è che Diodoro ſtoico filoſofo cieco imparò la Geometria, e la Muſica; Didimo Aleſſandrino

Figura 3.3: Descripción original del sistema Lana [165]

3.1.3. Stachelschrift

En 1809 Johann Wilhelm Klein (1765-1848), director del Instituto Klark, de Viena inventa un alfabeto compuesto de letras romanas que producían letras discontinuas de puntos muy finos, como si fueran hechos con la punta de un alfiler (figura 3.4).

Figura 3.4: Sistema Klein. [52]

El aparato Klein se compone de una caja de madera, una de cuyas mitades, subdividida en casilleros, sirve para colocar los tipos en orden alfabético. En las tres primeras filas, de arriba para abajo, van colocadas las letras. En la cuarta fila están los números. En la quinta y sexta se hallan los signos, entre los que van incluidos unos tipos en blanco, que sirven para la separación de palabras. La tapa de la caja se usa como tablilla para escribir, y con tal fin está cubierta en su lado interno con una felpa de tamaño adecuado sobre la que se coloca el papel, quedando fijo éste con un marco de madera. Dicho marco tiene en sus lados un riel de metal, el cual lleva dos agujeros equidistantes, en los que se fija la guía que sirve para escribir por medio de dos pequeños pernos que aquélla lleva en cada uno de sus puntos.

Para escribir se coloca la guía en los agujeros interiores del marco, produciéndose la escritura de abajo para arriba y de izquierda a derecha. Se van clavando los tipos uno al lado del otro de forma que, el que va quedando atrás se va retirando para poderlo usar en caso necesario. Terminada una palabra, se apoya sobre la última letra el tipo en blanco que se emplea para ese objeto, y a su lado puede colocarse la primera letra de la palabra siguiente. El alfabeto Klein, aunque fue anterior al sistema Braille, no alcanzó una difusión muy extensa entre los ciegos y solamente se popularizó en Alemania, Austria, Hungría y Checoslovaquia.

No obstante, en el Congreso Europeo de Educadores de Ciegos y Sordomudos celebrado en Viena en 1873 se estudió la problemática de estos discapacitados en las instituciones que les atendían, y se abogó por utilizar el sistema Klein y que el Braille (sistema que posteriormente se estudiará con detalle en esta memoria) tuviera un carácter auxiliar y secundario.

3.1.4. Método Valentin Haüy

Método de lectura en relieve para personas ciegas, ideado por el francés Valentin Haüy (1745-1822) y basado en la impresión en relieve de caracteres convencionales latinos. La idea de Haüy se basaba en que por procedimientos de impresión de las letras en relieve sobre papel, podrían editarse libros especiales susceptibles de ser leídos por las personas ciegas. Se lograría, por primera vez, imprimir no ya frases aisladas o mensajes sueltos, sino tratados amplios sobre infinidad de contenidos. Haüy y sus primeros alumnos advirtieron que los reversos de las páginas impresas presentaban letras legibles al tacto. En aquella época, los impresores utilizaban corrientemente papel húmedo para imprimir y así, éste quedaba grabado hasta cierto punto con la forma de las letras.

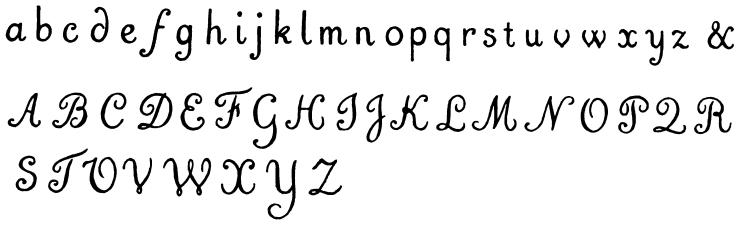


Figura 3.5: Alfabeto Haüy [121]

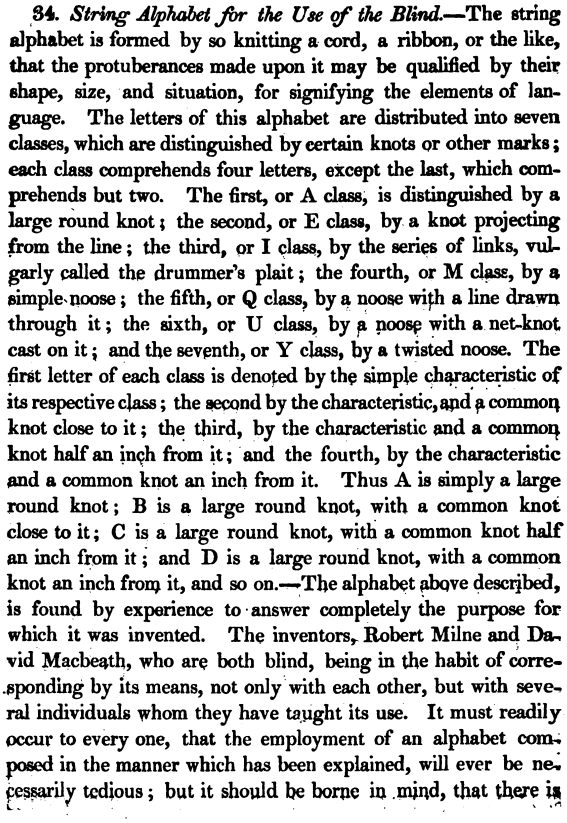
Haüy imprimía letras al revés, de modo que cuando se imprimían sobre papel húmedo dejaban impresiones táctiles en la posición correcta y en el orden debido. Hasta llegar al tamaño mejor percibido por el tacto, Haüy probó diferentes gruesos de letra, distintas tipografías y hasta ideó un sistema de abreviaturas. No obstante, aunque en el *Institut national des jeunes aveugles* que creó se utilizó su procedimiento –también llamado «impresión de París»–, los textos obtenidos no resultaron totalmente eficaces ya que los caracteres así obtenidos no permitían el reconocimiento inmediato de cada signo, pues había que recorrerlos íntegramente para saber de qué letra se trataba en cada caso. La lectura se convertía así en un constante y poco ágil deletreo, por lo que luego el Braille demostró ser más eficaz.

Si bien con su hallazgo iba a ser posible la lectura, no se había logrado aún plenamente el procedimiento adecuado para la escritura. Para escribir, sus alumnos utilizaban una pluma metálica con una punta redonda para obtener letras realzadas en reverso, en la parte posterior de grueso papel.

Este sistema de lectura y escritura fue utilizado en la escuela residencial de París hasta 1854 y en todas las primeras escuelas de Europa y América.

3.1.5. Alfabeto de nudos

Alfabeto ideado a principios del siglo XIX por David MacBeath y Robert Milne –ciegos del Asilo de Ciegos de Edimburgo– que recuerda a los quipus incas, y que ya en 1858 no estaba en vigor. Se trata de uno de los primeros intentos de idear un tipo de escritura para facilitar la comunicación de las personas ciegas. Con siete nudos diferentes, realizados en un cordel, y por medio de un nudo adicional, se formaban siete clases de signos con los que expresar el alfabeto completo.



34. *String Alphabet for the Use of the Blind.*—The string alphabet is formed by so knitting a cord, a ribbon, or the like, that the protuberances made upon it may be qualified by their shape, size, and situation, for signifying the elements of language. The letters of this alphabet are distributed into seven classes, which are distinguished by certain knots or other marks; each class comprehends four letters, except the last, which comprehends but two. The first, or A class, is distinguished by a large round knot; the second, or E class, by a knot projecting from the line; the third, or I class, by the series of links, vulgarly called the drummer's plait; the fourth, or M class, by a simple noose; the fifth, or Q class, by a noose with a line drawn through it; the sixth, or U class, by a noose with a net-knot cast on it; and the seventh, or Y class, by a twisted noose. The first letter of each class is denoted by the simple characteristic of its respective class; the second by the characteristic, and a common knot close to it; the third, by the characteristic and a common knot half an inch from it; and the fourth, by the characteristic and a common knot an inch from it. Thus A is simply a large round knot; B is a large round knot, with a common knot close to it; C is a large round knot, with a common knot half an inch from it; and D is a large round knot, with a common knot an inch from it, and so on.—The alphabet above described, is found by experience to answer completely the purpose for which it was invented. The inventors, Robert Milne and David Macbeath, who are both blind, being in the habit of corresponding by its means, not only with each other, but with several individuals whom they have taught its use. It must readily occur to every one, that the employment of an alphabet composed in the manner which has been explained, will ever be necessarily tedious; but it should be borne in mind, that there is

Figura 3.6: Alfabeto con cuerdas para uso de los ciegos. [19]

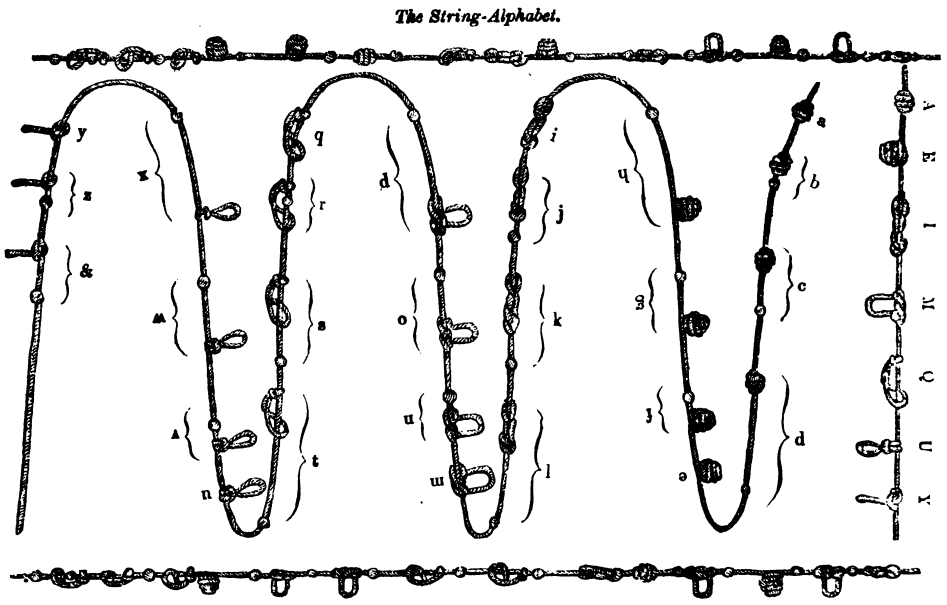


Figura 3.7: Alfabeto con cuerdas para uso de los ciegos. [3]

3.1.6. Sistema Barbier

La sonografía fue inventada hacia 1808 por el capitán francés Charles Barbier de La Serre (1767-1841), y era un sistema de escritura abreviada, a base de puntos en relieve, para pasar mensajes cifrados en el campo de batalla. Se trata de un sistema cuyos signos están basados en los sonidos franceses y no en las letras del alfabeto.

Al tener como finalidad la posibilidad de lectura al tacto en la oscuridad de la noche, su autor pensó que podía aplicarse a la comunicación entre y con personas ciegas, si bien presenta algunas limitaciones e imperfecciones:

- es un sistema fonético, que no permite tener en cuenta la ortografía.
- no hay combinaciones de puntos para representar los signos de puntuación, las cifras, notas musicales, símbolos matemáticos, etc.
- la altura de la rejilla (2×6 puntos) no permite leerla de una sola vez con un dedo.
- hay desaprovechamiento de la información: una rejilla de 12 puntos permite teóricamente representar hasta 2^{12} (4096) símbolos.

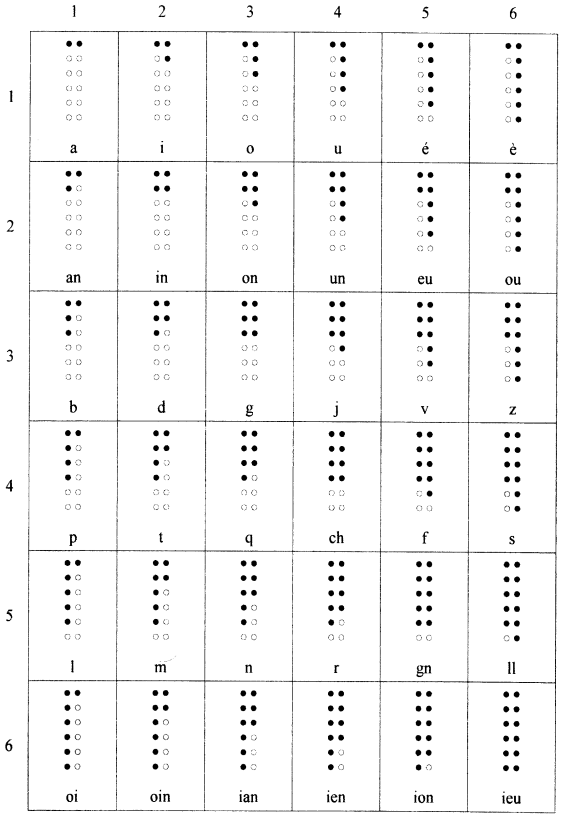


Figura 3.8: Alfabeto Barbier [69]

Louis Braille asistió en 1821 a una presentación de la sonografía realizada por Barbier en la *Institution royale des jeunes aveugles*, de la que tomó la idea de los puntos en relieve para crear su propio sistema de seis puntos.

3.1.7. Alfabeto Gall o triangular

James Gall (1784?-1874), director de la Escuela de Ciegos de Edimburgo, su ciudad natal, fue uno de los aspirantes a la medalla de oro ofrecida por la Sociedad de Arte de la capital escocesa, presentando al certamen su sistema, consistente en tipos gráficos en relieve, que eran una modificación angular de la letra romana; y en ella, Halle trataba de combinar las ventajas de dicha letra ordinaria con un sistema arbitrario, pero no tuvo apoyo financiero alguno.

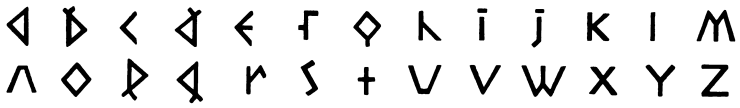


Figura 3.9: Alfabeto Gall [121]

Él era impresor y consiguió publicar varios libros por su cuenta. Su primer experimento comenzó en 1817 y escribió en 1828 *El Evangelio según San Juan*, utilizando, exclusivamente, letras mayúsculas, siendo la primera obra en relieve que se imprime en Inglaterra para uso de los ciegos. *El Evangelio según San Juan* vuelve a publicarlo en 1834, pero esta vez en el sistema Gall revisado. En 1830 anunció Gall que había inventado el tiflógrafo, aparato con el cual era posible escribir una página limpia y ordenada, utilizando su sistema gráfico para ciegos. Explicó el sistema Gall y el manejo del tiflógrafo, aparato con el cual se hacía una especie de letra gótica, en un librito [63] publicado en Edimburgo en el año 1837 por su autor.

3.1.8. Sistema Fry

En 1832 la Sociedad del Arte de Edimburgo, ofreció una recompensa en metálico –y al año siguiente una medalla de oro– a quien idease los mejores tipos gráficos en relieve para la lectoescritura de los ciegos. Se presentaron seis competidores el primer año, siendo desestimado su trabajo por el jurado de la sociedad y 15 fueron los candidatos el segundo, de los cuales 12 propusieron sistemas muy arbitrarios en lugar de letras, no obstante era general la preferencia por los signos de rayas.

La medalla de oro, con valor de 20 soberanos, se le concedió al, entonces ya difunto, doctor Edmund Fry (1754–1835) por su sistema de letras mayúsculas romanas *sans-serif* (sin gracias o remates).

El comité consideró que los ciegos tendrían grandes obstáculos en aprender unos caracteres que no eran familiares a los videntes, y que por tanto tales alfabetos no podrían adoptarse ampliamente en Europa y América [83]. Lorimer [94] señala sobre esta ignorancia de los problemas perceptivos y cognitivos involucrados:

> El lector con vista siempre encuentra difícil evaluar las cualidades de un código destinado a ser leído al tacto, e incluso una persona ciega necesita práctica antes de poder dar una opinión imparcial.

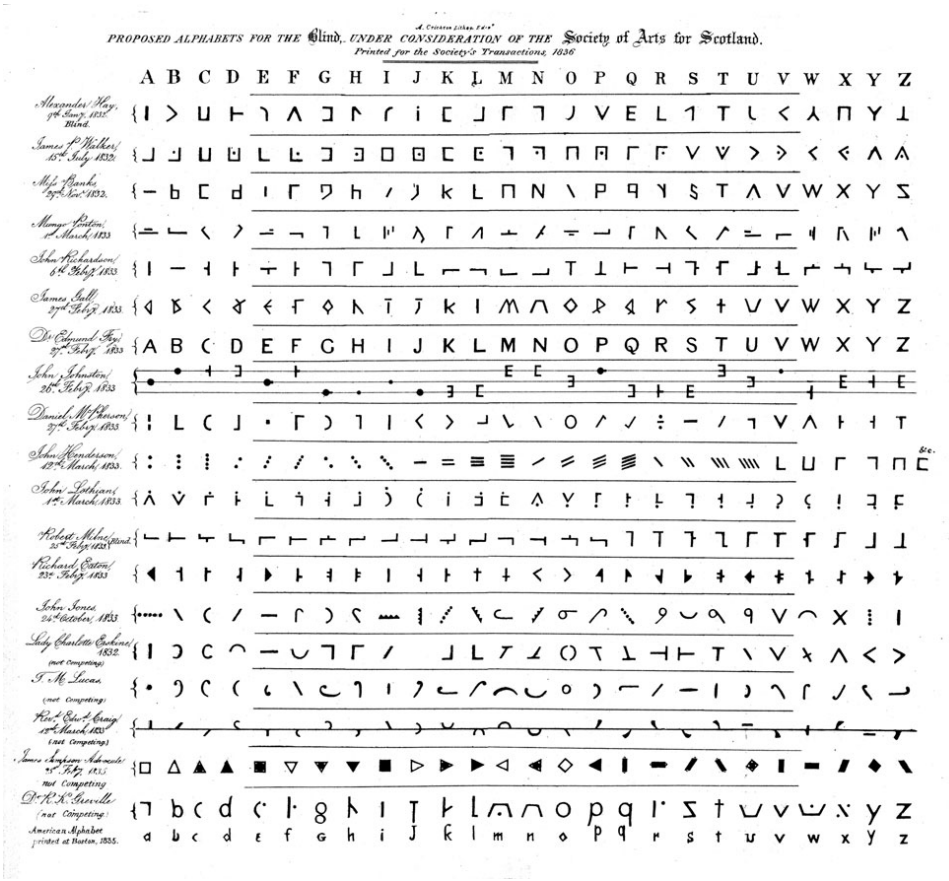


Figura 3.10: Alfabetos propuestos para los ciegos, a consideración de la *Society of Arts for Scotland* [82]

3.1.9. Sistema Alston

El asilo para ciegos de Glasgow, dirigido por John Alston (1778-1846), se convirtió en el defensor principal de los tipos Fry, aunque algo modificados por su máximo responsable, es decir, del sistema Alston, y pronto aparecieron libros en este procedimiento que se distribuyeron por toda Inglaterra, e incluso llegaron a Estados Unidos, pues consta que la Institución de Ciegos de Filadelfia escribió para felicitar a Alston por su sistema y decirle que ya habían empezado a publicar libros en el mismo.

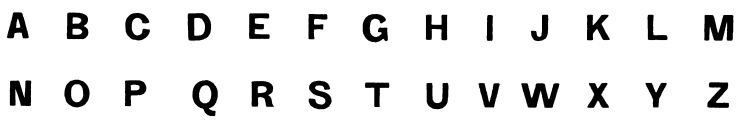


Figura 3.11: Alfabeto Alston [121]

Animado por la buena acogida que tenían las publicaciones en su sistema, John Alston hizo un llamamiento, pidiendo dinero para establecer una imprenta Alston permanente, aludiendo al primer libro para faltos de vista aparecido en Norteamérica, el *Evangelio según San Marcos*, impreso en Filadelfia, y a los libros publicados en Boston, argumentando que, si las gentes de estas dos ciudades estadounidenses apoyaban esta impresión de libros para los invidentes, seguramente que en Escocia no se permitiría que su proyecto languideciera por falta de dinero. Alston tuvo éxito en su llamamiento, porque su imprenta comenzó a trabajar con normalidad en enero de 1837. Su primer trabajo de importancia fue la impresión en relieve de las Sagradas Escrituras, que se terminó en 1840. Se sabe que en 1838 envió una voluminosa partida de sus libros al otro lado del Atlántico.

3.1.10. Método Snyder

Jacob Snider, Jr., de la *Pennsylvania Institute for the Instruction of the Blind*, produjo en 1833 el primer libro en relieve impreso en los Estados Unidos (el *Evangelio de San Marcos*), usando pequeñas letras romanas en mayúsculas y minúsculas. Aunque su método producía letras claras, debía resultar muy caro, porque al año siguiente se le denegaron fondos para publicar el Evangelio de San Juan. El sistema fue abandonado y posteriormente los libros se imprimirían siguiendo el método europeo convencional [53].

3.1.11. Boston Line Type

Desarrollado por Samuel Gridley Howe (1801–1876), fundador de la *New England School for the Blind* (posteriormente *Perkins School for the Blind*) en Massachusetts. Dado que en aquellos momentos no había ningún medio de lectura para la gente afectada de ceguera, Howe desarrolló un alfabeto romano angular simplificado y sin mayúsculas, similar al de Gall, que llamó tipos Boston Line. Publicó el primer libro en letras Boston Line en 1834, y este tipo continuó siendo el principal código de lectura táctil usado en los Estados Unidos durante los siguientes 50 años. La *American Printing House for the Blind* empezó publicando libros en Boston line type, y fue el código oficial usado por los estudiantes en Perkins hasta 1908.

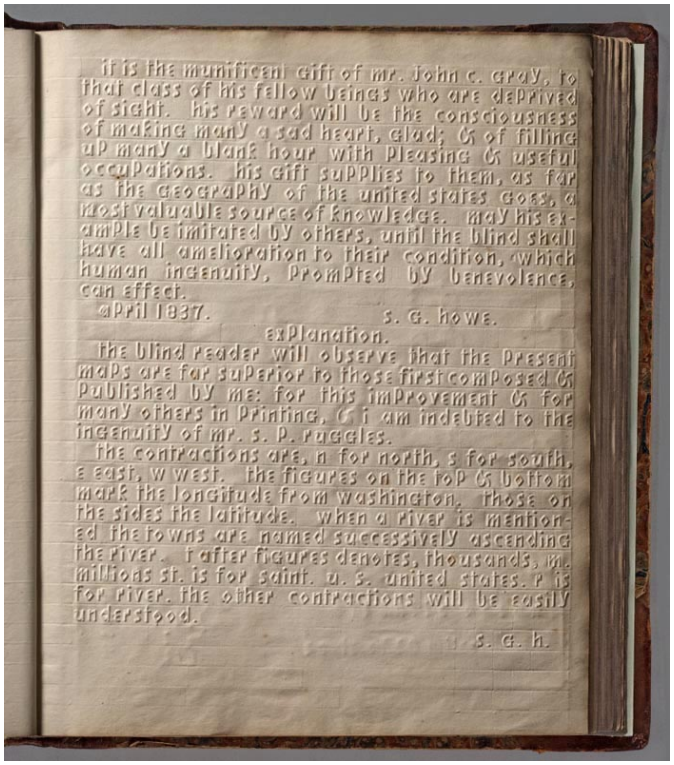


Figura 3.12: Texto introductorio al *Atlas of the United States Printed for the Use of the Blind* publicado en 1837 [152]

Julius Reinhold Friedlander (1803–1839) creó Philadelphia Line, sólo con mayúsculas, similar al sistema Alston, y que se usó en la *Pennsylvania Institution for the Instruction of the Blind in Philadelphia*. William Chapin, de la misma institución, combinó las letras minúsculas de Boston Line con las mayúsculas de Philadelphia Line, formando el «sistema combinado», que fue usado en los libros impresos por N. B. Kneass, Jr.

3.1.12. Sistema Lucas

En 1832 Thomas Mark Lucas (1764-1838), de Bristol (Inglaterra) abrió en esta ciudad una escuela para niños ciegos [113], y en 1837 anunció en Londres que había inventado un sistema de lectura en relieve, que según él, constituiría un medio universal de lectura para los ciegos. La base de este invento era estenográfica, siendo los elementos de sus caracteres las líneas rectas y las curvas, además de algunos puntos sueltos, y una vez más fue la Biblia el primer libro impreso, el cual se terminó en 1843.

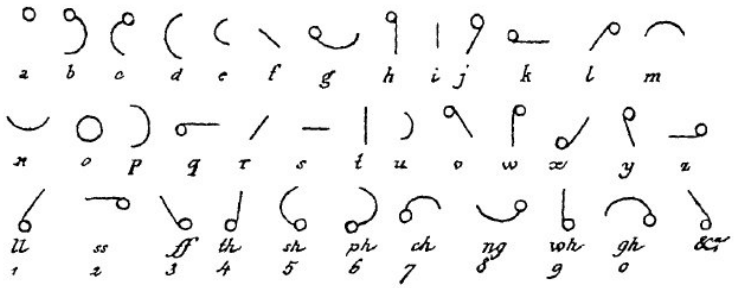


Figura 3.13: Sistema Lucas [91]

En 1838 fundó la *London Society for Teaching the Blind to Read*, que sigue funcionando hoy con el nombre de *Royal London Society for the Blind* [149].

Los partidarios del sistema de Lucas sostenían que los caracteres estenográficos facilitan la lectura mediante los dedos y contaban la historia del clérigo ciego, que habiendo estado durante doce años intentando leer con rapidez los caracteres comunes en relieve, no lo consiguió, pero desde que conoció el sistema estenográfico de Lucas leía con soltura en este alfabeto y podía desempeñar sus deberes ministeriales sin ayuda alguna y dirigía personalmente las dos celebraciones del domingo con la misma eficiencia y comodidad que antes de perder la vista.

3.1.13. Sistema Frere

Hacia 1838 James Hatley Frere (1779–1866) presentó un sistema fonético para enseñar a los ciegos a leer. Sus caracteres consistían en líneas rectas, semicírculos, líneas en forma de gancho, y ángulos de 45 grados, junto a un círculo hueco o relleno. Sus libros se escribían en bustrófedon, leyéndose una línea de izquierda a derecha y la siguiente de derecha a izquierda. En las líneas de regreso, las letras aparecían al revés. La mayoría de las vocales no se representaban, o lo hacían únicamente con unos puntos en distintas posiciones siguiendo un complejo conjunto de reglas [4]. Aunque útil para lograr que las personas sin ninguna educación aprendieran a leer en un corto espacio de tiempo, el sistema tendía mucho a viciar la pronunciación y en 1883 ya sólo se enseñaba en una escuela [145]. Moon enseñó el sistema de Frere en Brighton antes de crear el suyo propio.

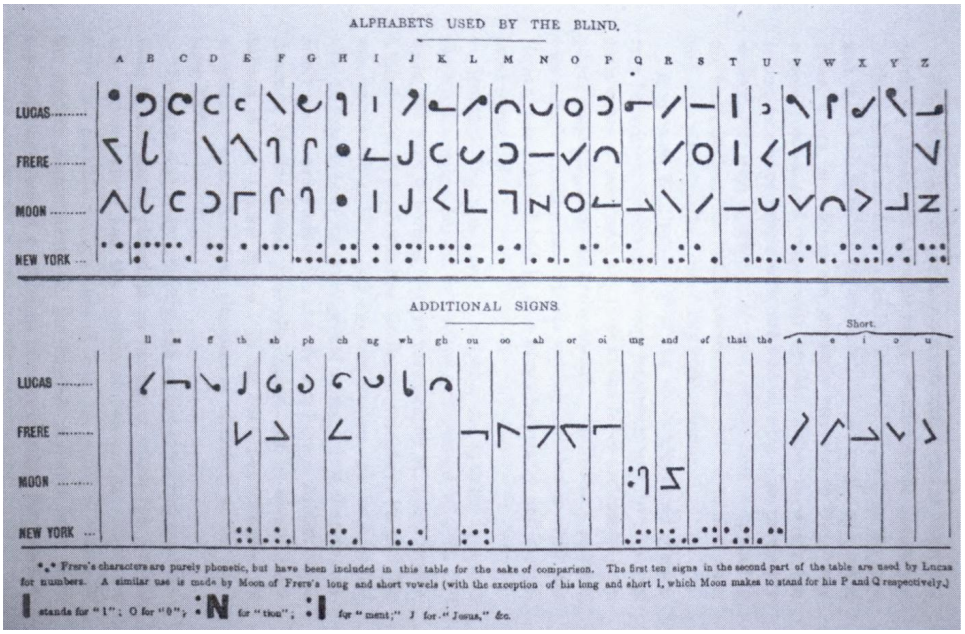


Figura 3.14: Alfabetos usados por los ciegos: Lucas, Frere, Moon y New York Type [33]

3.1.14. Sistema Moon Type

De todos los sistemas introducidos en esta época de rivalidad tipográfica, solamente uno ha sobrevivido hasta nuestros días difundiéndose por los países de lengua inglesa. Fue inventado por el inglés William Moon (1818-1894), de Brighton, que perdió gran parte de su visión en la infancia debido a la escarlatina. Después de finalizar sus estudios, el Dr. William Moon experimentó con una variedad de alfabetos en relieve para enseñar a leer y a escribir a los estudiantes ciegos. Finalmente creó el Moon Type en 1845, un código de líneas en relieve basado en letras impresas.

El alfabeto Moon, básicamente utiliza las letras latinas, dentro de un cuadrado, un mismo signo colocado en cada uno de los cuatro ángulos del paralelogramo, da cuatro caracteres diferentes. Otros signos sólo pueden adoptar dos posiciones dentro del cuadrado y los restantes mantienen una posición única. Hubo seis caracteres que Moon no pudo adaptar, y entonces inventó seis signos arbitrarios. Los partidarios del Moon Type, que aún es usado en Gran Bretaña para las personas con problemas de aprendizaje o dificultades motrices finas y para aquellas que han perdido la vista más tarde en su vida, creen que este sistema es más fácil de aprender y más simple de discriminar en forma táctil que el Braille.

Figura 3.15: El alfabeto Moon [185]

Aunque es prácticamente desconocido en Estados Unidos, están disponibles libros en Moon Type del *Royal National Institute for the Blind* [150], que también están disponibles en Canadá, Australia y en Gran Bretaña. Tiene el inconveniente de que al escribirlo manualmente el nivel de velocidad lectora suele ser muy lenta. En España y en los países hispanohablantes no se ha utilizado nunca.

Figura 3.16: Diferencia del bustrófedon en Frere y Moon, usando el alfabeto latino [10]

Hoy día, el código Moon puede ser generado con un software de ordenador. Todavía se producen libros en Moon a través de un proceso modificado de configuración de la página. Actualmente, también se generan materiales de lectura con Moon Writers, máquinas de termoformateado, tipos de letra de Moon para computadoras impresas en papel de microcápsulas (*swell paper*) y grabadores en relieve y software de traducción en Moon. También se puede escribir a mano el código Moon con un punzón sobre láminas de

papel plástico con una guía de borde, de una manera similar al uso de una pauta y un punzón para producir Braille. El *Royal National Institute for the Blind* [150] en Gran Bretaña tiene disponible un currículum de enseñanza de Moon.

3.1.15. American Modified Braille

En 1827, Louis Braille publicó el sistema que lleva su nombre, y que debido a su importancia se estudiará en una sección aparte (sección 3.2). Joel W. Smith, un profesor de afinación de piano de la Escuela para los Ciegos Perkins de Massachussets, desarrolló el Código Braille Norteamericano Modificado en la década de 1870. Cuando desarrolló este sistema, Smith diseñó caracteres que pensaba serían rápidos de leer y usarían eficientemente el papel. Este código fue usado en 19 escuelas para los ciegos en Estados Unidos, incluyendo Perkins. El American Modified Braille asignaba la menor cantidad de puntos a los caracteres que ocurren con mayor frecuencia en el idioma inglés. Los caracteres siguen siendo de tres puntos de alto y dos puntos de ancho, pero con configuraciones de puntos que corresponden a diferentes letras impresas y combinaciones de letras que el del Braille estándar de hoy.

3.1.16. New York Point

William Bell Wait (1839-1916) desarrolló un código de puntos para lectores ciegos, que usaba caracteres que tenían dos puntos de altura y uno, dos, tres y cuatro puntos de anchura. Cuando trabajaba en el *New York Institute for the Blind*, Wait comenzó a enseñar este sistema a los estudiantes e inventó una máquina de escribir de 12 teclas llamada *kleidograph*, que permitía producir texto fácilmente sin usar la tableta y el punzón. El New York Point se usó ampliamente en las escuelas para ciegos de Estados Unidos a finales del siglo XIX.

Wait sostenía que su sistema era más lógico que el Braille inglés y el americano, y estos sistemas compitieron en lo que se conoció como *War of the Dots* («guerra de los puntos»). Durante un tiempo, la *American Printing House for the Blind* publicaba simultáneamente en Boston Line Type, Braille modificado y New York Point.

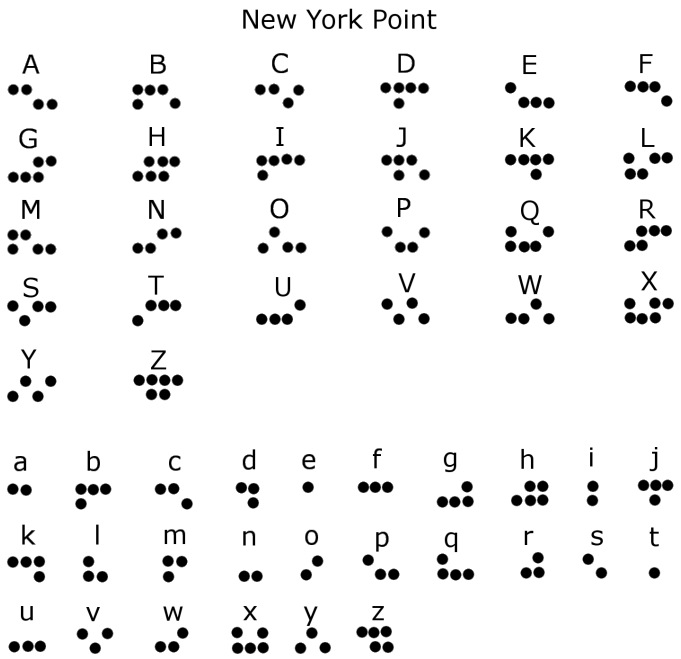


Figura 3.17: New York Point [186]

En una carta dirigida a la convención de 1905 de la *American Association of Workers for the Blind*, Charles W. Holmes, presidente de la *Perkins Alumni Association*, rogó la adopción de un sistema uniforme [79].

> Con el fin de valerse de la gama completa de la literatura (que siendo generosos es lamentablemente limitada) el lector ciego debe aprender y recordar todos estos códigos... ¿Cuánto tiempo aguantarían nuestros amigos videntes semejante estado de cosas en la impresión en tinta? Imaginen por un momento la situación ridícula que se produciría si los diarios publicados en Boston tuviesen un sistema de caracteres totalmente diferente de los utilizados por los editores de Nueva York, y que un hombre de Filadelfia no pudiese leer ninguno de los dos sin entrenamiento especial, porque su propia ciudad había adoptado un tercer sistema, tan diferente de los otros como los caracteres chinos de los romanos

A pesar de la elocuente defensa de Holmes, tomó un gran esfuerzo y muchos años que el código universal se hiciera realidad.

El *Uniform Type Committee*, formado en 1905, convino en que debería existir un código único y uniforme para todos los lectores de habla inglesa, y que los alfabetos en relieve debían ser discontinuados. El comité diseñó una prueba de legibilidad y descubrió que el Braille británico era superior al

Braille Modificado y al código New York Point [110].

3.1.17. Sistema Llorens

Este sistema de escritura y notación musical fue ideado por el español Pedro Llorens y Llatchós para la enseñanza de la lectura y la música. Consistía en una combinación de signos compuestos por líneas rectas y curvas en relieve (véase la figura 3.18).

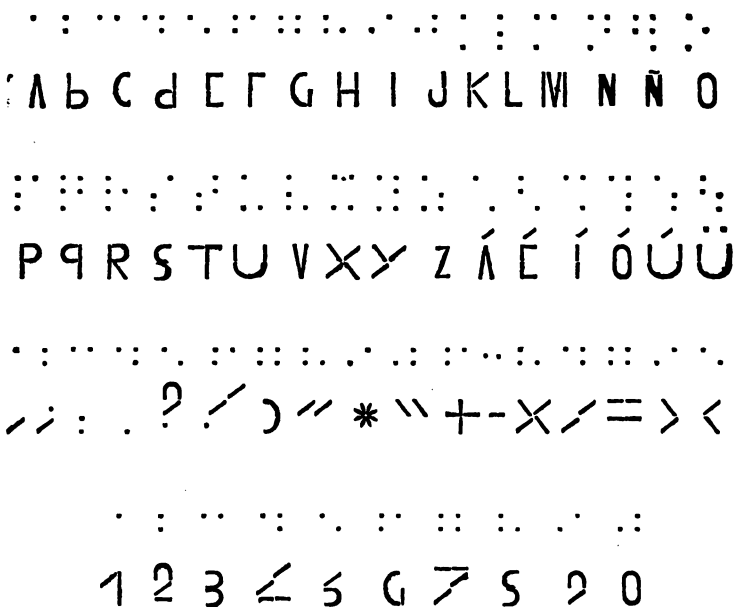


Figura 3.18: El alfabeto Llorens enfrentado al Braille [93]

3.1.18. Alfabeto Lorm

Alfabeto creado por Heinrich Landesmann, alias Hieronymus Lorm (1821-1902) utilizado por las personas sordociegas en la República Checa y en algunas zonas de Centro Europa. En cuanto sistema de comunicación dactilológico, el procedimiento de transmisión del mensaje consiste en dibujar una a una sobre la palma de la mano de los interlocutores, cada una de las letras del alfabeto; cada una de ellas se expresa por un punto o una línea que se dibuja sobre alguna parte de la mano, (por ejemplo, la letra «L» se dibuja como una línea recta desde la punta del dedo corazón, a la base de la muñeca).

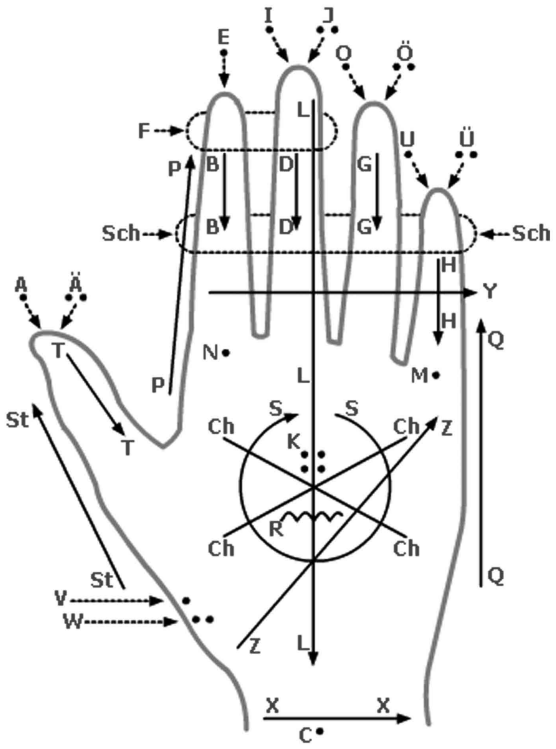


Figura 3.19: Alfabeto Lorm. [153]

3.1.19. Sistema Mulot

Sistema de escritura para ciegos, en relieve y caracteres latinos, inventado en 1886 por la francesa Mlle. Louise Mulot. Suscitó desde el principio una polémica nacional entre los defensores del sistema Braille –más ventajoso por su rapidez y eficacia para la lectura táctil– y defensores de la escritura en caracteres visuales –más ventajosos por la posibilidad de comunicación inmediata con las personas videntes– como base de enseñanza de las personas ciegas. Véase como se deshace en elogios el artículo *Une grande éducatrice des aveugles: Mlle Mulot* [133], también publicado en el *Journal des Instituteurs* en 1899.

Su mecanismo, concebido como guía para escritura en caracteres latinos tradicionales, estaba compuesto por dos planchas metálicas dobles (que se plegaban en forma de libro y tamaño de cuartilla). Cada una de las planchas se dividía en un gran número de ventanas o células rectangulares, en el interior de cada una de las cuales se trazaba una letra en relieve susceptible de lectura al tacto, por parte de las personas con ceguera. Para las personas con visión normal, una hoja de calco permitía teñir en azul o en negro los caracteres en relieve. Acompañaba la pauta una varilla metálica, en forma de estilete, que servía para escribir de derecha a izquierda y leer de izquierda

a derecha.

El sistema llegó a reconocerse como superior al Braille en la región de Mainet-et-Loire (Francia) y a implantarse en la Escuela de Jóvenes Ciegos de Angers, donde Mlle. Mulot era profesora, siendo definitivamente abandonado en 1908 ante la evidencia de la acogida universal del Braille.

3.1.20. Sistema Ballu

Sistema de escritura en relieve ideado por Victor Ballu (1829-1907), profesor de la Institución Imperial de Jóvenes Ciegos de París a finales del siglo XIX. Creó tres alfabetos de distintos tamaños, basados en el romano y ejecutados en relieve punteado. Su uso estuvo generalizado hasta la primera mitad del siglo XX.

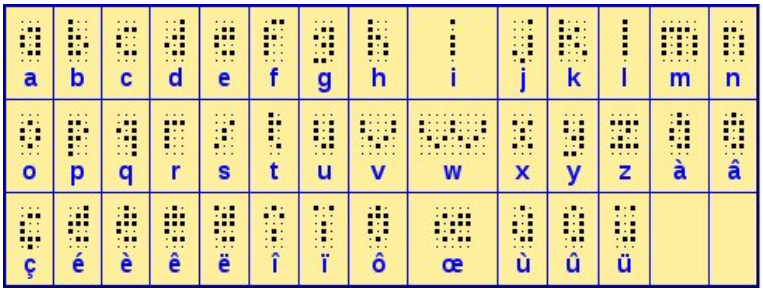


Figura 3.20: Letras minúsculas del alfabeto mediano de Ballu. [12]

3.1.21. Trait-point

El Dr. Ernest Vézien, de Dunkerque, presentó en abril de 1891 a la *Société Dunkerquoise* la primera versión de una escritura que serviría tanto a ciegos como a videntes, que posteriormente mejoraría [182], denominada «Traitpoint», y que defendió en el *Congrès International pour l'Amélioration du sort des Aveugles* celebrado en Bruselas del 6 al 10 de agosto de 1902 [6].

Su signo generador tenía 3 puntos de anchura y 4 de altura, lo que proporciona 4096 combinaciones, frente a las 64 del Braille. Sin pretender sustituir al Braille, Vézien pensaba que su sistema podía ser útil a quienes perdieran la vista tras su escolarización, y para la correspondencia entre ciegos y videntes.

3.1.22. Sistema Mascaró

Aniceto Mascaró Cos (1850-1906) fue un médico oculista español, nacido en Lladó (Gerona). Cursó la carrera de Medicina en Barcelona, trasladándose a América, pasando más tarde a vivir a Lisboa, donde en 1889 funda una escuela para educar a 15 ciegos. El Gobierno francés le condecoró por sus

relevantes trabajos en la investigación ocular y otros meritorios servicios a favor de la Medicina y la cirugía ocular, pues llevó también a cabo una gran labor divulgativa con sus artículos, libros y conferencias. Fue inventor de un sistema de lectura y escritura en relieve para uso de los ciegos; método que intentó unificar los sistemas de lectoescritura de Haüy y Braille, pues las letras y los signos ortográficos y numéricos están formados por puntos en relieve para que los ciegos puedan leerlos con el tacto, representando los caracteres visuales muy esquematizados; y tiene líneas sin relieve que, uniendo los puntos, forman el trazado de las letras romanas para ser leídas por los videntes.

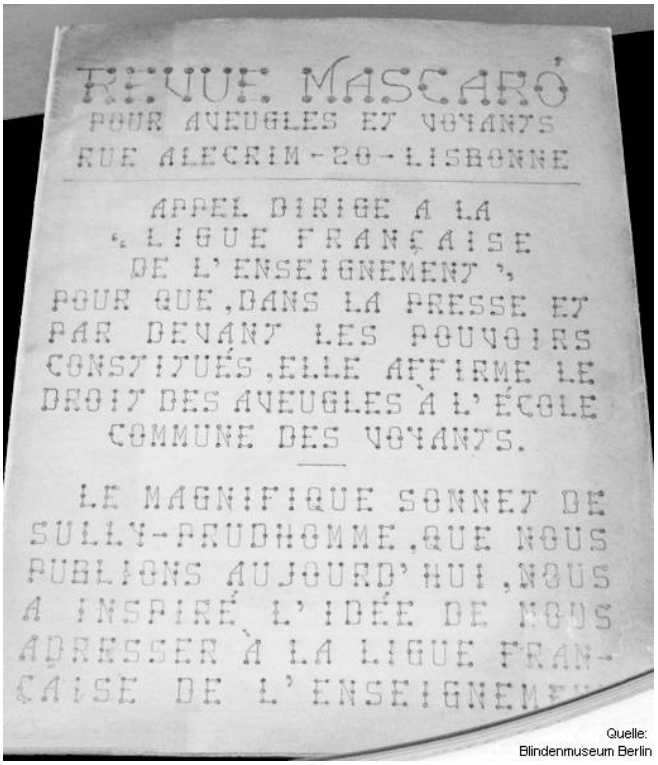


Figura 3.21: Ejemplo del sistema Mascaró [52]

Este sistema Mascaró facilita a los videntes la lectura de los libros en relieve usados por los ciegos, pero también permite a los invidentes escribir a mano los caracteres visuales sin grandes complicaciones y de manera que puedan leerlos quienes tienen vista. Aniceto Mascaró no tuvo éxito con su sistema de lectoescritura para ciegos, que fue olvidado muy pronto; en cambio, la escuela de ciegos por él fundada sí perduró, ya que siempre contó con la protección del Ministro y Consejero D. José Cándido Branco Rodrigues (1865-1918).

3.1.23. Alfabeto Malossi

Alfabeto dactilológico creado por Eugenio Malossi (1885-1930) para personas sordociegas. No se usa en España, sino fundamentalmente en Italia. En este sistema cada una de las letras del alfabeto se corresponde con un punto de los dedos de la mano de la persona sordociega, por ejemplo: la P está ubicada en la punta del dedo pulgar, la O está en la punta del dedo índice, etc. La persona sordociega coloca su mano, ya sea boca abajo, boca arriba o vertical con la punta de sus dedos hacia arriba, y su interlocutor irá deletreando el mensaje punteando o pellizcando los distintos puntos según la letra que desee «escribir».

Igualmente, la persona sordociega que utilice este sistema tiene que conocer la estructura de la lengua oral.

Cuando la persona sordociega está comenzando a aprender este sistema o cuando desea comunicarse con un interlocutor que lo desconoce, se suele utilizar un guante que lleva escritas o cosidas las letras en el punto que le corresponde. La persona que va a recibir el mensaje se coloca el guante y puede ir siguiendo el mensaje visualmente [66].

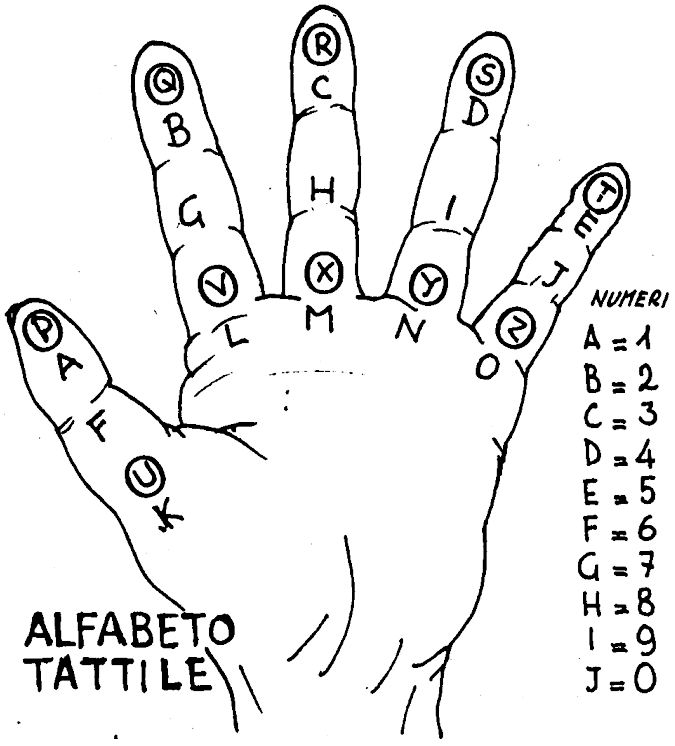


Figura 3.22: Alfabeto Malossi [134].

3.1.24. Alfabeto Fishburne

Código elaborado por S.B. Fishburne en 1972 y popularizado por Shafrath [159], como alternativa al código Braille y dirigido a sujetos que pierden la visión a edad avanzada y que tienen grandes dificultades para aprender el Braille. Fue diseñado para etiquetas, y no para ser usado en materiales de lectura.

Este sistema utiliza cuatro patrones de símbolos diferentes –puntos, líneas verticales, líneas horizontales y líneas diagonales–; a cada seis letras del alfabeto corresponde un símbolo diferente y se reconocen por la posición y el número de símbolos de un mismo tipo.

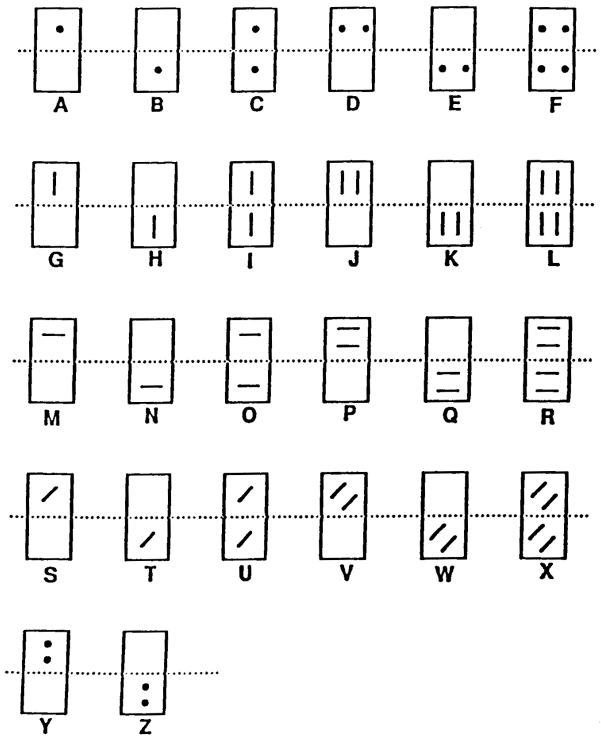


Figura 3.23: El alfabeto Fishburne [122]

3.1.25. Código C5

Código alternativo al Braille, elaborado por Willard R. Thurlow [175]. Está formado por símbolos similares al Moon y a los símbolos utilizados por Bliss y colaboradores en 1966. Las letras del alfabeto se representan tanto por puntos como por trazos verticales, horizontales y diagonales que intentan simular el contorno de las letras del alfabeto romano.

3.1.26. ELIA®

El alfabeto *Elementary Imprint Assistance* (ELIA) se creó como una alternativa al Braille para los ciegos que encuentran el Braille difícil (especialmente los ancianos). El diseño del código se basó en principios de ingeniería; usa un marco alrededor de los símbolos para ayudar a su identificación,y los símbolos se basan en el alfabeto romano. Este código está patentado y es propiedad de ELIA Life Technologies, así que por razones de coste es improbable que se produzca a gran escala.

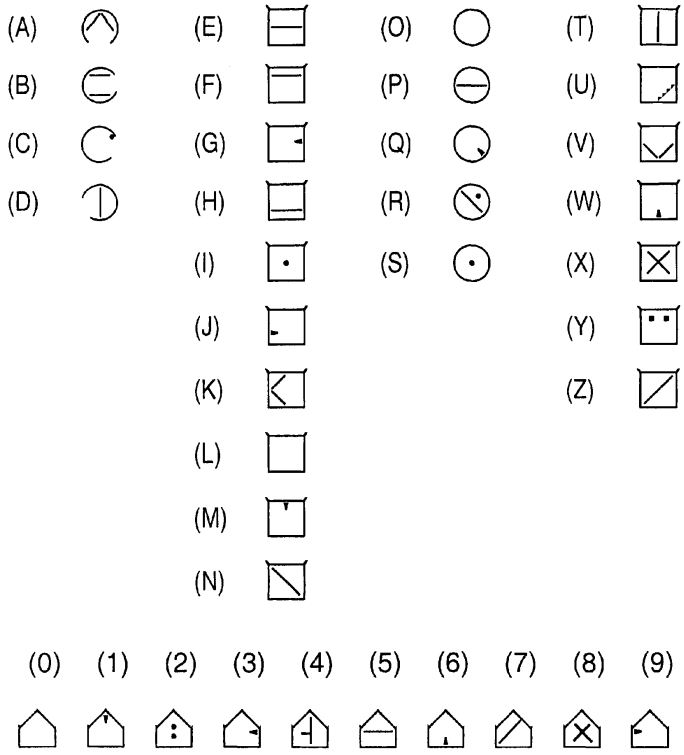


Figura 3.24: El alfabeto ELIA® [32]

3.2. El sistema Braille

3.2.1. Un poco de historia

Louis Braille (1809-1852), estudiante ciego del instituto fundado por Haüy, analiza el sistema de Barbier y realiza varias modificaciones: reduce el tamaño de los signos e inventa un alfabeto. En 1827 se publica, finalmente, el código de lectoescritura para ciegos: el sistema Braille. Además, Luis Braille adapta el sistema a las matemáticas, la música y las ciencias, y desarrolla un sistema de abreviaturas. Inventa también una pauta que permite la escritura.

Como curiosidad, es interesante saber que el título de la presentación del sistema Braille es exactamente: *Procedimiento para la escritura de palabras, música y canto llano por medio de puntos para uso de los ciegos y arreglado por ellos*. Intentaba, también, facilitar a las personas sin visión la lectura y escritura de partituras musicales (de hecho, Luis Braille era profesor de música).

Figura 3.25: Sello conmemorativo del bicentenario de Louis Braille, con puntos en relieve.

El sistema Braille no fue aceptado ni difundido fácilmente ya que suponía un cambio drástico con respecto a la tendencia anterior. Muchos consideraban que usar un alfabeto tan diferente al de los videntes aislaría a los ciegos. Finalmente, en el *Congrès Universel pour l'amélioration du sort des aveugles et des sourds-muets* celebrado en París en 1878, y tras un apasionado debate, se decide que el sistema Braille no modificado representa el mejor modo de enseñanza de la lectura y la escritura a los ciegos, y que se debía generalizar su uso [8].

Fue introducido en España por Jaime Bruno Berenguer, profesor de la Escuela Municipal de Ciegos de Barcelona en el año 1840. La Real Orden Ministerial del 23 de diciembre de 1896 publicó un código Braille castellano que declaró oficial y único en toda España [116].

Fernández del Campo sintetiza de esta forma la importancia del sistema Braille [22]:

> ¿Fue consciente Louis Braille de la potencia de su sistema, del instrumento que ponía en manos de los ciegos?... las posibilidades de su invento superan las necesidades que venía a solucionar y, sin pérdida de coherencia, permite responder a un sinnúmero de retos no vislumbrados en el momento de la creación. Como tantas otras veces a lo largo de la historia, la obra, cual dotada de vida propia, honraba a su creador, rindiendo frutos inesperados.

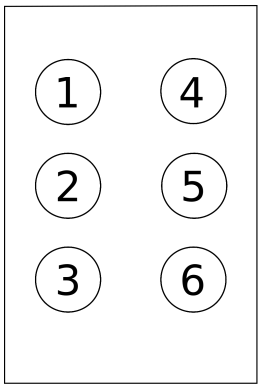
La difusión del sistema Braille como método universal de comunicación escrita para personas ciegas ha sido un factor decisivo en favor de la integración social y educativa de las personas con discapacidad visual. Hoy en día, el acceso a la información de estas personas es una realidad gracias, sobre todo, al sistema Braille.

3.2.2. Estructura del sistema

El sistema Braille se adecua estructural y fisiológicamente a las características del sentido del tacto. Se adapta perfectamente a las terminaciones nerviosas de las yemas de los dedos, y así los signos son transmitidos al cerebro, como una totalidad.

Este sistema está diseñado para ser utilizado a través del tacto, por medio de puntos en relieve. La unidad básica o signo generador es el cajetín o celdilla: un espacio en el que se sitúan los 6 puntos en relieve, distribuidos en dos columnas de tres puntos cada una.

Cada letra o signo se representa en un solo cajetín, en el que aparecen o no los 6 puntos en relieve, que son percibidos a través del tacto por las yemas de los dedos. En un texto en Braille los cajetines no están presentes, siendo visibles sólo los puntos. Para identificar los puntos, se les atribuye un número del 1 al 6.



Para que un texto en Braille sea legible (que la información quepa dentro de la yema del dedo) es preciso que tenga unas medidas determinadas, respetando las distancias que hay entre los puntos que forman un carácter, las que hay entre caracteres contiguos y las que debe haber entre renglones.

Diferentes organizaciones han establecido varios parámetros, prácticamente idénticos entre sí. La *European Carton Maker Association* [48] especificó el «ECMA Euro Braille», mientras que las empresas farmacéuticas utilizan la especificación «Marburg Medium» para cumplir con los estándares EN 15823:2010 (*Packaging - Braille on packaging for medicinal products*)

[11] e ISO 17351:2013. Alemania creó en 2007 la norma DIN 32976 (*Blindenschrift - Anforderungen und Maße*) [41].

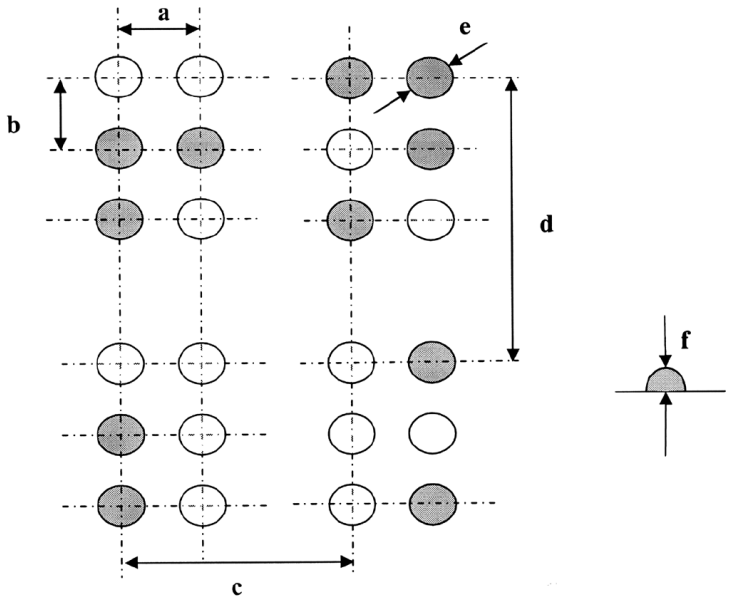


Figura 3.26: Parámetros del Braille [35]

Con carácter general, las medidas del cajetín Braille son:

a) Distancia horizontal entre los centros de puntos contiguos de la misma celda: 2.5 mm.

b) Distancia vertical entre los centros de puntos contiguos de la misma celda: 2.5 mm.

c) Distancia entre los centros de puntos idénticos de celdas contiguas: 6.0 mm.

d) Distancia entre los centros de puntos idénticos de líneas contiguas: 10.0 mm (con una tolerancia de +0.0 mm/-0.1 mm).

e) Diámetro de la base de los puntos: 1.3 - 1.6 mm.

f) Altura del relieve de los puntos: 0.5 mm.

Mediante las diferentes combinaciones de puntos en un mismo cajetín se pueden obtener 64 formas distintas de disposición de los puntos, incluyendo el cajetín en blanco, que se utiliza para separar las palabras. Como el número de posibilidades es limitado, por economía del sistema, un mismo signo Braille puede significar cosas distintas, según el contexto donde lo utilicemos o si le

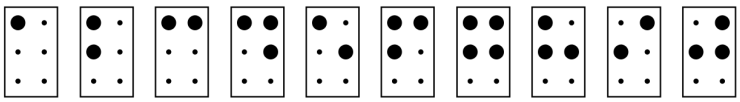
antepongamos otro signo. Por esta razón, el Braille es un sistema y no un simple alfabeto ya que, utilizando sus 64 combinaciones se han desarrollado distintos códigos: para matemáticas, ciencias, música, estenografía[1] (Braille abreviado), signografía específica para diferentes idiomas, etc.

En Braille cada letra se representa con una combinación de puntos en relieve. Para dejar espacios en blanco entre palabras, se emplean también un espacio o cajetín en blanco. Al comenzar a escribir se dejan dos espacios en blanco (sangría) y, entre párrafo y párrafo, es conveniente dejar un renglón en blanco, igual que en tinta. Estos espacios son muy útiles para que el lector pueda localizar fácilmente el inicio de cada párrafo y, así, darle facilidades para ubicarse en el texto.

Figura 3.27: Detalle de las manos de una persona leyendo en Braille [126]

El código está diseñado de manera lógica, mediante **series**:

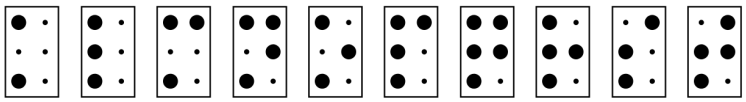
1ª serie se utilizan únicamente los cuatro puntos superiores (1,2,4,5) y con ellos se forman las diez primeras letras del alfabeto



a b c d e f g h i j

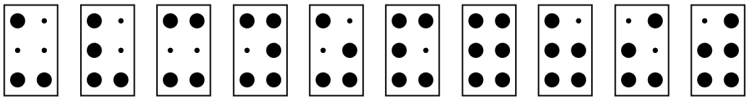
2ª serie se forma con los puntos de la primera serie, añadiéndoles el punto número 3 y, así, obtenemos las siguientes letras, a excepción de la letra ñ (conviene recordar aquí que Luis Braille era francés)

[1] Estenografía: Sistema de escritura Braille abreviada en el que un único signo puede representar una palabra o grupo de letras, para ahorrar espacio y aumentar la velocidad.



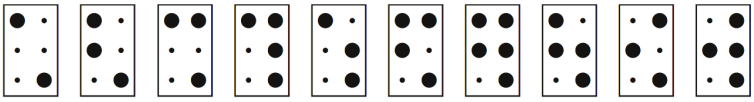
k l m n o p q r s t

3ª serie se forma con los puntos de la 1ª serie, añadiendo los puntos número 3 y 6:



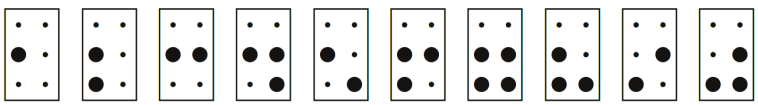
u v x y z ç signo generador á é ú

4ª serie son los elementos de la 1ª serie, añadiendo el punto número 6. La mayoría son signos propios del francés:



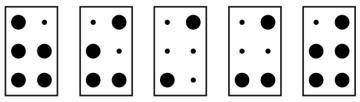
â (î ô û ë ñ ü < w

5ª serie es como la primera serie, pero utilizando los puntos de la mitad inferior de la celdilla o cajetín (puntos 2, 3, 5 y 6). Así, conseguimos los signos de puntuación:



, ; : ÷ ¿? ¡! = « » * °

Las **vocales con tilde** se representan mediante estas combinaciones de puntos:

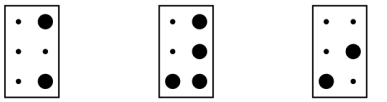


á é í ó ú

Otros **signos de interés** son:

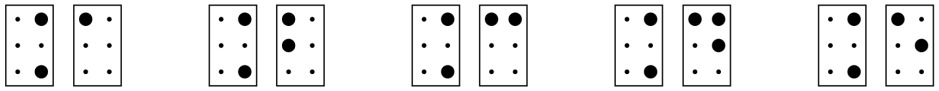
guión punto paréntesis puntos suspensivos

Como las 64 combinaciones posibles son insuficientes para la formación de todos los grafemas necesarios, es necesario utilizar signos complementarios que, antepuestos a una determinada combinación de puntos, convierten una letra en mayúscula, cursiva, número o nota musical:



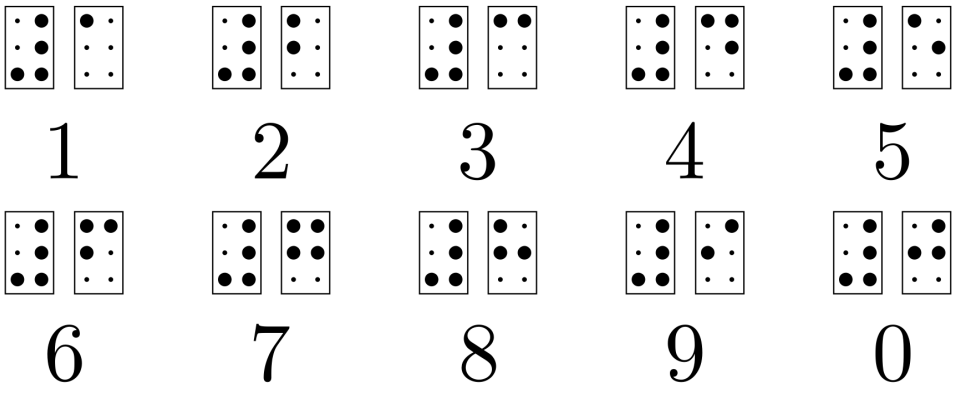
mayúscula número cursiva

Es decir, anteponiendo el signo de mayúscula, formado por los puntos 4 y 6, a cualquier letra, obtenemos las letras mayúsculas. Ejemplos:

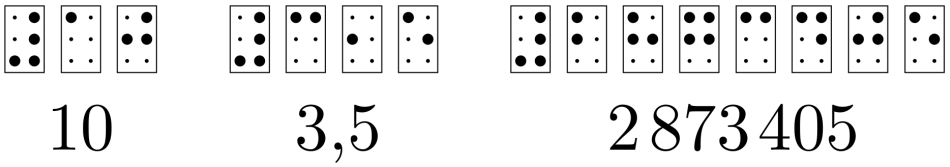


A B C D E

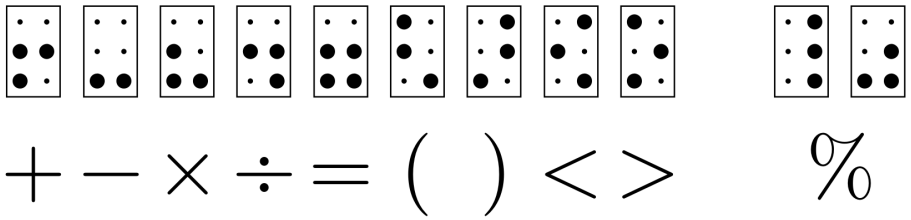
De la misma forma, anteponiendo el signo de número, formado por los puntos 3, 4, 5 y 6, a la primera serie, obtenemos los números del 1 al 0:



Para cantidades de dos o más cifras o números decimales sólo se coloca el signo de número delante de la primera cifra. La coma decimal se representa con el punto 2. Para números altos puede utilizarse el punto 3, como en vista, para el punto de separación de órdenes de unidades:

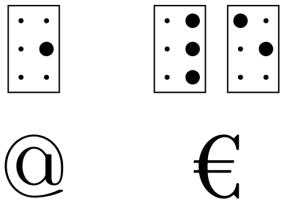


Para terminar, estos son los signos matemáticos básicos:



3.2.3. Otras consideraciones

El sistema Braille está universalmente aceptado y está «vivo»: es flexible y se adapta a las necesidades que van surgiendo. Así por ejemplo, presentamos dos signos de reciente creación:



Existen organismos internacionales que trabajan para consensuar y aplicar universalmente los cambios y novedades en el sistema. En España, la entidad encargada de este trabajo es la Comisión Braille Española [34].

Una mayor matización y conocimiento de la signografía Braille se puede encontrar en *Manual Digital de Signografía Braille* [20] o *Guías de la Comisión Braille Española. Signografía básica* [35].

3.2.4. Braille de 8 puntos

Código Braille actualmente utilizado en informática para obtener, en relieve, la representación de los 256 caracteres del código ASCII, y estandarizado en los informes ISO/TR 11548-1:2001 [76] e ISO/TR 11548-2:2001 [77]. El código ASCII es un código de representación de la información (basado en bytes de 8 bits), que nació para unificar la representación de la información entre los distintos sistemas de ordenadores.

El origen del Braille de 8 puntos se sitúa en el siglo pasado en España en donde el músico ciego Gabriel Abreu creó un sistema musical en relieve con 8 puntos, conocido hoy como Sistema Abreu, en honor a su inventor [45].

En los años 70, 川上泰一 (Taiichi Kawakami, 1917-1994), profesor de una escuela para ciegos de Osaka, desarrolló 漢点字 (*kantenji*), un código de ocho puntos para representar los *kanji* (caracteres japoneses) [123]. Las tres filas inferiores corresponden al bloque Braille japonés estándar para el silabario Hiragana, y los dos puntos extra de la parte superior se usan para marcar «radical 1 (de un kanji de dos bloques)» (●○), «radical derecho» (○●) o «kanji de un único bloque» (●●). Por ejemplo, añadir ●● sobre el Braille para /ki/ produce el kanji 木 («árbol», que en japonés se lee /ki/). Añadir ●● sobre el Braille para /me/ nos da 目 («ojo», /me/). Añadir ●○ sobre /ki/ y ○● sobre /me/ y combinar los dos produce 相 («aspecto», que no se pronuncia /ki/ o /me/ pero se dibuja como una combinación de 木 y 目).

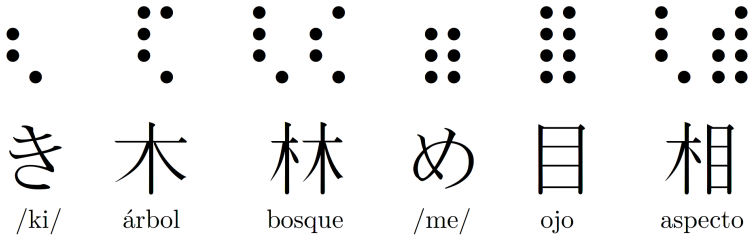


Figura 3.28: Funcionamiento de Kantenji

長谷川貞夫 (Sadao Hasegawa) publicó otro sistema en 1966, llamado *tenkanji*, que usaba tres bloques Braille de 6 puntos para representar cada carácter [18].

3.2.5. Características de la lectura

El aprendizaje de la lectura Braille es lento, y requiere alto grado de disciplina y concentración. Mientras que el niño vidente puede captar con un golpe de vista la figura de la letra o de la palabra, el niño ciego tiene que analizar con sus dedos (índices generalmente) cada una de las letras (o puntos), separando unas de otras. Los dedos no globalizan, no pueden captar de una vez una palabra porque la superficie de contacto de la yema de los dedos es muy pequeña.

En Braille, el lector lee, fundamentalmente, con los índices, con un movimiento que sólo aparentemente es continuo ya que se hacen pausas irregulares. No hay fijación, porque el tacto percibe sólo si se le imprime cierto movimiento.

Por esta razón, la lectura Braille suele ser más lenta que la visual. Sin embargo, hay investigaciones que demuestran que no hay diferencias en cuanto la comprensión.

Es necesario tener en cuenta que es contraproducente que el alumno «cuente» los puntos que conforman cada signo. Lo que se percibe al tacto es una figura bajo la yema del dedo, de tal forma que tanto los puntos como la ausencia de los mismos, configuran los signos.

Es aconsejable exigir a los alumnos que lean con ambas manos a la vez, ya que esto permite, en líneas generales, aumentar la velocidad, por lo menos en un 30 %. Obviamente, si leemos con los dos índices duplicamos el área perceptiva. Si leemos con los índices y los corazones, la volvemos a duplicar y, como consecuencia, aumentamos la velocidad. Cuanta más cantidad de letras leídas, más velocidad.

Figura 3.29: Se aconseja exigir al alumnado que lea con ambas manos a la vez, ya que se aumenta la velocidad, por lo menos en un 30 %. [126]

Los errores más comunes que se cometen en la lectura Braille son los siguientes [101]:

1. Confusión en alguna letra por omisión o adición de algún punto (por ejemplo, «p» en lugar de «m»; «a» mayúscula en lugar de «m», etc. Esta circunstancia resulta especialmente desmotivante para el lector adulto al comenzar el aprendizaje. Por eso, es conveniente enseñar esta particularidad y avisar de las posibles confusiones de antemano.
2. Confusión de letras simétricas (e-i, f-d, j-h, q-ñ, etc.) Por esta razón, es conveniente la enseñanza de las letras simétricas de forma distanciada. Por ejemplo, una vez que el niño ha interiorizado correctamente la letra «e», presentaremos la letra «i». Después se realizarán ejercicios de discriminación de ambas letras, ya que su parecido hace muy frecuente su confusión.
3. Pérdidas o saltos de renglón. Por esta razón, en algunos métodos de enseñanza del sistema Braille, se comienza dejando renglones en blanco para conseguir mayor separación entre ellos.
4. Terminación errónea de ciertas palabras de uso corriente.
5. Se leen más fácilmente las palabras cortas y conocidas.
6. Al leer se realizan tres tipos de movimientos: horizontales, verticales y de presión. Sólo los horizontales son los imprescindibles, por lo que los otros deberían desaparecer, porque hacen disminuir la velocidad y la comprensión y aumentan la fatiga.

3.2.6. Escritura del sistema Braille

En el sistema Braille, la escritura es más rápida que la lectura y suele presentar menos dificultad. Hace algunos años, la escritura del sistema Braille se

enseñaba desde los primeros niveles, con pauta y punzón. Una vez aprendida la escritura manual, se pasaba a la escritura a máquina. Ahora, se enseña a escribir desde el primer momento con la «máquina de punto positivo», una de las más utilizadas es la producida por *Perkins School for the blind*, de Massachussetts (EEUU), que en adelante llamaremos «máquina Perkins». Más adelante, cuando el alumno tiene adquiridas las técnicas y destrezas mínimas, se le presenta la pauta como un material complementario, que va a ser utilizado por el alumno de forma muy esporádica, para realizar pequeñas anotaciones cuando no tenga cerca la máquina (anotar un teléfono en la calle, etc.)

3.2.6.1. Escritura manual

La escritura manual se realiza con un punzón, con el que se perfora el papel, que está colocado sobre un soporte llamado pauta y con la ayuda de una rejilla.

La pauta consta de una plancha rectangular metálica o de plástico, del tamaño de la hoja en que se va a escribir (cuartilla o folio), horizontalmente atravesada por surcos o perforada con agujeros agrupados de seis en seis que se corresponden con las celdillas Braille. Sobre su perímetro se superpone un marco o bastidor que se une en la parte superior mediante una bisagra.

Perpendicularmente, y a lo largo de los lados laterales, se desliza una rejilla, lámina metálica o de plástico que contiene dos o tres líneas de cajetines, en cada uno de los cuales se escribe un signo Braille. Actualmente, la pauta y la rejilla vienen formando un solo cuerpo, en diversos tamaños y formatos, cuya denominación genérica es la de regleta.

El punzón se compone de un mango que puede ser de madera, metal o plástico, y de una punta metálica redondeada para no perforar el papel. Para manejar el punzón se coloca el dedo índice de la mano derecha sobre la parte superior del mango y los dedos pulgar y corazón sobre el borde que existe entre el mango y la punta.

Para empezar a escribir con la pauta se coloca el papel entre el marco y la plancha, de forma que el papel quede firmemente sujeto. A continuación, se coloca la rejilla (cuando esta es móvil) en los orificios laterales del marco y se comienza a escribir en el cajetín superior derecho de la primera línea.

Para escribir en la pauta es preciso escribir los signos invertidos, es decir, se escribe de derecha a izquierda invirtiendo el orden de la numeración de los puntos dentro del cajetín, lo cual hace que su aprendizaje sea mucho más difícil, al menos, en los primeros años, ya que exige al alumno tener bien definida su lateralidad y el concepto de reversibilidad[2].

[2]Reversibilidad: Capacidad de comprender y asimilar la propiedad característica del sistema Braille escrito en pauta, por la que es preciso dar la vuelta al papel para leer al derecho lo escrito.

Al escribir en la pauta, el punzón ha de estar perfectamente perpendicular al papel. Es conveniente pinchar los puntos de forma ordenada. El dedo índice de la mano que no escribe precede al punzón sobre la línea de escritura (no olvidar que se escribe de derecha a izquierda), ayudando a localizar el cajetín y a calcular los espacios que quedan libres para no cortar una palabra incorrectamente. Antes de comenzar a escribir letras conviene adquirir mecánicamente precisión en el punteado, ejerciendo la misma presión en todos los puntos.

Figura 3.30: Detalle de las manos de una persona escribiendo en Braille [126]

3.2.6.2. Escritura a máquina

La máquina para escribir en Braille consta de 9 teclas:

- Seis teclas, una por cada uno de los puntos Braille.
- Tecla espaciadora.
- Tecla para retroceder un espacio.
- Tecla de cambio de línea.

A cada punto de la celdilla Braille le corresponde una tecla. Para escribir una letra determinada presionamos, a la vez, las teclas que conforman dicha letra (así, por ejemplo, si queremos escribir la «m», deberemos pulsar las teclas correspondientes a los puntos 1,3 y 4).

Con el espaciador separamos las palabras. La tecla de retroceso se utiliza para volver a una posición anterior para corregir errores, al igual que sucede en las máquinas de escribir convencionales. El espaciador lineal se pulsa cada vez que se termine de escribir un renglón y/o se desee pasar al siguiente. La máquina tiene un timbre que avisa cuando la cabeza Braille se aproxima al margen derecho.

Figura 3.31: Máquina Perkins [126]

Para llegar a alcanzar una velocidad de trabajo aceptable, es necesario practicar mucho y una buena técnica que permita una mayor velocidad: cada tecla debe escribirse con un dedo determinado, para que se realice con la máxima rapidez y el mínimo esfuerzo:

- La tecla de avance de línea con el meñique izquierdo.
- La tecla 3 con el anular izquierdo.
- La tecla 2 con el corazón izquierdo.
- La tecla 1 con el índice izquierdo.
- La tecla espaciadora con los pulgares.
- La tecla 4 con el índice derecho.
- La tecla 5 con el corazón derecho.
- La tecla 6 con el anular derecho.
- La tecla de retroceso con el meñique derecho.

Al cabo de un tiempo no se necesita recordar qué puntos forman cada letra sino que, por memoria muscular, se marcan mecánicamente las teclas correspondientes a los puntos que forman cada letra.

El aprendizaje de la escritura con la máquina tiene grandes ventajas:

- Permite una mayor rapidez, los puntos de cada letra se pulsan a la vez, y no uno a uno como en la pauta. Es posible conseguir una velocidad similar a la de cualquier máquina de escribir, por lo que a la hora de realizar trabajos en el aula (dictados, toma de apuntes o exámenes), el ritmo puede ser igual (a veces, superior) al del resto de los compañeros.

- Además, la máquina marca el relieve de los puntos hacia afuera, en positivo, por lo que se puede leer inmediatamente lo que se escribe, sin necesidad de sacar el papel de la máquina, ni darle la vuelta. Permite así continuar sin dificultad una frase o palabra que se haya quedado sin terminar.

- Se escribe igual que se lee, de izquierda a derecha (en la pauta se escribe de derecha a izquierda y en espejo, para que se pueda leer lo escrito por el lado por donde aparece el relieve de los puntos, dando la vuelta al papel).

- Permite una mejor calidad de los puntos y facilita la asociación lectura-escritura.

- La máquina es muy útil para realizar operaciones matemáticas, sin tener que usar la caja de aritmética.

Figura 3.32: La máquina Perkins permite conseguir igualar e incluso superar el ritmo de escritura al resto de la clase. [126]

La máquina Perkins dispone de un sistema accesorio electrónico (Braille-n-print) para adaptarlo por debajo de la máquina que permite la conversión de un texto Braille escrito con la máquina a tinta, a través de una impresora convencional.

La utilización de la máquina tiene también algunas desventajas: su precio, peso, tamaño y el ruido que produce.

Como alternativa, sobre todo a partir del segundo y tercer ciclo de educación primaria, existen distintos modelos de anotadores parlantes (Braille'n Speak, Braille hablado, etc.) que ofrecen la posibilidad, además, de procesar la información y crear ficheros como un ordenador. Tienen un teclado similar al de la máquina Perkins pero, en lugar de grabar en un papel, lo hacen electrónicamente, en su memoria o en un disquete. Luego, se puede leer lo

escrito a través de un dispositivo de voz sintética o imprimirlo, en Braille o en tinta. Es muy útil para aquellas personas que pierden la vista en edad adulta ya que no es necesario que lean al tacto lo que escriben en Braille.

Figura 3.33: Detalle de las manos de una persona utilizando el Braille'n Speak [126]

De la misma forma que se indicó en el apartado dedicado a la lectura Braille, al escribir, tanto en pauta como a máquina, no se necesita recordar qué puntos forman cada letra sino que, por memoria muscular, se marcan mecánicamente las teclas (en la máquina) o los puntos que forman cada letra (en la pauta) correspondientes.

3.2.7. Algunos inconvenientes

El sistema Braille presenta algunas dificultades que es importante conocer, porque repercuten en la metodología de la enseñanza del sistema [101]:

- Las vocales acentuadas son diferentes a las no acentuadas, lo cual puede ser frustrante si dejamos para el final el aprendizaje de estas letras, que además son percibidas con dificultad. Conviene, por tanto, ir introduciéndolas poco a poco, desde el inicio del aprendizaje.
- En Braille no hay subrayados, márgenes, negritas, epígrafes, resúmenes, cuadros, tablas, etc. Por ejemplo, si comparamos un libro de texto de infantil y primaria en tinta con uno en Braille encontramos grandes diferencias. En tinta, están llenos de color, cuadros, subrayados, cambios de tipos y tamaños de letra, dibujos, etc. En Braille, por el contrario, resultan más aburridos y áridos, ya que el sistema no permite la utilización de recursos que, normalmente, hacen más atractivo el aprendizaje. Sólo podemos jugar con las sangrías y los espacios en blanco. Ni siquiera podemos variar el tamaño del cajetín (esto último sólo sería recomendable al inicio del aprendizaje, para el conocimiento del signo generador y ubicación de los puntos).

- Son necesarias signografías especiales para química, física, matemáticas, algunos idiomas, informática o música, por lo que es necesario el aprendizaje de muchas combinaciones.

  En general, la signografía especial se soluciona dando a cada signo un significado diferente, en función del contexto. Por ejemplo, el signo de admiración, representa también, en el contexto de las matemáticas, el signo de sumar. Otra forma de economizar signos es mediante la combinación de dos o más signos, lo que les hace tener un significado diferente. Por ejemplo, en Braille, las nueve primeras letras del abecedario son también los números del 1 al 9, si les ponemos delante un signo especial.

  Como ya hemos hecho referencia, con respecto a la signografía musical e informática se ha solucionado el problema ampliando el cajetín a 8 puntos, lo que multiplica las combinaciones posibles. En cuanto a las diferentes lenguas, se sustituyen algunos signos originales del sistema, propios del idioma francés, o se crean otros nuevos para los grafemas peculiares en otras lenguas. Por ejemplo, en castellano, fue necesario crear un signo especial para la letra ñ.

- Los textos en Braille ocupan más espacio que en tinta. Por ejemplo, la transcripción de una página en tinta al sistema Braille triplica su extensión. Un libro como *El Quijote* puede ocupar 14 volúmenes de unas 200 páginas cada uno. Un libro de texto de un alumno de educación secundaria puede ocupar más de 6 volúmenes. Esto ocasiona varios problemas: transporte y almacenamiento de los libros, organización de apuntes, etc.

  Para solucionar este aspecto, se idearon dos fórmulas:

  1. **Escritura interpunto**: es decir, escribir por los dos lados de la hoja, porque los puntos de un lado coinciden con los espacios entre los puntos del otro. Esto que resulta tan fácil y evidente en tinta, en Braille es necesario disponer de una impresora especial. De esta forma, se reduce a la mitad el volumen de las hojas (aunque hace difícil la lectura del Braille para una persona que lo lea con la vista).
  2. **Estenografía** (o escritura abreviada): Serie de normas que sirven para abreviar algunas sílabas y terminaciones finales. La finalidad es aumentar la velocidad lectora y reducir el espacio. También tiene signos propios. Por ejemplo: en estenografía, «que» se escribe «q». En español la estenografía no es muy utilizada, sin embargo, en francés y en inglés, es muy corriente. También se denomina Braille grado II.

- A medida que se lee un texto en Braille, los puntos se van degradando, por lo que en función del uso y trato que le demos, los puntos pueden irse desgastando en detrimento de la legibilidad y la velocidad lectoras. Por esta razón, es conveniente almacenar los libros en **vertical**, evitando apilarlos unos encima de otros.

- En cuanto a la **velocidad** más lenta del Braille con respecto a la lectura en tinta, podemos decir que un buen lector en tinta alcanza de 300 o 350 palabras por minuto. Sin embargo la velocidad media de los buenos lectores Braille es, salvo excepciones, de unas 150 o 200 palabras por minuto. Esto supone una clara desventaja. Efectivamente el carácter analítico asociado a las características de la percepción táctil hace que la velocidad de lectura disminuya considerablemente al leer en Braille. No obstante, existen algunos estudios que defienden que la menor velocidad en Braille es a causa de la falta de técnicas y didácticas adecuadas y que es posible la globalización y, por lo tanto, aumentar significativamente la velocidad.

  Para compensar esta menor velocidad, se utilizan aparatos como, por ejemplo, el *libro hablado*, que permite a la persona ciega acceder a los textos impresos mediante grabaciones especiales con las que se consigue una velocidad similar a la que se obtiene leyendo en vista. Son muchos los libros de texto de asignaturas de letras que se graban, en lugar de transcribirse al Braille.

Figura 3.34: Reproductor portátil de libros en formato Daisy que soporta diversos formatos. [126]

3.3. Didáctica del sistema Braille

En el capítulo anterior se han estudiado los métodos de aprendizaje de la lectoescritura usando el código visual. En este apartado vamos a analizar los

mecanismos que se ponen en marcha para el aprendizaje, didáctica, peculiaridades perceptivas y actividad psicomotora y táctil que requiere el sistema Braille.

Hay varios libros dedicados a esta cuestión, de los que destacaremos *Educación inclusiva: discapacidad visual* [61], *Guía didáctica para la lectoescritura braille* [101], *El niño ciego en la escuela* [30], *El Braille en la escuela: una guía práctica para la enseñanza del Braille* [47] y *La escuela y el niño ciego. Manual práctico* [39].

3.3.1. Elección del código

Una decisión importante con la que se encuentran los profesionales dedicados a la educación de personas con discapacidad visual es la elección del código de lectoescritura (tinta o Braille) que se debe enseñar a los alumnos cuando conservan un cierto resto visual.

Esta decisión se debe tomar valorando cada caso de forma individual, teniendo en cuenta, entre otros, los siguientes criterios [101]:

- Diagnóstico, pronóstico y evolución de la discapacidad visual
- Edad y competencia curricular del alumno.
- Desarrollo psicomotor (lateralidad, estructuración espacio-temporal, esquema corporal)
- Desarrollo de la percepción táctil, visual y auditiva.
- Motivación del alumno para el uso de un determinado sistema.
- Necesidades futuras. Tener en cuenta que cada vez son mayores las posibilidades de acceso a otros sistemas y medios de comunicación (posibilidades que aporta, actualmente, la tiflotecnología) y que requieren del conocimiento del sistema Braille.

En caso de duda, hay varias tendencias:

- iniciar al alumno en el código Braille, ya que resulta más sencillo pasar de la lectoescritura en Braille a la visual que a la inversa.
- trabajar en tinta, por ser la medida más normalizadora.
- enseñar los dos sistemas, ya que el código alternativo podrá servirle, en el futuro, como sistema de apoyo complementario.

Una vez realizada la elección de un código determinado, se deberá llevar a cabo un seguimiento del alumno para revisar la decisión tomada y estudiar la posibilidad de modificarla, si la evolución y el rendimiento así lo aconsejan.

En este trabajo se tratará solamente la lectoescritura de los discapacitados visuales mediante el código Braille.

3.3.2. Especificidades del sistema Braille

Martín-Blas [99] señala una serie de especificidades que distinguen el código Braille del visual, y que es preciso tener en cuenta:

- Se ejecuta en un espacio micrométrico. Esta circunstancia –que puede variarse en los primeros momentos de los aprendizajes, con el fin de facilitar el reconocimiento de los grafemas– exige un alto nivel de sensibilidad y discriminación táctil.

- Es un código muy rígido de formas. El error que pueda cometerse en la ubicación o localización de algunos de sus puntos es muy significativo, lo cual lo hace distinto del visual, en el que los rasgos pueden ser redundantes e, incluso, en ocasiones, superfluos.

- Es un código «frío» de formas. Si con el código visual (también denominado «en tinta») se pueden utilizar recursos didácticos y metodológicos basados en las posibilidades cromáticas, formales y flexibles que pueden componerse para hacer más atractivo su aprendizaje, en el Braille, esta posibilidad está ausente. No obstante, existen otros recursos metodológicos que facilitan e incentivan los procesos previos para la instrucción: juegos corporales y espaciales, puzzles, encajes, etcétera.

- Es un código en el que es significativa la direccionalidad. Las propias estructuras de muchos de sus grafemas representan signos «en espejo» o «simétricos» a otros. Por otro lado, cuando las letras en Braille se representan, en la escritura, por procedimientos manuales (no mecánicos) el sistema presenta la especificidad de su «reversibilidad», que consiste en que la escritura se lleva a cabo con dirección de derecha a izquierda, punzando en el papel con un punzón sobre una regleta o pauta, para después leer lo escrito, de izquierda a derecha, una vez que se ha dado la vuelta al papel.

- Presenta una difícil globalización. Muchos de los métodos de lectura visual se basan en el reconocimiento global de palabras, para proceder después al análisis de las mismas, métodos que tienen muy en cuenta el campo (central y periférico) con el que actúa el sentido de la vista. La percepción táctil es sumamente concreta, analítica y secuencial, lo que hace que, en el terreno del reconocimiento de palabras, se perciban éstas letra a letra. Niveles más altos de globalización aparecerán más tarde por diversas vías, como, por ejemplo, por destrezas lectoras adquiridas, por técnicas de anticipación basadas en el desarrollo lingüístico y cultural del individuo, etcétera.

3.3.3. Aspectos metodológicos

En la enseñanza del Braille habrá que tener en cuanta algunos principios didácticos [30]:

1. El principal objetivo que nos planteamos es la enseñanza de la lectoescritura, a través de un código concreto, el Braille. No es, pues, el sistema el objeto principal de nuestra labor, sino la habilidad para leer y escribir. Es esencial fomentar el interés del niño por estas habilidades, de igual manera que se hace con el niño vidente. Recordemos siempre que estamos enseñando a leer y escribir, y no enseñando Braille.

2. De los métodos de enseñanza al uso, se han descartado los llamados analíticos (véase la sección 2.2) en razón al hecho de que la información táctil es secuencial. Los dedos que se usan en la lectura tienen poca capacidad para recibir una gran cantidad. de información simultánea, al contrario de lo que ocurre con la visión. El niño vidente puede apreciar con un «golpe de vista» una frase entera, algo imposible para la lectura táctil. Ello no implica que en estados avanzados del aprendizaje, el lector de Braille no utilice mecanismos, como el uso del contexto y la experiencia táctil, que facilitan la rapidez lectora y dan la impresión de «globalización». Aunque ha habido alguna propuesta para usar un enfoque constructivista, como la de Izquierdo y Lleida [80], la mayoría de los métodos para la enseñanza del Braille en nuestra lengua presentan una metodología sintética.

3. Desde el principio se plantea la enseñanza del Braille integral. El Braille abreviado o estenográfico se emplea sólo en publicaciones de cierto nivel y nunca en libros de texto.

4. La escritura Braille deberá iniciarse en la máquina antes que en la pauta, y ello por tres razones principales:

   a) Se evita la duplicidad de códigos (uno de lectura y otro de escritura), máxime cuando ello requiere un buen sentido de orientación espacial en edades en que el niño todavía no lo tiene bien adquirido.

   b) Permite la simultaneidad de lectura y escritura.

   c) El uso de la máquina exige menos esfuerzo y precisión que el uso del punzón y la pauta, logrando una escritura mas rápida.

5. Se debe procurar planificar el inicio de la enseñanza del sistema Braille con suficiente antelación, de manera que el niño disponga en el colegio de una máquina.

6. No debemos olvidar que el uso de la pauta es necesario en múltiples situaciones de la vida del ciego, por lo que se plantea que conozca también la técnica en este instrumento en etapas educativas superiores, cuando el niño ya domina bien el sistema Braille.

7. Se evidencia que, independientemente de la bondad de un método concreto, la capacidad del niño, la actitud del profesor y el ambiente familiar son variables a tener muy en cuenta en el proceso.

3.3.4. Requisitos básicos

El niño con ceguera puede y debe iniciar el aprendizaje de la lectoescritura Braille al mismo tiempo que sus compañeros videntes se inician en la lectoescritura en tinta.

El aprendizaje del sistema Braille es lento y requiere motivación, atención y concentración. Por tanto, su enseñanza necesita unas condiciones previas y una actitud positiva del niño hacia el aprendizaje. Se debe comenzar con actividades simples, que puedan ser realizadas sin dificultad y con éxito.

Un niño vidente ve de forma continua y natural libros, carteles en la calle, periódicos... Encuentra una serie de estímulos en su vida diaria que le llevan a interesarse por la lectura. El niño con discapacidad visual no suele disponer de estos estímulos. Por tanto, es conveniente adaptar en Braille y con dibujos en relieve, los carteles y letreros que están en su aula o en casa y ponerlos a su altura, así como dejar a su alcance libros y cuentos en relieve.

Es muy significativo para el niño que alguien de su entorno aprenda también el sistema Braille. El apoyo de la familia, amigos o compañeros de clase es importante, sobre todo si el niño está escolarizado en un centro donde no hay más niños con discapacidad visual.

La ceguera no implica, automáticamente, tener un tacto más sensible, en todo caso, el tacto puede ir desarrollándose con la práctica. Existen algunas enfermedades que pueden provocar ceguera, como la diabetes que, incluso, conllevan una disminución de la sensibilidad táctil.

Como cualquier otro niño, el invidente necesita haber adquirido un cierto grado de desarrollo madurativo (esquema corporal asimilado convenientemente), así como un entrenamiento táctil, que necesitará para manejar los instrumentos de escritura y para acceder a la información en la lectura.

3.3.4.1. Actividades para desarrollar las destrezas previas

El desarrollo táctil del niño ciego es un proceso que se inicia desde su nacimiento. El apresto Braille está dirigido a estimular en el niño todas aquellas habilidades que le permitan desarrollar movimientos finos de las manos y dedos, reconocer objetos, formas y figuras táctiles con variación progresiva del tamaño y complejidad, e incorporar conceptos básicos vinculados a la direc-

cionalidad, a la secuenciación, a la comparación de igualdades y diferencias, nociones de cantidad, etc.

Presentamos algunas actividades que, en general, pueden ayudar a desarrollar las destrezas previas a la enseñanza del Braille:

1. Desarrollo de la motricidad gruesa y fina, realizando actividades de:

   - Automatización de desplazamientos de brazos de izquierda a derecha y viceversa.
   - Coordinación dígito-manual: encajar bloques; ensartar piezas y bolas; introducir objetos en recipientes; picar sobre papel; modelar con plastilina; arrugar, rasgar, doblar y recortar papeles; apilar diferentes materiales; pellizcar; pegar; enroscar; ensamblar; abrochar botones; abrir y cerrar cremalleras, pintar con los dedos, modelar con arcilla y plastilina, etc.
   - Reconocimiento de objetos tridimensionales y formas.
   - Picado con punzón con límites, por ejemplo dentro de figuras geométricas o de figuras de diferentes formas y dimensiones.
   - Seguimiento de líneas continuas y discontinuas, discriminación de puntos en el papel, localización de puntos, habilidades básicas de encajes y ensambles, técnicas elementales de presión y prensión de las manos y los dedos de objetos de diferentes tamaños, destrezas para picado, etc.
   - Ejercicios de disociación manual: abrir y cerrar alternativamente las manos, mover rítmicamente la posición de las manos (palma derecha arriba y palma izquierda abajo), golpear la mesa con cada mano en posición diferente (de canto, de plano, etc.), accionar de diferente forma cada mano (mientras que una mano golpea, la otra traza círculos, o bien, una traza líneas verticales y la otra, horizontales, etc.)
   - Ejercicios de separación de dedos: movimientos de oposición del pulgar a los otros dedos, golpear cada dedo con su pareja, levantar o flexionar separadamente cada dedo, teclear sobre la mesa, etc.
   - Ejercicios de adiestramiento general de dedos: marcar el paso con los dedos, mantener el equilibrio de una moneda en un dedo, clavar chinchetas, abrir y cerrar pinzas de la ropa con el pulgar y cada uno de los dedos, imitar gestos con los dedos, girar manivelas, enroscar tuercas, romper macarrones, escurrir el agua de esponjas, usar una grapadora, etc.

2. Aprendizaje de conceptos básicos:

- Reconocer, clasificar, emparejar y ordenar objetos de diferente tamaño, forma o textura.
- Conocer conceptos espaciales básicos: arriba, abajo, delante, detrás, al lado de, en medio, izquierda, derecha.
- Nociones de cantidad: más, menos, uno, lleno, vacío, ninguno, pocos, muchos, y cantidades de 1 a 10.
- Conocer conceptos sobre cualidades: relaciones de semejanza, de diferencia, de tamaño, peso, textura, rugosidad, forma y grosor.

3. Desarrollo senso-perceptivo:

   - A través del tacto el niño puede conocer su cuerpo, las texturas, la temperatura, tamaño, forma, etc. No sólo la experiencia táctil se limita a las manos, sino que se debe relacionar con cualquier parte del cuerpo: andar descalzo sobre texturas diferentes, etc.
   - Identificar diversas fuentes de sonidos, discriminar y repetir secuencias de sonidos, ritmos, etc.
   - Percepción y reconocimiento del espacio: reconocimiento derecha, izquierda, arriba, en medio, abajo, etc.
   - Relaciones espaciales con respecto a sí mismo, con respecto a los demás y entre los objetos y las personas.

4. Desarrollo de la memoria, atención y observación:

   - Actividades relacionadas con cuentos, narraciones, descripciones, dramatizaciones y adivinanzas.

Hemos ofrecido toda una serie de actividades para desarrollar las destrezas previas necesarias para el aprendizaje del sistema Braille. No obstante, aunque no es lo más ortodoxo, hay alumnos que no consiguen estas destrezas y, sin embargo, llegan a leer y escribir sin problema.

3.3.4.2. Actividades específicas de iniciación a la lectoescritura Braille

Una vez adquirida e interiorizada la estructura espacial del signo generador y conseguida la suficiente madurez dígito-manual, mediante la realización de las actividades propuestas en el epígrafe anterior, proponemos aquí actividades relacionadas directamente con el aprendizaje del sistema Braille:

- Reconocimiento de líneas de puntos de distinta longitud y dirección.

- Jugar con el muñeco Braillín. Muñeco que tiene en el cuerpo los 6 puntos del signo generador Braille. Favorece, por una parte, el desarrollo educativo, al ser un elemento de familiarización y de iniciación al sistema Braille, pero además, favorece la aceptación y enriquecimiento ante la diversidad al poder ser utilizado también por los niños videntes.

Figura 3.35: Virginia Pérez de Vallejos, creadora del muñeco Braillín [1]

- Enhebrado de cuentas, insertado de objetos o clavijas en agujeros, uso del punzón en superficies más o menos limitadas (pinchar dentro de un círculo, sobre una línea, etc.)
- Discriminación de posiciones de los puntos. Existe material específico para la iniciación al Braille como por ejemplo: signo generador, regletas de iniciación y regletas de preescritura. El objetivo en todos estos materiales es ampliar el tamaño de la celdilla Braille para que resulte más asequible el aprendizaje de la ubicación de los puntos.
- Ejercitación en el paso de páginas.
- Ejercicios de ubicación de los seis puntos utilizando material tridimensional: hueveras de media docena, tablero de 6 pivotes, cajas con 6 compartimentos, etc. Se pueden llevar a cabo diferentes ejercicios: localizar cada uno de los puntos a indicación nuestra, indicar qué puntos faltan, realizar dictados de puntos, etc. El objetivo es que aprenda el nombre de cada punto y su ubicación.
- Lectura de puntos Braille sobre papel, con fichas realizadas especialmente, en las que sólo aparezcan combinaciones de 1 ó 2 puntos. El

objetivo inicial es que sólo discrimine cuántos puntos hay en cada espacio y si están uno al lado del otro o uno encima del otro.

- Progresivamente se va complicando el ejercicio, con combinaciones de los puntos 1, 2, 3 y 4, para continuar después introduciendo paulatinamente los puntos 5 y 6.

- Una vez que el niño conoce los nombres de los puntos y sabe localizarlos en cualquier combinación, es el momento de comenzar a darles significado: aprender poco a poco las letras y sus combinaciones, como en cualquier otro método de lectura en tinta.

- Al comenzar el aprendizaje es conveniente dejar más espacio (dos espacios, por lo menos) entre cada renglón a leer, para facilitar al alumno el seguir la línea sin bajar o subir a la de al lado.

Existen varios métodos y cartillas para el aprendizaje del sistema Braille que, precisamente, comienzan trabajando la prelectura y preescritura Braille en la línea expuesta:

1. «**Percibo y trazo**», de Jerónima Ipland García [78]. Láminas en Thermoform realizadas en el CRE «Luis Braille» de Sevilla, para el desarrollo de la percepción táctil, que contiene seriaciones, cuantificaciones, clasificaciones y correspondencias.

2. Fichas para la «**Adquisición y desarrollo de conceptos básicos**», de Trinidad González de Cara, Socorro Sánchez Crisol y Milagros Suárez Vilar [24]. Se pretende desarrollar el aprendizaje de técnicas básicas de discriminación táctil, reconocimiento de formas, constancia perceptual, relaciones espaciales y posición en el espacio. Además se proponen actividades para desarrollar la percepción táctil, potenciar el resto visual. Se ofrece un programa para elaborar con diferentes texturas las láminas para la consecución de los objetivos anteriores.

3. **Método Alameda**. Véase la sección 3.4.2.

4. **Punto a punto**. Véase la sección 3.4.3.

3.3.5. Aprendizaje de las primeras palabras

En esta etapa se persigue un objetivo básico: que el niño comprenda el significado de la lectura y la escritura como actividad que les permite comunicarse.

Para ello deberán tenerse en cuenta un conjunto de particularidades que van a facilitar el proceso de enseñanza-aprendizaje [47]:

Letras de forma simétrica También llamadas letras «en espejo».

Cualquiera que sea la metodología utilizada para la enseñanza de la lectoescritura Braille, debería evitar una presentación ordenada de las letras del alfabeto tal y como aparece en algunos métodos alfabéticos en tinta. Será mucho más fácil evitar confusiones si no mostramos las vocales a la vez, por su semejanza espacial en la distribución de los puntos (e-i-o).

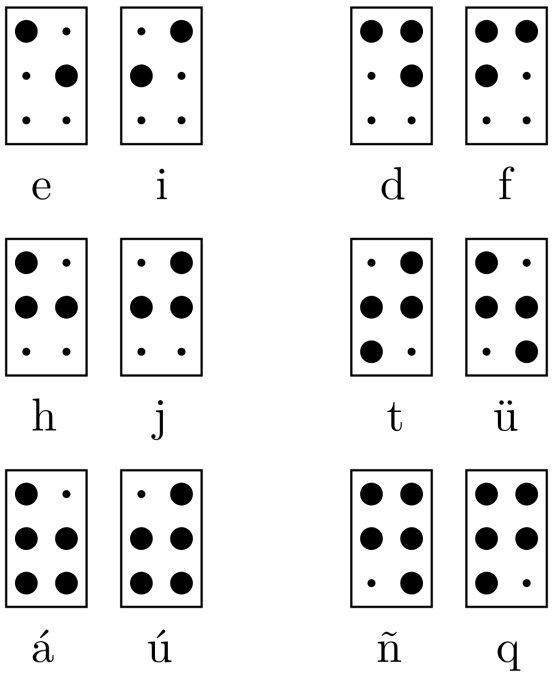


Figura 3.36: Letras simétricas

Letras con estructura semejante Es muy importante no enseñar simultáneamente, desde un principio, letras muy parecidas, para evitar posibles confusiones que dificultarían el proceso de enseñanza, y que en muchas ocasiones son difíciles de corregir una vez que se han adquirido.

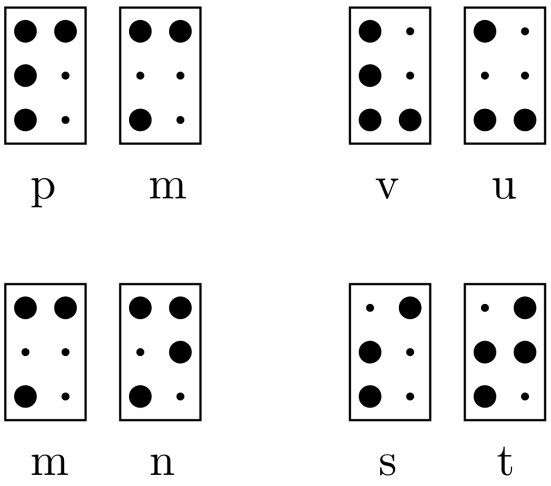


Figura 3.37: Letras con estructura semejante

Letras con estructura espacial simple En la primera etapa del proceso de enseñanza-aprendizaje deberán presentarse palabras con letras de estructura espacial simple. Tal es el caso de las que llevan puntos en una misma dirección o en los extremos:

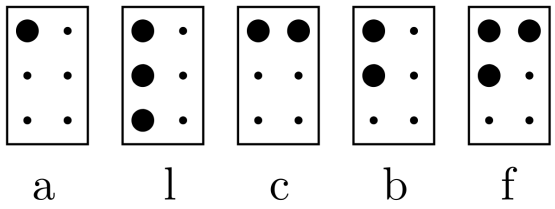


Figura 3.38: Letras con estructura espacial simple

Cuando el niño sea capaz de reconocer este tipo de letras, le será mucho más fácil aprender las que distribuyen sus puntos de forma discontinua, dejando huecos entre las filas del cajetín:

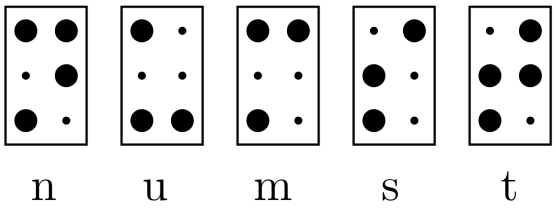


Figura 3.39: Letras con huecos intermedios

Fonemas idénticos con grafías diferentes Las semejanzas entre algunos

sonidos y la diferente forma que tienen de escribirse es otro de los aspectos importantes a tener en cuenta, al igual que en el aprendizaje de la lectura en tinta:

- ca - que - qui - co - cu
- za - ce - ci - zo - zu
- ga - gue - gui - go - gu
- ja - ge - je - gi - ji - jo - ju

La escritura de este tipo de fonemas debe enseñarse al niño cuando haya aprendido los más conocidos. En primer lugar, los que no presenten confusión a la hora de escribirlos. Por ejemplo, se le presentarán las sílabas ca - co - cu, para más adelante llegar a que - qui - ce - ci. Al igual ocurrirá con ga - go - gu y za - zo - zu.

3.3.5.1. Cómo enseñar las primeras palabras

Cuando el alumno ya conoce lo que es una regleta, el nombre de los puntos y es capaz de hacer dictados de sus posiciones sin apenas equivocarse, es hora de aprender a leer y escribir palabras.

Teniendo en cuenta la secuencia sugerida por Susana Crespo [39] y de acuerdo con su propia experiencia, Begoña Espejo [47] presenta un grupo de palabras que el niño puede aprender en una primera etapa del aprendizaje de la lectura y la escritura:

- ala (lala, la)
- pala (papa, lapa)

Aspectos importantes:

- No existe un tiempo determinado para la realización de los ejercicios que vamos a ver a continuación. Hay niños que aprenden una letra por día y otros que necesitan dos, tres o más. El resultado es siempre el mismo: aprenden a leer con facilidad. No se debe tener prisa.
- Nunca se enseñarán dos letras el mismo día en esta primera etapa.
- No se enseñará una letra nueva sin haber superado todos los ejercicios de la anterior.
- No debemos decirle al niño el nombre de las letras, sino el sonido del que forman parte. Por ejemplo:

Incorrecto	Correcto
la «l» con la «a» es LA	LA
la «b» con la «i» es BI	BI

El niño debe aprender la lectura/escritura del sonido, no el nombre de cada letra por separado.

3.3.5.2. Secuencia de palabras a seguir

Tras el aprendizaje de las palabras de la primera fase (ala - pala), se continuará con el mismo procedimiento y el siguiente orden, de acuerdo con este modelo:

Palabra tipo	Letra nueva	Palabra	Letra	Palabra	Letra
ola	o	mesa	s	niño	ñ
bola	b	goma	g	yo	y
palo	p	uva	v	tomate	t
mamá	m	ojo	j	cocina	ce - ci
caca	c	llave	ll	magia	ge - gi
pelo	e	pipí	i	chocolate	ch
dedo	d	rosa	r	guitarra	gue - gui
peluca	u	perra	rr	queso	que - qui
café	f	zumo	z		
nene	n	hoja	h		

3.4. Métodos de enseñanza

Una vez conseguidas la capacidad táctil previa y alcanzadas las destrezas generales y específicas propuestas en los epígrafes anteriores, se puede empezar a abordar la enseñanza del sistema Braille propiamente dicha.

Cada método defiende su propio orden en la presentación de las letras. Sin embargo, podemos decir que lo más aconsejable es seguir, siempre que sea posible, el mismo método de aprendizaje de la lectoescritura en tinta establecido en el aula para el resto de los compañeros videntes del alumno con ceguera, ya que este es el caso de la mayoría de los alumnos con discapacidad visual.

Muchos métodos se basan en que el niño debe conocer previamente el espacio rectangular que ocupan los 6 puntos de la celdilla y su ubicación. Para ello, se pueden utilizar algunos materiales que reproducen la celdilla Braille en grande y facilitan al niño el aprendizaje. Existen, por ejemplo, pizarras «macrobraille», con celdillas grandes en las que hay que insertar pivotes con los que el alumno se va familiarizando con los signos. No obstante, nos sirve cualquier material de desecho que tenga esta forma (hueveras de media docena, etc.).

Figura 3.40: El muñeco Braillín se puede utilizar como material para enseñar Braille ya que en su cuerpo se reproduce la celdilla Braille en grande. [126]

Nuestro idioma favorece la utilización de métodos sintéticos para el aprendizaje de la lectoescritura, es decir letra a letra o, como mucho, sílaba a sílaba, que es, precisamente la metodología que presentan la mayoría de los métodos de aprendizaje del sistema Braille.

El método analítico o globalizado no sería adecuado para el aprendizaje del Braille, ya que parte de la frase, para ir descendiendo a la palabra, la sílaba y la letra. No obstante, con mucha práctica, es posible llegar a reconocer en Braille algunos grupos de sílabas o palabras (imagen táctil) lo cual hace aumentar considerablemente la velocidad de lectura.

Como hemos ido analizando, existen diferentes factores que inciden en la capacidad lectora: la motivación, la cantidad de estímulos lectores que recibe el niño, la edad de comienzo de la lectura, el grado de desarrollo de las destrezas previas, la maduración del alumno, los apoyos que encuentra en el entorno y la metodología empleada. En función de todas estas variables, debemos elegir el método a utilizar.

No obstante, para respetar la inclusión del niño en su centro educativo, se debería utilizar el método con el que se esté trabajando en el aula realizando las adaptaciones de material que fueran necesarias.

A continuación presentamos los **métodos más utilizados para niños**, y tras ellos los métodos para adultos (sección 3.4.5).

3.4.1. Tomillo

Es un método de iniciación a la lectura Braille realizado por Rosa María Lucerga Revuelta y María Jesús Vicente Mosquete [95], y dirigido, especialmente, a la población infantil. Apoya la presentación de contenidos significativos, al mismo tiempo que respeta las peculiaridades de la exploración táctil. Además, se adecua a la edad a la que va dirigido, utilizándose palabras y frases cortas con sentido, con estructuras lingüísticas familiares para el niño.

Se emplean materiales atractivos para estas edades, con representaciones en relieve.

Consta de dos volúmenes: el volumen 1 comienza directamente con las vocales, menos la «i» (para evitar inversiones con la «e») presentándolas una por página con un objeto bidimensional que comienza por la vocal o letra a estudiar. Puesto que son dos volúmenes con anillas, se pueden sacar las fichas de forma independiente según la que se quiera trabajar. Al final de cada unidad hay una serie de ejercicios de repaso.

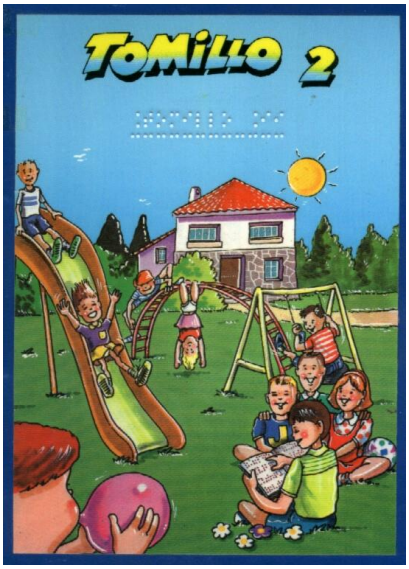


Figura 3.41: Portada del método Tomillo [95]

Representando las dificultades específicas del sistema Braille de reversibilidad, similitud de signos, dificultades de percepción, así como otras características de la lengua castellana: dificultades fonéticas y ortográficas, pretendiendo compatibilizar, en la medida de lo posible, la utilización de este método con los más comúnmente usados en las escuelas ordinarias, sigue esta secuencia de presentación de las letras: a, o, u, e, l, p, á, b, c (en los sonidos /ka/, /ko/, /ku/), d, m, signo de mayúscula, punto, i, n v, ó, s, g (en los sonidos /ga/, /go/, /gu/), t, f, r, í, ll, j, z, ñ, é, h, y, x, ch, ú, q, rr, r (en su sonido débil), gu (en los sonidos /gue/, /gui/), g (en los sonidos /ge/, /gi/), c (en los sonidos /ce/, /ci/), sílabas directas dobles. Es decir, se presentan, en primer lugar, las letras que se perciben más fácilmente al tacto, las que no presentan dificultades fonéticas y se evita unir letras simétricas. Se van introduciendo desde el principio las vocales con tilde. Utilizan doble espacio para facilitar la lectura y el cambio de línea.

3.4.2. Alameda

Es un método para la maduración lectoescritora de alumnos ciegos y deficientes visuales de 3 a 6 años, escrito por Julia Fuentes Hernández, profesora de apoyo del equipo interdisciplinar MEC-ONCE-IMSERSO.

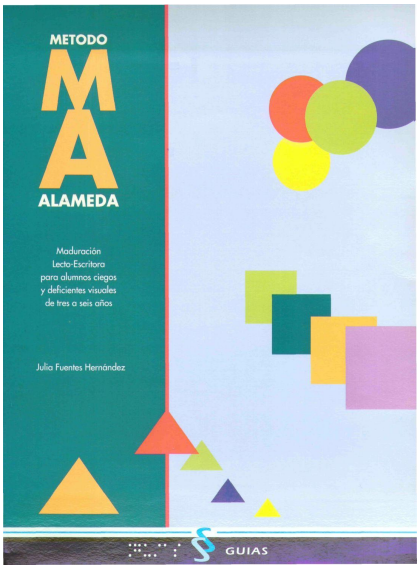


Figura 3.42: Portada del método Alameda [62]

Este método [62] está constituido por seis bloques. Por cada uno de estos bloques hay una maleta de actividades y material para su desarrollo de fácil utilización por parte del profesor tutor del aula.

El primero de los bloques está dedicado al desarrollo de la percepción táctil; el segundo a la discriminación de figuras geométricas; el tercero a los conceptos básicos espaciales; el cuarto, a las técnicas de seguimiento digital y rastreo; los bloques quinto y sexto comienzan ya con el aprendizaje del alfabeto Braille y la máquina Perkins.

El método dispone también de unas hojas de control para seguimiento del programa: objetivos específicos, tareas, actividades, nivel de logro, materiales y observaciones.

En este método el orden de introducción de las letras es variable en función de la programación seguida por el profesor tutor en clase.

3.4.3. Punt a punt

El método «Punt a Punt» [109] fue realizado por Maite Mañosa Mas y Josefina Miret Serra, del equipo del Centro de Recursos Educativos (CRE) de la ONCE «Joan Amades» de Barcelona, dirigidas por María Ángels Esteban Picó, y producido por el Centro de Producción Bibliográfica de la

ONCE. Se presenta en castellano y en catalán. Consta de dos series de 5 y 4 tomos, respectivamente. La primera serie presenta un programa de prelectura y preescritura y la segunda se dedica a la enseñanza del sistema Braille propiamente dicho:

Primera serie: en los tres primeros tomos se ofrecen una serie de ejercicios de prelectura, para el reconocimiento de formas (cuadrado, círculo, triángulo, rectángulo) y tamaños, líneas horizontales y verticales, y seguimiento de líneas y orientación espacial con cuadrados, líneas y series de varios elementos (conjuntos de puntos), para proceder a su discriminación, aun sin darles un significado. En el cuarto tomo es cuando se inicia el aprendizaje del sistema Braille: el signo generador, en grandes caracteres con el objetivo de ir disminuyendo el tamaño e ir reconociendo las diferentes posiciones y las primeras letras: a, b, l, e, o. También se comienza la preescritura. En el último tomo se hacen consideraciones metodológicas para el profesor.

Segunda serie: en los tres primeros tomos, se van presentando una a una todas las letras del alfabeto, se hacen ejercicios de reconocimiento táctil, identificación y discriminación, combinando con las letras ya sabidas, para pasar a la lectura de sílabas, palabras y frases. El orden de presentación de las letras es el siguiente: a, o, u, e, l, p, b, m, n, f, i, signo de mayúscula y punto, r, s, apóstrofe, t, ll, c, admiración, d, interrogación, g, j, á, í, ú, v, coma, x, h, q, punto y coma, ñ, z, dos puntos, é, ó, ü, t y k; (en la versión en catalán, se añaden las letras è, ò, ç, ï). Es decir, se presentan, en primer lugar, las letras que se perciben más fácilmente al tacto, las que no presentan dificultades fonéticas y se evita unir letras simétricas. Se van introduciendo desde el principio las vocales con tilde. Utilizan doble espacio para facilitar la lectura y el cambio de línea. El cuarto tomo está dedicado a los maestros, donde se les explica en qué consiste el método y cómo utilizarlo.

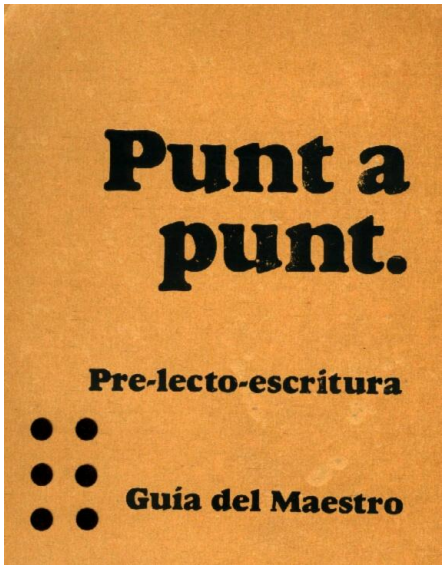


Figura 3.43: Portada del método Punt a punt [109]

Se acompaña de dibujos en relieve para motivar al alumno y de ejercicios para reforzar la discriminación de las letras. Se diferencia del método Tomillo en que trae muchas actividades de preescritura y prelectura, más que la lectura en sí, a la que dedica únicamente el cuaderno cuatro.

3.4.4. Otros métodos

En el Primer Congreso Estatal sobre prestación de servicios para personas ciegas y deficientes visuales, celebrado en Madrid en 1994, se presentaron varios métodos más:

- Almazara [180], de Carlos Valbuena García, Gregorio Cruz Santa Bárbara y Juan C. Rodríguez Ortega.

- A punto [97], de Ángel Martín-Blas Sánchez.

- Seis Puntos [64], de María Araceli Garcés Castillo y Blas Garcés Lázaro.

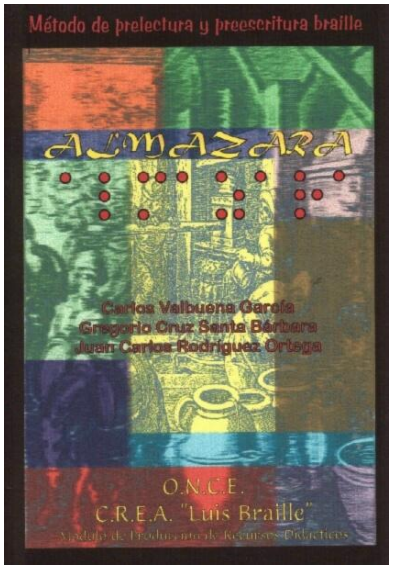


Figura 3.44: Portada del método Almazara [181]

3.4.5. Braille para personas adultas

Un adulto que acaba de perder la visión necesita aprender un nuevo código de lectoescritura: el código Braille. Para ello, deberá dominar las destrezas necesarias para acceder al sistema y tener las condiciones y capacidades necesarias para iniciar el aprendizaje, es decir:

- Desarrollar destrezas manipulativas y táctiles (entrenar el sentido del tacto).
- Adaptación psicológica a la nueva situación: evidentemente si la persona aún no tiene asumida su nueva situación puede sentirse muy bloqueado ante el aprendizaje del sistema por lo que podríamos provocar rechazo. En cambio, a otras personas les puede suponer un alivio descubrir que el aprendizaje del sistema les abre expectativas y posibilidades que creían perdidas. Siempre habrá que valorar la situación entre los distintos profesionales y el alumno.

No parece adecuado enseñar el sistema Braille a una persona ciega adulta con la misma metodología que se enseña a un niño. El adulto, en primer lugar, debe adaptarse a su nueva situación. Además, hay que tener en cuenta que va a tener menor facilidad para discriminar táctilmente las letras, ya que no tiene desarrollada la percepción táctil.

La enseñanza del Braille a las personas adultas debe formar parte de un programa de rehabilitación individual.

Se debe evitar que el alumno llegue a rechazar el sistema a causa de las dificultades con la discriminación. Incluso, hay autores que defienden la

posibilidad de representar el sistema con tablillas que representen las celdillas a mayor tamaño.

Si la persona aún conserva un buen resto visual puede ser aconsejable, en un primer momento, aprender el sistema Braille utilizando la vista. Hay investigaciones que afirman que, de esta forma, es más fácil, después, acceder al Braille a través del tacto.

Debido a las especiales características de los adultos (alfabetizados o no), se han creado para ellos diferentes métodos, como los que se exponen a continuación.

3.4.5.1. Alborada

Es una cartilla para el aprendizaje de la lectura. Presenta las letras en un orden bastante lógico, con frases de creciente complejidad. Aunque el contenido de las frases ha quedado algo desfasado, resulta un método fácil de utilizar y motivador para los alumnos adultos ya que, desde las primeras páginas, leen palabras y frases con significado. El orden de presentación de las letras tiene en cuenta la sencillez o complejidad de los signos: a, o, u, e, l, p, i, b, m, s, n, v, d, ñ, g, t, f, ll, r, c, y, j, q, h, z, x, ch, k, punto, signo de mayúscula, sílabas trabadas, á, é, ó, coma, punto y coma, dos puntos, guión, í, ú, ü, w, interrogación, admiración y signo de número [65].

Este método cumplió durante décadas un papel intermedio entre las primeras transcripciones textuales de las cartillas para niños y los materiales o métodos más modernos en la actualidad.

3.4.5.2. Bliseo

Es un método para el aprendizaje del sistema Braille destinado a adultos alfabetizados. Empieza profundizando en el conocimiento especial del signo generador y va introduciendo las letras de la primera serie (de la «a» a la «j»), para seguir con la siguiente serie, añadiendo el punto 3 (de la «k» a la «t», excepto la «ñ») y las 5 últimas letras, añadiendo el punto 6 [157].

Está diseñado pensando en adultos ciegos alfabetizados, por lo que superada la fase de habituación al tacto, sigue el orden alfabético tradicional en la sucesión de letras del abecedario, terminando con una serie de lecturas de contenido literario.

3.4.5.3. Flash

Flash [138], de Rosa María Quílez García y Santi Moese Ruiz fue otro de los métodos presentados al Primer Congreso Estatal sobre prestación de servicios para personas ciegas y deficientes visuales, celebrado en Madrid en 1994.

3.4.5.4. Pérgamo

El método Pérgamo [13] va dirigido a la persona adulta que necesita utilizar el código Braille y que se encuentra incluida en alguno de los distintos tipos de analfabetismo:

- **Inicial o absoluto**: el que presentan aquellas personas que, por diversas razones, nunca asistieron a la escuela, o la abandonaron prematuramente y carecen por tanto de las capacidades básicas de lecto-escritura.
- **En retroceso**: originado por el olvido y el desgaste de conocimientos adquiridos no actualizados.
- **Funcional**: manifestado por el déficit en conocimientos básicos que la sociedad demande para desenvolverse adecuadamente en ella.

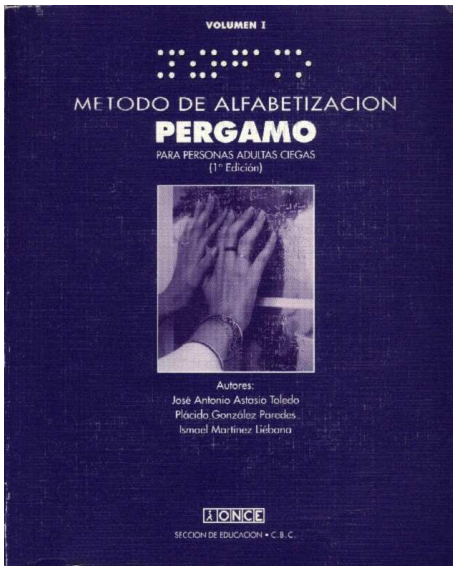


Figura 3.45: Portada del método Pérgamo [13]

Consta de **dos partes** claramente diferenciadas: elementos de prelectura y el método de alfabetización propiamente dicho. Además, incluye una interesante guía didáctica dirigida a los profesionales de la enseñanza.

Elementos de prelectura El objetivo de estos elementos es favorecer el desarrollo de la capacidad táctil del educando ciego adulto, con vistas a la utilización del sistema Braille.

Partiendo de la base de que los caracteres del código Braille son, ante todo, formas espaciales diferentes, para cuya conformación el método de puntos de las mismas constituye sólo un factor secundario, el entrenamiento del

adulto en su percepción y reconocimiento táctiles representa una actividad prioritaria y esencial.

El principio por el que se rige la presentación de las formas Braille (carentes todavía de significación gráfica determinada para el adulto) es el método analítico-sintético, basado en el modo espontáneo de acceso cognoscitivo al objeto. Tal procedimiento comprende dos partes complementarias: por un lado, la descomposición del todo en sus partes constituyentes y, por otro, la recomposición de estas en el todo.

Así, según esto, son tres las fases principales en la aparición de las diferentes formas Braille:

1. Presentación inicial del todo (los seis puntos Braille) como forma completa y referencial, a partir de la cual han de construirse, por análisis, el resto de formas del sistema.

2. Descomposición progresiva del todo en sus partes, hasta llegar finalmente a la forma atómica del sistema: el punto.

3. Por último, el procedimiento analítico-sintético culmina en su fase recompositiva, mediante la cual, partiendo de la unidad elemental, se llega en sentido inverso al punto de inicio: los seis puntos Braille.

El desarrollo de las diversas formas Braille se estructura en diez bloques, conteniendo cada uno de ellos diversas series. Las distintas formas aparecen siguiendo un orden establecido que se mantiene fijo para todos los bloques:

- Secuencia discontinua (formas aisladas).
- Secuencia continua (formas encadenadas).
- Secuencia combinada de las dos anteriores.

Los diez bloques son:

1. Se estructura en torno al signo generador completo (los seis puntos); de él se extraen puntos de diversas posiciones, creando en todos los casos formas de cinco puntos.

2. La forma referencial está constituida por cuatro puntos, los números 1 3 4 6 –las cuatro esquinas del signo generador–; a base de cambiar de posición uno de los cuatro puntos mencionados se obtienen formas Braille en ángulos rectos.

3. Arranca de la forma referencial constituido por los puntos 2345; manteniendo fijos el 2 y el 5 y variando la posición de los otros dos, se desarrollan diferentes formas Braille.

4. Se basa en la forma obtenida por los puntos 2 3 4 5; esta forma básica se invierte horizontal y verticalmente, dando origen a cuatro formas diferentes.

5. Se estructura en base a un conjunto de tres puntos, los números 1 3 4; mediante inversiones horizontales y verticales se obtienen las distintas formas que, en todo caso, incluyen «tres esquinas» del signo generador.

6. La referencia la constituyen los puntos 1 2 3; variando horizontalmente de posición uno, dos o los tres, se obtienen diferentes formas.

7. La forma base la constituye un conjunto de dos niveles horizontales del signo generador –los puntos 1 2 4 5 ó 2 3 5 6–; haciendo desaparecer uno de los puntos, se producen diversas formas en distintas posiciones.

8. La forma referencial está constituida por un solo nivel del signo generador –los puntos 1 4, 2 5 ó 3 6–; variando de nivel uno o los dos puntos de la forma referencial se obtienen las diversas formas del bloque.

9. La forma referencial la constituyen los puntos 1 3; las diversas formas del bloque se obtienen cambiando horizontalmente uno o ambos puntos.

10. La referencia es un único punto en sus seis ubicaciones posibles.

Las distintas actividades a desarrollar con este material de prelectura, han de tener como objetivo que el adulto reconozca cuántos puntos aparecen y en qué posiciones se presentan, todo ello tomando como referencia el signo generador.

Método de alfabetización Analizaremos tanto la fundamentación teórica en la que se basa el método, como la estructura del mismo:

- La lectura Braille supone un proceso de percepción táctil y por tanto debe basarse en un planteamiento asociacionista que supone partir de elementos significativamente simples (letras, sílabas) para avanzar hacia unidades más complejas (palabras, frases).

- Partir de elementos simples en el caso del Braille (letras, sílabas, etc.), supone hacerlo de elementos complejos: conjunto de puntos organizados espacialmente.

- Las vocales aparecen al principio unidas a los primeros fonemas consonánticos: l, s. . .

- Los diferentes sonidos consonánticos aparecen divididos en distintos bloques:

  1. Formado por aquellos en los que no existe una dificultad de relación biunívoca sonido-representación escrita. El orden es el siguiente: l, s, p, m, f, d, m, t, ñ, c (sonido ca, co, cu), h. Al final del bloque se introducen dos vocales acentuadas: á, é.

  2. Formado por aquellos que requieren una memoria táctil, siendo conveniente acometer su aprendizaje conjunto realizando ejercicios de discriminación. El orden es el siguiente: b, v, ll, y (sonido consonántico), r, rr. Al final del bloque se introducen tanto las vocales acentuadas í, ó, ú, como y (sonido vocálico).

  3. Formado por aquellos sonidos que, en determinados casos, son iguales y, sin embargo, poseen representaciones distintas: g, j, c (sonido ce, ci), z. Asimismo, se incluyen las mayúsculas y el punto, junto a la g; y la coma, junto a la c.

  4. Formado por una serie de sonidos que, bien por su dificultad, bien por su uso escaso, es aconsejable acometer su aprendizaje al final del proceso. El orden es el siguiente: x, q, ch, k, w, ü.

  5. Formado por el conjunto de sílabas trabadas más usuales: pl, cl, bl, fl, gl, pr, fr, dr, tr, cr, br, gr. Asimismo, se introduce el signo de número junto con los siguientes signos de puntuación: guión, dos puntos, punto y coma, interrogación, admiración, comillas, paréntesis.

3.5. Materiales e instrumentos

A continuación, se detallan los materiales e instrumentos más utilizados por los alumnos.

Para obtener mayor información, se puede visitar la página del catálogo con todos los materiales que pueden ayudar al alumnado: http://cidat.once.es

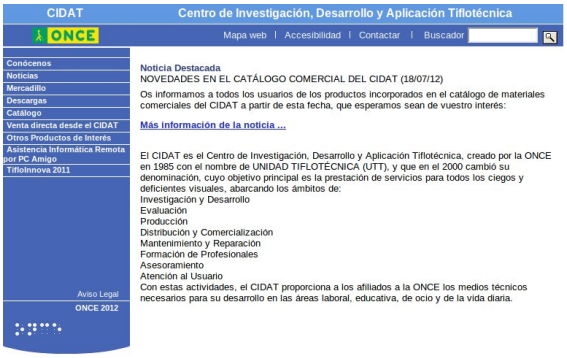


Figura 3.46: Página web del CIDAT [31]

Ábaco: Instrumento para realizar operaciones matemáticas con rapidez. Es una caja rectangular que consta de 24 (o 12) varillas que llevan ensartada 4 bolas móviles en la parte inferior y 1 en el parte superior. Tiene como inconveniente que no se pueden repasar los cálculos intermedios.

Figura 3.47: Ábaco Chino. [126]

Anotadores parlantes (Braille'n speak, Braille hablado, PC Hablado, PAC MATE –Braille o teclado qwerty–, etc.): Sistema portátil de almacenamiento, procesamiento y edición de la información. Se introducen los datos mediante teclado Braille y la salida se produce a partir de síntesis de voz. Se puede almacenar la información en disquetes, imprimir en Braille y tinta, etc. Incluye funciones de cronómetro, reloj, alarma, calendario, directorio telefónico y calculadora.

Figura 3.48: Pac Mate con teclado Braille [126]

Braille'n Print: Sistema electrónico para incorporar en la parte inferior de la máquina Perkins, que permite la conversión de un texto Braille a tinta, a través de una impresora convencional.

Caja de aritmética: Caja de madera para realizar operaciones aritméticas. En una de sus tapas tiene una retícula para insertar las fichas de plástico que llevan escritos los símbolos Braille (existe un modelo que lleva los signos en Braille y en tinta) y, la otra tapa, está subdividida en pequeños compartimentos donde se colocan las fichas. Es muy sencilla de utilizar, realizándose las cuentas de la misma forma que en tinta. Tiene como desventaja la lentitud, ya que hay que colocar las fichas en su lugar correspondiente, cada vez que se utiliza.

Calculadora parlante: Calculadora para realizar operaciones matemáticas, con voz. Existen diferentes modelos, ofreciendo diferente grado de funciones matemáticas (elementales o de nivel superior) Todas tienen auriculares. Tiene además función de reloj con alarma y fecha.

Figura 3.49: Calculadora parlante [126]

Cassette y grabadora de 4 pistas: permite la grabación y reproducción de cintas de cassette, duplicando la capacidad de cintas convencionales,

tiene varias velocidades y se pueden hacer marcas sonoras para ubicar los contenidos. Existen muchos modelos (portátil, de mesa, con radio, etc.)

Cuaderno de falsilla para escritura en tinta: Consta de una plancha de cartón de tamaño folio a la que van adheridas hojas de papel en blanco y una tapa superior de cartón a modo de falsilla con 16 renglones o ventanas que guían la escritura en tinta.

Estuche de dibujo: Contiene diferentes elementos adaptados para posibilitar el dibujo en relieve: goniómetro, compás, escuadra y cartabón, regla con celdas Braille, portaminas, portarruletas, punzón, sello para producir superficies rugosas, ruletas de diferentes dentados, tablero de dibujo y plantilla de dibujo. Cada elemento puede adquirirse también por separado.

Explorador Jaws: Instrumento electrónico de lectura y acceso a la información del ordenador. Es un producto software que permite trabajar en el entorno Microsoft Windows, ofreciendo respuesta de voz o Braille.

Hojas de dibujo positivo: Hojas de plástico especial que, colocadas sobre una plancha de goma o fieltro, y utilizando un punzón o la punta de un bolígrafo, realza el relieve de lo que se quiere representar (dibujos, esquemas...).

Horno Fúser: Aparato para la producción rápida de material en relieve a partir de láminas en tinta fotocopiadas en papel especial (capsular), el cual, mediante calor, provoca el relieve de los trazos marcados. Es sencillo de utilizar, en cuanto a la realización de las plantillas, pero la calidad del relieve no es buena y resulta excesivamente caro.

Impresora Braille: Impresora para Braille, que funciona con la información enviada desde un ordenador personal.

Figura 3.50: Impresora Braille. [126]

Instrumentos adaptados para la vida diaria: existen muchos y muy diversos instrumentos adaptados para la vida diaria del discapacitado visual, tales como dosificadores de medicinas, bastones de movilidad, termómetro parlante de cuerpo o de ambiente, rotuladores perfumados, medidor de glucosa, bastones de movilidad, relojes y despertadores parlantes, balanza de cocina, indicador de nivel de líquidos, detectores de luz de contraste sonoro o vibrante, etiquetas para ropa en Braille, enhebradores automáticos, grabadora de mensajes, brújula parlante, etc.

Figura 3.51: Reloj adaptado en relieve [126]

Libro hablado: Magnetófono de 4 pistas y velocidad regulable, para la lectura de obras literarias, libros de texto, etc.

Línea Braille: Periférico para el ordenador personal que permite ir leyendo en Braille (Braille efímero) la información que aparece en la pantalla del ordenador.

Figura 3.52: Línea Braille. [126]

Lupa TV: Instrumento electrónico de lectura y acceso a la información mediante un sistema de ampliación de imagen por monitor, que posibilita la

ampliación de las imágenes y otros cambios (de contraste, iluminación...) para las personas con resto visual. Existen muchos modelos con distintas posibilidades: en color, blanco y negro, etc.

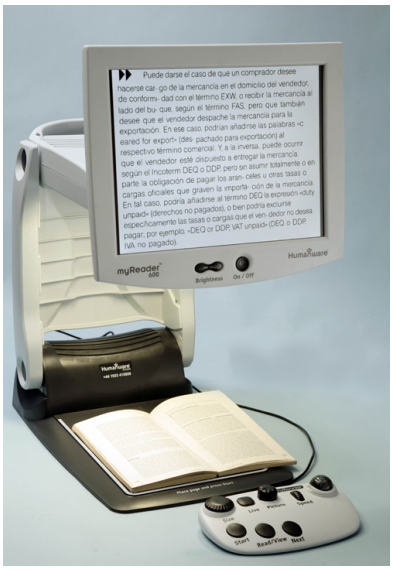


Figura 3.53: Lupa TV. [126]

Magnificador de pantalla de ordenador (Zoomtext): Programa para el acceso a la información del ordenador a través de la ampliación de la información que aparece en la pantalla y síntesis de voz. Es compatible con Jaws.

Figura 3.54: Zoomtext [126]

Máquina de escribir Perkins: Facilita la escritura del Braille de forma rápida y eficaz. Permite la lectura sin necesidad de sacar el papel, por lo que es posible realizar operaciones de cálculo con más facilidad que con la pauta. Véase la sección 3.2.6.

Figura 3.55: Máquina Perkins. [126]

Pauta: Instrumento para la escritura en Braille, de plástico o metal, de tamaño cuartilla o folio y que consta de dos planchas. La plancha de abajo está dividida en celdillas o surcos horizontales y la de arriba está formada por filas de cajetines Braille. Existen diferentes modelos. Entre ambas planchas se coloca el papel y, mediante un punzón, se graba el signo Braille en cada cajetín.

Pizarra Braille: Tablero perforado que permite la composición de los signos Braille de mayor tamaño que los originales, mediante clavitos que se insertan en los agujeros creados al efecto, con la forma del cajetín Braille.

Plantilla o plancha de dibujo positivo: Plancha de goma sobre las que se colocan hojas de papel o plástico, que permiten de forma rápida y eficaz, presionando ligeramente con un bolígrafo, ruleta o punzón, realizar cualquier tipo de dibujo obteniéndose los trazos en relieve «positivo» (es decir, el relieve se obtiene por el mismo lado por el que se dibuja, por lo que no es necesario darle la vuelta al papel). Es muy sencillo de utilizar en el aula, económico y muy eficaz.

Programas OCR (reconocimiento óptico de caracteres): Programa para reconocimiento de textos a partir de imágenes. Detecta mediante un escáner las formas gráficas presentadas (letras, números, etc.) y las almacena en un fichero que puede ser, después, utilizado por el lector a través de una pantalla ampliada, con síntesis de voz o por medio de línea Braille. La información puede ser archivada en otro dispositivo.

Punzón: Especie de lezna para escribir Braille a mano. La punta es de acero redondeado para que no rompa el papel. Hay diversos modelos, en plástico, madera o metal. Existe un punzón borrador, para realizar correcciones en la escritura Braille.

Regleta: Pauta de bolsillo para escribir Braille.

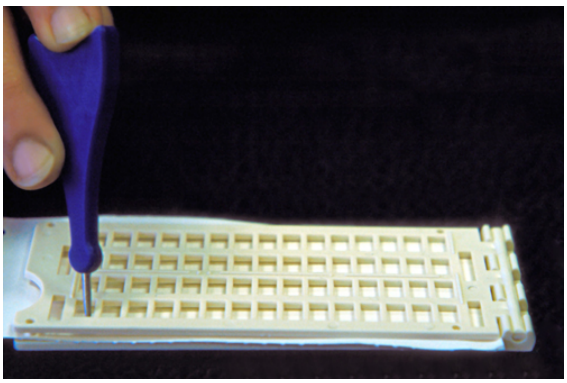


Figura 3.56: Detalle de las manos de una persona utilizando la regleta y el punzón [126]

Regleta de iniciación al sistema Braille: Instrumento utilizado en los primeros momentos del aprendizaje de la lectoescritura Braille. Es útil para el desarrollo de la percepción táctil y el aprendizaje de la lectura Braille. Consta de una serie de espacios o cajetines perforados con los seis puntos del signo generador, de tamaño superior al real, para insertar clavitos que van formando las distintas combinaciones de puntos, simulando el proceso de escritura con pauta y punzón.

Rotuladora en Braille DYMO: Rotuladora en Braille de cinta DYMO, provista de una ruleta con los caracteres del abecedario en tinta y Braille. Admite cinta para rotular de 6, 9 y 12 mm.

Ruedas (o ruletas) dentadas para dibujar: Instrumentos de dibujo con mango y ruedas dentadas con distintas terminaciones, para dibujar en relieve negativo sobre cualquier superficie blanda y papel de dibujo, plástico, cartulina... Son muy útiles y económicas (sirven las típicas ruedas dentadas que emplean los sastres y modistas para la realización de patrones). Existen varios modelos.

Tablero de dibujo negativo: Tablero de madera con superficie de fieltro, para realizar dibujos en relieve, aunque el relieve se obtiene en «negativo» (es decir, es necesario darle la vuelta para apreciarlo) Para facilitar la toma de medidas, la parte superior del marco tiene unos clavitos situados a intervalos regulares de un centímetro.

Thermoform: Aparato que sirve para la reproducción rápida de copias en relieve en papel plastificado de cualquier material (escritura en Braille, gráficos, dibujos, esquemas, etc.) a partir de maquetas en tres dimensiones. Existe también un equipo de instrumentos necesarios para la preparación de cualquier maqueta, después hacer copias en el Thermoform. Existen una completa gama de colecciones de láminas en Thermoform ya elaboradas como mapas, guías y planos, obras representativas de la historia del arte, láminas de ciencias naturales (esqueleto, sistema circulatorio...)

Existe un equipo de instrumentos para la preparación de cualquier tipo de matriz para el Thermoform.

Unidad de disco para Braille Hablado: Unidad de disco externa de formato 3 y medio, para formatear, almacenar y recuperar información en discos, orientada especialmente para ser conectada al Braille hablado.

Unilock: Juego de refuerzo para el aprendizaje del sistema Braille que consta de un tablero de plástico con renglones guía para colocar fichas donde están representadas las letras del alfabeto en tinta y en Braille.

3.6. Conclusiones

Como se ha visto, la amplia adopción del sistema Braille no es casual: después de múltiples propuestas a lo largo de la historia, ésta ha demostrado ser la que mejor responde a las posibilidades y necesidades de los discapacitados visuales. Así, por ejemplo [21]:

1. El sentido del tacto percibe y reconoce el punto como estímulo más adecuado que el trazo continuo.

2. Cada signo se forma mediante una disposición bien definida, con el mínimo de elementos necesario.

3. Las dimensiones de los signos, permiten que se perciba de forma instantánea y global, con la yema del dedo.

4. El número de 6 puntos en cada celda es justo el necesario (las combinaciones con 5 puntos hubieran permitido sólo 31 signos y con 7 puntos hubieran sido demasiados, 127)

Figura 3.57: Las dimensiones de los signos en Braille permiten que se perciba de forma instantánea y global, con la yema del dedo. [126]

Desde que en 1878, en el *Congrès Universel pour l'amélioration du sort des aveugles et des sourds-muets*, en París, se llegó a la conclusión de que el sistema Braille era superior a todas las demás formas de caracteres en relieve y debía adoptarse como escritura universal para ciegos el sistema ha seguido evolucionando, y se han desarrollado y creado nuevos signos para acomodarlo a las necesidades de cada lengua y del momento.

La difusión del sistema Braille como método universal de comunicación escrita para personas ciegas ha sido un factor decisivo en favor de la integración social y educativa de las personas con discapacidad visual. Hoy en día, el acceso a la información de estas personas es una realidad gracias, sobre todo, al sistema Braille. Para ellas, el braille es indispensable a todos los niveles: personal, escolar y social.

El aprendizaje de la lectoescritura mediante el código Braille, si bien tiene muchos puntos en común con su equivalente en tinta, presenta algunas particularidades que se deben tener en cuenta. No obstante, el niño con discapacidad visual puede y debe iniciar el aprendizaje de la lectoescritura Braille al mismo tiempo que sus compañeros videntes se inician en la lectoescritura en tinta. Debe tener cierto desarrollo de la motricidad gruesa y fina, aprendizaje de conceptos básicos, desarrollo senso-perceptivo, desarrollo de la memoria, atención y observación. Se han publicado diversos métodos tanto para el apresto como para la adquisición de la lectoescritura en sí.

Hace algunos años, la escritura del sistema Braille se enseñaba desde los primeros niveles con pauta y punzón. Una vez aprendida la escritura manual, se pasaba a la escritura a máquina. Ahora, se enseña a escribir desde el primer momento con la «máquina Perkins». Para los niños más mayores existen distintos modelos de anotadores parlantes (Braille'n Speak, Braille hablado, etc.) que ofrecen la posibilidad, además, de procesar la información y crear ficheros como un ordenador.

En definitiva, el alumno discapacitado puede conseguir una velocidad de lectura y escritura aceptables, y su código de lectoescritura ser operativo para el trabajo en el aula.

Capítulo 4

La filosofía del software libre

Resumen: En este capítulo se estudia en detalle la filosofía del software libre: su origen, características, tipos de licencia, metodología de desarrollo, etc.

4.1. Introducción

A día de hoy, mucha gente ha oído hablar de «Linux» y sabe que es una alternativa a Microsoft Windows, gratuita y libre de virus malignos. A bastantes les suena también la expresión «software libre», pero todavía no saben muy bien de qué se trata. Sin embargo, el software libre es tan antiguo como las propias computadoras, y sus raíces son todavía más profundas, pues se hunden en una tradición secular entre los hombres de ciencia: la de compartir los logros de cada uno con el resto de sus colegas.

A lo largo de la historia, la ciencia se ha desarrollado como búsqueda del conocimiento y de mejora de nuestras condiciones de vida. Desde la antigua Grecia, los científicos han considerado que el conocimiento era patrimonio de la humanidad. Podían ganar dinero de sus descubrimientos, pero no era ésa su principal motivación, sino satisfacer su curiosidad, contribuir a la sociedad y lograr el reconocimiento de sus semejantes. Para ello se apresuraban a publicar sus teorías y experimentos, poniéndolas a disposición de sus colegas, que las podían emplear para profundizar en el tema y hacer nuevos descubrimientos[1].

La filosofía *hacker* no es sino una actualización de la de los científicos de épocas anteriores. Básicamente consiste en creer que toda la información útil, que sirva para ayudar a comprender cómo funciona el mundo, debe ser

[1] Isaac Newton expresó su gratitud hacia los trabajos previos de Copérnico, Tycho Brahe, Galileo y Kepler en su conocida frase «*If I have seen further it is by standing on ye shoulders of giants*».

libre y accesible para todos, y que se debe usar el conocimiento ya disponible para crear más conocimiento.

4.2. Un poco de historia

4.2.1. Los primeros *hackers*

La cultura hacker tiene su mítico origen en los años 50. El *Tech Model Railroad Club* era (y sigue siendo) un club de estudiantes del prestigioso *Massachusetts Institute of Technology* (MIT) aficionados a las maquetas de trenes. Un grupo de miembros del TMRC formaba el subcomité de *Signals and Power*, que se ocupaba de arreglar, mejorar y redistribuir los innumerables cables, interruptores, relés, etc que hacían funcionar el complicado circuito que tenían, que ocupaba toda una habitación. Dedicaban a esta tarea incontables horas, y con el tiempo fueron desarrollando su propia jerga: por ejemplo llamaban *hack* a algo que se hacía no sólo por su (in)utilidad, sino también por el simple placer que suponía plantearse retos que exigían cierta innovación, estilo y técnica.

Algunos de aquellos *hackers* tomaron una asignatura recién creada: Programación de computadoras. Su profesor era el matemático John McCarthy, que acuñó el término «inteligencia artificial» e inventó el lenguaje de programación LISP. Inevitablemente, los hackers no tardaron en plantearse desafíos y poner en la programación la misma pasión que habían puesto en perfeccionar el circuito de trenes.

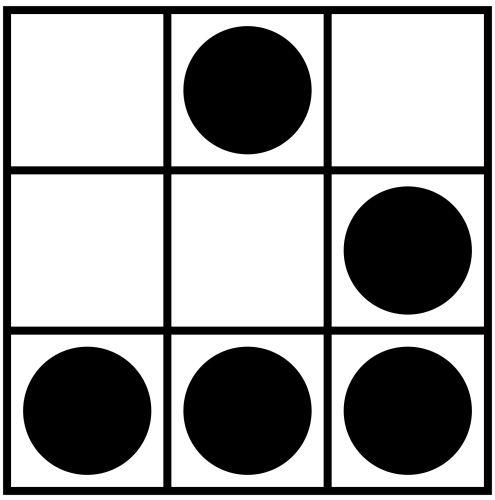


Figura 4.1: El *glider* (planeador) del juego de la vida de Comway se usa como emblema para representar la cultura hacker [140]

Aquel pequeño grupo de hackers dio inadvertidamente cuerpo a una filosofía y ética propias:

- Se debe desconfiar de la autoridad y promover la descentralización. Las burocracias crean reglas para consolidar el poder establecido, y ven el impulso constructivo de los hackers como una amenaza. La mejor manera de promover el libre intercambio de información son los sistemas abiertos, aquellos que no levantan fronteras artificiales entre el hacker y la información que necesita. Esto permite una mayor creatividad en general, y evita tener que reinventar la rueda una y otra vez.
- La valía de un hacker debe juzgarse por sus *hacks*, no por criterios *estúpidos* como calificaciones académicas, edad, raza o posición. Un hacker puede crear arte y belleza con una computadora, pero no sólo en el resultado producido: el propio código de un programa puede ser bello, si está escrito con maestría, es innovador y aprovecha al máximo los recursos disponibles. Además, las computadoras pueden mejorar nuestras vidas, incluso las de quienes no son hackers. Son herramientas poderosas con las que se puede hacer casi cualquier cosa que uno desee.
- Para los hackers, el trabajo y el dinero no son fines en sí mismos: el tiempo de ocio es más importante, y el dinero es básicamente un medio para poder dedicarse a actividades más afines a sus intereses personales o inquietudes intelectuales. Cuando trabajan en un hack, no es el dinero su principal motivación, sino la pasión de hacer algo interesante y creativo, y el reconocimiento del mismo por parte de los demás. Los resultados se ponen a libre disposición del resto de la comunidad, para que sean criticados o mejorados en un esfuerzo colectivo de aprendizaje. Defienden la libertad de expresión en la red, la privacidad, la libertad individual y el uso de la red como herramienta de denuncia y lucha contra situaciones de abuso e injusticia producidas en cualquier lugar del mundo. Entienden que las redes deben ser un elemento de inclusión y entendimiento, y no un instrumento que aumente las brechas sociales provocadas por la exclusión de personas y regiones en función de intereses políticos y económicos.

4.2.2. Origen de la Fundación para el Software Libre

Inicialmente, las computadoras eran herramientas que servían para procesar datos, y los programadores se ayudaban entre sí compartiendo el código que escribían. Sin embargo, poco a poco las empresas decidieron convertir los programas informáticos en un producto comercial y prohibir su libre copia y modificación, lo que llevó al desmembramiento de la comunidad *hacker*.

Richard Matthew Stallman[2] [164], del Laboratorio de Inteligencia Artifi-

[2]También es conocido por iniciales, RMS.

cial del MIT, veía a principios de los años 80 como la comunidad hacker que constituía su vida empezaba a disolverse bajo la presión de esta comercialización de la industria de software. En particular, otros hackers del Laboratorio de IA fundaron la empresa Symbolics, que activamente intentaba reemplazar el software libre del Laboratorio con su propio software privativo. Durante dos años Stallman consiguió duplicar en solitario cada avance que creaba el equipo de programadores de Symbolics, para cuya desesperación Stallman liberaba como software libre, en castigo por haber destruido la comunidad que él amaba.

Figura 4.2: Richard Stallman

Por aquel entonces, sin embargo, él era el último de su generación de hackers en el laboratorio. Finalmente se planteó crear una nueva comunidad, en la que compartir y ayudar a los demás no fuera ilegal. Para ello deci-

dió escribir un nuevo sistema operativo completo, compatible con Unix (un potente sistema operativo), pero libre para todos.

El 27 de septiembre de 1983 anunció en Usenet (grupos de discusión de la red) su proyecto, al que bautizó como GNU (*GNU's Not Unix*[3]), aunque no lo acometería hasta enero de 1984, pues antes de ponerse manos a la obra, RMS decidió dejar su puesto en el MIT, para evitar que la institución académica pudiese reclamar posteriormente algún tipo de derechos sobre su trabajo.

En 1985 publicó el «Manifiesto GNU», que define y explica sus objetivos y motivaciones, y poco tiempo después fundó la organización sin ánimo de lucro *Free Software Foundation* para coordinar el proyecto, al que poco a poco se iba uniendo más gente.

Figura 4.3: Logotipo de la FSF [56]

Escribir un sistema operativo no es tarea sencilla, y hacerlo tan completo como Unix, la convertía en titánica. RMS empezó escribiendo piezas capaces de funcionar sobre los Unices existentes: un editor de texto (Emacs), herramientas para programar como un compilador (gcc) y un depurador (gdb), etc. Un mérito tan importante o más que sus impresionantes logros como programador fue el inventar el concepto de *copyleft* («izquierdos de autor»), que implementó en la Licencia Pública General de GNU (conocida generalmente como «GPL»).

La influencia de Stallman ha sido esencial para establecer el marco de referencia moral, político y legal del movimiento del software libre como alternativa al desarrollo y distribución de software privativo. Ha recibido numerosos premios y reconocimientos por su trabajo, entre ellos el *genius grant* de la MacArthur Foundation en 1990, un doctorado honorario del *Royal Institute of Technology* de Suecia en 1996, y la membresía en la *American Academy of Arts and Sciences* en 2003.

[3]Es decir, «GNU No es Unix». Los hackers son aficionados a estos juegos de palabras autorreferenciales.

Figura 4.4: El logotipo del proyecto GNU [170]

Hacia 1990 el sistema GNU estaba casi completo; el único componente esencial que faltaba era lo que se llama *kernel* o núcleo[4], al que denominaron Hurd[5]. La *Free Software Foundation* decidió (quizás equivocadamente) escribirlo siguiendo un diseño tan innovador como complejo: en vez de escribir un núcleo monolítico al estilo tradicional de Unix, optaron por implementar el núcleo como una colección de procesos servidores (o «manada de ñus») que se ejecutarían sobre un micronúcleo y se ocuparían de las tareas del núcleo Unix [183].

Como micronúcleo, tras probar TRIX (desarrollado en el MIT), decidieron usar Mach (desarrollado en la Carnegie Mellon University), que ahora es usado también por OS X, el sistema operativo de Apple. El inicio del desarrollo se demoró un tiempo mientras esperaban que Mach se publicase como software libre, tal y como se había prometido.

La implementación de este diseño resultó ser mucho más difícil de lo que se esperaba, y con un equipo de desarrollo muy pequeño, los avances fueron muy lentos. Se intentó superar las limitaciones del viejo GNU Mach por un micronúcleo más moderno, llamado L4, pero sin llegar a buen puerto. Posteriormente se intentaría con Coyotos y Viengoos [58].

[4]El núcleo o *kernel* es el responsable de la distribución de recursos, interactuar a bajo nivel con el hardware, seguridad, acceso al sistema de ficheros, protocolos de red, etc.

[5]Aunque originalmente iba a llamarse Alix, el núcleo del sistema GNU fue bautizado por Thomas Bushnell como HURD [59]. «Hurd» significa *Hird of Unix-Replacing Daemons* (Hird de diablillos que reemplazan a Unix), y donde «Hird» significa *Hurd of Interfaces Representing Depth*, es decir: Hurd de interfaces que representan profundidad. ¡Si el nombre del sistema GNU es un acrónimo recursivo, el nombre del núcleo son dos acrónimos mutuamente recursivos!

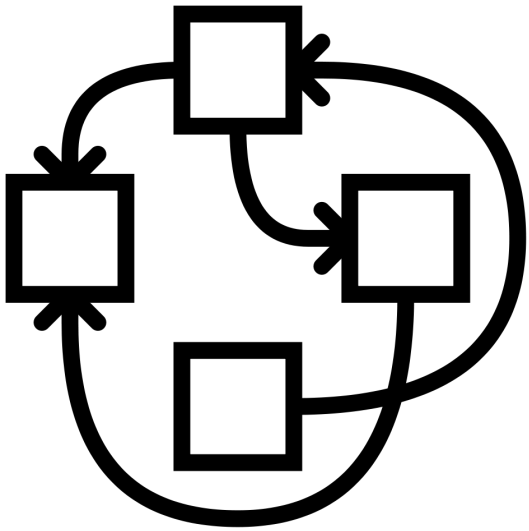


Figura 4.5: Logotipo de Hurd [59]

A día de hoy, el Hurd es funcional, y puede ejecutar muchos de los programas típicos de un entorno Unix. Sin embargo, todavía no ha alcanzado la madurez necesaria para ser usado en entornos de producción y poderse publicar la versión 1.0. Afortunadamente, no ha hecho falta esperar a la publicación del Hurd para poder disfrutar de un sistema completamente libre, gracias a la aparición de Linux.

4.2.3. Linux, *just for fun*

Unix es una familia de potentes sistemas operativos desarrollada a partir de 1969 en los Bell Labs de la *American Telephone and Telegraph company* (AT&T) por Kenneth Thompson y Dennis MacAlistair Ritchie (quien de paso creó el lenguaje de programación C) en un equipo dirigido por Doug McIlroy. Tras varios años de uso interno, AT&T empezó en 1974 a conceder licencias gratuitas o por un pago simbólico a las instituciones académicas, con lo que Unix se convirtió en la base de muchas clases y proyectos de investigación. Poco a poco se iría convirtiendo en un producto comercial, y finalmente se prohibió el uso de su código fuente con fines educativos, con lo que la distribución del libro *A Commentary on the UNIX Operating System* que John Lions había escrito explicando cada fragmento del código fuente pasó a ser clandestina[6].

En la asignatura «Sistemas operativos: diseño e implementación» que se impartía en la Universidad Libre de Amsterdam habían estado usando el

[6]En agosto de 1996, dos años antes de la muerte de Lions, el libro pudo finalmente ser publicado legalmente. Para entonces, unas fotocopias legibles eran un pequeño tesoro.

sistema Unix como ejemplo, pero ahora necesitaban buscar una alternativa. Para la mayoría de los usuarios, los Macintosh de Apple tenían un precio prohibitivo, y tendían a comprar PC basados en procesadores de la familia x86 de Intel, que funcionaban con el endeble MS-DOS (una copia mediocre del CP/M de Gary Kildall).

El profesor de la asignatura, Andrew Stuart Tanembaum, decidió ante esta situación escribir desde cero un sistema operativo tipo Unix para los ordenadores domésticos con procesadores x86, al que llamó Minix. Lo acompañó de un libro de texto explicativo que llevaba el mismo título que la asignatura, que hoy es un clásico en su campo. A pesar de que su código fuente estaba disponible, Minix no era libre, pues se seguía precisando una licencia y no se podía copiar. Tampoco era un sistema operativo excepcional: su único propósito era ser didáctico, por lo que el sistema era deliberadamente sencillo y con pocas funcionalidades; la claridad tenía más importancia que la potencia y la eficiencia.

Figura 4.6: La mascota de Minix [171]

El libro de Tanembaum fue devorado por miles de estudiantes de todo el mundo, que querían aprender como se escribía y se hacía funcionar un sistema operativo, ahora que todos los productores de software guardaban su código fuente en secreto. Entre estos estudiantes se encontraba un finlandés llamado Linus Benedict Torvalds.

Como hacker que era, Linus quería sacar el máximo partido posible de su 386, y a falta de otra alternativa (la FSF acababa de empezar a trabajar en el Hurd, que se preveía tardaría un tiempo en salir) decidió aplicar lo que había aprendido con el libro y escribir un nuevo núcleo que superase las limitaciones de Minix. Lo hizo por mera diversión, y aprovechando las herramientas del proyecto GNU.

Sin embargo, no fue ésta la verdadera genialidad de Linus, sino lo que hizo con lo que en principio no pasaba de ser un entretenimiento privado: lo

puso en la red a disposición de todo el que quisiera jugar con él, y solicitó la ayuda de quien quisiera colaborar.

El 25 de agosto de 1991 envió un mensaje al grupo de discusión de Usenet comp.os.minix explicando su proyecto, y señalando que ya había hecho funcionar sobre él algunas de las utilidades del proyecto GNU: el intérprete de órdenes bash y el compilador gcc. Aprovechaba para pedir comentarios sobre lo que la gente odiaba o le gustaba de Minix, ya que lo estaba tomando como modelo. Aclaraba que lo hacía por *hobby*, y que aquello no iba a ser un sistema grande y profesional como GNU.

A mediados de septiembre publicó, sin hacer mucho ruido, la versión 0.01. Su intención era que se llamara «Freax» (*free* + *freak* + *X*), pero Ari Lemmke, un amigo suyo que le ofreció espacio en su servidor, decidió publicarlo como Linux. El 5 de octubre Linus anunció la versión 0.02 con un histórico mensaje[176] en comp.os.minix, en el que animaba a la gente a descargarlo, probarlo y modificarlo para satisfacer sus necesidades. Para entonces ya era capaz de ejecutar más herramientas de GNU, como make y sed. Durante el resto del año salieron las versiones 0.03, 0.10 y 0.11, todas ellas bajo una licencia que prohibía su uso comercial. La versión 0.12, publicada en enero de 1992, fue la primera bajo la GNU GPL, y también la primera suficientemente estable (tras esta versión se saltó a la 0.95).

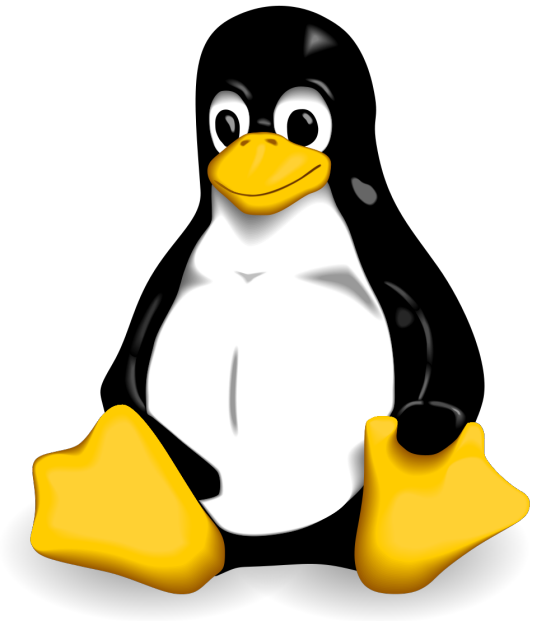


Figura 4.7: Tux, la mascota de Linux [49]

Lo revolucionario de Linux no está en su diseño (que no es especialmente innovador) ni en su filosofía (que la *Free Software Foundation* llevaba años

predicando), sino en su metodología. Efectivamente, hasta entonces el software se escribía en grupos cerrados y de carácter vertical, mientras que Linus inauguró un nuevo modelo, distribuido y muy abierto, en el que cualquiera podía participar. A estos métodos tan diferentes se les ha denominado modelo catedral y modelo bazar, respectivamente, y los estudiaremos con más detalle más adelante.

Estos hackers, como el propio Linus, empezaron a trabajar en Linux simplemente por diversión y para aprender. No tenían grandes ambiciones en él, sino que más bien lo consideraban como un juguete hasta que, al cabo de unos pocos años, se lanzara el sistema GNU o una versión libre de BSD (otra familia de sistemas de la que hablaremos después). Sin embargo, ese juguete que era entonces Linux es usado hoy por millones de personas y prácticamente cualquier gran empresa.

4.2.4. GNU/Linux: La unión hace la fuerza

Linux empezó a aparecer en servidores FTP de Finlandia y otras partes del mundo. Los mensajes sobre Linux en el grupo de discusión comp.os.minix eran cada vez más numerosos, y finalmente Tanenbaum, el profesor que había escrito el Minix, intervino, haciendo notar que el diseño monolítico de Linux era obsoleto y que el futuro estaba en los micronúcleos, y criticando que no fuera portable a otros procesadores. A esto siguió una tremenda y encendida discusión[7], tras la que Linux pasó a tener un grupo de discusión propio.

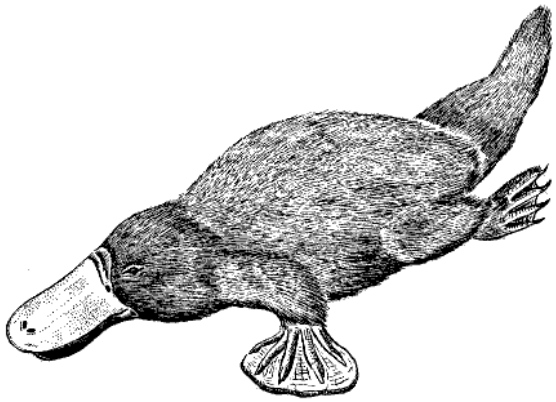


Figura 4.8: En los inicios de Linux, antes de la adopción de Tux como mascota oficial, hubo otras, como este ornitorrinco [2]

[7]Comparando la evolución de Hurd y de Linux, vemos que si bien la teoría nos dice que un micronúcleo es mucho más potente y flexible, también es mucho más difícil de implementar en la práctica. Hoy en día Linux ya no está atado al x86, sino que ha sido adaptado a muchas arquitecturas, y tiene un diseño muy modular, por lo que podría decirse que ocupa un espacio intermedio entre los micronúcleos y los núcleos monolíticos.

Los usuarios de Minix fueron pasando a utilizar Linux, porque tenía más funcionalidades y porque cuando arreglaban fallos o escribían nuevas funcionalidades que necesitaban, podían enviarle a Linus las modificaciones (llamadas parches) para que las incluyera en la siguiente versión (Tanenbaum no lo hacía para que no se resintiera la sencillez y portabilidad de Minix). Además no tenían que molestarse en escribir las otras partes del sistema, sino que simplemente adaptaban mutuamente el software GNU ya existente y el nuevo núcleo Linux, hasta que finalmente se obtuvo un sistema operativo libre completo y funcional: el sistema GNU/Linux (al que con frecuencia, y de manera incorrecta, se llama simplemente Linux).

Desgraciadamente, instalar en una computadora este nuevo sistema era un tanto complicado. No bastaba descargar y compilar[8] Linux, sino que había que realizar todo un proceso partiendo de las herramientas de GNU. También había que buscar el software libre de terceras partes que se quisiese usar, como el entorno gráfico X.

Fue Owen LeBlanc, del *Manchester Computing Centre*, quien, después de tener una réplica del código fuente de Linux en sus servidores, empezó en febrero de 1992 a publicar sus propios disquetes con binarios del núcleo y utilidades extra, con lo que se facilitaba extraordinariamente la instalación del sistema. Nacía así la primera *distribución* de GNU/Linux, llamada MCC Interim Linux. Poco después aparecieron otras distribuciones, como SLS (*Softlanding Linux System*) de Peter MacDonald, TAMU (*Texas A&M University*) e Yggdrasil.

El papel de una distribución es:

- instalación inicial del sistema
- empaquetar software (tomar programas de diferentes fuentes, compilarlos y configurarlos)
- estar atento a la publicación de nuevas versiones de los programas
- integración de los distintos paquetes de software
- facilitar la gestión de paquetes
- distribución fiable del software
- recepción de informes de fallos por parte de los usuarios, comprobación y reenvío a la fuente original

El resultado final es un sistema integrado, estable, seguro y fácil de instalar y actualizar. Hoy día hay más de 300 distribuciones activas, algunas de

[8] Procesar el código fuente escrito por el programador para generar un programa ejecutable por la computadora.

ellas empresas de notable envergadura: SuSE (hoy Novell), Red Hat, Canonical, Mandriva, etc.

A primera vista, a un observador externo le podría parecer que el mundo del software libre está disgregado y dividido. Sin embargo, la descentralización no implica dispersión: esta variedad resulta muy beneficiosa para los usuarios. La fuerte competencia obliga a las distribuciones a avanzar y mejorar continuamente, para intentar ser mejor y más atractiva que las demás. Al mismo tiempo, y al tratarse de software libre, cada distribución puede mirar las nuevas funcionalidades y mejoras que han desarrollado sus competidores, e incorporarlas a su versión, con lo que al final acaban siendo bastante similares, y son los usuarios los principales beneficiados.

En esta carrera por ser la distribución más moderna y atractiva, muchas distribuciones han optado por incluir también algo de software privativo para diferenciarse de las demás. En cualquier caso, tampoco pueden permitirse alejarse mucho del terreno común, pues a los usuarios no les gusta tener que invertir mucho esfuerzo en migrar y aprender un nuevo sistema, y los programadores no están dispuestos a perder tiempo y esfuerzo en solventar pequeñas incompatibilidades innecesarias. En general, un programa libre funcionará indistintamente sobre cualquier distribución de GNU/Linux (así como sobre *BSD).

Una distribución que merece una mención especial es Debian, por varios motivos:

- no es una empresa, sino que está constituida por un millar de voluntarios de todo el mundo, organizados en torno a un contrato social y una constitución.

- todo el software que incluye es totalmente libre (no obstante se pueden instalar paquetes no oficiales con software privativo).

- es la más completa: Debian Wheezy, de inminente aparición, contiene casi 36000 paquetes de software, sometidos a una estricta política de calidad.

- no se limita al núcleo Linux, sino que también desarrolla distribuciones basadas en Hurd y el núcleo de FreeBSD.

- no sólo funciona sobre Intel IA-32 (i386) e Intel EM64T/x86-64 (amd64), sino también sobre ARM EABI, PowerPC y otros procesadores más esotéricos (MIPS, S/390, SPARC...).

- es la base de unas 150 distribuciones que derivan de ella, tales como Ubuntu, Linux Mint, Damn Small Linux, Xandros, MEPIS, etc.

Figura 4.9: Logotipo de Debian [167]

Otras distribuciones que, entre otras particularidades, también son desarrolladas por voluntarios, son Gentoo y Arch.

Hay también distribuciones (llamadas *live*) capaces de funcionar desde un CD-ROM o *pendrive*, sin necesidad de instalar nada en el disco duro. Resultan útiles tanto para mostrar GNU/Linux a alguien que no lo conozca, así como para poder utilizar nuestras herramientas favoritas en una computadora que tenga instalado otro sistema.

Algunas de estos CD en vivo, como Knoppix o Ubuntu, tienen carácter general, mientras que otras están orientadas a un uso particular. Por ejemplo, Musix o dyne:bolic están pensadas para satisfacer las necesidades de activistas y artistas, y son una herramienta práctica para la producción de multimedia.

El usuario aguerrido que desee instalar GNU/Linux manualmente, desde cero y sin la ayuda de una distribución, puede visitar las páginas web del proyecto *Linux from scratch*, donde se explica el proceso paso a paso, a lo largo del cual tendrá ocasión de aprender mucho sobre el funcionamiento interno del sistema.

4.2.5. La tortuosa historia de BSD

El software libre es un concepto que no se limita al sistema GNU/Linux. El ejemplo más conocido son los BSD, una familia de sistemas muy similares a GNU/Linux en cuanto a su funcionamiento y calidad.

Como se ha explicado, AT&T no consideraba inicialmente Unix como un producto comercial, y lo compartía con terceras partes, como la Universidad de California en Berkeley, donde Ken Thompson pasó un año sabático. Allí, estudiantes como William N. Joy y Chuck Haley, empezaron a escribir software para él, como un intérprete de Pascal y el editor de texto vi, y a distribuir estos programas en cintas, bajo el nombre de *Berkeley Software Distribution* (BSD)[9].

Figura 4.10: Beastie, la mascota de BSD [111]

En 1979, la *Defense Advanced Research Projects Agency* (DARPA) decidió usar Unix como su sistema operativo estándar. Al enterarse, el catedrático de Berkeley Robert Fabry escribió una propuesta para que la Universidad desarrollara una versión mejorada de BSD que cubriera sus necesidades. Tras conseguir un contrato de 18 meses, se creó para este fin el *Computer Systems Research Group* (CSRG), con Bill Joy como líder del proyecto. Después de negociar con AT&T unos términos aceptables para todos, en octubre de 1980 publicaron 4BSD. Se vendieron unas 150 cintas, pero la licencia no era por máquina, sino por institución, con lo que el número de sistemas instalados

[9]BSD y 2BSD se basaron en Unix versión 6 sobre máquinas PDP-11, y 3BSD en Unix 32/V sobre máquinas VAX. El núcleo de 3BSD incorporaba una funcionalidad de memoria virtual que necesitaban y había desarrollado Ozalp Babaoglu, otro estudiante.

era varias veces mayor.

Las siguientes versiones se numerarían 4.1, 4.2, etc, pues AT&T objetó que 5BSD podría dar lugar a confusión con su propio System V. En junio de 1981 se publicó 4.1BSD, que entre otras mejoras, incluía las que Bill Joy había hecho sobre el núcleo para aumentar su rendimiento. DARPA estaba satisfecha, y les concedió un nuevo contrato por dos años más y casi el quintuple de fondos. Joy acabó yéndose a la empresa Sun Microsystems, pero el desarrollo continuó, y en agosto de 1983 se publicó la 4.2, que añadía un sistema de ficheros más rápido y soporte para los protocolos de red TCP/IP, lo que supuso una importante ventaja sobre el System V de AT&T. A partir de entonces, AT&T incorporaría éstas y otras mejoras desarrolladas por BSD a su System V.

A la versión 4.3 (junio 1986) le siguió 4.3BSD-Tahoe (junio 1988), en la que el núcleo había sido dividido en partes dependientes del procesador, y partes portables. Esto facilitaría extraordinariamente la posterior adaptación de BSD a otras arquitecturas. Hasta ese momento, todos los receptores de BSD tenían que adquirir previamente una licencia de código fuente de AT&T, pues Berkeley nunca publicaba su sistema solamente de forma binaria, sino siempre acompañado de su código fuente. A raíz del aumento del coste de estas licencias, Berkeley empezó a recibir peticiones para que publicara las utilidades y el código de red de TCP/IP que había desarrollado en una cinta aparte, que no requiriese dicha licencia.

Así, en junio de 1989 se publicó la *Networking Release 1*, el primer código libremente redistribuible de Berkeley. Berkeley cobraba 1000 dólares por cinta, pero la licencia era muy liberal, pues los usuarios podían modificar y redistribuir el código, incluso de forma binaria y bajo otra licencia. Las únicas exigencias eran que se mantuvieran las notas de copyright en los ficheros de código y que se añadiera una nota de reconocimiento a la Universidad y sus contribuidores en la documentación. A pesar de que el código no tardó en estar disponible de forma gratuita en servidores FTP, cientos de instituciones compraron copias, ayudando así a financiar futuros desarrollos.

Constatado este éxito, Keith Bostic propuso al CSRG reimplementar todas las utilidades, bibliotecas y el núcleo para poder publicar una versión de BSD libremente distribuible. En eventos públicos como Usenix, Bostic empezó a pedir a la gente que reescribiera las utilidades de Unix desde cero, basándose únicamente en su descripción pública. La única recompensa sería que su nombre aparecería en la lista de contribuidores junto a la utilidad que había reescrito. 18 meses después, prácticamente todas las utilidades y bibliotecas importantes habían sido reescritas. Bostic, Michael J. Karels y Marshall Kirk McKusick dedicaron los siguientes meses a examinar uno a uno todos los ficheros de la distribución, eliminando el código procedente del sistema 32/V de AT&T. Finalmente les quedaron 6 ficheros que no era fácil reescribir, y tras pedir autorización a la Universidad, en junio de 1991

publicaban la *Networking Release 2*, bajo los mismos términos que la anterior.

Viendo que las subvenciones al CSRG iban pronto a llegar a su fin, varios de sus miembros crearon una empresa llamada *Berkeley Software Design, Incorporated* (BSDi) para desarrollar una versión del código con soporte comercial. Uno de ellos, William Frederick Jolitz, descontento por algunas de las decisiones comerciales de la empresa, decidió abandonarla. Jolitz llevaba desde 1989 trabajando en portar BSD a la plataforma 386, trabajo que explicaba en una serie de artículos en la prestigiosa *Dr. Dobb's Journal*. En marzo de 1992 había conseguido reescribir los seis ficheros restantes, y publicó en la red 386BSD, la versión para PC que había preparado con su esposa, Lynne Greer Jolitz.

¿Por qué no triunfó este sistema operativo, libre y gratuito, cuando Linux todavía estaba en pañales? Aparte de los problemas legales que pronto rodearon a los BSD, otra importante razón fue que los Jolitz adoptaron desde el principio un modelo de desarrollo vertical clásico. Mientras Linus animaba a la gente a que le enviase parches, ellos no tenían tiempo o interés en atender los fallos y mejoras que aparecían.

Al cabo de un tiempo, dado este lento ritmo de desarrollo, un grupo de usuarios creó el proyecto NetBSD, que además de presentar un modelo de desarrollo más abierto, pretendía también soportar tantas plataformas como fuera posible. Otro grupo, formado por Jordan K. Hubbard, Rod Grimes y Nate Williams, publicaba un conjunto de parches llamado *Unofficial 386BSD Patchkit*, pero ante la falta de apoyo de Jolitz, finalmente decidieron crear su propia rama, que llamaron FreeBSD.

Por su parte, BSDi comercializó y ofreció servicios de soporte para su versión del sistema, llamada BSD/386 (y después como BSD/OS), a un precio muy inferior al de AT&T. Ahí fue cuando las cosas se complicaron: *Unix System Laboratories* (la filial que tras USG había creado AT&T para desarrollar y comercializar Unix) interpuso una demanda contra BSDi y la Universidad de California por violación de copyright y divulgación de secretos comerciales. La Universidad de California contraatacó con otra demanda, argumentando que a su vez USL estaba usando el código de BSD sin respetar la licencia (la nota de reconocimiento a la universidad en la documentación y publicidad).

Poco después Novell compró USL, y su directivo, Raymond J. Noorda, prefirió negociar a seguir un proceso judicial de resultados impredecibles. En enero de 1994 llegaron a un acuerdo confidencial: la universidad aceptó retirar tres de los 18000 ficheros que componían la *Networking Release 2*, hacer algunos cambios menores y añadir notas de copyright de USL a otros 70 ficheros. El resultado se publicó en junio de 1994 como *4.4BSD-Lite*, y USL se comprometió a no demandar a nadie que lo usara como base. Los detalles del acuerdo han permanecido en secreto durante más de diez años, hasta que en noviembre de 2004 el miembro de Groklaw «dburns» consiguió que se

(a) Logotipo de NetBSD [172]

(b) Logotipo de FreeBSD [169]

Figura 4.11: NetBSD y FreeBSD fueron las primeras distribuciones hechas por la comunidad.

hiciera público [68], con lo que se cerró una de las páginas más importantes de la historia de los Unices libres.

BSDi, NetBSD y FreeBSD (referidos en conjunto como *BSD) tuvieron pues que desandar lo que habían hecho durante esos tres años y volver a empezar a partir de esta versión. No obstante, eso les dio la oportunidad de incorporar las mejoras que habían hecho los demás grupos. En junio de 1995 se publicó *4.4BSD-Lite, Release 2*, y el CSRG se disolvió, dejando el desarrollo en manos de los demás proyectos.

Los diferentes *BSD eran y son unos sistemas maduros, estables y muy eficientes, pero para cuando se aclaró su situación legal, GNU/Linux ya era el sistema libre más popular. No obstante, siguen en plena vigencia y creciendo en número de usuarios y funcionalidades.

Por otra parte, la filosofía BSD, si bien apuesta firmemente por el software libre, no se opone al software privativo como hace la FSF. Esto se refleja en su licencia (que muchos encuentran demasiado permisiva) y en que sus desarrolladores y usuarios no hacen tanto *ruido* como los de GNU/Linux.

NetBSD sigue manteniendo como prioridad que su distribución funcione sobre el mayor número de plataformas posible, y hoy es capaz de comportarse exactamente igual sobre una vertiginosa lista de arquitecturas. De ahí su lema: *Of course it runs NetBSD*. NetBSD se usa por muchos grupos de investigación, desde IPv6 hasta la Estación Espacial Internacional.

FreeBSD, que en principio prefirió concentrarse en los procesadores x86, ahora funciona también sobre alpha, amd64, sparc y otros. Es un sistema operativo sólido como una roca y tremendamente eficiente, que obtiene el máximo rendimiento de la máquina. Otro de los objetivos iniciales era hacerlo más accesible a usuarios menos técnicos, y de hecho es el BSD libre más extendido.

En octubre de 1995, surgió de NetBSD otro grupo llamado OpenBSD, liderado por Theo de Raadt. OpenBSD decidió enfocar su trabajo a la se-

(a) Logotipo de OpenBSD [173]

(b) Logotipo de DragonFlyBSD [168]

Figura 4.12: Siguen apareciendo nuevas distribuciones basadas en BSD.

guridad, pero incorporando también la idea de facilidad de FreeBSD. Al ser Theo canadiense podía incluir en el sistema el software criptográfico que las leyes estadounidenses impedían exportar. OpenBSD puede vanagloriarse de ser el sistema más seguro del mundo, por lo que es muy empleado para montar cortafuegos.

Los últimos miembros importantes de la familia son DragonFly BSD y el OS X de Apple, que a su núcleo BSD (Darwin) añade la evolución de las API OpenStep de NeXT (Cocoa) y una atractiva interfaz gráfica (Aqua). Contando a OS X, hay más sistemas de escritorio funcionando con *BSD que con GNU/Linux.

4.3. Las definiciones de software libre

4.3.1. Definición de software libre de la FSF

Como hemos visto, al principio los hackers se intercambiaban sus programas y el código circulaba libremente. No existía una noción de software libre, pues todo el software lo era. El concepto de software libre no empezó a tomar forma hasta que las empresas comenzaron a restringir el acceso al código fuente, a prohibir la copia de los programas y a cobrar por licencias de uso.

Stallman fue el primero en presentar un análisis de la situación, dar cuerpo a una filosofía y formular una definición de software libre. La *Free Software Foundation* entiende por software libre aquél que concede cuatro libertades a sus usuarios[10]:

Libertad 0 La libertad de usar el programa, con cualquier propósito.

[10]Por extraño que pueda parecer, las computadoras no empiezan a contar por uno, sino por cero. Muchos hackers hacen lo mismo.

Libertad 1 La libertad de estudiar cómo funciona el programa, y adaptarlo a sus necesidades.

Libertad 2 La libertad de distribuir copias.

Libertad 3 La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie.

La libertad 0 garantiza que podamos usar el programa donde queramos y para lo que queramos. Esta libertad no es obvia, pues el software privativo suele poner limitaciones a donde podemos usarlo (normalmente una única computadora y tras el pago de una licencia) y cómo podemos usarlo. Hay quien ha impuesto limitaciones de uso a sus programas, no permitiendo que sean usados comercialmente, o en determinados países, o por los que ejercen ciertas profesiones u otras singularidades. Esta práctica presenta serios inconvenientes a la hora de combinar dos o más programas, y estos programas no se consideran libres, pues el software libre ha de serlo para todos.

La libertad 1 nos permite modificar el programa o, si no tenemos conocimientos técnicos para hacerlo, pagar a alguien para que nos lo haga. No tenemos que negociar con el productor original del programa, sino que podemos contratar a quien más confianza y mejor precio y servicio nos dé. Mientras la posibilidad de adaptar un programa a sus necesidades es importante para las empresas, los usuarios quizás estén más interesados en saber que puede ser traducido a su idioma, por minoritario que sea.

Con la libertad 2, el autor nos da su permiso para hacer copias del programa y dárselas a quien queramos. Y no sólo podemos regalarlo o intercambiarlo por otro programa, sino incluso venderlo, si encontramos a alguien que nos pague por él. En el caso del software libre esto no es ilegal, pues el autor no sólo lo autoriza, sino que además anima a ello. Al difundir el programa estaremos ayudando no sólo a otras personas, sino también al autor: su programa llegará a más gente sin ningún esfuerzo por su parte, y al haber más gente usándolo, recibirá más sugerencias para mejorarlo, y surgirá más gente dispuesta a ayudarle a desarrollarlo.

La última libertad nos permite redistribuir el programa con las modificaciones que hayamos hecho. Si lo hemos traducido a nuestro idioma, posiblemente queramos que nuestros paisanos puedan beneficiarse también de nuestro trabajo. Generalmente lo más recomendable es enviar nuestras mejoras a los autores originales, para que las incorpore al programa y así no tengamos que volver a hacer nuestras modificaciones en futuras versiones. Pero quizás queramos usar parte del código de ese programa en un programa propio, o hacerle cambios importantes y crear un programa nuevo.

Hay veces que un grupo de usuarios o desarrolladores no está de acuerdo con el rumbo que está tomando un programa y decide emprender un *fork* o bifurcación: partir del código ya existente pero darle una orientación di-

ferente. Generalmente esto no es negativo, sino que proporciona una mayor diversidad y cubre necesidades que un sólo programa no podría cubrir.

Estas cuatro libertades son las que definen el modelo del software libre, y todas sus demás características son consecuencia de ellas. Para que un programa sea considerado software libre, debe facilitar todas y cada una de estas libertades. Por ejemplo, hay software privativo que nos permite ver su código fuente y así comprobar que no hace nada malicioso, pero sin concedernos las demás libertades mencionadas. El *freeware* o software gratuito normalmente tampoco es libre, pues aunque nos permita usarlo y distribuirlo, no nos permite modificarlo. El *shareware* no es más que una forma de distribución de software privativo: normalmente se nos permite copiarlo, pero no usarlo más allá de un periodo de evaluación.

Por otra parte, el software libre no tiene porqué ser gratuito. La empresa productora o distribuidora puede cobrar la cantidad que estime oportuna por proporcionar una copia. Sin embargo, la propia naturaleza del modelo tiende a la gratuidad, pues una vez que un usuario tiene el programa, goza de todas las libertades del software libre, y tiene perfecto derecho a revenderlo más barato, copiárselo a sus amigos o publicarlo en Internet para cualquiera que lo necesite.

4.3.2. Directrices de software libre de Debian

Debian fue iniciada por Ian A. Murdock en agosto de 1993 y es, junto con Slackware, la distribución de GNU/Linux más antigua que sigue viva. En abril de 1996 pasó a ser coordinada por Bruce Perens, y después por otros. Debian incluye únicamente software libre, por lo que fue patrocinada por la FSF durante un año (de noviembre de 1994 a noviembre de 1995).

Sin embargo, Debian tenía algunas dificultades para determinar si un determinado programa era libre. Había que interpretar cuidadosamente la definición de la FSF y considerar las restricciones de la licencia del programa para ver si encajaban tanto en el espíritu como en la letra. Finalmente, en junio de 1997 Bruce elaboró una serie de características precisas que debía tener un programa para ser considerado libre y poder formar parte de Debian. Las directrices fueron discutidas y refinadas durante un mes, y en julio de 1997 se publicaron las *Debian Free Software Guidelines*. Las DFSG facilitan la clasificación de un programa como libre o privativo comparando su licencia con estas directrices.

Debian define tres pruebas hipotéticas que se pueden usar para comprobar si una licencia es libre o no en ciertos casos límite:

Test de la isla desierta El software debe poder ser usado, modificado y compartido por una persona aislada en una isla desierta en la que resulte imposible subir archivos a ningún sitio de Internet. Esto elimina la posibilidad de cumplir cláusulas como la exigencia de hacer públicas

las modificaciones al código o enviar parches a un sitio concreto. Más allá de metáforas concretas, lo que indica este test es que el uso y modificación del software debe estar permitido incluso en el caso de que no sea posible su distribución.

Test del disidente El software debe poder ser usado, modificado y compartido con sus compañeros por una persona disidente en un estado totalitario, sin necesidad de revelar su verdadera identidad, las modificaciones realizadas o incluso la posesión del programa a su gobierno. Cualquier exigencia de compartición de código con alguien más que aquellas personas que hayan recibido un binario pondría al disidente en peligro y no debería ser una exigencia de la licencia.

Test de los tentáculos del mal La licencia no debe permitir que ni siquiera los autores originales puedan retirar las libertades ya concedidas para los programas que ya hayan sido publicados. Se establece el supuesto de que la persona que ha realizado el programa fuera contratada por una gran corporación maligna que quisiera perjudicar a los usuarios del software (hacer sus vidas miserables, obligarles dejar de usar el programa, atacarles legalmente, hacer que el programa deje de ser libre, descubrir sus secretos, etc.), o que una corporación sea comprada por otra más grande que pretenda destruir el Software Libre.

4.3.3. Definición de *Open Source* de la OSI

Con el tiempo, el software libre empezó a suponer una alternativa de bajo coste y alta calidad al software privativo. Sin embargo, el discurso filosófico y moral de la FSF no era del agrado de todos. Para algunos, el software libre simplemente suponía un sistema más eficiente de desarrollo y uso del software, y que proporcionaba una serie de interesantes ventajas que estudiaremos un poco más adelante. También estaba el problema de que, en inglés, *Free Software* puede significar tanto software libre como software gratuito.

Con estos argumentos, Eric Steven Raymond (conocido por ser autor del artículo «La catedral y el bazar» y otros textos y programas), Bruce Perens y algunas otras personas idearon un nuevo concepto: el *Open Source* o software de código abierto. Con esta nueva denominación pretendían ganar la atención del mundo empresarial, en el que la palabra «libertad» y el ideario de la FSF no despertaba entusiasmos.

Para este fin fundaron la *Open Source Initiative* (OSI). Tras eliminar las referencias a Debian de las DFSG y realizar algún otro cambio menor, publicaron la definición del software de código abierto. Además registraron una marca de certificación (*OSI Certified*) que un programa puede ostentar si su licencia ha sido aprobada por la OSI como conforme con la definición de software de código abierto.

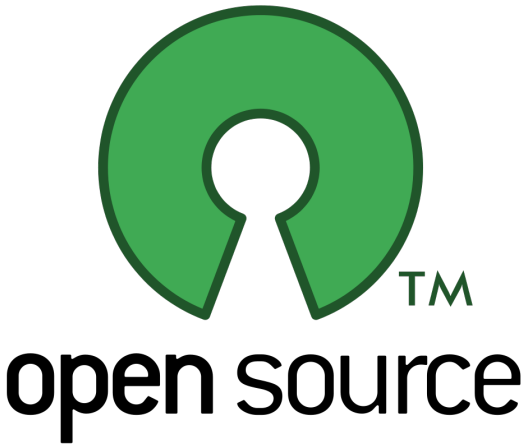


Figura 4.13: Logotipo de la Open Source Initiative [124]

En la práctica, el software libre y el *Open Source* son lo mismo, pero mientras un término hace énfasis en los valores éticos de beneficio de la comunidad, libertad, cooperación, etc, el otro recalca las ventajas prácticas, que son, en definitiva, económicas.

Aunque es difícil determinar la importancia que tuvo la nueva denominación, lo cierto es que ese año el software libre empezó a despertar el interés de las empresas, y podemos asumir que no fue por los motivos éticos, sino por las ventajas que proporcionaba. En febrero de 1999, Bruce Perens anunciaba [131] que abandonaba la dirección de la *Open Source Initiative*, pues consideraba que ya habían conseguido la atención del mundo no-hacker, y que había llegado el momento de volver a hablar de software libre y de la importancia de la libertad.

4.4. Licencias

4.4.1. ¿Propiedad? intelectual: copyright, patentes y marcas

Todos conocemos el concepto «propiedad». Sabemos que, si alguien se cuela subrepticiamente en nuestra casa y se come la tarta que tenemos en la nevera, nos está causando un perjuicio, pues él la disfrutará y nosotros no. También sabemos que podemos ver y tocar nuestras posesiones, pero que, desafortunadamente, la mayoría de las veces no podremos duplicarlas.

En cambio, compartir la receta de nuestra tarta no nos provocaría ninguna pérdida (al contrario, nos proporcionará la gratitud de muchos golosos). Aun después de que dar nuestra receta a todas sus amistades, la seguiremos teniendo, y podremos continuar usándola, mejorándola y compartiéndola durante toda nuestra vida. Observamos además que la podemos reproducir de

manera indefinida, sin más coste que un poco de lápiz y papel.

Cabe preguntarse pues si tiene sentido hablar de «propiedad» en estas condiciones. Evidentemente no se trata de algo inherente a las ideas, sino de un concepto artificial de reciente creación.

Es indiscutible que, aunque determinados trabajos sean inmateriales, sus creadores deben ser recompensados por él, como lo son otros profesionales. En la búsqueda de un mecanismo para estimular a los autores a que aporten nuevas creaciones, la sociedad decidió concederles determinados derechos sobre su obra, durante un tiempo limitado. Sin embargo, hoy se nos quiere hacer creer que no se trata de algo formal, sino que el autor ostenta una propiedad absoluta, equiparable a los objetos físicos, y se utiliza una expresión tan confusa como «propiedad intelectual». En realidad, la mal llamada «propiedad intelectual» agrupa varios tipos de derechos sobre de lo intangible que poco tienen que ver entre sí. Confundir copyright, patentes, y marcas registradas nos puede llevar a suposiciones incorrectas.

El copyright, tal y como hoy lo conocemos, nació en Estados Unidos en el año 1790, con el fin de impulsar el desarrollo de las artes y las ciencias. El copyright protege las obras literarias, artísticas e intelectuales en su forma, pero no protege las ideas contenidas en esas obras. Consiste en conceder en exclusividad, y durante un tiempo limitado, ciertos derechos de control al autor: copia de la obra, hacer obras derivadas, distribución de la obra, ejecución de la obra...

Diferentes países fueron adoptando legislaciones similares, que desde la convención de Berna en 1881, se han ido uniformando paulatinamente. Hoy el copyright se aplica automáticamente y, si no se indica lo contrario, se entiende que el autor se reserva todos los derechos.

Estos derechos, por otra parte, están sometidos a ciertas limitaciones, como los derechos de «uso justo». Este derecho permite usar una obra con propósitos de crítica, comentario, información de novedades, investigación, etc. Pasado el tiempo legislado, la obra pasa a ser «de dominio público», y ya no hay ninguna restricción en como se puede usar, modificar o distribuir la obra.

En Estados Unidos, este tiempo era inicialmente de 14 años, pero se ha ido extendiendo repetidamente. Actualmente es la vida de autor más 70 años, o de 95 años si se trata de una empresa. Si bien el propósito original del copyright era recompensar al autor, ahora no sólo le recompensa de por vida, sino también a sus nietos y hasta a los nietos de sus nietos[11]. Se dice que Disney, que no quiere perder el control sobre Mickey Mouse, tiene algo

[11]Stephen Joyce, el ya septegenario nieto de James Joyce, defiende agresivamente los derechos que ha heredado, hasta el punto de prohibir varios actos del 100 aniversario de la novela *Ulises*, que se celebró en Dublín en junio de 2004. Finalmente, para evitar sus desaforadas exigencias y poder exhibir los manuscritos originales, el parlamento irlandés tuvo que modificar la ley con carácter de urgencia.

que ver con esto.

Otra forma de protección es la «propiedad industrial», entre las que se incluyen las patentes y los signos distintivos. Las patentes tienen por propósito el fomento de la investigación y el desarrollo tecnológico, y protegen una invención particular, que debe cumplir con ciertos requisitos como originalidad, no trivialidad y aplicación industrial.

Una patente reconoce el derecho de explotar en exclusiva la invención patentada, impidiendo a otros su fabricación, venta o utilización sin consentimiento del titular. Mientras el secreto industrial hace que los competidores tengan que *inventar* por segunda vez los aparatos que ya existen (una forma ineficiente de usar los recursos), con las patentes el inventor tiene un incentivo económico para hacer públicos sus descubrimientos. Al cabo del plazo de validez de una patente (20 años), la sociedad dispone de las especificaciones del invento, que puede usar libremente.

Presentan el inconveniente de que si dos personas desarrollan de forma independiente el mismo invento, sólo una podrá patentarlo, y la otra ni siquiera tendrá derecho a usarlo sin el pago que le exija la otra. Tampoco es raro que las pequeñas empresas no dispongan de los recursos necesarios para patentar y hacer valer sus invenciones[12].

Los signos distintivos (marcas y nombres comerciales) son un nombre o signo único que identifica a un producto, servicio o empresa y evita que se pueda confundir con otro. Ocasionalmente esta protección puede dar lugar a abusos, aunque no es frecuente.

Curiosamente, ésta es la razón por la que GNU/Linux y *BSD no son Unix. NetBSD lo explica de esta manera: si algo parece un pato, camina como un pato y hace cuac como un pato, ¿qué es? ¡La respuesta depende de si la palabra «pato» es una marca registrada! Si lo es, a lo más que nos podemos acercar, sin permiso del propietario de la marca, es a que «es como un pato». Unix es una marca registrada de The Open Group, y para poder usarla en un producto, además de pasar unas pruebas, hay que pagar una importante cantidad de dinero. Por eso se les llama «sistemas de tipo Unix».

4.4.2. La licencia BSD: libertad sin control

La propiedad de los objetos físicos la conocemos bien. Cuando compramos una silla, sabemos que podemos usarla para sentarnos, pero también subirnos sobre ella para alcanzar los estantes más altos, regalársela a un amigo, destrozarla si nos cansamos de ella o, mejor, pintarla de otro color.

[12] El caso del teléfono es paradigmático: lo inventó el inmigrante italiano Antonio Meucci, quien no tenía los 250 dólares necesarios para obtener una patente definitiva, así que el 28 de diciembre de 1871 se limitó a pagar la solicitud preliminar, renovable anualmente por 10 dólares. Tres años después ni siquiera disponía los 10 dólares necesarios para renovarla. Así, Alexander Graham Bell pudo conseguir la patente en 1876 y amasar una fortuna. La disputa entre Meucci y Bell llegó a los tribunales, pero Meucci murió durante el proceso.

En definitiva podemos hacer lo que queramos con ella, pues es nuestra y para eso la hemos pagado.

En cambio, cuando vamos a una tienda y pagamos por un CD de música, no pasamos a ser los dueños de esas canciones: únicamente estamos adquiriendo un soporte físico, quizás un pequeño libreto y unos determinados derechos de uso no exclusivo. Dicho de otra manera: a cambio de nuestro dinero, aparte de un poco de plástico y papel, lo que obtenemos es un permiso para usar de cierto modo, y bajo ciertas condiciones, el contenido del CD.

Los programas informáticos, al igual que la música, se protegen mediante el copyright. El propietario del copyright de un programa tiene el control legal de quién y cómo puede usar dicho programa.

Una licencia es un documento por el cual el propietario de la obra concede (frecuentemente tras el abono de cierta cantidad de dinero) ciertos derechos bajo determinadas limitaciones. Por ejemplo, puede disponer que el licenciatario sólo lo puede usar en una computadora, que sólo puede hacer una copia de seguridad, y que no puede redistribuirlo a otras personas.

Naturalmente, también se pueden conceder licencias más permisivas, como son las del software libre. La licencia BSD original [143] bajo la que se publicó el sistema 4.4BSD-Lite era muy sencilla. Tenía 4 puntos, siendo el tercero la cláusula de publicidad con la que la Universidad de California había contraatacado a USL. La licencia concedía a los licenciatarios libertad para usar el código, copiarlo, modificarlo y distribuirlo como quisieran, con lo que podían incluso usarlo para crear binarios bajo una licencia privativa.

Básicamente la única restricción era que cualquier tipo de publicidad que mencionara funcionalidades o uso de ese software debía dar crédito a la Universidad de California y sus colaboradores. Inicialmente esto no era un problema, pero cuando el desarrollo salió del CSRG, los nuevos contribuidores usaron esta misma licencia para las líneas de código que añadían, reemplazando la referencia a la universidad por su propio nombre o el de su institución. En consecuencia, un programa podía requerir que se mostrasen varios reconocimientos. Al combinar varios de estos trabajos (en una distribución, por ejemplo) podía resultar que hicieran falta páginas enteras de reconocimientos. Esto no es una exageración: en agosto de 2012, NetBSD incluía casi 250 de estas frases.

La FSF observó este problema, y se dirigió a los desarrolladores que usaban la licencia BSD para solicitarles que suprimiesen esta cláusula. FreeBSD lo hizo (sobre su propio código) en 1996. La Universidad de California derogó [72] esta clausula con efectos retroactivos el 22 de julio de 1999. Todavía hay quien usa la licencia original, pero la mayoría de la gente usa la licencia modificada. No obstante, todavía hay trabajos que mantienen esta pequeña pero molesta restricción.

FreeBSD simplificó aún más la licencia, eliminando también el cuarto

punto y reduciéndola a dos [60]. Básicamente consiste en que el código puede ser usado por cualquiera y para cualquier propósito, con dos únicas condiciones: no proclamarse como su autor, y no denunciar al proyecto en caso de fallos. Desde junio de 2003 OpenBSD usa la licencia ISC, escrita por el *Internet Systems Consortium*, que es funcionalmente equivalente, pero aún más breve.

4.4.3. GPL: nace el copyleft

La FSF tuvo que enfrentarse a una situación paradójica. Si publicaban su software como de dominio público o bajo una licencia muy permisiva, como la BSD modificada, corrían el riesgo de que alguna empresa lo tomara, modificara y redistribuyera como software privativo. Ya había pasado con el sistema de ventanas X, que aunque había sido desarrollado en el MIT como software libre, normalmente llegaba a los usuarios bajo una licencia privativa.

El objetivo de la licencia BSD era promover el desarrollo de la industria y el uso de estándares abiertos. Sin embargo, la meta del proyecto GNU no era ser popular y tener muchos usuarios, sino proporcionar libertad a los usuarios. Por tanto idearon unos términos de distribución que impidieran que el software GNU pudiera ser convertido en software privativo. Haciendo un nuevo juego de palabras, bautizaron al método como *copyleft* o izquierdos de autor, un término que había sido acuñado por Don Hopkins en oposición al *copyright* o derechos de autor tradicionales.

El copyleft es un *hack* de las leyes internacionales del copyright. Se basa en ellas, pero las usa de tal modo que, en vez de servir para mantener un software privativo, sirven para mantenerlo libre. La idea central del copyleft es autorizar a cualquiera para usar el programa, copiarlo, modificarlo y distribuir versiones modificadas, pero con la restricción de que no se permite añadir más restricciones a dichas versiones. De esta manera, las libertades concedidas por el autor original se convierten en inalienables, y ese código ya no puede ser mutado a privativo. Hay quien objeta que este tipo de licencias son menos libres, pues tienen una restricción importante (prohibido prohibir), pero para la mayoría esto no es un problema, sino una atractiva funcionalidad.

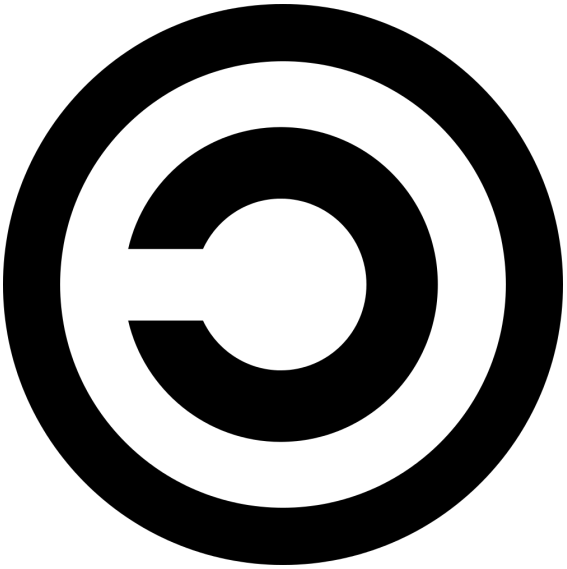


Figura 4.14: Símbolo del copyleft [184]

Al principio, la FSF implementó esta idea en una licencia individual para cada programa. Esto tenía el inconveniente de que había que adaptarla para cada paquete (una para Emacs, otra para NetHack, etc), y hacer todavía más modificaciones para que otros programadores pudieran emplearla en sus propios programas.

Finalmente, en febrero de 1989 publicaron la primera versión de la Licencia Pública General de GNU [57], o GNU GPL. Esta licencia, escrita en un intrincado lenguaje legal, puede emplearse en cualquier programa sin necesidad de modificaciones, sin importar quien lo publique. Sólo hace falta una breve nota en el propio programa, indicando que se publica bajo dicha licencia.

En Junio de 1991 se publicó la versión 2 de la GPL, que muchos consideran una auténtica obra de arte o de ingeniería legal, para la que no ahorran elogios. El sistema GNU oficial se compone de unos pocos cientos de paquetes, pero hay miles de programas libres que funcionan sobre él, y la inmensa mayoría de ellos están publicados bajo esta licencia, al igual que el núcleo Linux.

Redactada por el abogado Eben Moglen, su solidez ha levantado airadas críticas de empresas como Microsoft, que han arremetido contra ella calificándola de «antiamericana», «comunista», «cáncer para la industria», etc. El caso es que el copyleft les impide llevar a cabo una de sus prácticas habituales: la de comprar las empresas competidoras.

Figura 4.15: Logotipo de la licencia GNU GPL 3 [56]

El 29 de junio de 2007, tras una larga discusión pública y cuatro borradores, se publicó la versión 3 de la GPL que resuelve algunas ambigüedades, aumenta la compatibilidad de la GPL con otras licencias, y proporciona cierta protección contra la «gestión digital de derechos» (DRM) y las patentes de software.

4.4.4. Otras licencias

Hay muchas otras licencias de software libre, tanto con copyleft (*GNU Affero General Public License*, *GNU Lesser General Public License*, *European Union Public Licence 1.1*, *IBM Public License*, *Mozilla Public License*, etc) como sin él (*Apache License 2.0*, *MIT License*, *Eiffel Forum License 2*, Zlib, etc). Algunas de ellas son muy sencillas, mientras que otras pueden ser bastante complicadas, pero todas proporcionan las cuatro libertades básicas del software libre. La FSF mantiene en su web una lista de licencias que cumplen la definición de software libre, y la OSI otra de las que cumplen la definición de *Open Source*. Miriam Ruiz [151] analiza las características de las más comunes.

La OSI reconoce en septiembre de 2012 69 licencias, y una decena más en desuso. Esta proliferación de licencias tiene el inconveniente de que sus términos pueden fácilmente ser incompatibles entre sí, y que por tanto un proyecto no pueda reutilizar el código de otro que esté bajo otra licencia.

4.4.5. Elección de una licencia

Las licencias se pueden clasificar en tres grandes grupos:

Regalo Son tan poco restrictivas como sea posible, permiten usar el código en software privativo.

Copyleft fuerte No se puede usar el código en software privativo: cualquier programa que lo use está obligado a usar la misma licencia de distribución.

Copyleft débil Un programa privativo puede usar el código, pero éste, y las mejoras que se le hagan, debe mantener la licencia original.

Para evitar el problema de la proliferación de licencias, Bruce Perens recomienda [132] limitarse a una licencia en cada uno de estos grupos: Apache License 2.0, GNU GPL v3 y GNU LGPL v3 respectivamente. Todas las demás licencias son similares a estas tres, pero no están tan al día con respecto al entorno actual (Internet, dispositivos embebidos, etc) y a la legislación y jurisprudencia de patentes y copyright.

Sólo cabe añadir una cuarta licencia: la Affero GPL3, diseñada específicamente para el software como servicio (SaaS), es decir, aquél que no se distribuye, sino que se usa a través de la red (aplicaciones web, etc).

4.4.6. Más allá del software

El espíritu del software libre se puede llevar a otros tipos de información, pues ésta siempre se puede duplicar a un coste prácticamente nulo. La FSF constató que, además de software libre, también hacía falta documentación libre que explicara como funciona ese software. Para ello creó la *GNU Free Documentation License*, que está pensada para documentación técnica (aunque no necesariamente de software). Además de la GNU FDL, existen otras licencias libres de documentación como la *FreeBSD Documentation License* y la *Apple's Common Documentation License*.

La FDL sirve igualmente para libros de texto o diccionarios. Así, numerosos voluntarios han creado a través de Internet Wikipedia, una completa enciclopedia bajo esta licencia. Se basa en el uso de un tipo de programa llamado *wiki*, que permite a cualquiera editar cualquier artículo de manera sencilla. En caso de que alguien hiciera modificaciones malintencionadas, es posible deshacer los cambios y devolver el artículo a su estado anterior.

A pesar de que hoy el mundo avanza mucho más rápido que en 1790, el periodo de protección del copyright se ha multiplicado increíblemente. Deberíamos plantearnos si hay un equilibrio entre lo que el autor aporta a la sociedad y lo que ésta cede: no parece normal que algunos artistas se hagan multimillonarios por poco más de media hora de música.

Por otra parte, lo más frecuente actualmente es que el copyright, más que recompensar a los autores (ya sean éstos músicos, escritores o programadores), sirva para generar pingües beneficios a las empresas intermediarias, a las que los autores se ven obligados a ceder sus derechos.

Son estas empresas las que han presionado a los gobiernos y logrado que las leyes sean cada vez más extensivas, empezando a afectar gravemente nuestra vida cotidiana. La imprenta fue un vehículo extraordinario para la expansión de la cultura, pero ahora, cuando más fácil y económico es acceder a ella, no hacen sino poner trabas, llegando al extremo de gravar con un canon el préstamo de libros en las bibliotecas públicas. Cada vez que un

avance tecnológico amenaza su imperio, intentan detenerlo. En su momento pretendieron ilegalizar los aparatos domésticos de vídeo, y ahora intentan criminalizar el uso de Internet, la grabación de CD-ROM, etc. Últimamente están desarrollando e intentando imponer unas tecnologías llamadas DRM (*Digital Rights Management*, Gestión de Derechos Digitales), que podrían hacernos perder el control de nuestras computadoras. Básicamente pretenden someter el mundo de la cultura a sus intereses comerciales, polarizarlo de modo que haya unos pocos creadores bajo su control, y el resto seamos consumidores pasivos de su cultura.

Figura 4.16: Imagen creada por Nina Paley para una campaña contra la DRM [158]

Por todo esto, otros colectivos han llevado la filosofía del software libre a diversos tipos de manifestaciones artísticas (literatura, música, fotografía, cine, etc), para lo que han creado licencias específicas. El sitio web Creative Commons [38] es un importante punto de encuentro para artistas con estas inquietudes, que prefieren usar las nuevas tecnologías para dar a conocer su obra al mayor público posible.

Figura 4.17: Logotipo de Creative Commons [38]

Creative Commons ha publicado media docena de licencias, combinando diferentes opciones:

Reconocimiento («by», o *Attribution*) El material creado por un artista puede ser distribuido, copiado y exhibido por terceras personas si se muestra en los créditos.

No Comercial («nc», o *Non commercial*) El material original y los trabajos derivados pueden ser distribuidos, copiados y exhibidos mientras su uso no sea comercial. Incumple la libertad 0 del Software Libre y la directriz 6 de Open Source.

Sin Obra Derivada («nd» o *No Derivate Works*) El material creado por un artista puede ser distribuido, copiado y exhibido pero no se puede utilizar para crear un trabajo derivado del original.

Compartir Igual («sa» o *Share Alike*) El material creado por un artista puede ser modificado y distribuido pero bajo la misma licencia que el material original. Incumple la libertad 3 del Software Libre y la directriz 3 de Open Source.

Al usar una licencia Creative Commons, es importante señalar explícitamente de cuál de ellas se trata. De todas las licencias Creative Commons, solamente dos son libres: CC by 3.0 y CC by-sa 3.0. Las versiones anteriores a la 3.0 incluyen cláusulas problemáticas que, por su redacción, hacen que no puedan considerarse libres según las DFSG.

Figura 4.18: Las licencias de Creative Commons que son libres ostentan este emblema [38]

Otro nuevo campo que está naciendo con fuerza es el del hardware libre, con Arduino como la plataforma más conocida. Ya se ha publicado una declaración de principios y definición de *Open Source Hardware* (OSHW) [7], así como algunas licencias específicas para hardware. Las principales son la CERN Open Hardware License (CERN OHL) de la Organización Europea para la Investigación Nuclear [125] y la TAPR Open Hardware License (TAPR OHL) [177] de la organización de radioaficionados Tucson Amateur Packet Radio.

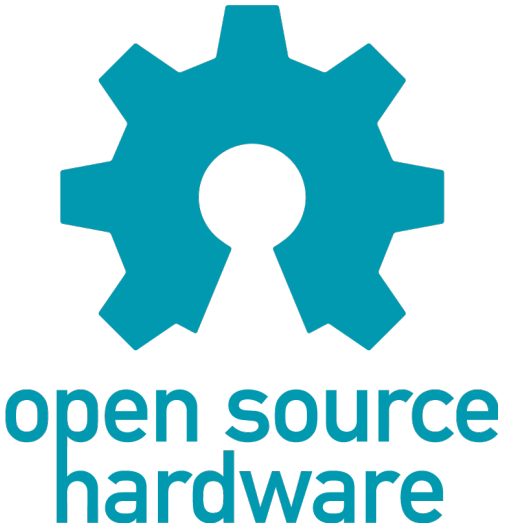


Figura 4.19: Logotipo del hardware abierto [162]

En términos mas generales, ha nacido el concepto de conocimiento libre, con las siguientes características (equivalentes a las cuatro libertades

esenciales del software libre):

0. Puede ser libremente adquirido y libremente usado, con cualquier propósito y sin necesitar permiso de nadie.

1. Puede adaptarse libremente a las necesidades del adquisidor.

2. Puede compartirse libremente con los demás.

3. Es tal que puede mejorarse y sus versiones adaptadas y mejoradas pueden compartirse libremente con los demás, para que así se beneficie la comunidad entera.

El acceso a una fuente modificable del conocimiento es una precondición para los puntos 1 y 3.

4.4.6.1. Mal uso de los términos «libre» y «copyleft»

El proyecto Creative Commons ha tenido bastante éxito en hacer notar que las leyes tradicionales de derechos de autor son brutalmente restrictivas. Sin embargo, no ha tenido tanto éxito en lograr que los autores concedan las mencionadas libertades para crear obras culturales libres, y además ha contribuido a sembrar mucha confusión sobre el significado los términos «libre» y «copyleft» (mal usados a propósito por algunas personas, y debido a la ignorancia por otras).

Mucha gente pone que su web «tiene licencia CC», sin explicitar cuál de ellas. Otros utilizan licencias CC que no permiten el uso comercial ni hacer obras derivadas, por lo que no son mucho mejores que el copyright tradicional. Incluso hay quien utiliza el término copyleft para obras que no sólo no son copyleft, sino que ni siquiera son libres.

Para aclarar esta situación se creó la definición de Obra Cultural Libre (*Free Cultural Works*). Para que una obra cultural pueda considerarse libre, su licencia debe otorgar las siguientes libertades sin cortapisas:

0. La libertad de usar y ejecutar la obra: Se debe permitir que el licenciatario haga todo tipo de uso, privado o público, de la obra. Para los tipos de trabajo en los que sea relevante, esta libertad debe incluir todos los usos derivados («derechos relacionados») como ejecutar o interpretar la obra. No debe haber excepciones que consideren, por ejemplo, cuestiones políticas o religiosas.

1. La libertad de estudiar la obra y aplicar la información: Se debe permitir que el licenciatario examine la obra y utilice en cualquier manera el conocimiento obtenido de la obra. La licencia no puede, por ejemplo, restringir la «ingeniería inversa».

2. La libertad de redistribuir copias: Las copias pueden venderse, intercambiarse o darse gratis, como parte una obra mayor, una colección, o de forma independiente. No debe haber límite a la cantidad de información que se puede copiar. Tampoco debe haber límites sobre quien puede copiar la información o donde puede copiarse.

3. La libertad de distribuir obras derivadas: Con objeto de dar a todo el mundo la posibilidad de mejorar una obra, la licencia no debe limitar la libertad de distribuir una versión modificada (o, para obras físicas, una obra derivada de alguna manera del original), sin importar la intención y propósito de dichas modificaciones. Sin embargo, se pueden aplicar algunas restricciones para proteger estas libertades esenciales o el reconocimiento de los autores.

La libertad 0 implica que una obra libre se debe poder distribuir, comercialmente o no, sin tener que pedir permiso a nadie. Así que los materiales que se distribuyen con condiciones de uso que no permitan su utilización comercial, no son libres (en particular, las licencias CC by-nc, CC by-nc-sa y CC by-nc-nd no son libres).

La libertad 1 implica que materiales distribuidos en formatos no modificables no sean libres si no van acompañados de sus archivos fuente (a partir de los cuales fueron generados).

La libertad 2 implica que los materiales que se pueden usar y copiar en ciertos ámbitos (por ejemplo, el educativo), y sólo en esos ámbitos, no sean libres.

La libertad 3 implica que los materiales que se distribuyen con condiciones de uso que no permitan generar obras derivadas, no sean libres (por ejemplo, las licencias CC by-nd y CC by-nc-nd no son libres).

Figura 4.20: Logotipo de la definición de obra cultural libre [119]

Por último, hay personas que usan el término «copyleft» para indicar que su trabajo se puede copiar, lo cual es sólo una de las cuatro libertades. Como se ha explicado (sección 4.4.3), Copyleft es una clausula de algunas licencias libres, y significa que las condiciones de uso de una obra derivada deben mantener las mismas libertades de la obra original, sin ninguna restricción adicional. De este modo, cualquier obra que derive de una obra copyleft, ha de ser necesariamente libre.

4.5. Praxis: cómo y porqué se desarrolla el software libre

4.5.1. Porqué: valores éticos y motivos prácticos

Cuando RMS inició el proyecto GNU, no pensaba que el software libre iba a contar con millones de usuarios y sería empleado en la mayoría de las empresas. Lo que ha pasado es que, con el tiempo, ha resultado que el software libre no sólo es mejor desde una perspectiva filosófica, sino también práctica, que es la que publicitan los seguidores del *Open Source*.

Desde hace algunos años se nos viene enseñando que «copiar es malo», sin más: quien lo hace es un «pirata», un delincuente de la peor calaña. Pero si prestamos atención a nuestro entorno, o si nos remontamos atrás en el tiempo, comprobaremos que no es cierto. Constantemente redistribuimos chistes, noticias, rumores, etc. En tiempos pasados también los libros se copiaban (entonces era una labor tediosa, pues se tenía que hacer a mano). En cambio, hoy en día la biblioteca de Alejandría sería considerada la sede de una organización criminal, sería objeto de intervención policial y los fantásticos conocimientos que albergaba serían destruidos en nombre de la propiedad *intelectual*. Exactamente eso es lo que paso en 2012 con el sitio web Megaupload.

No cabe duda de que ayudar y colaborar con nuestros amigos y vecinos es bueno para la sociedad, de lo que se deduce que publicar, copiar, mejorar o traducir programas informáticos es lo correcto, mientras que guardárnoslos para nosotros es antiético.

Por ello el software libre no precisa necesariamente de financiación: muchos programadores lo escriben en su tiempo libre, simplemente por el placer de programar y colaborar con otros hackers o bien para satisfacer una necesidad personal, y después lo publican para ayudar a quien le pueda ser útil, pues no pierden nada al hacerlo. Así mismo hay mucha gente que, sin saber programar, colabora haciendo traducciones, dibujando iconos o escribiendo documentación, en agradecimiento por haber recibido como software libre un programa que les resulta particularmente útil. Estas maneras de contribuir están al alcance de cualquiera que disponga de un poco de tiempo. Quien no lo tenga, aún puede ayudar usando software libre difundiéndolo, informando

de fallos, aportando sugerencias o haciendo una aportación económica.

Tampoco todos lo hacen por altruismo, las motivaciones no directamente económicas son muchas: pueden hacerlo por aprender, practicar y adquirir experiencia, darse a conocer[13], etc o simplemente mejorar un programa que también ellos utilizan, por lo que se unen al grupo de desarrollo. Se habla entonces de *colaboración egoísta*.

Es sabido que el principal objetivo de cualquier empresa es ganar dinero, y que algunas de las empresas de informática más grandes del mundo (IBM, Novell, Sun, etc) están invirtiendo enormes sumas de dinero en el software libre. En general, podemos descartar la filantropía, así que, ¿por qué lo hacen? ¿por qué cada vez más empresas usan y desarrollan software libre?

Los mecanismos económicos del software libre todavía están en estudio, y no será aquí donde se explique como ganar dinero con él, pero sí se intentará explicar algunas de sus ventajas y posibles modelos de negocio.

Las prácticas de las empresas de software privativo han conducido a situaciones de monopolio en las que no hay competencia, y en las que por tanto las empresas líderes aplican márgenes abusivos y no necesitan trabajar para mejorar la calidad de su producto ni atender a las peticiones y quejas de sus clientes, pues éstos no tienen ninguna alternativa viable.

De hecho, la mayoría de las pequeñas empresas de software se dedican a escribir software a medida, por encargo, y no intentan comercializar programas, pues es prácticamente imposible que una empresa consiga vender un programa alternativo al dominante (sea éste un procesador de textos, un maquetador, o cualquier otro), aunque sea mejor y más barato, pues generalmente los usuarios prefieren comprar el programa que usa todo el mundo y no el que les ofrece una pequeña empresa local. De intentarlo, esta empresa tendría además que invertir previamente una importante cantidad de dinero en *marketing* para lograr introducirse en los principales canales de distribución. Lo más probable es que acabe arruinada o, si consigue tener cierto éxito, sea absorbida por sus grandes rivales. Esto provoca que el software privativo avance lentamente y sea caro de producir.

En cambio, en el caso del software libre la situación es más equilibrada, y se compite en una cierta igualdad de oportunidades. Para empezar, un nuevo proyecto puede apoyarse en la miríada de bibliotecas y funciones ya existentes, ahorrándose la necesidad de volver a programar lo que otros ya han hecho antes. Además, cada proyecto puede tomar ideas e incluso código de los programas rivales, basarse en otro ya existente, etc. con lo que todos se benefician y desarrollar un programa es más rápido y más económico.

El software libre no tiene porqué ser gratuito, aunque con frecuencia lo sea. En general, las empresas desarrolladoras no obtienen sus ingresos de

[13] En ocasiones, cuando un proyecto destaca por ser especialmente interesante, alguna empresa contrata a su principal desarrollador y le paga para que siga trabajando, ahora a tiempo completo, en su proyecto.

la venta de licencias, sino de servicios como implantación, soporte, mantenimiento, cursos de formación, certificación de profesionales, consultorías, adaptación del programa a casos particulares, e incluso *merchandising*.

El hecho de que cualquiera pueda acceder al código fuente de un programa hace que los programadores sean más cuidadosos[14] con su código y eviten hacer chapuzas de las que podrían avergonzarse después. Esto no significa que todos y cada uno de los programas libres sean de mayor calidad que los privativos, pero sí tienen un mayor potencial. Para una empresa de este tipo esto significa un importante ahorro de costes en su modelo de negocio.

Para las empresas usuarias de software, además del menor o ninguno coste de las licencias, el software libre tiene la ventaja de que pueden elegir quien le proporcionará los servicios de mantenimiento y soporte, pues las opciones ya se limitarán a la empresa productora. Una vez más, esto redunda en un mejor precio y servicio, pues la mayor proximidad de los programadores y la existencia de competencia implica que sus quejas y sugerencias sean atendidas. En consecuencia, los programas libres se adaptan mejor a las necesidades de los usuarios y tienen menos fallos que sus contrapartidas privativas. Por todo ello, el software libre ofrece generalmente a las empresas usuarias unos índices de coste total de propiedad y de retorno de inversión más atractivos que el software privativo.

Algunas empresas ofrecen sus productos bajo dos licencias, una libre y otra privativa[15]. Gracias a la primera pueden ser empleados por usuarios y desarrolladores de software libre, mientras que la segunda exige el pago de una licencia a los desarrolladores de software privativo. Cuando alguien contribuye un parche a un proyecto de este tipo, para que entre en la rama principal se le pide que acepte que se pueda publicar también bajo la licencia privativa.

También muchas empresas de hardware están interesadas en el software libre. Hoy en día, el coste de las licencias de los programas instalados en una computadora suele ser muy superior al coste de la propia máquina. Estas empresas apoyan el software libre porque si se reduce el coste del software, las empresas usuarias podrán permitirse comprarles más y mejores máquinas.

Lo cierto es que durante los últimos 30 años se ha invertido muchísimo dinero en software privativo y poquísimo en software libre, y lo cierto es que, a día de hoy, el software libre es comparable al privativo (muy superior en algunas áreas, todavía a la zaga en otras), de lo que se concluye que, económicamente, el modelo del software privativo es muy ineficiente en comparación

[14] Un estudio de la empresa Reasoning de diciembre de 2003 estudió la calidad del código de varias bases de datos, y concluía que el del proyecto libre MySQL era seis veces superior al de los productos privativos [141].

[15] El ejemplo más conocido es la empresa noruega Troll Tech. De 2000 a 2009, sus bibliotecas Qt estuvieron disponibles bajo dos licencias libres (GPL y QPL), por lo podían ser empleadas por proyectos como KDE, pero también lo estaban bajo una licencia privativa, que muchas empresas de desarrollo de software cerrado adquirían por su altísima calidad.

con el del software libre.

4.5.2. Cómo: la catedral y el bazar

El software libre, además de sus valores éticos, ha supuesto un nuevo sistema de desarrollo de software. Siguiendo la ingeniería del software tradicional, los programas se desarrollaban de manera muy estructurada por grupos cerrados de gente liderados por un analista. Cuando la FSF inició el desarrollo del sistema GNU, si bien invitaba a los voluntarios a entrar en el círculo, siguió prácticamente el mismo modelo, y a cada persona se le asignaba una tarea muy específica. Fue el joven Linus Torvalds quien tuvo la genial intuición de lanzar su mensaje a USENET y dejar que la gente aportase lo que considerase oportuno en vez de seguir un plan preestablecido. Prácticamente todos los proyectos de software libre han adoptado esta misma metodología. Eric S. Raymond estudió este fenómeno, denominando al método centralizado «catedral» y al distribuido «bazar».

Examinemos rápidamente las características del modelo bazar. Lo primero que llaman la atención es que los proyectos suelen estar desarrollados por grupos de hackers distribuidos por todo el mundo, que no se conocen entre sí, sino que simplemente comparten un interés común, de modo que forman una comunidad más o menos difusa. Cualquier persona con los mismos intereses puede colaborar e incorporarse al equipo de desarrollo. Sin duda, el hecho de trabajar en algo que les gusta les lleva a poner más esmero e incluso cariño en lo que hacen.

Para coordinarse, utilizan listas de distribución de correo electrónico. Mediante ellas intercambian sus objetivos, puntos de vista, críticas, etc. En cada proyecto suele haber un *dictador benevolente* (normalmente el fundador del proyecto) que debe coordinar y motivar a los demás desarrolladores, así como buscar consensos. En proyectos grandes no suele haber un único líder, sino un grupo de ellos elegidos por los demás desarrolladores en base a sus méritos (cuanto contribuyen al proyecto). En casos de desacuerdo irreconciliable, el coordinador tiene la última palabra, pero nadie está sometido a ella, sino que puede iniciar su propio proyecto a partir de lo hecho hasta el momento (lo que se denomina *fork* o bifurcación). Cuando un líder no puede o sabe cumplir adecuadamente su papel, debe ceder el testigo a otro desarrollador, para que el proyecto continúe con vigor.

Para poder trabajar varias personas a la vez sobre el código de un programa se han desarrollado sistemas de versiones concurrentes a través de Internet, como Subversion o Git, que al tiempo que permiten que cualquier desarrollador pueda añadir sus aportaciones, permiten retirarla y volver al estado anterior en caso de que contuviera algún error. Es bien sabido que un sistema libre como GNU/Linux o *BSD tiene muchos menos fallos de seguridad y peligro de ser atacado por un virus informático que otro privativo como Microsoft Windows. Esto se debe también al carácter colaborativo del

desarrollo. Por una parte se discuten y analizan las ventajas e inconvenientes de los distintos diseños posibles, y por otra, si un programador comete un fallo, es muy probable que haya otro que lo detecte y solucione.

Además, los programas no son publicados cuando las exigencias del *marketing* indiquen, sino cuando se considera que los programas están preparados. Es frecuente que haya dos ramas en paralelo del programa: una estable, apta para su uso en entornos productivos, y otra de desarrollo, para quienes quieran estar a la última. El ciclo de publicación de las versiones de desarrollo suele ser muy rápido, y es habitual que cada año se publiquen varias versiones del programa. Esto permite que los usuarios adopten rápidamente las nuevas versiones, encuentren los inevitables fallos y sugieran nuevas mejoras que serán atendidas por los desarrolladores y se repita al ciclo. Cuando la versión de desarrollo tiene cierta cantidad de nuevas funcionalidades y es estable, pasa a ser la versión de producción y se crea una nueva rama de desarrollo.

Aún con todo, el desarrollador de Linux Alan Cox ha hecho notar que este modelo tampoco está exento de problemas. En concreto ha descrito lo que ha denominado el «consejo municipal» [37], que se produce cuando en un proyecto hay muchas personas que dan ideas y críticas, pero pocas o ninguna realmente hace algo. Este efecto *opinódromo* puede afectar especialmente a proyectos que empiezan de cero, sin un código base del cual partir, por lo que es importante empezar a publicar código cuanto antes, aunque todavía no sea muy funcional.

Poul-Henning Kamp, de FreeBSD, describió un fenómeno similar que denominó *discutir por el color del cobertizo para bicicletas* [85]. Explica que las cosas grandes y complicadas se llevarán a cabo sin mucha discusión, precisamente por su complejidad, mientras que las cosas sencillas recibirán un montón de atención por parte de gente que querrá hacerse notar y discutirá sus más insignificantes detalles, provocando finalmente una discusión recurrente en la que nunca se alcanzará un consenso.

Otro obstáculo es que con demasiada frecuencia el diseño del programa no está documentado, lo que supone un cierto esfuerzo de familiarización al programador que entra en un proyecto ya avanzado.

A pesar de estos posibles inconvenientes, el modelo bazar ha demostrado ser extraordinariamente eficiente y capaz de producir programas de mayor calidad con menos recursos. Los partidarios del término *Open Source* esgrimen esto como su principal motivación, mientras que los seguidores de la filosofía del software libre explican que no es el objetivo final, sino simplemente una consecuencia de la libertad del software, que es lo realmente importante.

4.5.3. Apto para todos los públicos

Inicialmente, el software libre estaba escrito por y para informáticos. La mayoría de los programas se usaban desde la línea de órdenes y los entornos gráficos eran francamente espartanos. Sin embargo, esto hace ya tiempo que empezó a cambiar, y surgieron magníficos entornos de escritorio como KDE y GNOME, y programas para llevar a cabo todas las tareas corrientes: procesadores de textos, hojas de cálculo, grabación de CD, visualización de DVD, escucha de MP3 y Ogg, fotografía digital, etc.

Android, un sistema operativo para teléfonos móviles y *tablets* que utiliza Linux como núcleo, es el líder indiscutible de su sector. Así, en el último par de años cientos de millones de usuarios han empezado a usar Linux, sin siquiera saberlo.

Figura 4.21: Logotipo de Android [67]

La manera más simple de asomarse al software libre es quizá empezar a usarlo sobre Windows. En efecto: también hay programas libres para Windows. Sobre esta cuestión hay un debate abierto en la comunidad: unos piensan que crear software libre para Windows es contraproducente, argumentando que disponer de software libre para Windows reduce los alicientes de cambiar de sistema operativo, mientras que otros afirman que esto les permitirá saborear el software libre fácilmente y les motivará a dar el salto. Hay una importante cantidad de programas libres para Windows, como la *suite* ofimática LibreOffice o el navegador web Mozilla Firefox.

Otra posibilidad más interesante es probar una distribución *live* como Ubuntu o Knoppix, que nos permitirá usar GNU/Linux en nuestro PC sin necesidad de instalar nada en él.

Instalar una distribución de GNU/Linux o *BSD no es especialmente complicado, y el usuario capaz de instalar Microsoft Windows no debería tener mayor problema. La facilidad de uso de los entornos KDE y GNOME es tal hoy en día que una vez instalados probablemente no necesitemos ninguna ayuda para manejarnos con ellos.

Sin embargo, es probable que nos pique la curiosidad y queramos aprender y sumergirnos en los secretos de nuestro nuevo sistema. La comunidad ha generado una asombrosa cantidad de documentación: las distribuciones de *BSD incluyen un exhaustivo y bien escrito manual, y el Proyecto de Documentación de Linux alberga numerosísimos manuales, cursos y documentos HOWTO (que explican cómo llevar a cabo una tarea específica). En las librerías hay también bastantes títulos en castellano, para todos los niveles.

Si disponemos de acceso a Internet, hay abundantes foros y listas de distribución de correo en las que los usuarios se ayudan unos a otros. Antes de plantear nuestras dudas en estos recursos, debemos recordar leer sus documentos FAQ, en los que se responden las preguntas más frecuentes. Así mismo, en la mayoría de las ciudades importantes hay grupos de usuarios (denominados LUG) o *hacklabs* (laboratorios de hacking), cuyos miembros, además de ayudarse mutuamente, organizan quedadas, jornadas, cursos y otras actividades a nivel local.

4.5.4. Obstáculos en el camino

Hace unos años, las empresas de software privativo menospreciaban al software libre y no lo percibían como una amenaza. Sin embargo, cada vez hay más y mejor software libre, que cuenta hoy con millones de usuarios. Muchas de las empresas y gobiernos más importantes del mundo se han rendido a la evidencia y han empezado también a emplearlo e incluso potenciarlo.

Hay que reconocer no obstante que todavía no ha llegado a absolutamente todas las parcelas posibles, y que hay algunas necesidades que no están debidamente cubiertas. La falta de madurez de las aplicaciones libres en algunos campos muy especializados (diseño asistido por computadora, por ejemplo) puede impedir que algunos profesionales puedan migrar completamente. La percepción general es que es simplemente una cuestión de tiempo.

Las empresas de software privativo están viendo así como el mundo del software está cambiando, por lo que han empezado a reaccionar en diferentes frentes. Algunas están adaptándose en mayor o menor medida a los nuevos tiempos, mientras que otras han preferido el combate, en diferentes frentes.

El primer obstáculo para la migración es en realidad un defecto del soft-

ware privativo, donde cada programa usa un formato propio no documentado, con lo que una aplicación, como un procesador de textos, no entiende y tiene problemas para abrir un documento creado por otra (o incluso por otra versión de ella misma). A la hora de cambiar a otro programa, es probable encontrarse con que los datos y documentos que se han acumulado durante años están en un formato cerrado, que el nuevo programa no conoce. Con demasiada frecuencia, el viejo programa se negará incluso a exportar nuestros documentos a un formato común de intercambio.

Evidentemente esta política es una estrategia deliberada de las casas de software, destinada atrapar a los sufridos usuarios en su programa, impedirles que puedan migrar a otro programa de la competencia, pues el coste de hacerlo subiría extraordinariamente. La situación es especialmente grave en caso de que dicha empresa cierre y el programa desaparezca.

En cambio, en el mundo del software libre se presta gran atención al uso de estándares, y todos los formatos están bien documentados, de modo que es imposible que nuestro trabajo quede atrapado en manos de una única empresa.

Otra dificultad similar es que algunas empresas usan aplicaciones hechas a medida que fueron escritas sin pensar en la portabilidad, de modo que solamente funcionan sobre Microsoft Windows, por lo que para migrar tendrían que reescribir el programa o usar un emulador de este sistema.

Otro arma que emplean son las técnicas FUD (*Fear, Uncertainty, and Doubt*): consiste en sembrar miedo, incertidumbre y duda mediante campañas de desinformación plagadas de medias verdades y datos obsoletos que confundan a la gente con menores conocimientos técnicos y les hagan desconfiar de la posibilidad de migrar a software libre.

Así, nos dirán que el software libre es una cosa de aficionados, que no tiene garantías y no se puede utilizar en entornos profesionales. O, por el contrario, pretenderán convencernos de que es muy difícil y que sólo sirve para expertos. Muchas veces estos ataques se camuflan como informes o artículos supuestamente imparciales, pero que están financiados por la empresa atacante. Algunos de estos FUD consiguen arraigar entre la gente y convertirse en auténticos mitos difíciles de desterrar, por más que se muestren pruebas que demuestren su falsedad.

Pero el mayor peligro que acecha al desarrollo del software libre es el de las patentes de software. Recordemos que, mientras el copyright controla la reproducción de una expresión concreta, las patentes otorgan durante un tiempo limitado la exclusividad para la explotación de una invención. Esto tenía cierto sentido cuando se inventó, pues se aplicaba a invenciones mecánicas. Pero aplicar el mismo concepto a ideas, como son los algoritmos matemáticos y por tanto los programas informáticos, lleva a situaciones absurdas. Por poner una analogía, el copyright controla la letra de la canción *Happy birthday to you* [70], pero no impide que la cantemos con otra letra.

En cambio, una patente controlaría la idea misma de felicitar un cumpleaños, como quiera que lo hagamos.

El artículo 52 de la Convención de la Patente Europea excluye expresamente los programas de ordenador «como tales», pero mediante el subterfugio de presentarlos junto a algún elemento hardware, se han estado concediendo patentes de software desde los años 80.

Además, algunos *lobbies* han estado presionando para que se adopte una legislación similar a la estadounidense. Esta adopción sería catastrófica no sólo para el software libre, sino también para la pequeña y mediana empresa informática europea.

El último peligro es interno, y es olvidar la importancia de la libertad y aceptar el uso de software privativo sobre sistemas libres, como hacen algunas distribuciones de GNU/Linux.

4.6. Conclusiones

Como hemos visto, la competencia salvaje de la industria del software privativo había llevado a una situación de monopolios, precios desorbitados, falta de innovación, abusos como documentos cautivos en formatos cerrados, etc,

Afortunadamente Stallman no fue el último hacker. En parte por la difusión de las computadoras personales, y en parte por su propio trabajo al crear el proyecto GNU y difundir su filosofía, han aparecido nuevas generaciones de hackers que han adoptado la ética del hacker y dedican su talento a escribir software y documentación libre.

En la actualidad, el software libre ha alcanzado tal grado de desarrollo que ya no es necesario convencer a nadie de que es posible. Ya no es una cuestión de fe, la existencia de productos de altísima calidad como el sistema GNU/Linux y programas como Mozilla Firefox, LibreOffice, Apache, Perl, Python, PHP, MySQL, PostgreSQL, etc es indiscutible. La filosofía de apoyo mutuo del modelo colaborativo del software libre ha demostrado así que lo que es mejor éticamente, también lo es en la práctica, y que compartir la información y el conocimiento nos beneficia a todos: un magnífico ejemplo para la educación en valores.

Como se ha explicado, el modo de funcionamiento del modelo bazar es muy asambleario; por las listas de distribución de correo electrónico se discuten todas las ideas, se estudian sus ventajas y fallos, y se busca el consenso sobre cual es la mejor: una suerte de democracia directa.

El software libre también está produciendo cambios en la economía. No deja de ser sorprendente que Bill Gates se haya convertido en una de las personas más ricas del mundo vendiendo algo inmaterial desarrollado por trabajadores asalariados anónimos. El software libre está trayendo una mayor equitatividad al sector, y permitiendo que los programadores recuperen el

control de su trabajo. Quizás ya no sea posible hacerse rico repentinamente, pero sin duda el conjunto de la sociedad sale ganando. Para los países en vías de desarrollo el software libre supone también una oportunidad tecnológica que no necesita de grandes inversiones.

Si bien hace unos años un sistema GNU/Linux o *BSD podía resultar de difícil manejo para un usuario final, el software libre ha ampliado sus horizontes y proporciona ahora programas de excelente calidad que satisfacen las necesidades de la mayoría de los usuarios, los actuales entornos de escritorio KDE y GNOME no sólo no tienen nada que envidiar al software privativo en facilidad de uso, sino que lo dejan claramente atrás en aspectos como accesibilidad y capacidad de personalización.

Para la mayoría de los usuarios ya es posible llevar a cabo todas sus tareas usando únicamente software libre, y solamente es cuestión de querer dar el salto y aprender algunas cosas nuevas. En cambio, a un usuario sin conocimientos informáticos le supondrá el mismo esfuerzo aprender a manejarse en un entorno libre que en uno privativo.

Para muchos usuarios, las nuevas libertades que les da este software les abre un nuevo campo en el que jugar y aprender, con lo que acaba convirtiéndose en una pasión. En cualquier caso, y aunque sólo sea como curiosidad, es un mundo que merece la pena explorar, y que el profesional de la informática no puede ignorar.

Capítulo 5

Apoyos tecnológicos: normativa que deben cumplir

Resumen: En este capítulo se pormenorizan los requisitos de accesibilidad que deben cumplir las aplicaciones informáticas para personas con discapacidad (norma UNE 139802:2009).

En julio de 2009 el comité técnico AEN/CTN 139 *Tecnologías de la Información y las Comunicaciones para la Salud* publicó la norma UNE 139802:2009, es la versión oficial, en castellano, de la Norma Europea EN ISO 9241-171:2008, que a su vez adopta la Norma Internacional ISO 9241-171:2008. Esta norma anula y sustituye la ISO 139802:2003, que a su vez anulaba y sustituía la Norma Experimental UNE 139802 EX de enero de 1998.

5.1. Introducción

El objetivo de esta parte (la -171) de la Norma ISO 9241 es proporcionar directrices para el diseño del software de sistemas interactivos con el fin de alcanzar el máximo nivel posible de accesibilidad. El diseño de sistemas para aumentar la accesibilidad contribuye a mejorar la eficacia, la eficiencia y la satisfacción de las personas que tienen una amplia variedad de capacidades y preferencias. La accesibilidad está, por tanto, estrechamente relacionada con el concepto de usabilidad (véase la Norma ISO 9241-11).

Los principales procedimientos para mejorar la accesibilidad de las interfaces persona-sistema son:

- adoptar un enfoque de diseño centrado en la persona (véase la Norma ISO 13407);
- seguir un proceso de diseño centrado en el contexto;

- proporcionar la capacidad de individualización (véase la Norma ISO 9241-110);
- ofrecer al usuario instrucciones y formación personalizadas.

Es importante que los objetivos y las características de accesibilidad se incorporen al diseño tan pronto como sea posible, cuando es relativamente económico, en comparación con el coste de modificar productos ya diseñados para que sean accesibles, que es mucho mayor. Esta parte de la Norma ISO 9241, además de proporcionar directrices para conseguir el propósito anterior, responde a la creciente necesidad de considerar las exigencias sociales y legislativas para garantizar la accesibilidad, mediante la supresión de barreras que impiden que las personas participen en actividades cotidianas tales como el uso de entornos, servicios, productos e información.

Esta parte de la Norma ISO 9241 se aplica al software de sistemas interactivos utilizados en el hogar, en actividades de ocio, en lugares públicos y en el trabajo. Proporciona requisitos y/o recomendaciones en cuanto a diseño, apariencia y comportamiento del sistema, así como cuestiones específicas de accesibilidad. Por tanto, complementa a las Normas ISO 9241-11, ISO 9241-12, ISO 9241-13, ISO 9241-14, ISO 9241-15, ISO 9241-16, ISO 9241-17, ISO 9241-110 e ISO 14915, asimismo refleja los objetivos establecidos en la Guía ISO/IEC 71. También, es importante cumplir con estas normas internacionales para poder lograr la accesibilidad.

Esta parte de la Norma ISO 9241 se basa en el conocimiento que existe actualmente en cuanto a las características de las personas que tienen una diversidad funcional física, sensorial y/o cognitiva. Sin embargo, la accesibilidad es una cuestión que afecta a muchos grupos de personas. Los usuarios potenciales de sistemas interactivos son consumidores o profesionales - personas en casa, estudiantes, ingenieros, empleados, vendedores, diseñadores de páginas Web, etc. Las capacidades físicas, sensoriales y cognitivas de las personas pertenecientes a estos grupos varían considerablemente y cada grupo de usuarios objetivo incluirá a personas con capacidades diferentes. Por lo tanto, las personas con discapacidad no forman un grupo específico que puede ser aislado y posteriormente descartado. Las diferencias entre las capacidades pueden deberse a diversos factores que limitan la capacidad para participar en las actividades de la vida diaria, y que constituyen una «experiencia humana universal». Por tanto, la accesibilidad se refiere a un amplio grupo de usuarios, que incluye a:

- personas con deficiencias físicas, sensoriales y con problemas cognitivos desde su nacimiento o adquiridos a lo largo de la vida;
- personas mayores que podrían beneficiarse de nuevos productos y servicios pero ven disminuidas sus capacidades físicas, sensoriales y cognitivas;

- personas con discapacidad temporal, como una persona con un brazo roto o alguien que ha olvidado sus gafas; y
- personas que en determinadas situaciones tienen dificultades, como una persona que trabaja en un entorno ruidoso o tiene las manos ocupadas por otras actividades.

A la hora de diseñar y evaluar sistemas interactivos, hay otros términos que a menudo se asocian con la accesibilidad. Los términos «diseño para todos» en Europa y «diseño universal» en Estados Unidos, afrontan el objetivo de lograr la máxima accesibilidad posible para la mayor cantidad y diversidad de usuarios, independientemente de su nivel de habilidad, idioma, cultura, entorno o discapacidad. Esto no significa que cada producto podrá ser usado por todos los consumidores. Siempre habrá una minoría de personas con deficiencias graves o múltiples que necesitarán adaptaciones o productos especializados. La accesibilidad, tal y como se define en esta parte de la Norma ISO 9241, se centra en los objetivos de maximizar el número de usuarios y aumentar el nivel de usabilidad experimentado por esos usuarios.

Esta parte de la Norma ISO 9241 reconoce que algunos usuarios de software necesitarán ayudas técnicas (productos de apoyo) para poder utilizar un sistema. El concepto de diseño de software accesible incluye la capacidad de un sistema para proporcionar conexiones con las ayudas técnicas y permitir su integración con éxito, a fin de aumentar el número de personas que pueden utilizar un sistema interactivo. Se proporcionan directrices para el diseño de software que integre las ayudas técnicas de la forma más eficaz posible. Es importante señalar que accesibilidad se puede proporcionar a través de una combinación de software y hardware controlado por software. Las ayudas técnicas, normalmente, proporcionan posibilidades de salidas y entradas no proporcionadas por el sistema. Ejemplos de software serían los emuladores de teclado que aparecen en la pantalla para sustituir a los teclados físicos, los magnificadores de pantalla que permiten a los usuarios ver su pantalla con diferentes niveles de aumento, y el lector de pantalla que permite a los usuarios ciegos navegar a través de las aplicaciones, determinar el estado de los controles, y leer el texto a través de síntesis de voz. Los ejemplos de hardware incluyen ratones o punteros de cabeza que permiten el control del cursor y dispositivos de salida Braille en lugar de un monitor. Hay muchos otros ejemplos. Si los usuarios emplean software y/o hardware de apoyo complementario, la usabilidad mejorará siempre que los sistemas y aplicaciones puedan integrar esas tecnologías. Por este motivo las plataformas (incluidos los sistemas operativos) deben proporcionar servicios de programación para permitir que el software pueda funcionar de forma eficaz con software y hardware de apoyo complementario, tal como se especifica en esta parte de la Norma ISO 9241. Si los sistemas no dan soporte a las ayudas técnicas, aumenta la probabilidad de que los usuarios se encuentren con problemas de compatibilidad, rendimiento y usabilidad.

Esta parte de la Norma ISO 9241 está destinada a los siguientes tipos de usuarios:

- diseñadores de herramientas de desarrollo de interfaces de usuario y de guías de estilo destinadas a diseñadores de interfaces;
- diseñadores de interfaces de usuario, que aplicarán las directrices durante el proceso de desarrollo;
- desarrolladores, que aplicarán las directrices durante el diseño e implementación de la funcionalidad del sistema;
- aquéllos que sean responsables de aplicar soluciones para satisfacer las necesidades de usuarios finales;
- compradores, que tomarán como referencia esta parte de la Norma ISO 9241 para las compras públicas;
- evaluadores, que sean responsables de garantizar que los productos son conformes con esta parte de la Norma ISO 9241.

El beneficiario último de esta parte de la Norma ISO 9241 será el usuario final del software. Aunque es poco probable que los usuarios finales lean esta parte de la Norma ISO 9241, su aplicación por parte de los diseñadores, desarrolladores, evaluadores y compradores debería proporcionar interfaces de usuario que sean más accesibles. Esta parte de la Norma ISO 9241 se refiere al desarrollo de software para interfaces de usuario. Sin embargo, la norma también puede ser útil para aquellos que participen en el diseño de interfaces de usuario hardware cuando se considere la interacción entre software y hardware.

La Norma ISO 9241 se desarrolló originalmente como una norma internacional con diecisiete partes sobre los requisitos ergonómicos para trabajos de oficinas con pantallas de visualización de datos. Como parte del proceso de revisión de las normas, se acordó una importante reestructuración de ISO 9241, para ampliar su campo de aplicación, para incorporar otras normas pertinentes y hacerla más usable. La revisión del título general de ISO 9241, «Ergonomía de la interacción hombre-sistema», refleja estos cambios y está en consonancia con el título y el campo de actuación del Comité Técnico ISO/TC 159/SC 4. La norma, compuesta de varias partes, ha sido revisada y está estructurada como una serie de normas numeradas por «centenas»: Por ejemplo la serie 100 está dedicada a interfaces de software, la serie 200 a diseño centrado en el hombre, la serie 300 a pantallas de visualización, la serie 400 a los dispositivos físicos de entrada, y así sucesivamente.

5.2. Objeto y campo de aplicación

Esta parte de la Norma ISO 9241 proporciona directrices y especificaciones de ergonomía para el diseño de software accesible para su uso en el trabajo, en el hogar, en la educación y en lugares públicos. Abarca cuestiones relacionadas con el diseño de software accesible para personas con la más amplia gama de capacidades físicas, sensoriales y cognitivas, incluyendo a personas con discapacidades temporales y a las personas mayores. Esta parte de la Norma ISO 9241 se centra en la accesibilidad del software, complementando al diseño general de usabilidad tratado en las Normas ISO 9241-110, ISO 9241-11 a 9241-17, ISO 14915 e ISO 13407.

Esta parte de la Norma ISO 9241 es aplicable a la accesibilidad de sistemas interactivos. Contempla una amplia gama de software (por ejemplo, de oficina, Web, de apoyo al aprendizaje y sistemas de bibliotecas).

Promueve el aumento de la usabilidad de los sistemas para una gama más amplia de usuarios. Aunque no abarca ni el comportamiento ni los requisitos de las ayudas técnicas (incluyendo software de apoyo), sí aborda el uso de las ayudas técnicas como un componente integrado en los sistemas interactivos.

Está dirigida a aquellos que son responsables de la especificación, diseño, desarrollo, evaluación y adquisición de software de plataforma y de aplicación.

5.3. Explicación y beneficios de la aplicación de la accesibilidad

La accesibilidad es una consideración importante en el diseño de productos, sistemas, entornos y servicios porque afecta al rango de personas que pueden usarlos y al hecho de que puedan utilizarlos fácilmente. Cuanto más accesible sea un diseño, mayor será la gama de personas que lo encontrarán usable.

La accesibilidad se puede mejorar mediante la incorporación de características y atributos que resultan beneficiosos para usuarios con necesidades especiales. Para determinar el nivel de accesibilidad alcanzado es necesario medir la eficacia, la eficiencia y la satisfacción de los usuarios que trabajan con un producto o que interaccionan con un entorno, considerando la máxima diversidad de usuarios. La medición de la accesibilidad es especialmente importante, dada la complejidad de las interacciones con el usuario, los objetivos, las características de la tarea y los demás elementos del contexto de uso. El nivel de accesibilidad de un producto, sistema, entorno o instalación puede variar notablemente según su contexto de uso.

Planificar de la accesibilidad como una parte integral del diseño y del proceso de desarrollo, implica identificar sistemáticamente los requisitos de

accesibilidad, incluyendo medidas de accesibilidad y criterios de verificación dentro del contexto de uso. Esto proporcionará los objetivos del diseño que servirán como base para evaluar el diseño resultante.

El enfoque adoptado en esta parte de la Norma ISO 9241 proporciona las siguientes ventajas:

- el marco de trabajo puede utilizarse para identificar los aspectos de accesibilidad y los componentes del contexto de uso que deben tenerse en cuenta a la hora de especificar, diseñar o evaluar la accesibilidad de un producto;
- el rendimiento y la satisfacción de los usuarios puede utilizarse para medir el grado en que un producto, sistema, entorno o instalación es accesible en un contexto especificado;
- la medida del rendimiento y satisfacción de los usuarios puede proporcionar una base para determinar y comparar la accesibilidad de productos con diferentes características técnicas que se utilizan en un mismo contexto;
- la accesibilidad prevista para un producto puede ser definida, documentada y verificada (por ejemplo, como parte de un plan de calidad).

5.4. Principios para el diseño de software accesible

Hay diferentes formas de diseñar software accesible. Esta parte de la Norma ISO 9241 no presupone ningún método de diseño o proceso específico, ni abarca todas las diferentes actividades necesarias para garantizar el diseño de sistemas accesibles. Es un complemento para los actuales métodos de diseño y proporciona una perspectiva de accesibilidad centrada en la persona, basándose en la Norma ISO 13407, que puede ser aplicada (para aumentar el número de personas que son capaces de utilizar el software) en cualquier proceso de diseño o contexto de uso específico. La orientación proporcionada en esta parte de la Norma ISO 9241 es aplicable a cualquier etapa del desarrollo de sistemas interactivos.

El diseño de software accesible debería adherirse a los siguientes principios.

Uso equitativo Las soluciones equitativas proporcionan los mismos medios de uso para todos los usuarios: idénticos siempre que sea posible o, en otro caso, equivalentes. Lograr un uso equitativo garantizará que las soluciones diseñadas para aumentar la accesibilidad no causarán una pérdida de privacidad, un aumento de riesgos para la seguridad y la protección, o la estigmatización de los individuos.

Adecuación para el máximo rango de usos La adecuación para el máximo rango de usos, implica diseñar con el objetivo de producir soluciones que serán útiles, aceptables y estén disponibles para la mayor variedad posible de usuarios dentro de la población objetivo, teniendo en cuenta sus habilidades especiales, variaciones en sus capacidades, la diversidad de sus tareas, y sus diferencias de entorno, económicas y sociales.

Robustez El software debería diseñarse para ser tan robusto como sea posible de forma que funcione con las ayudas técnicas actuales y futuras. Aunque no es posible hacer accesible todo el software sin agregar ayudas técnicas, estas directrices deberían ayudar a los diseñadores a desarrollar software que aumente la accesibilidad sin el uso de ayudas técnicas, y que, mediante el suministro de la información necesaria de la interfaz, permitan que el software y los dispositivos de ayuda puedan funcionar de manera eficaz y eficiente cuando se utilicen. El software puede promover la integración de las ayudas técnicas si proporciona información que pueda ser interpretada por éstas, y si facilita la comunicación a través de protocolos estándar de comunicación entre aplicaciones.

Es conveniente que las soluciones desarrolladas para que el software sea accesible estén basadas en la aplicación de directrices de diseño ergonómico: ISO 9241-12, ISO 9241-13, ISO 9241-14, ISO 9241-15, ISO 9241-16, ISO 9241-17 e ISO 9241-110, y aquellas provistas en las Normas ISO 14915-1, ISO 14915-2 e ISO 14915-3. Las directrices de estas normas internacionales se incluyen también en las disposiciones de otras normas y publicaciones específicamente dedicadas a mejorar la accesibilidad, tales como el Web Content Accessibility Guidelines 2.0 (WCAG)], principios de diseño universal/diseño para todos (DFA). Algunos factores que representan principios y buenas prácticas en materia de ergonomía y que son de especial importancia como base para el logro de la accesibilidad son los siguientes:

- la información debe ser perceptible por el usuario (ISO 9241-12, Principio nº 1 de WCAG 2.0);
- el contenido y los controles deben ser comprensibles (ISO 9241-12, ISO 9241-110, Principio nº 3 de WCAG 2.0);
- los elementos de la interfaz deben ser operables (ISO 9241-110, Principio nº 2 de WCAG 2.0);
- el software debería ser tolerante ante los errores (ISO 9241-110, DFA);
- el software debe ser flexible en el uso, permitiendo a los usuarios elegir entre una gama más amplia de alternativas de entrada y salida. (ISO 9241-110, DFA).

La orientación adicional facilitada en esta parte de la Norma ISO 9241 se refiere específicamente a cuestiones que surgen al proporcionar soluciones de diseño que han de satisfacer las necesidades de usuarios con una gran variedad de capacidades.

5.5. Requisitos

A continuación se enumeran los requisitos que deben cumplirse para poder verificar la conformidad con esta parte de la Norma ISO 9241. Para más detalles, notas y ejemplos puede consultarse la propia norma.

5.5.1. Recomendaciones y requisitos generales

5.5.1.1. Nombres y etiquetas de los elementos de interfaz de usuario

8.1.1 Proporcionar un nombre a cada elemento de interfaz de usuario.

8.1.4 Hacer que los nombres estén disponibles para las ayudas técnicas (AT).

5.5.1.2. Ajustes de preferencias de usuario

8.2.4 Facilitar la individualización del cursor y el puntero.

8.2.7 Permitir que el usuario controle el tiempo de respuesta.

5.5.1.3. Consideraciones especiales sobre ajustes de accesibilidad

8.3.1 Hacer que los controles de las características de accesibilidad sean fáciles de descubrir y operables.

8.3.3 Evitar interferir con las características de accesibilidad.

5.5.1.4. Pautas generales sobre control y uso

8.4.4 Proporcionar alternativas cuando las ayudas técnicas no estén operativas.

8.4.5 Permitir que el software pueda controlar la expulsión de medios.

8.4.9 Permitir que persistan los avisos o la información sobre errores.

5.5.1.5. Compatibilidad con las ayudas técnicas

8.5.2 Facilitar la comunicación entre el software y las AT.

8.5.3 Utilizar los servicios estándar de accesibilidad.

8.5.4 Hacer que la información de elementos de interfaz de usuario esté disponible para las ayudas técnicas.

8.5.5 Permitir que las ayudas técnicas cambien el foco de teclado y la selección.

8.5.6 Proporcionar descripciones de los elemento de interfaz de usuario.

8.5.7 Hacer que la notificación de eventos esté disponible para las ayudas técnicas.

8.5.9 Utilizar las entradas y salidas estándar del sistema.

8.5.10 Facilitar una presentación adecuada de tablas.

8.5.11 Permitir la instalación de emuladores de teclado o de dispositivos apuntadores.

8.5.12 Permitir que las ayudas técnicas supervisen las operaciones de salida.

5.5.1.6. Sistemas cerrados

8.6.1 Leer el contenido de sistemas cerrados.

8.6.2 Anunciar cambios en sistemas cerrados.

8.6.3 Operatividad a través de controles que sean reconocibles táctilmente.

8.6.4 Dejar pasar las funciones del sistema.

5.5.2. Entradas

5.5.2.1. Opciones alternativas de entrada

9.1.2 Permitir el control en paralelo de las funciones del dispositivo apuntador mediante el teclado.

5.5.2.2. Foco del teclado

9.2.1 Proporcionar cursores de foco de teclado y de texto.

9.2.2 Proporcionar cursores de foco de teclado y de texto de gran visibilidad.

5.5.2.3. Entrada de teclado

9.3.2 Facilitar el uso completo mediante teclado.

9.3.3 Permitir la entrada secuencial de combinaciones de teclas.

9.3.4 Permitir el ajuste del tiempo mínimo antes de la aceptación de pulsaciones de tecla.

9.3.5 Permitir el ajuste de la aceptación de la pulsación repetida de la misma tecla.

9.3.8 Permitir que los usuarios desactiven la repetición de teclas.

9.3.12 Reservar asignaciones de atajos de teclado de accesibilidad.

9.3.14 Separar la navegación con el teclado de la activación.

5.5.2.4. Dispositivos anotadores

9.4.2 Proporcionar control directo de la posición del puntero para dispositivos externos.

9.4.4 Permitir cambiar la asignación de funciones de los botones de dispositivos apuntadores.

9.4.6 Proporcionar la función de mantener pulsado un botón de dispositivo apuntador.

9.4.9 Permitir el ajuste de parámetros de clics múltiples.

9.4.10 Permitir el ajuste de la velocidad del puntero.

9.4.11 Permitir el ajuste de la aceleración de puntero.

9.4.13 Proporcionar un medio para encontrar el puntero.

9.4.14 Proporcionar alternativas a operaciones simultáneas de puntero.

5.5.3. Salidas

5.5.3.1. Recomendaciones generales sobre salidas

10.1.1 Evitar frecuencias de destello de que provocan ataques epilépticos.

10.1.2 Permitir que el usuario pueda controlar la presentación de información dependiente del tiempo.

10.1.3 Proporcionar alternativas accesibles para información audiovisual relevante para la tarea.

5.5.3.2. Salida/Output visual (pantalla)

10.2.4 Proporcionar acceso mediante teclado a información que se presenta fuera de la pantalla física.

5.5.3.3. Color

10.4.1 No transmitir información utilizando únicamente el color.

5.5.3.4. Aspecto y comportamiento de las ventanas

10.5.3 Permitir la navegación sin puntero entre ventanas.

10.5.4 Permitir usar ventanas «siempre delante» (1er plano).

10.5.5 Permitir al usuario controlar múltiples ventanas «siempre delante».

10.5.7 Permitir posicionar las ventanas.

10.5.10 Permitir que las ventanas eviten tener el foco.

5.5.3.5. Salida sonora

10.6.2 Permitir el control del volumen.

10.6.7 Permitir a los usuarios elegir alternativas visuales para salidas sonoras.

10.6.8 Sincronizar los equivalentes sonoros de eventos visuales.

10.6.9 Proporcionar servicios de síntesis de voz.

5.5.3.6. Equivalentes textuales del sonido (subtítulos)

10.7.1 Mostrar cualquier subtítulo proporcionado.

10.7.3 Dar soporte a la configuración global de los subtítulos.

5.5.3.7. Multimedia

10.8.1 Permitir que los usuarios puedan detener, iniciar y pausar.

5.5.4. Documentacion en línea, «ayuda» y servicios de soporte técnico

5.5.4.1. Documentación y «Ayuda»

11.1.2 Proporcionar documentación de usuario en formato electrónico accesible.

11.1.3 Proporcionar alternativas de texto en la documentación electrónica y en la «Ayuda».

11.1.5 Proporcionar documentación y «Ayuda» sobre las características de accesibilidad.

5.5.4.2. Servicios de soporte técnico

11.2.1 Proporcionar servicios de soporte técnico accesibles.

Capítulo 6

Descripción de los programas previos

Resumen: En este capítulo se expondrán los programas para la enseñanza-aprendizaje del Braille existentes con anterioridad, y sus características.

En los últimos años ha crecido el interés por la aplicación de la tecnología en la enseñanza-aprendizaje del Braille. Se han desarrollado diversos dispositivos físicos. Sin ánimo de ser exhaustivo se pueden mencionar:

- La propuesta de Cétares et al. [40] de un sistema de enseñanza del código Braille para niños con limitaciones visuales.
- *Automated Braille Writing Tutor*, dispositivo desarrollado en la Universidad de Carnegie Mellon y que desde el primer momento se probó en la India.
- La patente estadounidense US5902112 (Sally S. Mangold), con una matriz rectangular de zonas sensibles al tacto y un procesador acoplado a un sintetizador que pronuncia el carácter Braille.
- La patente alemana DE19547742 (René LEMOINE), en la que se describe un dispositivo y un método para practicar idiomas por escrito u oralmente sin necesidad de profesor. En concreto, reinventa un método para la introducción y muestra de signos visuales y sonoros a elección del usuario.
- La solicitud internacional de patente WO2008120303 con unidades de entrada de caracteres Braille para introducir un carácter Braille por medio de una pluralidad de unidades de presión y de unidades de confirmación para la salida de los caracteres Braille introducidos.

- La patente estadounidense US576963, especialmente destinada a la enseñanza de operaciones de cálculo básicas.

Así mismo, y a medida que se avanzó en el tiempo, se fue desarrollando software específico para discapacitados visuales, que recibe el nombre de *tiflosoftware*. La ONCE mantiene un directorio [127] de programas tales como magnificadores, lectores de pantalla, cuentos y juegos para niños, utilidades de telefonía, etc.

Sin embargo, las aplicaciones informáticas para el aprendizaje del Braille son muy escasas. Éstas se pueden dividir en dos grupos, según vayan dirigidas a videntes o personas con discapacidad visual. Nela se enmarca en el segundo grupo, que por tanto se estudiará con mas detalle.

6.1. Programas para videntes

6.1.1. Curso básico de autoaprendizaje del Braille

Es un programa *online* realizado en Macromedia Flash por la ONCE [148], con un guión de Carmen Roig. Está diseñado para personas sin problemas de visión que, por su trabajo, circunstancias familiares o interés general, deseen conocer el sistema Braille. Por medio de animaciones un hada ciega (Braillinda) que visita a Louis Braille se introducen las letras, signos de puntuación y números (figura 6.1).

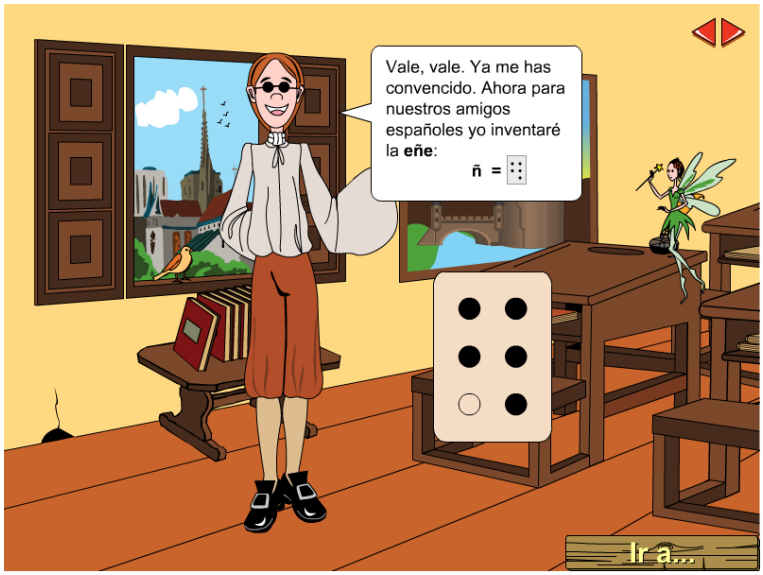


Figura 6.1: Captura de «Luis y Braillinda te lo cuentan» [148]

6.1.2. Braille virtual

Este curso se puede seguir *online* desde el navegador, o bien descargarse al PC. Fue producido por la Universidad de São Paulo (Brasil) en 2004, y esta disponible en portugués, castellano e inglés. También desarrollado en Flash, consiste en animaciones gráficas que presentan los símbolos divididos en grupos de 10. Tras la presentación, se pueden realizar ejercicios de memorización (figura 6.2).

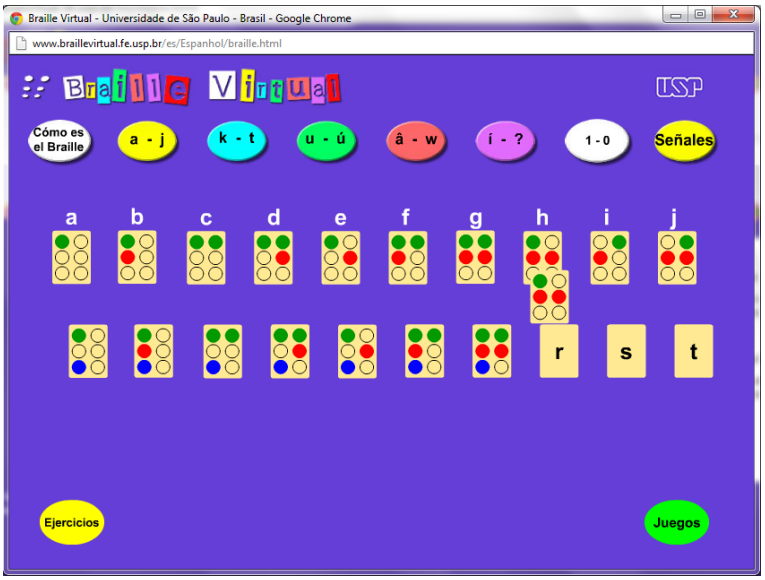


Figura 6.2: Captura del curso «Braille virtual» [178]

6.1.3. Detrás de cada punto

Fue un programa desarrollado por el Instituto Nacional para Ciegos de Colombia (INCI) [74] para Microsoft Windows 95, pero que actualmente (marzo 2013) ya no se encuentra disponible. Este programa daba seguimiento al proceso de adquisición de la escritura Braille a través de ejercicios y actividades de evaluación a personas videntes, por lo que era útil para educadores y familiares de niños con discapacidad visual. Contenía actividades de lectura y escritura de palabras y frases en Braille.

6.2. Programas para discapacitados

6.2.1. Salon Braille Virtual

Es un tutorial interactivo diseñado y desarrollado por José Manuel Álvarez, del Programa de Asistencia Tecnológica de la Universidad de Puerto

Rico, y programado en Macromedia Flash por Javier A. López Ramos en los años 2006–2009 [136].

Es un programa interactivo que sirve de apoyo tecnológico en el proceso tradicional de enseñanza del sistema Braille en el aula. El programa no pretende sustituir el aprendizaje del Braille, sino ser un refuerzo mientras los estudiantes ciegos tocan los puntos en relieve del Braille, así como utilizando la máquina Perkins y la regleta tradicional (figura 6.3).

Está dirigido tanto a niños como a adultos ciegos, así como a aquellos con un grado de visión limitado. Está disponible para Microsoft Windows y para Mac OS X.

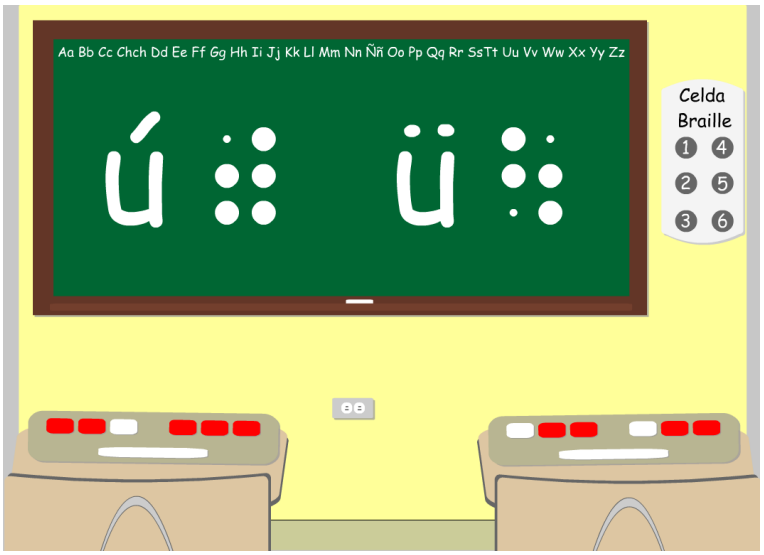


Figura 6.3: Captura de SBV [136]

A través de una serie de órdenes de voz (con un acento y vocabulario marcadamente portorriqueños), la persona puede ir aprendiendo las combinaciones de puntos que representan letras, números y signos de puntuación en el sistema Braille. También se aprende cómo marcar estas combinaciones en los diferentes instrumentos que se utilizan en la escritura del mismo.

El programa tiene las siguientes actividades (figura 6.4)

Letras Al pulsar en el teclado alguna letra aparece, en voz y en pantalla, la combinación de puntos que representa dicha letra.

Letras acentuadas Al pulsar una vocal, aparece la combinación de puntos de la misma al ser acentuada.

Signos de puntuación Al pulsar una tecla de un signo de puntuación, aparece la combinación de puntos para dicho signo.

Números Al pulsar una tecla numérica, aparece la combinación de puntos del mismo, precedido por la combinación que representa el signo numérico en el sistema Braille.

Explorar celda Braille Con las teclas cursores se pueden marcar diferentes puntos del cajetín Braille, y el programa indica qué representa dicha combinación.

Explorar maquinilla Perkins Similar a la actividad anterior, pero usando la máquina Perkins.

Explorar regleta Braille Similar a las dos anteriores, pero teniendo en cuenta que al escribir con una regleta los puntos se han de voltear horizontalmente.

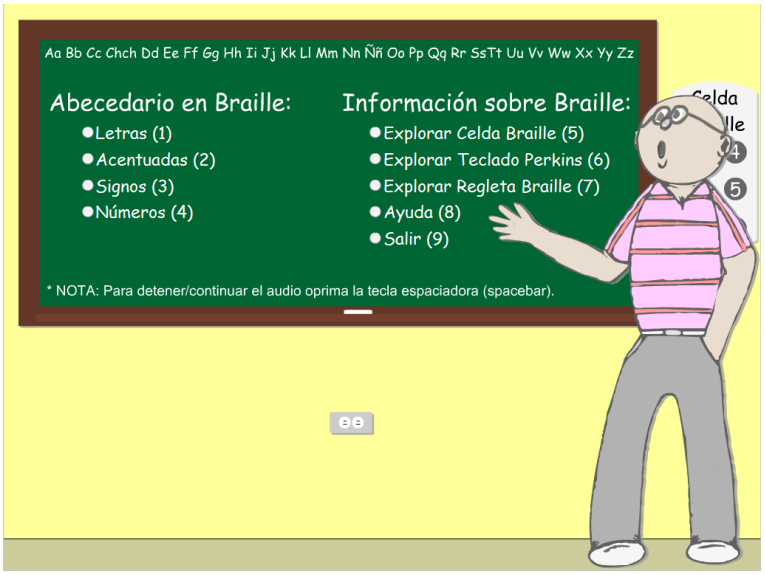


Figura 6.4: Menú de actividades de SBV [136]

6.2.1.1. Inconvenientes

- Para el usuario no alfabetizado, carece de método pedagógico para aprender a escribir: no enseña a combinar las letras. Es decir, el niño podrá aprender el nombre de la C (ce) o de la Q (cu), pero no sabrá escribir palabras como «casa» o «queso».

- Para el usuario alfabetizado, no presenta las letras una a una en un orden lógico (por frecuencia o alfabético, por ejemplo), sino que todas están disponibles desde el primer momento. No tiene ejercicios para comprobar los conocimientos ni reforzar las que no se recuerden bien.

- Obliga al usuario invidente a manejarse con cierta soltura con el teclado: necesita conocer al menos la ubicación de los números, las teclas cursores, control, borrar...
- Sólo está disponible en castellano, no está internacionalizado.
- Falta de flexibilidad. El docente no puede hacer ningún cambio, ni siquiera sustituir las grabaciones por otras en el acento local.
- No es software libre, aunque su descarga es gratuita. No se le pueden hacer correcciones o mejoras.

6.2.2. Pequén LeeTodo Braille

Pequen LeeTodo [98] es una obra multimedia, interactiva y parlante integrada por un escritorio con ocho aplicaciones que dan acceso a 31 programas. Facilita el acercamiento al mundo de la informática a usuarios de todas las edades, con o sin discapacidad visual, cognitiva, auditiva o motora. Ha sido realizada en Argentina por Marcelo H. Martinotti y Lidia B. Miño.

La aplicación Pequén LeeTodo Braille está diseñada para afianzar el aprendizaje del teclado Braille y agilizar la escritura de palabras y números con la máquina Perkins. Sólo se utilizan seis teclas del teclado del PC, simulando el teclado de una máquina Perkins. La práctica de escritura de manera virtual promueve el mejor aprovechamiento del consumo de papel.

Figura 6.5: Captura de Pequén LeeTodo: Práctica Braille - Dictado de palabras [98]

Dentro de LeeTodo, la aplicación de Braille presenta tres actividades:

Teclado Braille, Práctica Braille y Calculadora. La complejidad de las actividades aumenta gradualmente al pasar de una a otra.

Teclado Braille Esta actividad facilita el aprendizaje de la máquina Braille, que el programa simula con las teclas F-D-S y J-K-L. Permite escribir números, signos o símbolos y mayúsculas. Atendiendo a las necesidades de las personas con disminución visual se presentan las letras en braille en tamaño gigante, ocupando gran parte de la pantalla y la imagen de un teclado Braille virtual. Simultáneamente se indica en audio la letra pulsada.

Práctica Braille Tiene tres secciones:

Alfabeto En esta actividad Pequén dicta el alfabeto para el afianzamiento de la escritura en sistema Braille. Si el usuario escribe en forma incorrecta, el asistente le indica el número correspondiente a las teclas del teclado virtual Braille que debe pulsar (figura 6.6).

Números Pequén solicita la escritura de números. El software los muestra con signos grandes para disminuidos visuales. El usuario deberá pulsar las teclas correspondientes, si lo hace en forma correcta pasa al siguiente número. En caso contrario, el software le indica el número de las teclas que se deben pulsar, continuando de esta forma hasta concluir el nivel elegido. Si el usuario supera el nivel ingresado, el software pasa automáticamente al siguiente nivel (de una cifra a dos, de dos cifras a tres, etc.).

Dictado de Palabras Pequén dicta palabras para el afianzamiento de la escritura en sistema Braille. Cuenta con 22 listados de palabras, que van incorporando nuevas letras progresivamente. Si el usuario escribe en forma incorrecta el asistente le indica el número correspondiente a las teclas que se deben pulsar. Se registra el desempeño en cada práctica y el tiempo de realización, para que el docente o acompañante podrán hacer el seguimiento de la actividad. Pequén LeeTodo proporciona en pantalla la escritura en forma convencional y en Braille (figura 6.5).

Calculadora Este módulo permite la práctica de la escritura de números y de las operaciones matemáticas simples de suma, resta, multiplicación y división utilizando el teclado Braille.

Figura 6.6: Captura de Pequén LeeTodo: Práctica Braille - Alfabeto [98]

6.2.2.1. Inconvenientes

- Aunque tiene una interesante actividad de dictado de palabras, éstas no vienen acompañadas de imágenes que la hagan más interesante a los usuarios que mantengan un resto de visión.
- La secuencia de palabras no es óptima, porque las sílabas no se introducen en un orden pedagógicamente adecuado (primero las directas, después las inversas y por último las trabadas).
- Utiliza palabras ajenas al vocabulario infantil (oropel, magma, onírico, alabastro, agar agar, etc).
- Cada vez que se inicia el programa hay que iniciar el dictado desde la primera palabra, no sigue desde el punto alcanzado en la sesión anterior.
- Sólo está disponible para Microsoft Windows. En nuestras pruebas, la estabilidad de la versión actual en Windows 7 ha resultado mediocre.
- No es software libre, aunque su descarga es gratuita. No se le pueden hacer correcciones o mejoras.

6.2.3. Cantaletras

La primera versión de este programa se publicó en 1999 como software privativo. Posteriormente, en 2006, el Centro de Desarrollo de Tecnologías

de Inclusión (CEDETi) de la Pontificia Universidad Católica de Chile lo reescribió en C++ con las librerías Qt, publicándolo como software libre bajo licencia GNU GPL 2 [28] (figura 6.7).

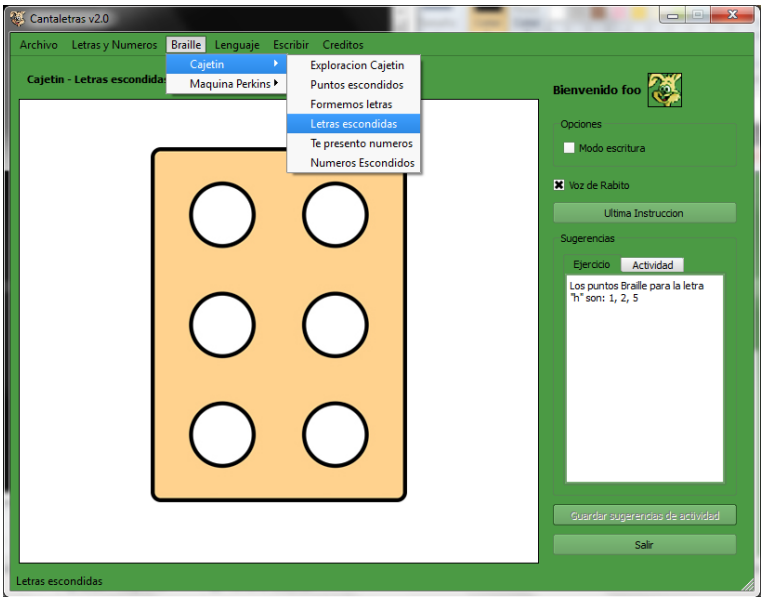


Figura 6.7: Captura de Cantaletras [28]

Cantaletras está orientado principalmente a niños ciegos o con baja visión, de entre 4 y 10 años. Tiene como objetivo permitirles traducir a sus modalidades sensoriales intactas (tacto y oído) la experiencia lectoescritora del vidente, apoyando la enseñanza de lectura y escritura inicial a través de actividades lúdicas que permiten la enseñanza del sistema Braille y el uso del teclado.

El software tiene una filosofía de inclusión, siendo su diseño atractivo para el trabajo conjunto de personas con discapacidad visual y personas videntes (figura 6.8).

Todas las actividades están diseñadas en forma de juego, y permite ejercitar e la escritura Braille a través de las modalidades cajetín y máquina Perkins. Asimismo, las actividades se enfocan al conocimiento de las letras y a la ubicación espacial de las letras en el teclado del computador.

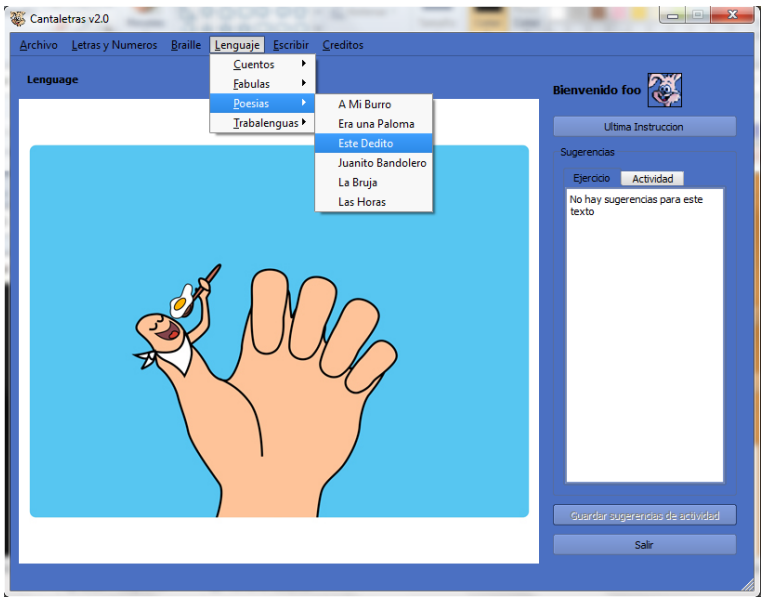


Figura 6.8: Captura de Cantaletras [28]

6.2.3.1. Inconvenientes

- Para el usuario no alfabetizado, carece de método pedagógico para aprender a escribir: no enseña a combinar las letras. Es decir, el niño podrá aprender el nombre de la G (ge) o de la R (erre), pero no sabrá escribir palabras como «guinda» o «pera».

- Para el usuario alfabetizado, no presenta las letras una a una en un orden sistemático (por frecuencia o alfabético, por ejemplo), sino que todas están disponibles desde el primer momento. Tiene un interesante juego de letras escondidas, pero no refuerza las que no se recuerden bien.

- No simula una máquina Perkins de forma realista: no hay que pulsar las teclas de forma simultánea, sino que se pulsan de una en una y al acabar se pulsa Intro.

- Internacionalización limitada: incluye un gestor de medios que facilita cambiar los ficheros de sonido e imagen, pero no es posible traducir los menús u otras partes del programa. Por ello el CEDETi ha tenido que publicar la versión en inglés como un programa diferente (Jumping Letters).

- Deficiente usabilidad. El programa pretende hacer muchas cosas muy diferentes: cuentos, poesías y trabalenguas, enseñar Braille, un editor

de texto... Muchas opciones, que con frecuencia resultan confusas (figura 6.9).

- Aunque formalmente es software libre, su código no esta disponible de forma inmediata, sino que hay que solicitarlo. A diferencia de lo que es habitual en el software libre, no está desarrollado de forma colaborativa en un sistema de control de versiones como Git o Subversion, sino por un equipo cerrado. La última versión es de 2009, y no se sabe si sigue en desarrollo.

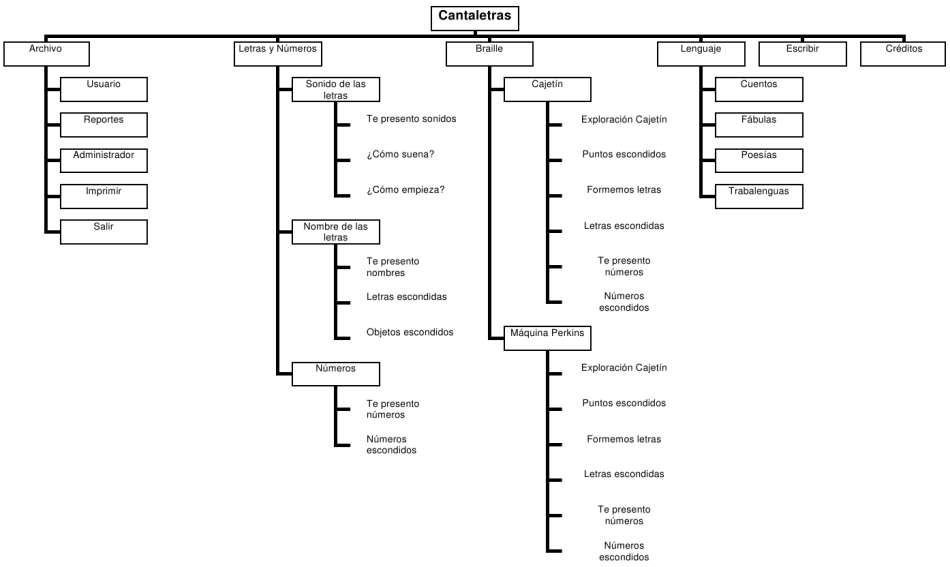


Figura 6.9: Menús de Cantaletras [155]

Capítulo 7

Definición de requisitos

Resumen: En este capítulo se recoge la especificación de requisitos de software para el entrenador de Braille para niños.

7.1. Introducción

A partir de las necesidades planteadas por las profesionales del sector (véase la sección 1.1) y del estudio previo del estado del arte (sección 6.2), se especificaron los requisitos a satisfacer por la aplicación, para que ésta cubriera la solicitud recibida solventando los puntos débiles de los programas existentes.

7.1.1. Propósito

La presente especificación de requisitos de software identifica y describe cada uno de los componentes de Nela, un entrenador de Braille para niños con discapacidad visual (ceguera o visión reducida).

Nela es un desarrollo que forma parte del Trabajo de Fin de Carrera del autor, estudiante de Ingeniería Técnica en Informática de Gestión en la Escuela Universitaria Politécnica de Teruel (Universidad de Zaragoza). Dicho TFC esta dirigido por la doctora Inmaculada Plaza, quien actúa como cliente de este proyecto.

7.1.2. Audiencia a la que se dirige, y sugerencias de lectura

Este documento se dirige tanto al estudiante que desarrolla el producto como al cliente del proyecto. En él se establecen las bases sobre las que se procederá al diseño y posterior construcción del producto.

Está estructurado en las siguientes secciones:

- Sección 2. Descripción global.
- Sección 3. Características del producto: descripción general del sistema a desarrollar, con el fin de indicar los factores que afectan al producto y sus requisitos.
- Sección 4. Requisitos de la interfaz. Describe las características lógicas de cada interfaz entre el producto software y los usuarios.

Se recomienda leer primero la sección 2, y después leer secuencialmente el resto de las secciones, o bien dirigirse a la sección concreta que resulte de interés.

7.1.3. Alcance del proyecto

Este proyecto producirá una aplicación multiplataforma para la enseñanza-aprendizaje de la escritura en código Braille para niños con discapacidad visual.

Objetivos:

- Proporcionar un programa que sea efectivo en la enseñanza-aprendizaje de la escritura en código Braille, fundamentado en sólidos principios pedagógicos.
- Conseguir que el discente sea autosuficiente, y pueda utilizar eficazmente el programa sin la supervisión de un adulto.
- Lograr que la aplicación sea atractiva para niños que mantengan un resto de visión.
- Obtener una aplicación flexible, que el docente pueda adaptar a sus necesidades particulares.
- Facilitar su uso en el sistema operativo libre GNU/Linux, además de sobre Microsoft Windows.
- Permitir la distribución y uso del programa, mediante una licencia de software libre.

7.1.4. Definiciones, acrónimos y abreviaturas

Braille Sistema de escritura para ciegos que consiste en signos dibujados en relieve para poder leer con los dedos.

Máquina Perkins Dispositivo mecánico similar a una máquina de escribir que permite producir Braille sobre papel.

7.1.5. Referencias

Esta especificación se ha escrito en conformidad con la IEEE *Recommended Practice For Software Requirements Specifications* (IEEE Std 830-1998). El contenido se ha creado a partir de las conversaciones mantenidas entre el desarrollador, el cliente y dos profesoras del sector.

7.2. Descripción global

7.2.1. Perspectiva del producto

La lectura del Braille al tacto requiere de un largo tiempo, particularmente en adultos, hasta que las yemas de los dedos desarrollan la sensibilidad necesaria. En cambio, la escritura (y la lectura visual) es mucho más fácil y se puede adquirir antes que la lectura.

Tradicionalmente, el Braille se ha escrito por medio de una regleta y un punzón, con el que se perforaba la hoja de papel. El niño debía aprender a escribir los caracteres del Braille volteados horizontalmente, para que al otro lado los puntos en relieve estuvieran correctamente colocados.

La máquina Perkins reproduce las letras sobre el papel tal y como se leen, de modo que el usuario puede leer inmediatamente lo escrito. Necesita de buena coordinación interdigital y bimanual: para formar cada letra se deben presionar simultáneamente las teclas correspondientes a los puntos que la constituyen.

Nela introducirá a los niños en el manejo del ordenador enseñándoles a escribir en código Braille tal y como se hace en una máquina Perkins, lo que posteriormente les permitiría trabajar con programas comerciales existentes actualmente en el mercado.

7.2.2. Características del producto

El niño colocará las manos sobre la fila central del teclado, con los dedos índices sobre las teclas F y J, que tienen unas marcas en relieve (si el niño tuviera dificultades para notar ese relieve, se puede pegar un trozo de fieltro sobre las teclas). Las teclas S-D-F y J-K-L corresponden respectivamente a los puntos 3-2-1 y 4-5-6 del cajetín Braille. Para escribir una letra se pulsan simultáneamente las teclas correspondientes a los puntos que la componen en el código Braille.

Nela solicitará al niño que escriba una palabra, y le comunicará, tanto de forma visual como sonora, si la introducción ha sido correcta o no, mostrando en la parte inferior de la pantalla la palabra escrita en Braille. El programa introduce automáticamente una nueva palabra una vez que el niño escribe habitualmente de forma correcta las palabras presentadas hasta el momento. Las palabras ya vistas seguirán presentándose ocasionalmente

para repasarlas. Cuando se cometa un error en una palabra, ésta será presentada con mayor frecuencia, para reforzarla. Del mismo modo, las palabras que no presenten dudas aparecerán más raramente.

7.2.3. Clases de usuario y características

Se considerará la existencia de dos tipos de usuario: niños que presenten ceguera completa, y niños con visión reducida. No se requerirán previos conocimientos de lectoescritura, pero sí haber alcanzado la madurez necesaria para abordar la capacidad lectoescritora: conceptos espaciales básicos (arriba/abajo, izquierda/derecha), nociones de cantidad (primeros números), coordinación dígito-manual, etc. Para adquirir estas habilidades se pueden usar materiales como el *Método Alameda* [62] o el programa informático «El toque mágico» [29].

No obstante, su utilización no se limitará a este público objetivo: podrá ser utilizado igualmente con niños sin discapacidad visual a los que se desee introducir en la enseñanza del código Braille, y por jóvenes o adultos con pérdida de visión debido a algún accidente o enfermedad.

7.2.4. Entorno operativo

El software se podrá ejecutar al menos sobre el sistema operativo Gnu/Linux) y sobre Microsoft Windows. No hará gestión de usuarios, sino que aprovechará la propia del sistema operativo.

A nivel de hardware requerirá de un equipo de sobremesa o portátil de gama baja, pues sus requisitos de procesador y memoria son reducidos para los estándares actuales. El equipo deberá estar dotado de altavoces.

Algunos teclados, particularmente en *netbooks*, no emiten correctamente los eventos de liberación de teclas cuando hay varias teclas pulsadas a la vez. Ante un hardware con esta limitación, la única solución un teclado diferente.

7.2.5. Restricciones de diseño e implementación

- En la medida en que sea posible, el software será conforme a la norma de accesibilidad ISO 9241-171:2008 (capítulo 5).

7.2.6. Documentación de usuario

Se redactará un manual de usuario, que estará disponible en formato PDF y/o HTML en la web del producto.

7.3. Requisitos funcionales

RQ.01 El programa presentará una palabra al niño, indicándole que la escriba usando la fila central del teclado como una máquina Perkins.

RQ.02 A continuación el programa comunicará si la introducción ha sido correcta o no.

RQ.03 Cuando el niño escriba habitualmente de forma correcta las palabras presentadas hasta el momento, el programa introducirá automáticamente una nueva palabra.

RQ.04 La secuencia de palabras introducirá paulatinamente nuevas sílabas y nuevas letras, con dificultad progresiva.

RQ.05 Las palabras ya vistas seguirán presentándose ocasionalmente para repasarlas.

RQ.06 Cuando se cometa un error en una palabra, ésta será presentada con mayor frecuencia, para reforzarla. Del mismo modo, las palabras que no presenten dudas aparecerán más raramente.

RQ.07 El programa hará uso de un vocabulario con significado afectivo y vivencial para el niño.

RQ.08 El docente podrá modificar la secuencia de palabras, alterando su orden, añadiendo o suprimiendo palabras.

RQ.09 El docente podrá sustituir las imágenes y sonidos empleados al presentar las palabras.

RQ.10 El programa estará internacionalizado (preparado para traducirse a otros idiomas).

RQ.11 Cada vez que se inicie el programa se reanudará la práctica desde el punto alcanzado en la sesión anterior.

7.4. Requisitos de la interfaz

RQ_I.1 La interfaz deberá ser especialmente sencilla, con el mínimo número de opciones posible.

RQ_I.2 Cada palabra irá acompañada de una imagen, para hacer el programa más atractivo a niños con resto de visión.

RQ_I.3 Las imágenes deberán ser sencillas (tipo *clip-art*), de gran tamaño y colores con contraste.

RQ_I.4 Toda comunicación con el usuario se realizará tanto de forma visual como sonora.

RQ_I.5 El programa no precisará del uso del ratón: todas sus características estarán accesibles vía teclado.

7.4.1. Boceto de una posible interfaz

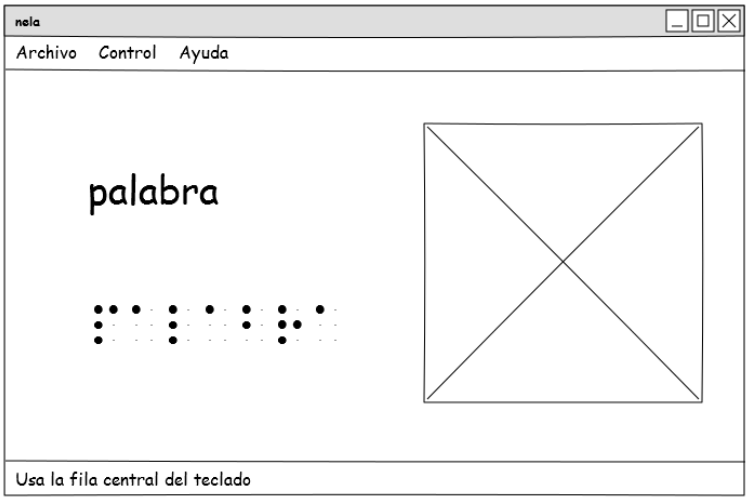


Figura 7.1: Boceto de una posible interfaz

Capítulo 8

Tecnologías y herramientas empleadas

Resumen: En este capítulo se abordan las cuestiones referentes a las tecnologías usadas: lenguaje de programación elegido, entorno de desarrollo, herramientas utilizadas, etc.

Todo el software empleado en la elaboración del este Trabajo de Fin de Carrera ha sido software libre: desde el compilador del código, hasta la edición de esta memoria. La única excepción ha sido el sistema operativo Microsoft Windows, que obviamente hubo que utilizar para producir la versión de Nela para él.

8.1. Lenguaje de programación

Uno de los requisitos de diseño fue que el software debía ser multiplataforma, y funcionase tanto en Microsoft Windows como en Gnu/Linux. Se pueden distinguir tres maneras diferentes de escribir software multiplataforma:

1. Haciendo uso de lenguajes interpretados, como Python.
2. Mediante compilación a bytecode intermedio, que será ejecutado en una máquina virtual, como Java.
3. Escribiendo el código fuente en un lenguaje estandarizado, que se compilará a código nativo en cada plataforma.

Cada método tiene sus ventajas y desventajas, tales como rapidez de ejecución y comodidad para el usuario o el desarrollador. En este caso se optó por la tercera opción, y desarrollar en C++ , por las siguientes razones:

Portabilidad C++ es un lenguaje programación estandarizado (Iso/Iec 14882:2011). Hay numerosas implementaciones del compilador, disponibles para prácticamente cualquier sistema operativo y hardware. Por ello, un programa escrito en código estándar será independiente de la plataforma, y el programa compilado se ejecutará a velocidad nativa. El lenguaje acepta múltiples paradigmas, tales como programación procedimental, orientada a objetos, funcional o genérica.

Bibliotecas externas C++ es un lenguaje con una larga historia y muy popular en la industria por ser prácticamente tan rápido como C, pero permitiendo un mayor nivel de abstracción. Por esta razón existen innumerables bibliotecas para facilitar la programación en este lenguaje. Para crear aplicaciones gráficas se ha de usar un *widget toolkit* como gtkmm, Fltk, Fox toolkit o Juce (o Swing en Java). Nela hace uso de las bibliotecas Qt que, además de proporcionar componentes gráficos, conforman una alternativa, más amplia, a la *Standard Template Library* de C++ . De hecho, gracias a su arquitectura modular, se pueden emplear para escribir aplicaciones no gráficas. Además de unas clases para interfaces gráficas con una buena integración con el entorno del usuario, las Qt proporcionan soporte para Unicode, *signals* y *slots* (una implementación del patrón de diseño *Observer* o *publish-subscribe*) y facilitan la recolección de basura.

Figura 8.1: Logotipo de las bibliotecas Qt [43]

Se ha comprobado el correcto funcionamiento del programa sobre Gnu/Linux y Microsoft Windows 7. Debería ser bastante sencillo adaptarlo también a Mac OS X. Además del sistema operativo, otros aspectos a considerar son el tamaño de palabra y la *endianness*. Nela se ha probado sobre 32 y 64 bits. Aunque sólo se disponía de sistemas *Little Endian* en los que probar, no se hacen operaciones que puedan suponer un problema en sistemas *Big Endian*.

8.2. Entorno de desarrollo

Para editar el código se suele utilizar un IDE (*Integrated Development Environment* o entorno de desarrollo integrado). Existen numerosos IDE para C++ en el mercado, tales como Eclipe, NetBeans, Microsoft Visual C++ Studio, etc. Se ha optado por Qt Creator, que es un IDE de manejo sencillo pero que incorpora todas las funcionalidades que se esperan de un IDE: gestión de proyectos (qmake/cmake), control de versiones (svn, cvs, git), integración con un depurador, etc. Qt Creator está optimizado para trabajar con las bibliotecas Qt (el propio programa está escrito con ellas), y está disponible tanto para GNU/Linux como para MS Windows.

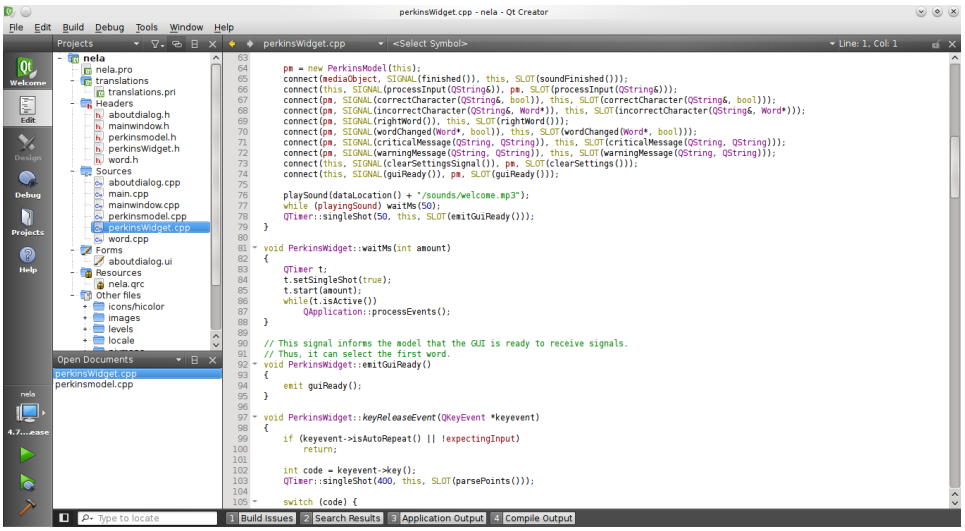


Figura 8.2: Captura de Qt Creator [43]

Como se ha indicado, hay muchos compiladores de C++ , como clang, Intel C++ Compiler, etc. Se ha usado el compilador de C++ de GNU (g++) en ambos sistemas, si bien se podría usar cualquiera de los aceptados por Qt. Adicionalmente se ha usado krazy para hacer una comprobaciones estáticas básicas del código (hay productos más completos, como cppcheck, Coverity, PVS-Studio o Valgrind).

8.3. Gestión del código

Para gestionar la evolución del código se suele usar un sistema de control de código fuente (*Source Code Management* o SCM). Un sistema de control de versiones permite a varias personas trabajar a la vez sobre un mismo código. No sólo almacena el código, sino también su historial, de modo que es posible ver quién y cuándo ha realizado un determinado cambio, o volver atrás al

estado en que se encontraba en cualquier momento. También se pueden crear ramas experimentales (y después fusionarlas con la rama principal o bien abandonarlas), ver los cambios introducidos entre dos versiones, etc.

Hay varias herramientas, tanto centralizadas (CVS, Subversion, Perforce...) como distribuidas (Bazaar, Git, Mercurial...). La participación en el VI Concurso Universitario de Software Libre imponía el uso de la forja de RedIRIS, que utiliza Subversion. De no ser así, probablemente se habría elegido Git.

8.4. Multimedia

Para editar las imágenes cuando ha sido necesario, se ha usado GIMP (GNU *Image Manipulation Program*), un programa que sirve para procesar gráficos y fotografías digitales en mapa de bits (JPEG, PNG, GIF, etc). Para este trabajo sólo ha sido necesario usar las funciones básicas (cambio de tamaño, recorte, etc), pero GIMP tiene características muy avanzadas. Cuenta además con un sistema de *plug-ins* y extensiones para ampliar el programa, como por ejemplo para la manipulación de fotografías en formato RAW.

Para la grabación y edición del audio se usó Audacity. Audacity puede grabar de un micrófono, la línea de entrada u otras fuentes (hasta 16 pistas a la vez) y crear grabaciones multipista. Importa y exporta en diversos formatos: Ogg Vorbis, MP2 y MP3 (mediante el codificador LAME), WAV, AIFF, AU, etc. Tiene una interfaz de edición muy sencilla, en la que se puede cortar, copiar, pegar y borrar, o aplicar efectos (ecualizar, eliminar ruidos, ajustar volúmenes, etc).

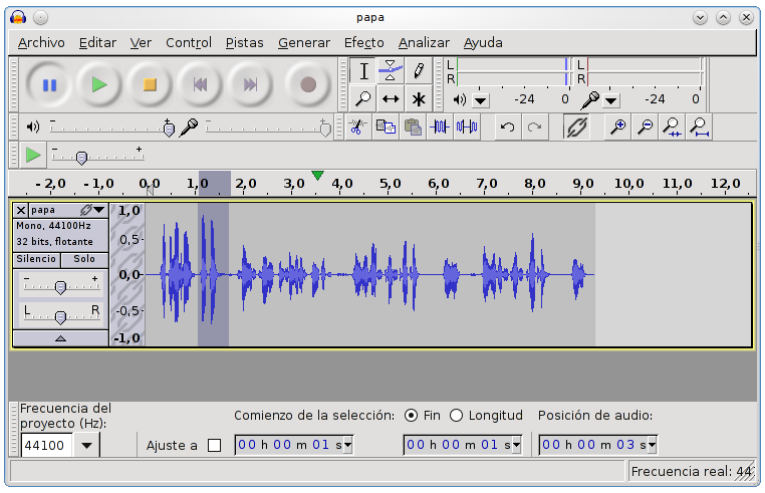


Figura 8.3: Captura de Audacity [166]

Las imágenes y sonidos empleados en el programa están en el dominio

público o tienen una licencia compatible con la GNU GPL 3. Fueron obtenidos de webs como openclipart.org y freesound.org. Desgraciadamente no se pudieron utilizar las imágenes del Portal Aragonés de la Comunicación Aumentativa y Alternativa (ARASAAC) [135] porque no son libres (usan la licencia Creative Commons BY-NC-SA), lo que imposibilita su uso en software GPL o sitios web como la Wikipedia (véase la sección 4.4.6.1).

8.5. Instalador

Si bien el código fuente de Nela se puede compilar en diversos sistemas, éste no es un método accesible a los usuarios finales. Desde la página web se puede descargar un instalador para Windows, que proporciona una instalación guiada que no ofrece ninguna dificultad. Hay diversos programas para crear instaladores para Windows (InstallShield, NSIS, WiX, etc). Se eligió Inno Setup, que es software libre, tiene todas las funcionalidades requeridas (como la capacidad de instalar tipos de letra) y está razonablemente documentado.

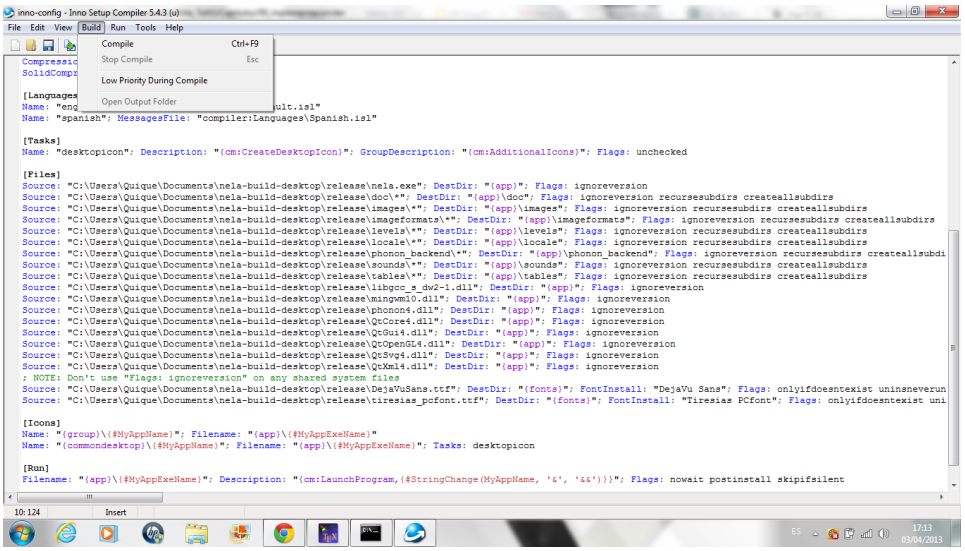


Figura 8.4: Captura de Inno Setup Compiler [154]

También se facilita un paquete `.deb` para sistemas GNU/Linux basados en Debian, como Ubuntu. La estructura de un paquete debian son las fuentes del programa y un directorio `debian` con ciertos ficheros específicos para el paquete. Algunos ficheros comunes son:

debian/copyright contiene la licencia del paquete.

debian/changelog mantiene el registro de cambios del paquete. Lista tanto

los cambios del programa como del propio paquete. Esta en formato GNU ChangeLog.

debian/control contiene la información que usará el gestor de paquetes (apt-get, aptitude, synaptic...), las dependencias para compilar el programa, los datos de la persona que mantiene el paquete, etc.

debian/rules es la esencia del paquete. Describe qué hacer an ciertas acciones (como compilarlo, qué hacer al instalarlo o desinstalarlo, etc).

Para crear el paquete se usaron las herramientas `dh_make` y `debhelper`. Se puede encontrar información detallada sobre la creación de paquetes Debian en la *Guía del nuevo desarrollador de Debian* [146].

Hermes Ojeda Ruiz, de México, conoció el programa a través de los medios de comunicación, y lo empaquetó para la distribución Chakra GNU/Linux en el repositorio CCR (*Chakra Community Repository*).

8.6. Documentación

Esta memoria se escribió usando LaTeX. TeX es un conjunto de herramientas creadas por Donald Knuth para procesar texto, muy utilizadas en entornos científicos (matemáticas, física e informática) ya que permiten escribir fórmulas de manera sencilla y obtener un acabado profesional. Es el formato preferido para escribir artículos científicos. LaTeX, creado por Leslie Lamport, añade ciertas características que permiten aumentar la potencia y sencillez de TeX. Una vez creado un fichero en TeX, es muy fácil obtener ficheros definitivos en formato PDF. Para gestionar la bibliografía se empleó BibTeX.

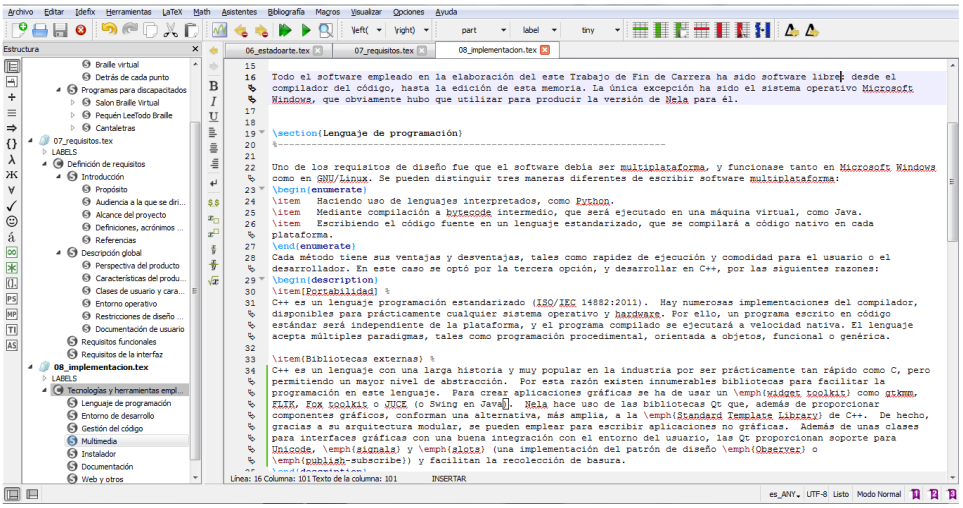


Figura 8.5: Captura de TeXstudio [174]

Para editar el código LaTeX se utilizó TeXstudio, un editor multiplataforma con una interfaz similar a Texmaker. Proporciona plegado de código, realce de sintaxis, corrección ortográfica, etc. Está escrito en C++ con las bibliotecas Qt.

8.7. Sistema operativo

El desarrollo se ha realizado en un sistema Debian. A lo largo del desarrollo se ha hecho uso de muchas de las herramientas disponibles en un sistema Unix (grep, etc). Como entorno de escritorio se usó Kde.

Capítulo 9

Pruebas, propuestas de mejora y líneas de trabajo futuro

Resumen: En este capítulo se expone el resultado de las primeras pruebas piloto, y las ampliaciones que se van a realizar o que podría ser interesante implementar.

A raíz de los premios recibidos, el proyecto apareció en diversos medios (prensa escrita, entrevistas radiofónicas, noticias en la web...). Gracias a esta repercusión fue descargado por profesionales de varios puntos de la geografía estatal, algunos de los cuales tuvieron la amabilidad de escribir mensajes de correo con sus impresiones.

Ixabel de Andrés, profesora del Creena (Centro de Recursos de Educación Especial del Gobierno de Navarra) probó uno de los primeros prototipos (v0.3.7), reportando que le parecía interesante y que creía que le sería de utilidad para uno de sus alumnos. Como inconveniente informó de una dificultad: cada vez que se iniciaba el programa había que empezar desde el principio.

Reconociendo que esta funcionalidad era necesaria, fue añadida al siguiente prototipo. Actualmente Nela guarda hasta que palabra de la secuencia ha llegado el estudiante, y el grado de dominio alcanzado en cada una de las palabras vistas hasta el momento.

Miguel Martín, instructor tiflotécnico de la Once en Valencia, probó el programa encontrándolo muy atractivo, y haciendo tres comentarios:

En primer lugar indicó que la grabación que dice los puntos o las palabras que se deben pulsar se oye poco. Lo cierto es que los sonidos se grabaron con el micrófono interno de un ordenador portátil, en el dormitorio del proyectista. Habría que volver a grabarlos con un micrófono de calidad, y en un estudio insonorizado. Sería perfecto si los pudiera realizar un actor o locutor, con una voz entrenada.

En segundo lugar, aunque le gustaba la grabación de aprobación al escribir correctamente una palabra, opinaba que quizá sería bueno acortar los

aplausos un poco, dado que al estar mucho rato con el programa podrían lleguen a cansar.

La observación me parece acertada, pero creo que otra posibilidad mejor seria contar con varios sonidos de aprobación diferentes, para que no suene siempre el mismo. La implementación a nivel de código sería bastante sencilla, y la mayor dificultad sería la obtención/grabación de los nuevos sonidos.

Por último, planteaba la posibilidad de crear también una versión de Nela para Mac OS X, argumentando que Apple lleva accesibilidad integrada en todos sus productos, y que cada vez hay más gente ciega que usa los productos de Apple.

No teniendo acceso inmediato a un ordenador Mac, no ha sido posible crear esta versión. No obstante, actualmente el código debería compilar en Mac sin necesidad apenas de alguna modificación. La mayor dificultad sería adquirir los conocimientos necesarios para crear el paquete instalador. Para una persona con experiencia en desarrollo sobre Mac OS X, preparar la versión de Nela para Mac apenas supondría trabajo.

Pilar Miró, que trabaja con niños ciegos desde 1989 en la ONCE, explicó que, en principio, el programa le parece interesante porque los niños pequeños tienen poca independencia digital y fuerza, y suele fatigarles el trabajo con la máquina Perkins. La posibilidad de avanzar en la escritura mediante el teclado le resulta tentadora, pero de entrada le parece complicado que estos mismos niños coordinen suficientemente bien la presión simultánea de las teclas.

No ha confirmado este extremo probando el programa con niños, pero si así fuera, dos posibles soluciones serían: a) dar más tiempo para pulsar las teclas (que la simultaneidad no sea tan precisa). b) ir pulsando las teclas -sin simultaneidad-, y cuando se hayan pulsado todas, pulsar Espacio.

Así mismo informó de problemas al escribir la palabra «pala». Esta palabra es la primera de la secuencia que incluye una letra (P) que se escribe en Braille con cuatro puntos (1-2-3-4). Se ha observado que algunos teclados (particularmente en netbooks) no generan correctamente los eventos de liberación de teclas cuando hay muchas teclas pulsadas simultáneamente. Es una limitación hardware que afecta también a otros programas comerciales que simulan una máquina Perkins.

Lucía Esteva, profesora especializada en alumnos con discapacidad visual, encontró el programa muy práctico, pero tropezó con el mismo problema. La única solución es conectar otro teclado. Se ha añadido esta información al manual de usuario.

Manuel Valentín, técnico en Tiflotecnología y Braille de la ONCE, valoró el programa para su uso por parte de los alumnos que comienzan el aprendizaje del Braille. A raíz de esta valoración, y cómo colaborador de la web www.compartolid.es, publicó en ella un artículo en el que, además de dar a conocer el proyecto y su utilidad, también aportó una valoración de aspectos

que tienen que ver con su accesibilidad, y que podrían ser de interés para futuras versiones.

Hace notar que no es posible configurar las teclas que simulan las pulsaciones Braille. Esto podría ser útil para algunos usuarios, pero sobre todo sería necesario para poder usar el programa en países que tienen otra distribución del teclado, como Francia.

Una interesante propuesta es añadir un modo Braille para usuarios con problemas motores y que sólo pueden usar una mano. Multiteclado Braille tiene esta opción, que funciona usando las mismas teclas de una de las manos, y dividiendo la escritura de una letra en dos pasos: primero la parte izquierda del cajetín y después la segunda.

Por ejemplo, en el caso de un usuario que sólo disponga de la mano izquierda, en el primer paso las teclas s-d-f corresponderían los puntos 3-2-1, y en el segundo paso a los puntos 6-5-4. Un semicajetín vacío se genera con la tecla g. Así, para generar una «n» (puntos 1-3-4-5), primero se pulsan las teclas f y s, se sueltan y en el segundo paso se pulsan las teclas f y d. Para generar una «a» (punto 1) se pulsa la tecla f, y a continuación la g.

Estas modificaciones son de mayor complejidad, y podrían ser la base de otro TFC. Otra línea de trabajo futuro, y que podría favorecer la integración en el aula, sería añadir un modo para ayudar a niños videntes a aprender a escribir, usando los caracteres latinos habituales y el teclado completo.

Capítulo 10

Conclusiones

Resumen: En este capítulo se reflexiona sobre las conclusiones generales del desarrollo del proyecto, se analizan los retos afrontados por el alumno y se termina con una pequeña conclusión a nivel personal.

10.1. Conclusiones generales

La primera conclusión que se extrae tras la realización de este proyecto es que se han cumplido con creces todos los objetivos marcados inicialmente. Se ha desarrollado una herramienta útil para el aprendizaje de la escritura mediante el código Braille, flexible y fácil de adaptar a diferentes metodologías, independiente del sistema operativo y capaz de funcionar en distintos idiomas. La herramienta ha sido evaluada por varios profesionales del campo, quienes la han valorado de forma muy positiva.

Adicionalmente el proyecto ha obtenido tres galardones:

- El primer premio en la fase final del VI Concurso Universitario de Software Libre, en el que compitieron a nivel estatal 99 proyectos de un total de 139 estudiantes.

- El premio especial de accesibilidad de dicho concurso.

- El primer premio en la fase local de Aragón, organizada por la Oficina de Software Libre de la Universidad de Zaragoza.

Figura 10.1: En Sevilla, con el personal de la ONCE que entregó el premio especial de accesibilidad.

Las agencias EFE y Europa Press difundieron la noticia de estos reconocimientos, que fue recogida por numerosos medios locales, nacionales y extranjeros, pudiéndose citar entre otros:

- Heraldo de Aragón
- El Periódico de Aragón
- Telecinco
- ABC
- 20 minutos
- El Diario Vasco
- El Norte de Castilla
- numerosas webs de noticias, incluso en otros países como Venezuela o Argentina.

El Diario de Teruel dedicó una página completa al proyecto.
También se efectuaron varias entrevistas radiofónicas:

- Onda Cero Teruel
- Gente de Aragón (Onda Cero Radio).

- La ventana Aragón (Cadena SER).

Y por supuesto tuvo repercusión en diversas webs especializadas, tales como

- Servicios Sociales ONCE Aragón.
- Observatorio de la Discapacidad, portal del Ministerio de Sanidad, Política Social e Igualdad.
- Centro Nacional de Referencia de Aplicación de las TIC basadas en fuentes abiertas, portal del Ministerio de Industria, Energía y Turismo.
- Aragón Investiga, portal del Gobierno de Aragón.
- Cátedra Telefónica-Universidad de Alicante Impacto de las Tecnologías del Lenguaje Humano en la inclusión social.
- Clúster de Entidades pro Software Libre de Aragón
- Barrapunto, de donde saltó a otras webs y blogs como Madres y Padres

Para finalizar, cabe destacar que, junto con mi directora y la profesora Nuria Tregón, he escrito un artículo que será publicado en el próximo número de *Novática*, de inminente aparición. *Novática* es la revista bimestral de la Asociación de Técnicos de Informática. Está incluida en numerosos catálogos e índices, nacionales e internacionales, de publicaciones técnicas y científicas, entre ellos los de muchas universidades de todo el mundo, especialmente españolas e iberoamericanas. Entre los índices más conocidos en los que aparece Novática están dblp.uni-trier.de (Alemania), CIRS (Francia), Scienceineurope.net, ICYT, Dialnet (España) y el principal catálogo en lengua española, Latindex (México). El artículo puede consultarse en el apartado de Anexos.

10.2. Retos afrontados

Para llevar a cabo este proyecto, he tenido que afrontar una serie de retos o problemas. A continuación se explican tanto los problemas como la forma en que se han afrontado.

10.2.1. Tecnologías desconocidas

En primer lugar, tuve que familiarizarme el lenguaje de programación C++ , que no se toca en ningún momento a lo largo de la carrera. Para ello he utilizado el libro *A Complete Guide to Programming in C++* [87]. C++ tiene fama de ser un lenguaje grande y complicado, pero no sospechaba hasta que

punto. Para dar una idea, se puede señalar que el libro *The C Programming Language* tiene 274 páginas, mientras que *The C++ Programming Language* es cinco veces mayor (1368 páginas).

Para realizar la interfaz gráfica, he tenido que aprender a usar las bibliotecas Qt, ayudándome del libro *An Introduction to Design Patterns in C++ with Qt*. Las Qt también han sido útiles para resolver otros problemas, como detectar la pulsación simultánea de hasta 6 teclas, la internacionalización del programa o mostrar caracteres Braille en pantalla.

Adquirir estos conocimientos requirió de un largo tiempo de estudio, antes siquiera de que pudiera empezar a trabajar en el proyecto, pero considero que el resultado final demuestra que mereció la pena.

Para escribir la documentación, he profundizado en el sistema de documentación LaTeX. En este caso no partía de cero, pues ya había escrito algún documento breve con él, pero escribir un trabajo de esta envergadura, con capítulos, referencias cruzadas, bibliografía, caracteres japoneses, etc ha sido bastante más complejo.

10.2.2. Materias ajenas a la informática

Para lograr un programa que realmente fuera útil, fue necesario estudiar la didáctica de la lectoescritura, tanto para niños videntes como para discapacitados visuales. Comprender este proceso (que sorprendentemente no se estudia en la carrera de Magisterio) fue fundamental para que el programa no se quedara en un mero trabajo académico sin aplicación real.

Escribir la sección de perspectiva histórica con las distintas soluciones que se han propuesto a lo largo del tiempo para el acceso de los discapacitados visuales a la lectoescritura (sección 3.1) requirió de una verdadera labor de investigación. Esta parte de la historia estaba básicamente sin documentar, y los tímidos intentos hallados contenían errores e importantes omisiones. Se puede afirmar sin temor a exagerar que este trabajo es la exposición más completa y documentada jamás escrita.

10.2.3. Continuidad del desarrollo

El reto final es que Nela siga evolucionando tras la presentación de este TFC. Se ha hecho todo lo necesario para que esto sea posible: licencia libre, desarrollo público en una forja [106], blog informativo [105], etc. El desarrollo está abierto a cualquier persona interesada: educadores que propongan mejoras, desarrolladores que deseen implementarlas, traductores que adapten el programa a otros idiomas, ilustradores que faciliten dibujos libres... Se pretende, de este modo, que la presente versión de Nela sea un punto de partida y un lugar de encuentro y colaboración.

10.3. Conclusión personal

A nivel personal, este proyecto ha sido una experiencia inesperadamente gratificante.

Por una parte, y como es de esperar para realizar un TFC, tuve que adquirir nuevos conocimientos, que posteriormente pueden resultar útiles en mi vida laboral. El tener que estudiar diferentes alternativas técnicas y tomar decisiones argumentables también es una experiencia valiosa.

Sin embargo, lo que lo hace emocionante es que profesionales de diversos puntos del estado deseen utilizar mi programa con sus niños. Realizar este proyecto ha tomado mucho más tiempo del que había pensado. Podría haber elegido un proyecto más sencillo para salir del paso y superar el trámite (por ejemplo presentar una de las webs en PHP que ya había programado), pero no me arrepiento en absoluto de la decisión tomada.

La participación en el VI Concurso Universitario de Software Libre también fue una gran experiencia, muy motivadora y educativa, que merece pena por sí sola. Los premios obtenidos, y la repercusión mediática subsiguiente, hacen que me sienta orgulloso de mi trabajo. Los dos días que pasé en Sevilla con los demás finalistas, los organizadores y los patrocinadores, fueron sencillamente fantásticos, un recuerdo que atesoraré siempre.

Como colofón, este proyecto me ha brindado la oportunidad de escribir por primera vez en una revista indexada (Novática), lo que es otro motivo de satisfacción, y quizás pueda resultar de ayuda en mi futuro profesional.

Mi reflexión final es que habría que recomendar a todos los estudiantes que se tomen el TFC en serio, haciendo un trabajo que sea útil en el mundo real, y animarles a publicarlo de forma libre.

Bibliografía

[1] Braillin, el muñeco que enseña braille. Disponible en `http://www.changemakers.com/es/node/129132/images`. (Último acceso, marzo 2013).

[2] The public's library and digital archive. Disponible en `http://www.ibiblio.org/pub/Linux/logos/platypus/`. (Último acceso, marzo 2013).

[3] *The Penny Cyclopaedia of the Society for the Difussion of Useful Knowledge*, vol. Volumen 4. 1835.

[4] The blind, their works and ways. *The Living Age*, (v. 40), 1854.

[5] *The Encyclopædia Britannica, or, Dictionary of arts, sciences, and general literature*. Número v. 4 en The Encyclopædia Britannica, Or, Dictionary of Arts, Sciences, and General Literature. Adam & Charles Black, 1854.

[6] *Congrès International pour l'Amélioration du sort des Aveugles ...: tenu à Bruxelles, du 6 au 10 août 1902*. Imprimerie de l'École Professionnelle de l'Institut de la Sainte Famille, Bruselas (Bélgica), Disponible en `http://www.archive.org/details/congrsinternat00roya`.

[7] Open Source Hardware (OSHW) definition. 2011. Disponible en `http://www.freedomdefined.org` (último acceso, marzo 2013).

[8] Agriculture et du Commerce, M. *Congrès universel pour lámélioration du sort des aveugles et des sourds-muets, tenu à Paris, du 23 au 30 Septembre, 1878*. Comptes rendus sténographiques des congrès internationaux. Impr. Nationale, Paris (Francia), 1879. Disponible en `http://www.archive.org/details/congrsuniverse00impr` (último acceso, marzo 2013).

[9] Amos Comenius, J. *Orbis sensualium pictus*. Núremberg (Alemania), 1658. Disponible en `http://www.hs-augsburg.de/~harsch/Chronologia/Lspost17/Comenius/com_o000.html` (último acceso, marzo 2013).

[10] ARMITAGE, T. R. *The education and employment of the blind*. 1871.

[11] ASOCIACIÓN ESPAÑOLA DE NORMALIZACIÓN Y CERTIFICACIÓN COMITÉ AEN/CTN 49. Envases y embalajes. Braille sobre envases y embalajes para medicamentos. 2011. UNE-EN 15823:2011.

[12] ASSOCIATION DU PETIT MUSÉE DU BRAILLE. Sitio web. Disponible en `http://petitmuseedubraille.free.fr`. (Último acceso, marzo 2013).

[13] ASTASIO TOLEDO, J. A., GONZÁLEZ PAREDES, P. y MARTÍNEZ LIÉBANA, I. *Método de alfabetización Pérgamo para personas adultas ciegas*. ONCE, Centro Bibliográfico y Cultural, Madrid, 1994.

[14] BARBOSA HELDT, A. *Cómo enseñar a leer y escribir*. Pax Mexico L.C.C.S.A., 2008. ISBN 9789688604342.

[15] BOIX HERNÁNDEZ, S., CODINA MIR, M., CORBELLA ROQUETA, M. T., MIRET SERRA, J., SOLÉ TORNÉ, A. y TAPIA MARTÍNEZ, I. Estratègies per a la inclusió de l'alumnat amb discapacitat visual. Disponible en `http://ateneu.xtec.cat/wikiform/wikiexport/cursos/escola_inclusiva/d232/index`.

[16] BOLAÑOS, B., CAMBRONERO, M. I. y VENEGAS, A. *Didáctica de la lecto-escritura I parte*. Euned, San José (Costa Rica), 1987. ISBN 9789977643663.

[17] BONNER, S. F. *Education in ancient Rome: from the elder Cato to the younger Pliny*. University of California Press, 1977. ISBN 9780520034396.

[18] BRADY, K. A braille tale: A historical perspective on Chinese braille and a proposed system of brailled characters. *The Harvard Undergraduate Research Journal*, vol. 4(2), 2011. Disponible en `http://thurj.org/ss/2012/01/3097/` (último acceso, marzo 2013).

[19] BREWSTER, D., OF EDINBURGH, R. S., JAMESON, R. y SOCIETY, W. N. H. *The Edinburgh Philosophical Journal*. v. 6. A. Constable & Company, 1822.

[20] BUENO MARTÍN, M. *Manual Digital de Signografía Braille*. I Congreso Virtual INTEREDVISUAL sobre el Sistema Braille, instrumento de acceso a la comunicación, la educación y la cultura de las personas ciegas, Málaga, Disponible en `http://www.juntadeandalucia.es/averroes/caidv/interedvisual/ftp/manual_digital_simb_braille.pdf`.

[21] FERNÁNDEZ DEL CAMPO, J. E. *Desafíos didácticos de la lectura braille*. ONCE, Madrid, 2001.

[22] FERNÁNDEZ DEL CAMPO, J. E. *Braille y matemática*. Guías (Organización Nacional de Ciegos Españoles). ONCE, 2004. ISBN 9788448401436.

[23] CANGA-ARGÜELLES CRUZ, E., MARTÍNEZ BELINCHÓN, P., SAHUQUILLO SAHUQUILLO, M. I. y GARCÍA GARCÍA, F. *Micho 1: Método de lectura castellana*. Bruño, 1989. ISBN 9788421604861.

[24] GONZÁLEZ DE CARA, T., SÁNCHEZ CRISOL, S. y SUÁREZ VILAR, M. *Adquisición y desarrollo de conceptos básicos*. ONCE, Madrid, 1997.

[25] CARRASCO, S. *Colección de aríticulos*. Biblioteca de autores uruguayos. A. Barreiro y Ramos, 1884. Disponible en `http://archive.org/details/coleccindeart00carr` (último acceso, marzo 2013).

[26] CARTON, C. L. *Le sourd-muet et l'aveugle*. Le sourd-muet et l'aveugle. Vandecasteele-Werbrouck, Brujas (Bélgica), 1837.

[27] CASTRO OROZCO, A. I. *Descripción gráfica y signografía braille del Código Matemático Unificado*. I Congreso Virtual INTERED-VISUAL sobre el Sistema Braille, instrumento de acceso a la comunicación, la educación y la cultura de las personas ciegas, Málaga, Disponible en `http://www.juntadeandalucia.es/averroes/caidv/interedvisual/ftp/cmu_88_98.doc`.

[28] CEDETI. Cantaletras. 2006. Disponible en `http://www.cedeti.cl/software-educativo/cantaletras/` (último acceso, marzo 2013).

[29] CEDETI. El toque mágico. 2008. Disponible en `http://www.cedeti.cl/software-educativo/el-toque-magico/` (último acceso, marzo 2013).

[30] CENTRO DE APOYO A LA INTEGRACIÓN DE DEFICIENTES VISUALES. *El niño ciego en la escuela*. Junta de Andalucía, 2 edición, 1997.

[31] CENTRO DE INVESTIGACIÓN, D. Sitio web. Disponible en `http://cidat.once.es`. (Último acceso, marzo 2013).

[32] CHEPAITIS, A. J. Alphanumeric font. 2005. Disponible en `http://www.google.com/patents/USD521054?hl=es` (último acceso, marzo 2013). Patente US D521,054.

[33] COLLIGAN, C. y LINLEY, M. *Media, Technology, and Literature in the Nineteenth Century: Image, Sound, Touch*. The Nineteenth Century Series. Ashgate, 2011. ISBN 9781409400097.

[34] COMISIÓN BRAILLE ESPAÑOLA. Sitio web. Disponible en http://www.once.es/new/servicios-especializados-en-discapacidad-visual/braille/comision-braille-espanola. (Último acceso, marzo 2013).

[35] COMISIÓN BRAILLE ESPAÑOLA. *Guías de la comisión Braille española: Signografía básica*. Guías de la Comisión Braille Española. Comisión Braille Española, ONCE, 2005. Disponible en http://www.once.es/serviciosSociales/index.cfm?navega=detalle&idobjeto=127&idtipo=1.

[36] COOPER, H. L. A brief history of tactile writing systems for readers with blindness and visual impairments. *See/Hear*, Disponible en http://www.tsbvi.edu/seehear/spring06/spring06.pdf.

[37] COX, A. Cathedrals, bazaars and the town council. 1998. Disponible en http://slashdot.org/features/98/10/13/1423253.shtml (último acceso, marzo 2013).

[38] CREATIVE COMMONS. Sitio web. Disponible en http://creativecommons.org. (Último acceso, marzo 2013).

[39] CRESPO, S. E. *La escuela y el niño ciego. Manual práctico*. Graficart, Córdoba (Argentina), 1980.

[40] CÉTARES SALAS, A., CORTÉS RIVERA, C. A. y SILVA OLARTE, L. F. *Sistema de enseñanza del código Braille para niños con limitaciones visuales*. Trabajo de grado, Facultad de Ingeniería, Pontificia Universidad Javeriana, Bogotá (Colombia), 2005.

[41] DEUTSCHES INSTITUT FÜR NORMUNG E. V NA 063-06-04 AA. Blindenschrift - anforderungen und maße. 2007. DIN 32976.

[42] DIBONA, C., OCKMAN, S. y STONE, M. *Open sources: voices from the open source revolution*. O'Reilly, 1999. ISBN 9781565925823. Disponible en http://www.oreilly.com/catalog/opensources/book/toc.html (último acceso, marzo 2013).

[43] DIGIA. Qt. Disponible en http://qt.digia.com. (Último acceso, marzo 2013).

[44] DIONYSIUS OF HALICARNASSUS y ROBERTS, W. R. *On Literary Composition: Being the Greek Text of the De Compositione Verborum*. Macmillan, Londres (Reino Unido), 1910. Disponible en http://www.archive.org/details/cu31924026465165 (último acceso, marzo 2013).

[45] DIXON, J. Eight-dot braille: A position statement of the Braille Authority of North America. Disponible en `http://www.brailleauthority.org/eightdot/eightdot.pdf`.

[46] DÍAZ SAEZ, Ó. *Entrenador para niños con discapacidad visual*. Trabajo de fin de carrera, EUPT, Universidad de Zaragoza, 2009.

[47] ESPEJO DE LA FUENTE, B. *El Braille en la escuela: una guía práctica para la enseñanza del Braille*. Guías (Organización Nacional de Ciegos Españoles). Organización Nacional de Ciegos Españoles, Madrid, 1993. ISBN 9788487277344.

[48] EUROPEAN CARTON MAKERS ASSOCIATION. Sitio web. Disponible en `http://www.ecma.org/`. (Último acceso, marzo 2013).

[49] EWING, L. Linux 2.0 penguins. Disponible en `http://isc.tamu.edu/~lewing/linux/`. (Último acceso, marzo 2013).

[50] EZQUERRA, W. *Nuevas letras: 1ª cartilla*. Rivadeneyra, S.A., Madrid, 1968.

[51] EZUST, A. y EZUST, P. *An Introduction to Design Patterns in C++ with Qt*. Prentice Hall Open Source Software Development Series. Pearson Education, Boston (Estados Unidos), 2011. ISBN 9780132851633.

[52] FAKOÓ, A. Alte und neue blindenschriften. Disponible en `http://www.fakoo.de/blindenschriften.html`. (Último acceso, marzo 2013).

[53] FARRELL, G. *The Story of Blindness*. Harvard University Press, 1956. Disponible en `http://books.google.es/books?id=HqocjtV44EwC`.

[54] FEDERACIÓN DE ASOCIACIONES DE EDUCACIÓN DE PERSONAS ADULTAS (FAEA). Metodologías globalizadoras para desarrollar las competencias básicas en el aprendizaje permanente. Disponible en `http://www.catedu.es/competencias`. (Último acceso, marzo 2013).

[55] FERREIRO, E. y TEBEROSKY, A. *Los sistemas de escritura en el desarrollo del niño*. Siglo XXI, México, 1979. ISBN 968-23-0426-1.

[56] FREE SOFTWARE FOUNDATION. Sitio web. Disponible en `http://www.fsf.org`. (Último acceso, marzo 2013).

[57] FREE SOFTWARE FOUNDATION. GNU General Public License, version 1. 1989. Disponible en `http://www.gnu.org/copyleft/copying-1.0.html` (último acceso, marzo 2013).

[58] FREE SOFTWARE FOUNDATION. Porting the Hurd to another microkernel. 2001–2011. Disponible en `http://www.gnu.org/s/hurd/history/port_to_another_microkernel.html` (último acceso, marzo 2013).

[59] FREE SOFTWARE FOUNDATION. Hurd. 2001–2013. Disponible en `http://www.gnu.org/software/hurd/hurd.html` (último acceso, marzo 2013).

[60] FREEBSD PROJECT, T. The FreeBSD Copyright. 1992. Disponible en `http://www.freebsd.org/copyright/freebsd-license.html` (último acceso, marzo 2013).

[61] LAFUENTE DE FRUTOS, Á. *Educación inclusiva: discapacidad visual*. Colección Formación en red / Instituto de Tecnologías Educativas. Ministerio de Educación, Cultura y Deporte. Subdirección General de Documentación y Publicaciones, 2011. ISBN 9788436951943.

[62] FUENTES HERNÁNDEZ, J. *Método Alameda: maduración lectoescritora para alumnos ciegos y deficientes visuales de tres a seis años*. Guías (Organización Nacional de Ciegos Españoles). ONCE, Departamento de Servicios Sociales para Afiliados, 1995. ISBN 9788487277993.

[63] GALL, J. *An account of the recent discoveries which have been made for facilitating the education of the blind: With specimens of the books, maps, pictures, &c. for their use*. published by James Gall, 1893.

[64] GARCÉS CASTILLO, M. A. y GARCÉS LÁZARO, B. Proyecto de investigación educativa sobre el método de lectura en sistema braille: "Seis Puntos". En *Actas del 1 Congreso estatal sobre prestación de servicios para personas ciegas y deficientes visuales*, páginas 289–298. ONCE, Madrid, 1996.

[65] GARCÉS LÁZARO, B. *Alborada: cartilla para el aprendizaje de la lectura*. ONCE, Madrid, 1969.

[66] GARCÍA DORADO, M. Sistemas de comunicación de personas sordociegas. En *La sordoceguera: un análisis multidisciplinar* (editado por D. Á. Reyes, J. A. A. Reyes, P. G. Viñas y E. R. Rey), Estudios (Organización Nacional de Ciegos Españoles. Departamento de Servicios Sociales para Afiliados). ONCE, Madrid, 2004. ISBN 9788448401429.

[67] GOOGLE, I. Android. Disponible en `http://www.android.com`. (Último acceso, marzo 2013).

[68] GROKLAW. The 1994 usl-regents of ucal settlement agreement - pdf and text. 2004. Disponible en `http://www.groklaw.net/article.php?story=20041126130302760` (último acceso, marzo 2013).

[69] HENRI, P., OSUNA, J. y FAJARDO, J. *La vida y obra de Luis Braille*. ONCE, Madrid, 1988. ISBN 9788440435811.

[70] HILL, B. A birthday that involves copyright infringement is an... unhappy birthday. Disponible en `http://unhappybirthday.com/`. (Último acceso, marzo 2013).

[71] HIMANEN, P. *La ética del hacker: y el espíritu de la era de la información*. Destino, 2002. ISBN 9788423336371.

[72] HOSKINS, W. Readme.impt.license.change. 1999. Disponible en `ftp://ftp.cs.berkeley.edu/pub/4bsd/README.Impt.License.Change` (último acceso, marzo 2013).

[73] ICKELSAMER, V. *Die rechte weis auffs kürtzist lesen zu lernen*. 1527.

[74] INSTITUTO NACIONAL PARA CIEGOS DE COLOMBIA. Sitio web. Disponible en `http://www.inci.gov.co`. (Último acceso, marzo 2013).

[75] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION TECHNICAL COMMITTEE 122. Packaging – Braille on packaging for medicinal products. 2013. ISO 17351:2013.

[76] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION TECHNICAL COMMITTEE 173. Communication aids for blind persons – identifiers, names and assignation to coded character sets for 8-dot braille characters – part 1: General guidelines for braille identifiers and shift marks. 2001. ISO/TR 11548-1:2001.

[77] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION TECHNICAL COMMITTEE 173. Communication aids for blind persons – identifiers, names and assignation to coded character sets for 8-dot braille characters – part 2: Latin alphabet based character sets. 2001. ISO/TR 11548-2:2001.

[78] IPLAND GARCÍA, J. *Percibo y trazo*. ONCE, Sevilla, 1985.

[79] IRWIN, R. B. *The war of the dots*. American Foundation for the Blind, 1933. Disponible en `http://www.afb.org/warofthedots/book.asp` (último acceso, marzo 2013).

[80] IZQUIERDO, D. y LLEIDA, I. Aprendizaje del braille: un enfoque constructivista. En *I Congreso Nacional de Educación y Personas con Discapacidad: conciencia, compromiso y mejora continua*, páginas 411–418. Gobierno de Navarra. Departamento de Educación y Cultura, Pamplona, 2003.

[81] IZQUIERDO NAJES, J. *Dispositivo de aprendizaje de código Braille basado en un microcontrolador PIC: comparación con el desarrollo basado en Arduino*. Trabajo de fin de carrera, EUPT, Universidad de Zaragoza, 2012.

[82] JAMESON, R. *The Edinburgh New Philosophical Journal*. 42. Disponible en `http://www.jmcvey.net/alphabets/index.htm`.

[83] JAMESON, R., ROYAL SOCIETY OF EDINBURGH y WERNERIAN NATURAL HISTORY SOCIETY. *The Edinburgh new philosophical journal: exhibiting a view of the progressive discoveries and improvements in the sciences and the arts*, vol. XXIII. A. and C. Black, 1837.

[84] JOHNSON, E. C. *Tangible typography: or, How the blind read*. J. Whitaker, 1853.

[85] KAMP, P.-H. Why should i care what color the bikeshed is? 1999. Disponible en `http://www.freebsd.org/doc/en_US.ISO8859-1/books/faq/misc.html#bikeshed-painting` (último acceso, marzo 2013).

[86] KAPUR, N. *The Paradoxical Brain*. Cambridge medicine. Cambridge University Press, 2011. ISBN 9780521115575.

[87] KIRCH-PRINZ, U. y PRINZ, P. *A Complete Guide to Programming in C++*. Jones and Bartlett computer science. Jones & Bartlett Learning, Sudbury (Estados Unidos), 2002. ISBN 9780763718176.

[88] KLAUWELL, A. y MARTIN, B. *Erstes Lesebuch*. F. Brandstetter, Leipzig (Alemania), 1908. Disponible en `http://archive.org/details/ersteslesebucher00klauuoft` (último acceso, marzo 2013).

[89] LESSIG, L. *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*. Penguin Press, 2004. ISBN 9781594200069. Disponible en `http://www.elastico.net/archives/001222.html` (último acceso, marzo 2013).

[90] LEVY, S. *Hackers: Heroes of the Computer Revolution*. Penguin Books, 2001. ISBN 9780141000510.

[91] LIESEN, B. El braille: origen, aceptación y difusión. *Entre dos mundos: Revista de traducción sobre discapacidad visual*, (19), Disponible en `http://www.once.es/appdocumentos/once/prod/SS-PUB-EDM-19.pdf`.

[92] LISSÓN, A. y PRATS-SOBREPERE, P. *Poquito a poco*. La Galera, S.A., Barcelona, 1981. ISBN 9788424606503.

[93] LLORENS Y LLATCHÓS, P. *Ventajas é inconvenientes de los sistemas de escritura ideados para uso de los ciegos, y en particular de los adoptados para su enseñanza*. Narciso Ramirez y Rialp, Barcelona, 1865.

[94] LORIMER, P. *A Critical Evaluation of the Historical Development of the Tactile Modes of Reading and an Analysis and Evaluation Or Researches Carried Out in Endeavours to Make the Braille Code Easier to Read and to Write*. Tesis Doctoral, University of Birmingham, 1996.

[95] LUCERGA REVUELTA, R. M. y VICENTE MOSQUETE, M. J. *Método Tomillo*. ONCE, Centro de Producción Bibliográfica, Barcelona, 1987.

[96] MARROU, H.-I. *Historia de la educación en la Antigüedad*. Akal universitaria. Ediciones Akal, 2004. ISBN 9788476000526.

[97] MARTÍN-BLAS SÁNCHEZ, Á. "A punto". Un método de lecto-escritura braille con máquina. En *Actas del 1 Congreso estatal sobre prestación de servicios para personas ciegas y deficientes visuales*, páginas 316–318. ONCE, Madrid, 1996.

[98] MARTINOTTI, M. H. y MIÑO, L. B. Pequén leetodo. 2012. Disponible en `http://www.leetodo.com.ar/` (último acceso, marzo 2013).

[99] MARTÍN-BLAS SÁNCHEZ, Á. El aprendizaje del sistema braille. En *Aspectos evolutivos y educativos de la deficiencia visual* (editado por M. R. Villalba Simón y I. Martínez Liébana). ONCE, 2000. ISBN 978-84-484-0222-8.

[100] MARTÍNEZ ABELLÁN, R. *Louis Braille. El acceso de los ciegos al conocimiento*. Eduforma: Grandes educadores. Editorial MAD, Alcalá de Guadaíra, 2009. ISBN 9788467620801.

[101] MARTÍNEZ-LIÉBANA, I. y POLO CHACÓN, D. *Guía didáctica para la lectoescritura Braille*. Guías (Organización Nacional de Ciegos Españoles). Organización Nacional de Ciegos Españoles, Madrid, 2004. ISBN 9788448401498.

[102] MARTÍNEZ MOCTEZUMA, L. *La Infancia y la Cultura Escrita*. Educación (Mexico City, Mexico). Siglo XXI de España Editores, S.A., 2001. ISBN 9789682323416.

[103] MARUNY CURTO, L., MINISTRAL MORILLO, M. y MIRALLES TEIXIDO, M. *Escribir y leer*. Edelvives, Zaragoza, 1998. ISBN 9788426332073.

[104] MATELLÁN OLIVERA, V., GONZÁLEZ BARAHONA, J., HERAS QUIRÓS, P. D. L. y ROBLES MARTÍNEZ, G. Sobre software libre. compilación de ensayos sobre software libre. 2004. Disponible en `http://gsyc.escet.urjc.es/~grex/sobre-libre/` (último acceso, marzo 2013).

[105] MATÍAS SÁNCHEZ, E. Blog del proyecto nela. Disponible en `http://nelaproject.blogspot.com`. (Último acceso, marzo 2013).

[106] MATÍAS SÁNCHEZ, E. Página de desarrollo de nela. Disponible en `https://forja.rediris.es/projects/cusl6-nela/`. (Último acceso, marzo 2013).

[107] MATÍAS SÁNCHEZ, E. FreeBSD, un secreto bien guardado. *Mundo Linux*, 2005. Disponible en `http://www.cronopios.net/Textos/freebsd_un_secreto_bien_guardado.pdf` (último acceso, marzo 2013).

[108] MATÍAS SÁNCHEZ, E. Una introduccion al software libre. En *Ciberactivismo: sobre usos políticos y sociales de la red* (editado por Reunión de Ovejas Electrónicas). Virus, Barcelona, 2006. ISBN 9788496044722.

[109] MAÑOSA MAS, M. y MIRET SERRA, J. *Punt a punt*. ONCE, Centre de Producció Bibliográfica, Barcelona, 2000.

[110] MCGINNITY, B. L., SEYMOUR-FORD, J. y ANDRIES, K. J. Books for the blind. 2004. Disponible en `http://perkins.pvt.k12.ma.us/museum/section.php?id=200` (último acceso, marzo 2013).

[111] MCKUSICK, M. K. History of the BSD Daemon. Disponible en `http://www.mckusick.com/beastie/index.html`. (Último acceso, marzo 2013).

[112] DE MEJÍA, P. *Silva de varia lección*. Bartolomé de Nagera, Zaragoza, Disponible en `http://books.google.es/books?id=4dp8nD-3nZ8C`.

[113] MILLS, P. V. Thomas Lucas and his embossed stenographic characters. *The British Journal of Ophthalmology*, vol. 49(9), páginas 485–489, Disponible en `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC506146/`.

[114] MINISTERIO DE EDUCACIÓN Y CIENCIA. *Método práctico de alfabetización: Fundamentado en palabras generadoras*. Ediciones Iberoamericanas, S. A., Madrid, 1967.

[115] MOINEAU, L. y PAPATHÉODOROU, A. Cooperación y producción inmaterial en el software libre. Elementos para una lectura política del fenómeno GNU/Linux. 2000. Disponible en `http://`

biblioweb.sindominio.net/telematica/cooperacion.html (último acceso, marzo 2013).

[116] MOLINA ORTÍN, M. C. La educación de los niños invidentes desde el siglo xix hasta el inicio de su integración en los centros ordinarios. *Participación educativa*, (18), páginas 198–210, Disponible en http://www.mecd.gob.es/revista-cee/pdf/n18-molina-ortin.pdf.

[117] MURCIANO IBÁÑEZ, D. *Evaluación y mejora de un dispositivo electrónico para la enseñanza-aprendizaje del código Braille*. Trabajo de fin de carrera, EUPT, Universidad de Zaragoza, 2010.

[118] MUÑINO GIL, L. y MONTERO DE ESPINOSA ESPINO, G. Leo con lula. Disponible en http://leoconlula.com/. (Último acceso, marzo 2013).

[119] MÖLLER, E. ET AL. Definition of free cultural works. 2008. Disponible en http://www.freedomdefined.org (último acceso, marzo 2013).

[120] NAHARRO, V. *Nueva arte de enseñar a leer a los niños de las escuelas*. Imprenta que fue de Fuentenebro, Madrid, 1824. Disponible en http://books.google.es/books?id=hOH48He7gncC (último acceso, marzo 2013).

[121] NATIONAL RESEARCH COUNCIL, S. *Evaluation of sensory aids for the visually handicapped: a report on a conference ... held at the National Academy of Sciences, Washington, D.C., November 11-12, 1971*. National Council of Sciences, 1972.

[122] NEWMAN, S. E. Children's learning of two alphabets used by the blind: braille and Fishburne. *British Journal of Visual Impairment*, vol. 10(1), páginas 21–23, 1992.

[123] NIHON KANTENJI KYOKAI. Kantenji. Disponible en http://www.kantenji.jp. (Último acceso, marzo 2013).

[124] OPEN SOURCE INITIATIVE. Sitio web. Disponible en http://www.opensource.org. (Último acceso, marzo 2013).

[125] ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE. CERN Open Hardware Licence. 2011. Disponible en http://www.ohwr.org/projects/cernohl/wiki (último acceso, marzo 2013).

[126] ORGANIZACIÓN NACIONAL DE CIEGOS ESPAÑOLES. Sitio web. Disponible en http://www.once.es. (Último acceso, marzo 2013).

[127] ORGANIZACIÓN NACIONAL DE CIEGOS ESPAÑOLES. Tiflosoftware. Disponible en ftp://ftp.once.es/pub/utt/tiflosoftware/. (Último acceso, marzo 2013).

[128] PAISEY, D. Learning to read: Friedrich Gedike's primer of 1791. 1978. Disponible en `http://www.bl.uk/eblj/1978articles/article11.html` (último acceso, marzo 2013).

[129] PALÁU FERNÁNDEZ, A. y OSORO PANTIGA, D. *Sistema Paláu: método fotosilábico. 1ª cartilla*. Anaya, Madrid, 1983. ISBN 9788420721262.

[130] PALOMINO, A. S. y GONZÁLEZ, J. A. T. *Educación especial II: Ámbitos específicos De intervención*. Pirámide, Madrid, 1999. ISBN 9788436813951.

[131] PERENS, B. It's time to talk about free software again. 1999. Disponible en `http://lists.debian.org/debian-devel/1999/02/msg01641.html` (último acceso, marzo 2013). Mensaje a la lista de correo electrónico debian-devel.

[132] PERENS, B. How many open source licenses do you need? 2009. Disponible en `http://www.datamation.com/osrc/article.php/3803101/Bruce-Perens-How-Many-Open-Source-Licenses-Do-You-Need.htm` (último acceso, marzo, 2013).

[133] PETIT, É. Une grande éducatrice des aveugles: Mlle Mulot. *L'école moderne*, Disponible en `http://gallica.bnf.fr/ark:/12148/bpt6k75830r`.

[134] POLTINI, E. *Codici e segnali*. Sussidi tecnici. Arte scout. Nuova Fiordaliso, 2008. Disponible en `http://costermano1.altervista.org/scoutcostermano1/segnalazione_files/alfabetotattile.htm`.

[135] PORTAL ARAGONÉS DE LA COMUNICACIÓN AUMENTATIVA Y ALTERNATIVA. Sitio web. Disponible en `http://www.catedu.es/arasaac/`. (Último acceso, marzo 2013).

[136] PROGRAMA DE ASISTENCIA TECNOLÓGICA DE PUERTO RICO. Salón braille virtual. 2009. Disponible en `http://pratp.upr.edu/template/informacion/accesibilidad` (último acceso, marzo 2013).

[137] PÉREZ MARINA, J. *La Llectura y la escritura en la educación especial: Mëtodo «Esperanza»*. Educación especial. Ciencias de la Educación Preescolar y Especial, Madrid, 1988. ISBN 9788485252183.

[138] QUÍLEZ GARCÍA, R. M. y MOESE RUIZ, S. Flash: método de aprendizaje del sistema braille para ciegos adultos alfabetizados. En *Actas del 1 Congreso estatal sobre prestación de servicios para personas ciegas y deficientes visuales*, páginas 285–288. ONCE, 1996.

[139] QUINTILIAN y BUTLER, H. E. *Institutio oratoria*. Número v. 1 en Loeb classical library. W. Heinemann, 1920. Disponible en `http://www.`

archive.org/details/institutioorato00butlgoog (último acceso, marzo 2013).

[140] RAYMOND, E. S. The glider: A universal hacker emblem. 2003. Disponible en http://www.catb.org/hacker-emblem/ (último acceso, marzo 2013).

[141] REASONING. Reasoning study reveals code quality of MySQL open source database ranks higher than commercial equivalents. 2003. Disponible en http://www.businesswire.com/news/home/20031215005408/en/Reasoning-Study-Reveals-Code-Quality-MySQL-Open (último acceso, marzo 2013). Nota de prensa.

[142] REEDER, R. R. *The Historical Development of School Readers and of Method in Teaching Reading*. Columbia University, 1900. Disponible en http://www.archive.org/details/historicaldevel02reedgoog (último acceso, marzo 2013).

[143] REGENTS OF THE UNIVERSITY OF CALIFORNIA, T. The 4.4BSD copyright. 1979. Disponible en http://www.freebsd.org/copyright/license.html (último acceso, marzo 2013).

[144] REX, E. J. *Foundations of Braille Literacy*. AFB Press, 1994. ISBN 9780891289340.

[145] RIPLEY, G. y DANA, C. A. *The Blind. The New American Cyclopædia*. Número v. 2 en The American Cyclopaedia: A Popular Dictionary of General Knowledge. D. Appleton and company, Disponible en http://chestofbooks.com/reference/American-Cyclopaedia-1/The-Blind.html.

[146] RODIN, J., AOKI, O., FERNÁNDEZ-SANGUINO PEÑA, J. ET AL. Guía del nuevo desarrollador de debian. 2013. Disponible en http://www.debian.org/doc/manuals/maint-guide/ (último acceso, marzo 2013).

[147] RODRÍGUEZ ÁLVAREZ, M. D. L. Á. y VELÁSQUEZ RODRÍGUEZ, K. V. Palabras normales versus onomatopeya. *Enseñar y aprender en las escuelas de la Patagonia*, páginas 155–172, Disponible en http://host26.200-45-54.telecom.net.ar/investigacion/educa/web_relee/archivos/alf_siglo_a5.pdf. (Último acceso, marzo 2013).

[148] ROIG, C. Curso básico de autoaprendizaje del braille. Disponible en http://www.once.es/otros/cursobraille/. (Último acceso, marzo 2013).

[149] ROYAL LONDON SOCIETY FOR THE BLIND. Sitio web. Disponible en http://www.rlsb.org.uk. (Último acceso, marzo 2013).

[150] ROYAL NATIONAL INSTITUTE FOR THE BLIND. Sitio web. Disponible en http://www.rnib.org.uk/. (Último acceso, marzo 2013).

[151] RUIZ GONZÁLEZ, M. Análisis de las licencias de software libre. 2008. Disponible en http://www.miriamruiz.es/slides/doc_licencias_de_software_libre.pdf (último acceso, marzo 2013).

[152] RUMSEY, D. David Rumsey map collection. Disponible en http://www.davidrumsey.com. (Último acceso, marzo 2013).

[153] RUPP, T. Lorm-alphabet. Disponible en http://www.lormer.de/pages/lorm-alphabet.php. (Último acceso, marzo 2013).

[154] RUSSELL, J. Inno setup. Disponible en http://www.jrsoftware.org/isinfo.php. (Último acceso, marzo 2013).

[155] SALINAS, M. Guía instalación y aplicación software educativo cantaletras 2.2. 2009. Disponible en http://www.docstoc.com/docs/49802364/ (último acceso, marzo 2013).

[156] SALUS, P. H. *A Quarter Century of Unix*. Addison-Wesley Unix and Open Systems. Addison-Wesley, 1994. ISBN 9780201547771.

[157] SÁNCHEZ HERRERO, Á. *Bliseo: método para el aprendizaje del código de lectoescritura Braille*. ONCE, Madrid, 1980.

[158] SELLIE, A. y GOINS, M. The readers'bill of rights for digital books. Disponible en http://readersbillofrights.info/. (Último acceso, marzo 2013).

[159] SHAFRATH, M. R. An alternative to braille labeling. *Journal of Visual Impairment and Blindness*, vol. 80(9), páginas 955–56, 1986.

[160] SIMÓN RUEDA, C., OCHAITA ALDERETE, E. y HUERTAS MARTÍNEZ, J. A. El sistema braille: Bases para su enseñanza-aprendizaje. *CL & E: Comunicación, lenguaje y educación*, (28), páginas 91–102, Disponible en http://dialnet.unirioja.es/servlet/articulo?codigo=2941799.

[161] SOLÉ, B. Propuestas para la enseñanza del lenguaje escrito. 2009. Disponible en http://es.scribd.com/doc/22827523/Belinda-Sole-Propuestas-para-la-ensenanza-del-lenguaje-escrito (último acceso, marzo 2013).

[162] STAFFORD, B. The open source hardware logo repository. Disponible en http://oshwlogo.com/. (Último acceso, marzo 2013).

[163] STALLMAN, R. *Software libre para una sociedad libre*. Traficantes de Sueños, Madrid, 2004. Disponible en `http://www.nodo50.org/ts/editorial/librospdf/free_software.pdf` (último acceso, marzo 2013).

[164] STALLMAN, R. M. Sitio personal. Disponible en `http://www.stallman.org`. (Último acceso, marzo 2013).

[165] LANA DE TERZI, F. *Prodromo ouero saggio di alcune inuentioni nuoue premesso all'arte maestra*. Per li Rizzardi, Brescia (Italia), 1670.

[166] THE AUDACITY PROJECT. Sitio web. Disponible en `http://audacity.sourceforge.net/`. (Último acceso, marzo 2013).

[167] THE DEBIAN PROJECT. Sitio web. Disponible en `http://www.debian.org`. (Último acceso, marzo 2013).

[168] THE DRAGONFLY BSD PROJECT. Sitio web. Disponible en `http://www.dragonflybsd.org/`. (Último acceso, marzo 2013).

[169] THE FREEBSD PROJECT. Sitio web. Disponible en `http://www.freebsd.org/`. (Último acceso, marzo 2013).

[170] THE GNU PROJECT. Sitio web. Disponible en `http://www.gnu.org`. (Último acceso, marzo 2013).

[171] THE MINIX 3 PROJECT. Sitio web. Disponible en `http://www.minix3.org/`. (Último acceso, marzo 2013).

[172] THE NETBSD PROJECT. Sitio web. Disponible en `http://www.netbsd.org/`. (Último acceso, marzo 2013).

[173] THE OPENBSD PROJECT. Sitio web. Disponible en `http://www.openbsd.org/`. (Último acceso, marzo 2013).

[174] THE TEXSTUDIO PROJECT. Sitio web. Disponible en `http://texstudio.sourceforge.net/`. (Último acceso, marzo 2013).

[175] THURLOW, W. R. An alternative to braille. *Journal of Visual Impairment and Blindness*, vol. 82(9), página 378, 1988.

[176] TORVALDS, L. B. Free minix-like kernel sources for 386-at. 1991. Disponible en `https://groups.google.com/d/msg/comp.os.minix/4995SivOl9o/GwqLJlPSlCEJ` (último acceso, marzo 2013). Mensaje al grupo de noticias comp.os.minix.

[177] TUCSON AMATEUR PACKET RADIO. The tapr open hardware license. 2007. Disponible en `http://www.tapr.org/ohl.html` (último acceso, marzo 2013).

[178] UNIVERSIDADE DE SÃO PAULO. Braille virtual. 2004. Disponible en `http://www.braillevirtual.fe.usp.br/es/index.html` (último acceso, marzo 2013).

[179] USERO ALIJARDE, A. *Letrilandia: Libro de lecturas 1*. Editorial Luis Vives (Edelvives), Zaragoza, 2007. ISBN 9788426355836.

[180] VALBUENA GARCÍA, C., CRUZ SANTA BÁRBARA, G. y RODRÍGUEZ ORTEGA, J. C. Metodo de prelectura y preescritura braille "Almazara". En *Congreso estatal sobre prestación de servicios para personas ciegas y deficientes visuales: Madrid, Septiembre 1994*, páginas 297–302. ONCE, 1996.

[181] VALBUENA GARCÍA, C., CRUZ SANTA BÁRBARA, G. y RODRÍGUEZ ORTEGA, J. C. *Almazara. Método de prelectura y preescritura braille*. ONCE-CREA Luis Braille, Sevilla, 1997.

[182] VÉZIEN, E. Écriture à l'usage des aveugles et des voyants. *Mémoires de la Société dunkerquoise pour l'encouragement des sciences, des lettres et des arts*, (v. 28), páginas 429–433, Disponible en `http://archive.org/details/mmoires09artsgoog`.

[183] WALFIELD, N. H. y BRINKMANN, M. A critique of the GNU hurd multi-server operating system. *ACM SIGOPS Operating Systems Review*, vol. 41(4), páginas 30–39, 2007.

[184] WIKIPEDIA. Copyleft — Wikipedia, the free encyclopedia. 2013. Disponible en `http://en.wikipedia.org/wiki/Copyleft` (último acceso, marzo 2013).

[185] WIKIPEDIA. Moon type — Wikipedia, the free encyclopedia. 2013. Disponible en `http://en.wikipedia.org/wiki/Moon_type` (último acceso, marzo 2013).

[186] WIKIPEDIA. New York Point — Wikipedia, the free encyclopedia. 2013. Disponible en `http://en.wikipedia.org/wiki/New_York_Point` (último acceso, marzo 2013).

Anexo: Artículo para la revista Novática

Resumen: Se adjunta a continuación el artículo escrito por invitación de Novática, decana de las revistas informáticas españolas, que ya ha sido revisado y aprobado para su inclusión en el próximo número de la publicación.

Nela: Aprende a escribir usando Braille

Enrique Matías Sánchez, Inmaculada Plaza García, Nuria Tregón Martín
quique@unizar.es, inmap@unizar.es, nuritre@unizar.es

Resumen

Nela es un programa informático que ayuda en el aprendizaje de la escritura usando el código Braille, simulando una máquina Perkins con el teclado del ordenador. Está dirigido a niños con discapacidad visual (visión reducida o ceguera).

Programado en C++ usando las bibliotecas Qt, es multiplataforma (funciona tanto en Microsoft Windows como en GNU/Linux) y está internacionalizado.

Ha sido valorado positivamente por varios educadores, y obtuvo el primer premio en la fase estatal del VI Concurso Universitario de Software Libre, así como el premio especial de accesibilidad.

Palabras clave

Aprendizaje, Braille, ceguera, discapacidad visual, enseñanza, escritura, escuela, formación, software educativo.

English

Title: Nela: Learn to Write Using Braille

Abstract: Nela is a computer program that helps in learning to write using the Braille code, simulating a Perkins brailler with the computer's keyboard. It is intended for children with visual disabilities (low vision or blindness).

Programmed in C++ using the Qt libraries, it is crossplatform (runs on both Microsoft Windows and GNU/Linux) and it is internationalized.

It has been favorably reviewed by several educators, and won first prize in the statewide 6th Free Software University Contest, as well as the special prize to accessibility.

Keywords: Blindness, Braille, Educational Software, Learning, School, Teaching, Training, Visual Impairment, Writing.

Autores

Enrique Matías es estudiante de Ingeniería Técnica en Informática de Gestión en la Escuela Universitaria Politécnica de Teruel. Nela es su Trabajo de Fin de Carrera.

Decidido defensor del Software Libre, en 2003 organizó una multitudinaria conferencia de Richard Stallman en Zaragoza. Ha sido traductor de KDE, colaborador de la revista *Mundo Linux* y miembro de diversas organizaciones de

carácter social. Actualmente trabaja en el Servicio de Informática de la Universidad de Zaragoza.

Inmaculada Plaza García es Profesora Titular de Universidad en la Universidad de Zaragoza. Investigadora Principal del grupo EduQTech (*Education-Quality-Technology*), primer grupo de investigación con sede en Teruel reconocido por el Gobierno de Aragón en el área Tecnológica. Dirige la «Cátedra en innovación y calidad tecnológica» de la Univ. de Zaragoza. Presidenta de la Sociedad de Educación del IEEE en España (2010–2012), actualmente es *Past-Chair* de la misma. Su investigación se centra en tres áreas: Calidad de vida/Salud, especialmente en el ámbito de las neurociencias, Calidad en educación y Calidad e innovación empresarial; siempre desde un enfoque de desarrollo en el medio rural. Promotora del software y hardware libre, intenta difundir su utilización en todos los proyectos y redes en los que participa.

Nuria Tregón Martín actualmente ejerce como maestra de apoyo del equipo ONCE de Aragón. Paralelamente es Profesora Asociada de la Universidad de Zaragoza, en el Departamento de Teoría e Historia de la Educación de la Facultad de Ciencias Sociales y Humanas en el Campus de Teruel. Ha finalizado el Máster de Cooperación al Desarrollo y su línea de investigación va encaminada al estudio de la discapacidad en países empobrecidos.

1. Introducción

Nela, el programa informático que se presenta en este artículo, tiene como principal objetivo servir de herramienta de apoyo en el aprendizaje de la escritura usando el código Braille.

Aunque en un inicio se diseñó pensando en su utilización por parte de niños con discapacidad visual (ceguera o visión reducida), su utilización no se limita a este público objetivo. Nela puede ser utilizado igualmente con niños sin discapacidad visual a los que se desee introducir en la enseñanza del código Braille, pero también por jóvenes o adultos que hayan perdido la visión (o tengan visión reducida) debido a algún accidente o enfermedad.

La importancia de utilizar las Tecnologías de la Información y Comunicación (TIC) en el ámbito de la discapacidad visual ya ha sido tratada previamente por diversos autores, como Lidner F. (1990), Toledo P. y Hervás C. (1992) [9], o el grupo ACCEDO, siendo este último apoyo fundamental para la creación e integración de nuevas tecnologías para personas con discapacidad y específicamente para personas con discapacidad visual.

Aunque ya es ampliamente reconocido el papel que desempeñan las nuevas tecnologías en el desarrollo cognitivo, concretamente por su gran motivación en el discente, hemos de remarcar la importancia de las TIC en el proceso de enseñanza-aprendizaje de los alumnos con discapacidad visual al ser una herramienta, imprescindible y a su vez complementaria, tanto para el acceso a la información como para la comunicación.

Para presentar Nela, el artículo se ha estructurado en los siguientes apartados: en la sección segunda se describe el contexto en el que surge este proyecto, que ha dado lugar a la realización de un trabajo fin de carrera cuyo resultado es el programa informático presentado. Tras esta contextualización, se pasará a describir con mayor detalle diferentes aspectos de Nela: su funcionamiento (sección 3), algunos aspectos a considerar en su utilización en la enseñanza (sección

4), aspectos técnicos (sección 5) y su licencia (sección 6). Para finalizar se comentará brevemente la primera prueba piloto (sección 7), concluyendo con un apartado de líneas de trabajo futuro (sección 8).

2. Contexto y antecedentes

La idea de desarrollar Nela surgió como parte de un proyecto mucho mayor, iniciado en el año 2007, cuando dos profesionales que trabajaban con niños con discapacidad visual en la provincia de Teruel (Lucía Azara e Inés Benedicto) se pusieron en contacto con el grupo de investigación EduQTech (Univ. de Zaragoza) planteando la necesidad de desarrollar herramientas que facilitasen la enseñanza y el aprendizaje de los caracteres del código Braille.

En general, los recursos existentes dirigidos a los discapacitados visuales no son numerosos y debido a su limitado interés comercial, caros. En concreto, los dispositivos o aplicaciones para la adquisición de la lectoescritura para niños en las primeras etapas de escolarización son muy reducidos, por lo que las profesionales indicaron la necesidad de ampliar el abanico de posibilidades desarrollando herramientas personalizables y que favorecieran el aprendizaje de un modo progresivo.

Para conocer el estado del arte se analizaron diferentes antecedentes. Sin ánimo de ser exhaustivos se pueden mencionar:

- La propuesta de Cétares et al. [1] de un sistema de enseñanza del código Braille para niños con limitaciones visuales.
- *Automated Braille Writing Tutor*, dispositivo desarrollado en la Universidad de Carnegie Mellon y que desde el primer momento se probó en la India.
- La patente estadounidense US5902112 (Sally S. Mangold), con una matriz rectangular de zonas sensibles al tacto y un procesador acoplado a un sintetizador que pronuncia el carácter Braille.
- La patente alemana DE19547742 (René Lemoine), en la que se describe un dispositivo y un método para practicar idiomas por escrito u oralmente sin necesidad de profesor. En concreto, reinventa un método para la introducción y muestra de signos visuales y sonoros a elección del usuario.
- La solicitud internacional de patente WO2008120303 con unidades de entrada de caracteres Braille para introducir un carácter Braille por medio de una pluralidad de unidades de presión y de unidades de confirmación para la salida de los caracteres Braille introducidos.
- La patente estadounidense US576963, especialmente destinada a la enseñanza de operaciones de cálculo básicas.

Así mismo, y a medida que se avanzó en el tiempo, se fueron analizando algunos programas informáticos como por ejemplo «Salón Braille Virtual» o «Cantaletras» y, como complemento, aplicaciones informáticas destinadas a personas no ciegas (familiares, profesionales) como el «curso básico de autoaprendizaje del Braille» de la ONCE (Organización Nacional de Ciegos Españoles),

«Braille Virtual» de la Universidad de São Paulo (Brasil), «Detrás de cada punto» del INCI (Instituto Nacional de Ciegos de Colombia), «Demo Braille» y «Alfabeto Braille en línea» de la FBU (Fundación Braille del Uruguay).

El análisis del estado del arte permitió, y ha seguido permitiendo a lo largo de estos años, detectar distintas necesidades no cubiertas. Como su descripción excedería el objetivo de esta publicación, se puede consultar la patente ES2366509 «Dispositivo electrónico para la enseñanza y el aprendizaje de la escritura de caracteres en código Braille» de la Universidad de Zaragoza, en la que se presenta un análisis más detallado y las conclusiones obtenidas.

De esta forma surgió el proyecto «TerBraille», que se definió pensando en dos etapas:

1. Una primera etapa en la que los usuarios pudieran aprender los caracteres alfanuméricos en código Braille a través de un dispositivo electrónico.

2. Una segunda etapa, en la que se introduciría a los usuarios en el manejo del ordenador, reforzando el aprendizaje de los caracteres alfanuméricos a la vez que extendiendo el aprendizaje a sílabas y palabras. Para ello se definiría un programa informático basado en el uso de la máquina Perkins[1].

El grupo EduQTech utiliza y fomenta el uso de software/hardware libre, por lo que los diferentes desarrollos se realizaron bajo esta filosofía.

La primera versión del dispositivo electrónico fue desarrollada, a través de sendos trabajos fin de carrera, por Óscar Díez Sáez [3] y David Murciano [8], dirigidos conjuntamente por los doctores Carlos Medrano e Inmaculada Plaza. El prototipo, basado en la plataforma Arduino fue probado durante un año en un colegio de Teruel, y posteriormente por profesionales de Zaragoza. En una segunda versión, realizada por Jorge Izquierdo [6], se añadió la comparación del dispositivo con una plataforma conocida basada en Pic. Finalmente, y como cuarto trabajo fin de carrera, se propuso la realización del software objeto de esta publicación.

De esta forma, Nela constituye la segunda etapa del proyecto. Junto con el dispositivo electrónico puede considerarse un kit de enseñanza/aprendizaje de la escritura, en el que cada elemento se puede utilizar de forma individual o complementaria.

3. Funcionamiento del programa

Como se ha indicado en la sección 1, Introducción, aunque el programa Nela puede ser utilizado por jóvenes y adultos, inicialmente se desarrolló para niños con discapacidad visual, que hayan alcanzado la madurez necesaria para abordar la capacidad lectoescritora.

Nela simula una máquina Perkins usando el teclado normal del ordenador. Una máquina Perkins viene a ser una especie de máquina de escribir para ciegos, que sólo tiene 6 teclas, una para cada uno de los seis puntos del cajetín Braille (más el espaciador, etc).

El niño colocará las manos sobre la fila central del teclado, con los dedos índices sobre las teclas F y J, que tienen unas marcas en relieve. Si el niño tiene dificultades para notar ese relieve, se puede pegar un trozo de fieltro sobre las

[1] http://en.wikipedia.org/wiki/Perkins_Brailler

teclas. Las teclas S-D-F y J-K-L corresponden respectivamente a los puntos 3-2-1 y 4-5-6 del cajetín Braille. Para escribir una letra se pulsan simultáneamente las teclas correspondientes a los puntos que la componen en el código Braille.

Figura 1: Interfaz gráfica de Nela.

Nela solicitará al niño que escriba una palabra, y le comunicará, tanto de forma visual como sonora, si la introducción ha sido correcta o no, mostrando en la parte inferior de la pantalla la palabra escrita en Braille.

El programa introduce automáticamente una nueva palabra una vez que el niño escribe habitualmente de forma correcta las palabras presentadas hasta el momento. Las palabras ya vistas seguirán presentándose ocasionalmente para repasarlas. Cuando se cometa un error en una palabra, ésta será presentada con mayor frecuencia, para reforzarla. Del mismo modo, las palabras que no presenten dudas aparecerán más raramente.

4. Aspectos a considerar en la enseñanza utilizando Nela

El objetivo de Nela es aprender a escribir, usando Braille como código. Por ello, antes de empezar a utilizarlo, es recomendable que el niño haya alcanzado cierta madurez lectoescritora: conceptos espaciales básicos (arriba/abajo, izquierda/derecha), nociones de cantidad (primeros números), coordinación dígito-manual, etc. Con este fin se puede usar el *Método Alameda* [5].

En la enseñanza de la lectoescritura Braille, debería evitar una presentación ordenada de las letras del alfabeto, tal y como aparece en algunos métodos alfabéticos en tinta. Es muy importante no enseñar simultáneamente, desde un principio, letras muy parecidas, para evitar posibles confusiones que dificultarían el proceso de enseñanza, y que en muchas ocasiones son difíciles de corregir si se han adquirido. Teniendo en cuenta la secuencia sugerida por Susana E. Crespo [2], y de acuerdo con su propia experiencia, Begoña Espejo [4] presenta un orden por el que introducir las letras. Para más detalles se puede consultar la documentación de Nela o la bibliografía.

Es esencial fomentar el interés del niño por estas la habilidad de leer y escribir, de igual manera que se hace con el niño vidente. Por ello, Nela hace uso desde el primer momento de un vocabulario con significado afectivo y vivencial (boca, cama, paloma...) seleccionado de un corpus de vocabulario infantil.

Paulatinamente se introducen nuevas palabras con nuevas sílabas y nuevas letras, con dificultad progresiva (sílabas directas, inversas, mixtas y trabadas). Si bien este método tiene una sólida base pedagógica, Nela no lo impone, y el docente que lo desee puede no obstante personalizar el programa para seguir otra metodología. La aplicación se adapta a las necesidades específicas de cada usuario, no introduciendo nuevas palabras hasta que no se hayan asimilado las anteriores, y reforzando las que le resulten más complicadas al niño.

5. Aspectos técnicos

Nela está dirigido a niños tanto a niños con visión reducida como con ceguera. Para permitir su uso en niños que presenten ceguera total, Nela no precisa del uso del ratón: todas las características del programa están accesibles vía teclado.

Para facilitar su uso en niños con baja visión, Nela emplea letras de gran tamaño. El tipo de letra elegido (Tiresias PCfont[2]) está diseñado para una legibilidad óptima por la Unidad de Investigación Científica del *Royal National Institute of Blind People*[3] (Real Instituto Nacional para los Ciegos) de Londres.

5.1. Multiplataforma

Uno de los requisitos de diseño fue que el software debía ser multiplataforma, y funcionase tanto en Microsoft Windows como en GNU/Linux. Se pueden distinguir tres maneras diferentes de escribir software multiplataforma:

1. Haciendo uso de lenguajes interpretados, como Python.
2. Mediante compilación a *bytecode* intermedio, que será ejecutado en una máquina virtual, como Java.
3. Escribiendo el código fuente en un lenguaje estandarizado, que se compilará a código nativo en cada plataforma.

Cada método tiene sus ventajas y desventajas, como rapidez de ejecución y comodidad para el usuario o el desarrollador. En este caso se optó por la

[2] http://www.tiresias.org/fonts/pcfont/about_pc.htm
[3] http://www.rnib.org.uk/

tercera opción, y desarrollar en C++ usando las bibliotecas Qt. Aunque mucha gente piensa en las Qt como un *toolkit* gráfico (como pueda ser Swing), sería más acertado entenderlas como una alternativa, más amplia, a la *Standard Template Library* de C++ . De hecho, gracias a su arquitectura modular, se pueden emplear para escribir aplicaciones no gráficas. Además de unas clases para interfaces gráficas con una buena integración con el entorno del usuario, las Qt proporcionan soporte para Unicode, *signals* y *slots* (una implementación del patrón de diseño *Observer* o *publish-subscribe*) y facilitan la recolección de basura.

Se ha comprobado el correcto funcionamiento del programa sobre GNU/Linux y Microsoft Windows 7. Debería ser bastante sencillo adaptarlo también a Mac OS X. Además del sistema operativo, otros aspectos a considerar son el tamaño de palabra y la *endianness*. Nela se ha probado sobre 32 y 64 bits. Aunque el autor sólo disponía de sistemas *Little Endian*, no se hacen operaciones que puedan suponer un problema en sistemas *Big Endian*.

Si bien el código fuente de Nela se puede compilar en diversos sistemas, éste no es un método accesible a los usuarios finales. Desde la página web se puede descargar un instalador para Windows, que proporciona una instalación guiada que no ofrece ninguna dificultad. Hay diversos programas para crear instaladores para Windows (InstallShield, NSIS, WiX, etc). Se eligió Inno Setup[4], que es software libre, tiene todas las funcionalidades requeridas (como la capacidad de instalar tipos de letra) y está razonablemente documentado.

También se facilita un paquete `.deb` para sistemas de tipo Debian como Ubuntu. Previamente habrá que tener instalados los paquetes de los que depende (libqt4-svg, libphonon4, etc). Conviene comprobar que no se esté usando el motor de Phonon basado en xine (que ha sido abandonado y ya no recibe mantenimiento), sino los *backends* actuales basados en VLC o gstreamer. El programa está disponible para la distribución Chakra GNU/Linux en el repositorio CCR (*Chakra Community Repository*).

5.2. Internacionalización

Otro requisito de diseño fue que el programa estuviese internacionalizado (preparado para traducirse a otros idiomas). En el caso de Nela, esto implica cuatro actividades diferenciadas:

- Traducción de la interfaz del programa.
- Elaboración de la secuencia de palabras.
- Creación de la tabla Braille del idioma en cuestión.
- Grabación de los sonidos.

Todos los textos que aparecen en el programa (menús, etc) se encuentran en un fichero en formato XML y extensión `.ts`. La traducción se puede realizar usando el programa Qt Linguist, o un simple editor de texto.

La secuencia de palabras que se presentan al niño debería seguir un orden definido con un criterio científico como el presentado anteriormente, adecuado a las características del idioma en cuestión. Esta secuencia se encuentra en un

[4] http://www.jrsoftware.org/isinfo.php

fichero en formato `csv` (valores separados por tabuladores). La primera columna es la palabra a presentar, la segunda columna es la cadena en ASCII de 7 bits que se usará para los nombres de los ficheros de las grabaciones correspondientes, y la tercera columna es la imagen que ilustra dicha palabra. Este fichero se puede editar con una hoja de cálculo o un editor de texto.

Inicialmente había una cuarta columna con la palabra escrita en Braille (caracteres Unicode 0x2800 - 0x28FF), para poder comparar con la respuesta introducida por el niño, y mostrar la respuesta correcta en pantalla. Sin embargo esto dificultaba la elaboración de este fichero, porque no hay ninguna manera sencilla de introducir esos caracteres desde el teclado. Finalmente se optó por generar la cadena en Braille a partir de la primera columna. La primera aproximación fue crear una tabla *hash* que asociaba cada letra con el número del carácter Unicode que le corresponde en Braille. Esta solución tampoco es óptima, porque los educadores o traductores que deseen editar este fichero probablemente no estén familiarizados con las tablas Unicode. La solución final fue añadir un nuevo fichero de texto con dos columnas: la primera con cada letra, y la segunda con los puntos del carácter Braille que le corresponden (por ejemplo: `g 1245`). A partir de dichos puntos, se calcula el número del carácter Unicode (10267 en el caso de la letra g) y se devuelve el cáracter en cuestión. Éste es el código de la función:

```
QChar PerkinsModel::pointsToUnicode(QString points)
{
    int code = 10240;
    foreach (const QString &digit, points) {
        code += pow(2, digit.toInt() - 1);
    }
    return QChar(code);
}
```

El código Braille es diferente en cada idioma, pues ha de adaptarse a los caracteres que emplee (en el caso del castellano, vocales acentuadas, letra eñe e interrogante y admiración inicial). Por ello, ha de crearse un fichero de correspondencias letra-puntos para cada idioma.

Un interesante problema técnico fue la simulación de la máquina Perkins, pues obviamente los sistemas no están pensados para detectar la pulsación simultánea de hasta 6 teclas. Además, los dedos de la mano no se mueven exactamente a la vez. El sistema emite un evento cada vez que se pulsa o libera una tecla, y lo que hace Nela es recoger en una lista los números del cajetín Braille correspondientes a las teclas soltadas en los 400 milisegundos siguientes a la primera liberación (valor escogido empíricamente). A continuación ordena esa lista, elimina duplicados si los hay, y se la pasa como una cadena de texto a la función mostrada anteriormente, que devuelve el carácter Braille correspondiente a las pulsaciones del usuario. Finalmente, se compara ese carácter con el esperado y se actúa en consecuencia. Se ha observado que algunos teclados, particularmente en *netbooks*, no emiten correctamente los eventos de liberación de teclas cuando hay varias teclas pulsadas a la vez. En caso de encontrarnos ante un hardware con esta limitación, la solución es conectar un teclado externo.

Las grabaciones sonoras se encuentran en formato MP3, aunque en el futuro se podría investigar la posibilidad de usar Speex[5], un formato de compresión de audio diseñado para la voz humana. Se pueden realizar con un micrófono doméstico y un programa como Audacity[6], que incluye una funcionalidad de reducción de ruido.

Las imágenes empleadas no se encuentran en formatos *rasterizados* como JPEG o PNG, sino en el formato vectorial SVG que, además de ocupar menos espacio en disco, se pueden escalar a cualquier tamaño de pantalla.

6. Licencia

Nela es Software Libre [7], con licencia GNU GPL 3 o posterior: se puede usar, copiar a otras personas, modificar y distribuir versiones modificadas de forma gratuita y legal.

El desarrollo está abierto a cualquier persona interesada.

7. Primeras pruebas piloto

Como prueba piloto inicial se contó con la colaboración desinteresada de cuatro docentes de distintos puntos de España que aportaron ideas y sugerencias de mejora para próximas versiones.

El problema más claro es la baja calidad del audio. Los sonidos fueron grabados inicialmente con el micrófono interno de un portátil. Es necesario volver a grabarlos en un estudio insonorizado, con un micrófono de calidad y a un volumen adecuado.

Otra propuesta es utilizar una mayor variedad de sonidos para el feedback, y hacerlos más breves, para evitar que puedan llegar a hacerse monótonos.

También han sugerido algunas mejoras de accesibilidad, por ejemplo para usuarios con problemas motores y que sólo puedan usar una mano. Por último, se nos ha planteado la posibilidad de crear una versión para Mac OS X.

8. Líneas de trabajo futuro

Como se ha indicado en el punto 6, Licencia, el desarrollo está abierto a cualquier persona interesada: educadores que propongan mejoras, desarrolladores que deseen implementarlas, traductores que adapten el programa a otros idiomas, ilustradores que faciliten dibujos libres...

Se pretende, de este modo, que la presente versión de Nela sea un punto de partida y un lugar de encuentro y colaboración. Para más información, consúltese:

Blog http://nelaproject.blogspot.com.es/

Desarrollo https://forja.rediris.es/projects/cusl6-nela/

[5] http://www.speex.org/
[6] http://audacity.sourceforge.net/

9. Agradecimientos

Los autores agradecen su colaboración a todos aquellos educadores que han intervenido en la prueba piloto de evaluación, enriqueciendo el trabajo con sus comentarios y sugerencias. Así mismo, agradecen las aportaciones de D. Hermes Ojeda Ruiz quien ha colaborado mediante parches, y empaquetamiento para Chakra Gnu/Linux.

Referencias

[1] Cétares Salas, A., Cortés Rivera, C. A. y Silva Olarte, L. F. *Sistema de enseñanza del código Braille para niños con limitaciones visuales*. Trabajo de grado, Facultad de Ingeniería, Pontificia Universidad Javeriana, Bogotá (Colombia), 2005. Disponible en http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis85.pdf (último acceso, marzo 2013).

[2] Crespo, S. E. *La escuela y el niño ciego. Manual práctico*. Graficart, Córdoba (Argentina), 1980.

[3] Díaz Saez, Ó. *Entrenador para niños con discapacidad visual*. Trabajo de fin de carrera, EUPT, Universidad de Zaragoza, 2009.

[4] Espejo de la Fuente, B. *El Braille en la escuela. Una guía práctica para la enseñanza del Braille*. ONCE, Madrid, 1993.

[5] Fuentes Hernández, J. *Método Alameda*. ONCE, Madrid, 1995.

[6] Izquierdo Najes, J. *Dispositivo de aprendizaje de código Braille basado en un microcontrolador PIC: comparación con el desarrollo basado en Arduino*. Trabajo de fin de carrera, EUPT, Universidad de Zaragoza, 2012.

[7] Matías Sánchez, E. Breve introducción al software libre. *Pueblos*, páginas 49–51, 2004. Disponible en línea en http://cusl6-nela.forja.rediris.es/Breve_introduccion_al_software_libre.pdf.

[8] Murciano Ibáñez, D. *Evaluación y mejora de un dispositivo electrónico para la enseñanza-aprendizaje del código Braille*. Trabajo de fin de carrera, EUPT, Universidad de Zaragoza, 2010.

[9] Villalba Simón, M. R. *Aspectos evolutivos y educativos de la deficiencia visual. Volumen II*. ONCE, Madrid, 2000.