ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Curso Académico 2009/2010

Proyecto de Fin de Carrera

REBECA: SOFTWARE EDUCATIVO DE INTRODUCCIÓN A LA PROGRAMACIÓN PARA HISPANO HABLANTES

Autor: Irene Montano Rodríguez

Tutor: David Miraut Andrés

"La mejor manera de que alguien aprenda algo consiste en hacerle creer que está aprendiendo otra cosa"

Randy Pausch

Agradecimientos

Vaya en primer lugar mi agradecimiento a todos aquellos que han hecho posible este proyecto, desde los creadores de *Alice* hasta los profesores del departamento de Gráficos y Realidad Virtual de la Universidad Rey Juan Carlos.

Muy especialmente a mi tutor David Miraut por su paciencia y dedicación y sobre todo por haber sabido animarme cuando más lo he necesitado.

A todas mis amigas porque llevan conmigo toda la vida y en especial a Marian por su colaboración en la creación de imágenes y a Carmen que me ha dedicado su tiempo y ayuda haciendo que finalmente el proyecto tomara forma.

A mis padres por haberme dado apoyo durante mis años de estudiante y por haber confiado siempre en mí.

A mi hermana Alicia porque ha prescindido de su ordenador durante meses confiando en que algún día acabaría. A mi hermana Ana porque llena mis días de alegría.

Y por supuesto a Sergio Ruiz Pérez, mi compañero durante estos años de Universidad y durante todo el proyecto, por su capacidad de trabajo, por su tesón y porque sin él esto no hubiera sido posible.

Muchas Gracias a todos.

Resumen

En el actual sistema es necesaria una adaptación de la educación a las características de nuestra época, en cuya construcción es deseable participar de forma creativa. El momento presente es propicio para esta necesaria renovación educativa en todas las etapas.

La importancia de la educación y del conocimiento en un mundo global es cada vez mayor, las elevadas tasas de paro existentes entre diplomados y licenciados universitarios, junto a la devaluación de una buena parte de las titulaciones que se imparten en las universidades, han llevado a las autoridades políticas y académicas de los países de la Organización para la Cooperación y el Desarrollo Económico (OCDE) a plantearse las razones por las que la enseñanza superior, parece no responder adecuadamente a las nuevas necesidades de la sociedad. Por todo ello se considera necesario reformar las enseñanzas previas a la universitaria y crear nuevos recursos educativos que atraigan a los futuros universitarios hacia una formación que posibilite su incorporación al mundo laboral de forma más eficaz.

En la búsqueda de nuevos recursos educativos que den respuesta al contexto anteriormente planteado, encontramos *Alice,* un proyecto de innovación educativa, liderado por la Universidad Carnegie Mellon, creado con el propósito de ayudar a los estudiantes a conocer los fundamentos de la programación orientada a objetos. Esta herramienta tiene el inconveniente de que durante su desarrollo no se ha contempló la posibilidad de dar soporte a otros idiomas, por lo que su interfaz está exclusivamente en inglés.

El proyecto *Rebeca* surge de la necesidad de acercar *Alice* a los estudiantes de habla no inglesa, y en particular a los estudiantes de España e Iberoamérica.

Vivimos en la era de la tecnología y de la información, esto nos permite compartir casi cualquier tipo de información desde puntos opuestos del planeta en un mismo instante. Por otra parte el código abierto nos permite poder leer, modificar y redistribuir el código existente que este sujeto a esta modalidad bajo

nuestra elección y criterio. En el caso concreto de *Alice*, existe la posibilidad de modificar el código ya que este es abierto, la decisión de internacionalizar el código de *Alice* surge con el objetivo de crear *Rebeca*, un software adaptable a cualquier idioma para que estudiantes de todo el mundo tengan oportunidad de poder conocer *Rebeca* en su propia lengua.

El proyecto *Rebeca s*e divide en varias etapas: Modificación del código fuente, donde se realiza el proceso de internacionalización y adaptación. En esta etapa también se modifican las imágenes del juego. Creación de ejecutables en distintas plataformas. Y la elaboración de la guía didáctica de Rebeca. En esta guía, al igual que en el conjunto del proyecto, se han cuidado todos los detalles para que aquellos que se animen a conocer *Rebeca* puedan tener una primera toma de contacto sencilla y agradable.

Con todo esto esperamos haber cumplido nuestro objetivo y que *Rebeca* sea disfrutado en castellano tanto por profesores como estudiantes y de esta manera lograr acercarles más a la informática.

Índice

1. INTRODUCCIÓN

La introducción a la programación ha sido siempre una tarea difícil para muchos alumnos y profesores, además los retos de la programación orientada a objetos suponen una dificultad añadida a la asimilación y aprendizaje de estos conceptos abstractos. Esto es especialmente preocupante cuando estudios recientes revelan que el número de estudiantes en informática ha disminuido un 60% entre 2000 y 2004 (Higher Education Research Institute, UCLA), y la tendencia en los últimos años en los países industrializados. Por este motivo, hemos decidido poner nuestro grano de arena para contribuir a evitar el desánimo y la frustración que pueden experimentar los estudiantes en su primera aproximación a la informática.

Los sistemas *Alice* y *Rebeca* suponen una alternativa para hacer la programación orientada a objetos más cercana al público joven: en *Alice* y en *Rebeca* los objetos son fácilmente reconocibles ya que están representados mediante muñecos, animales, escenografía, etc. cuyo estado puede cambiar gracias a métodos como "mover hacia adelante un metro" o "girar a la izquierda un cuarto de vuelta" es fácilmente entendible por cualquier estudiante.

Los profesores saben bien que si un alumno no está motivado para aprender, ni siquiera toda la pedagogía del mundo podrá ayudarle. La mejor técnica que tiene para aprender un alumno es encontrarse estimulado por sí mismo. Aunque podemos generar este sentimiento mediante incentivos, los sistemas *Alice* y *Rebeca* ofrecen una forma más directa: basan la programación en la actividad de contar historias.

Bajo este paradigma intuitivo y visual los alumnos desarrollan un conjunto de animaciones que cambian de estado. Es difícil de imaginar una manera más simple de transmitir el concepto del estado en que se encuentra un objeto y el uso de la programación para modificarlo. Esta es la meta principal de *Alice* y de *Rebeca*: que conceptos abstractos puedan convertirse en conceptos concretos a los ojos de nuevos programadores.

Gracias al uso de gráficos 3D, los sistemas *Alice* y *Rebeca* se comunican directamente con una generación que ha crecido inmersa en videojuegos y

películas de animación, proponiendo una alternativa eficiente en la transmisión de conocimiento capaz de rivalizar en motivación y profundidad con la enseñanza tradicional de la programación, que ha tenido escasas variaciones durante los últimos 30 años.

1.1. PRESENTACIÓN

A pesar del gran número de puestos de trabajo ofertados, las matriculaciones en las facultades de informática han caído a menos de la mitad de las que había en el año 2000, según un estudio publicado por el Higher Education Research Institute de la universidad de California-Los Ángeles en 2005.

Los profesores se enfrentan a una complicada tarea al intentar motivar a sus alumnos en el estudio. En el caso concreto de la programación orientada a objetos este problema se acentúa por tratarse de conceptos complejos y abstractos.

Los proyectos *Alice* y *Rebeca* surgen con la finalidad de paliar la falta de interés y la desmotivación que ha surgido entre los alumnos de las escuelas de informática en los últimos años, también pretende poner su granito de arena para que el número de estudiantes que optan por las carreras de informática no siga descendiendo y conseguir que surja de nuevo un interés por este tipo de ingenierías.

Alice y *Rebeca* hacen que el aprendizaje de la programación orientada a objetos se convierta en algo sencillo e intuitivo. Los objetos en Alice están representados por animaciones (animales, personajes, instrumentos, etc.) que cambian de estado a través de métodos, (mover hacia atrás, aumentar tamaño…).

1.2. ¿QUÉ ES ALICE?

Alice es un innovador entorno de programación 3D que hace que sea fácil crear una animación para contar una historia, un videojuego interactivo o un video para compartir en Internet. Es una herramienta de enseñanza de libre acceso, está pensada para facilitar una primera toma de contacto de los alumnos a la programación orientada a objetos y permite a los estudiantes adquirir los conceptos fundamentales de programación. En *Alice* y en *Rebeca* los objetos 3D (personas, animales, vehículos…) pueblan un mundo virtual y los estudiantes crean programas para animar estos objetos.

En la interfaz de *Alice* y de *Rebeca* los estudiantes arrastran y colocan elementos gráficos para crear un programa donde las instrucciones se corresponden con las sentencias estándar de lenguajes de programación orientada a objetos como pueden ser java, C++ y C#. *Alice* y *Rebeca* les permite ver, de manera inmediata, como se ejecutan sus animaciones programadas, y entender fácilmente la relación entre las sentencias de programación y el comportamiento de los objetos en sus animaciones.

Mediante la manipulación de los objetos en su mundo virtual los estudiantes adquieren experiencia en nociones típicas de un curso de introducción a la programación.

1.3. ORIGEN DE *ALICE*

Alice fue desarrollado en la universidad de Carnegie Mellon como parte de un proyecto de realidad virtual, con apoyo de la Fundación Nacional de Ciencia americana, DARPA e Intel entre otros colaboradores.

Entre los desarrolladores de Alice cabe destacar al fallecido Randy Pausch que fue el fundador del proyecto; Wanda Dann directora del proyecto, Dennos Cosgrove, investigador científico; Caitlin Kelleher que como parte de su tesis

doctoral en la universidad Carnegie Mellon dió un enfoque a *Alice* como herramienta para contar historias. Este nuevo enfoque se diseñó para captar el interés de las chicas estudiantes de instituto por la programación.

Se decidió el nombre del proyecto Alice en honor a la obra literaria *Alice's adventures in Wonderland* del autor *Lewis Carroll* (*Charles Lutwidge Dodgson*) matemático y escritor inglés del siglo XIX.

1.3.1 ORIGEN DE *REBECA*

Rebeca es el resultado de internacionalizar y adaptar el código de *Alice*. La idea surge en el departamento de Realidad Virtual de la Universidad Rey Juan Carlos, y fue ofrecido como posible proyecto fin de carrera para los alumnos de ingeniería Informática.

Después de buscar en varios departamentos algún proyecto que nos motivara Sergio Ruiz (compañero de proyecto) y yo decidimos tomar parte en el proyecto *Rebeca* y enfrentarnos a la internacionalización y adaptación del código de *Alice*.

Nuestra mayor motivación fue ser conscientes de la importancia que supone conseguir que *Rebeca* sea un software fácilmente adaptable a cualquier idioma. Consideramos que *Alice* es una potente herramienta educativa, puesto que trata de facilitar el aprendizaje a los alumnos, crear una versión adaptable supone eliminar el obstáculo del idioma a todos aquellos interesados en *Alice* que no tengan el inglés como primera lengua.

Nuestro propósito, por tanto, es eliminar la barrera del idioma y conseguir que *Rebeca* pueda ser utilizado por quién lo desee independientemente de su lengua.

1.4 VENTAJAS DE USAR *ALICE* Y *REBECA*

El enfoque de *Alice* y de *Rebeca* aporta soluciones a problemas habituales en el mundo de la enseñanza de la programación:

- El frágil mecanismo de creación de programas, particularmente la sintaxis:
  El editor de *Alice* y de *Rebeca* elimina la posibilidad de cometer errores frustrantes en las primeras etapas de la creación de programas, ya que los elementos arrastrados al editor son siempre válidos y permite a los estudiantes desarrollar una intuición para la sintaxis.

- La dificultad para ver resultados cuando se ejecuta el programa:
  Los depuradores de código son una buena herramienta, pero el enfoque de *Alice* y *Rebeca* permite comprobar el estado del programa de una forma mucho más visible. Para un estudiante es más sencillo ver que un objeto se ha movido hacia adelante en vez de hacia atrás, que ver si una variable se ha decrementado en lugar de incrementado.

- La falta de motivación para programar:
  En cursos piloto usando *Alice* y *Rebeca* se ha demostrado que los estudiantes realizan más ejercicios opcionales y asisten a más clases que con un sistema de aprendizaje tradicional. Curiosamente, grupos minoritarios en informática, como las chicas, muestran mayor interés.

- La dificultad de comprender la lógica compuesta y aprender técnicas de diseño:
  Los entornos *Alice* y *Rebeca* fomentan la creación de métodos y funciones. La analogía de hacer una película nos permite usar el concepto de storyboard, para crear un guión de texto y refinarlo hasta convertirlo en pseudocódigo.

1.5. OBJETIVOS

El objetivo de *Alice* y de *Rebeca* es que los estudiantes puedan asimilar de una forma simple y transparente los mismos conceptos que proporcionaría un curso de programación, considerando especialmente importantes las siguientes destrezas:

Pensamiento y expresión algorítmicos: ser capaz de leer y escribir en un lenguaje formal.

Abstracción: aprender cómo transmitir ideas complejas de una forma simple y descomponer el problema lógicamente.

Apreciación de elegancia: ser consciente de que, aunque hay muchas formas de resolver un problema, algunas son infinitamente mejores que otras

1.6. PROGRAMAS SIMILARES

Existen otros recursos y materiales en el panorama educativo entre los que podemos destacar *Scratch* y *Kodu*, ambas aplicaciones sirven para crear juegos a través de la programación, de manera intuitiva y sencilla.

Alice y *Rebeca* al igual que *Scratch* y *Kodu* persigue los mismos objetivos de iniciación a la programación. En un primer momento es más fácil "jugar" con *Scratch* o con *Kodu*, pero las limitaciones que tienen se alcanzan rápidamente. *Alice* y *Rebeca* suponen un mayor esfuerzo inicial, y es necesario un apoyo o referencia que acompañe al nuevo usuario para alcanzar a conocer todas las posibilidades que nos permite la aplicación.

La innovación que introduce *Alice* y *Rebeca* es utilizar un lenguaje de programación orientado a objetos directamente heredado de Java, facilitando el paso de *Alice* y de *Rebeca* a cualquier otro lenguaje de programación.

1.6.1 *SCRATCH*

Scratch pertenece al Media Lab del MIT y se creó con la finalidad de hacer la programación accesible a todo tipo de público. El objetivo principal de este software es permitir programar animaciones juntando ladrillos que contienen instrucciones de todo tipo. Con estas instrucciones podemos dar vida a elementos predeterminados creando así nuestros propios minijuegos, que pueden ponerse en el sitio web de *Scratch*.

Scratch ayuda a los estudiantes a adquirir, progresivamente, estrategias de pensamiento algorítmico que se podrán aplicar en la resolución de problemas. Es fácil comenzar a ver resultados desde el primer uso. Los estudiantes pueden realizar pequeñas actividades desde el principio y a medida que van conociendo la aplicación los estudiantes ganan en comprensión de conceptos matemáticos como expresiones booleanas, variables, coordenadas…

Scratch tiene ciertas limitaciones y no soporta los elementos que se mencionan a continuación:

- Métodos.
- Parámetros.
- Valores de retorno.
- Herencia y polimorfismo.

1.6.2 *KODU*

Es un lenguaje de programación visual hecho específicamente para crear juegos. Está diseñado para ser accesible a niños. Esta aplicación ha sido desarrollada por Microsoft y existe una versión para Xbox 360 y otra para PC.

Kodu permite diseñar mundos en tres dimensiones a partir de una serie de elementos configurados previamente por el programa. Para crear los videojuegos nos encontramos un lienzo vacío donde podemos colocar los distintos elementos, desde árboles hasta juguetes, frutas, vehículos o personajes, así como efectos de sonido y melodías.

Estos elementos interactúan con el entorno y con el jugador, desde el sistema de menús pueden establecerse distintas rutinas de comportamiento. De esta manera podemos hacer que los elementos del entorno se comporten de una u otra forma en función de la situación, delimitando acciones a partir de aspectos como el tiempo transcurrido.

Kodu puede expresar conceptos avanzados de diseño de juegos de una manera sencilla, directa e intuitiva aunque se prescinde de algunos de los elementos de programación más estrictos, como pueden ser los bucles, las subrutinas, procesos de ramificación, polimorfismo o incluso manipulación de tipos de datos como cadenas.

1.7 PROBLEMA

Alice es un software cuya interfaz está completamente en inglés. Cuando se desarrolló el código no se contempló la idea de hacer que éste fuera multilenguaje por lo que para aquellos que no dominan el idioma anglosajón *Alice* es una herramienta poco amigable. Comenzar a utilizar *Alice,* a pesar de ser intuitivo, no es demasiado sencillo, que la aplicación esté en otra lengua hace que el estudiante no se encuentre cómodo en el entorno y no se centre realmente en intentar entender el comportamiento de *Alice*, si no que se sienta abrumado y confuso por el idioma. El estudiante que acude a *Alice* con la idea de aprender programación, ve frustrado su interés al encontrarse con dificultades añadidas que echan para atrás su iniciativa.

Hay que tener en cuenta que el nivel de inglés que actualmente manejan los estudiantes de instituto en nuestro país no es lo suficientemente avanzado como para que no suponga ningún tipo de barrera a la hora de enfrentarse a una aplicación donde todo está en inglés. *Alice* esta principalmente dirigido a estos estudiantes pero para que el software tenga éxito en España y los países iberoamericanos es deseable ofrecerles un interfaz 100% en castellano, de esta

manera los estudiantes se sienten cómodos y pueden dedicarse plenamente a entender *Alice* desde el punto de vista de la programación y no del idioma.

En nuestro caso sabíamos de antemano que nos enfrentábamos a una tarea complicada, ya que el código fuente de *Alice* que proporciona la Universidad *Carnegie Mellon* ha sido generado por muchos desarrolladores a lo largo de diferentes etapas, durante más de 15 años. A lo largo de este tiempo se han escrito más de 17000 líneas de código implementadas sin cumplir una lógica homogénea y, lo más importante, sin haber tenido en cuenta que fuera a ser internacionalizado en algún momento.

Parte del trabajo necesario para internacionalizar y adaptar el extenso código de *Alice*, es realizar cambios en los paquetes de imágenes. Se han modificado todas aquellas imágenes que contenían textos en inglés, pero para obtener como resultado nuestro producto final *Rebeca* ha sido necesario cambiar también las imágenes propias de *Alice* (splash inicial de la aplicación, íconos…), y todas las imágenes que no han sido cedidas por los desarrolladores de la Carnegie Mellon. El equipo de *Alice* no nos dio el permiso para utilizar las imágenes de *Alice* porque consideró que la transformación en el código implicaba que ya no se trataba de la misma aplicación y por lo tanto no era posible utilizar la imagen corporativa de *Alice* registrada como marca en EEUU. En un primer momento nuestra idea fue llamar al código internacionalizado *Alicia*, pero tampoco fue posible debido al parecido con el original *Alice*. Debido a estos problemas de licencias el producto final se ha llamado *Rebeca a través del espejo*.

Para completar el proyecto se ha realizado una guía didáctica que sirva de apoyo tanto a profesores como a estudiantes que quieran introducirse en el mundo de la programación a través de *Rebeca*.

2. OBJETIVOS

2.1 DESCRIPCIÓN ESPECÍFICA DEL PROBLEMA

Para alcanzar nuestro objetivo principal, conseguir que *Rebeca* sea el resultado de internacionalizar y adaptar *Alice,* es muy importante que el código de *Alice* se internacionalice correctamente ya que pretendemos conseguir que una vez finalizado el proceso el software sea adaptable a cualquier idioma que se desee.

A continuación se detalla brevemente qué es la internacionalización. Se conoce con este nombre al proceso de diseñar software de manera que pueda adaptarse a diferentes idiomas y regiones sin la necesidad de realizar cambios de ingeniería en el código. Posteriormente a la internacionalización se realiza la adaptación (localización si se usa el termino directamente traducido del inglés, *localization*) que adecúa el código a las necesidades lingüísticas del mercado concreto al que está destinado.

Uno de los inconvenientes con el que nos encontramos es que el software original no permite ciertos elementos propios de la grafía del castellano, por lo tanto debemos adaptar el código para que se adapte a este juego de caracteres.

El código fuente original de *Alice* sobre el que comenzamos a trabajar pertenecía a la versión *Alice* 2.0 publicada en 2005 y contenía varios errores, también ha sido parte de nuestro trabajo solventarlos para que no se reproduzcan en nuestra versión adaptada al castellano.

Para finalizar hemos creado una guía didáctica que sirva como manual de aprendizaje tanto a los estudiantes como a los profesores a los que va dirigido *Rebeca*.

Este trabajo tiene por tanto como objetivo:

Introducir el software a institutos y universidades: El propósito de *Rebeca* es motivar a los estudiantes en el ámbito de la programación y por tanto incentivarles a la hora de elegir sus estudios. El software está orientado tanto a estudiantes de instituto con la finalidad de captar su interés a la hora de elegir una carrera, como a estudiantes de universidad para facilitarles la introducción en el mundo de la programación.

Incrementar los grupos minoritarios en las carreras técnicas: Estudios realizados en la universidad de Washington demuestran que las chicas muestran un gran interés cuando toman contacto con herramientas como *Alice* y *Rebeca*. Esto supone un reclamo para fomentar el número de mujeres estudiantes en las carreras técnicas, sector en el que hasta ahora vienen siendo grupo minoritario.

Aceptación del software y facilidad de uso: El idioma de *Alice* ha resultado ser un handicap a la hora de utilizar la herramienta. La Escuela Técnica Superior de Ingeniería Informática de la Universidad Rey Juan Carlos ha tratado de difundir *Alice* y *Rebeca* en varios eventos con un resultado muy positivo. La adaptación del software al castellano supone un reto para eliminar las dificultades causadas por el idioma y atraer a un mayor número de usuarios.

Publicidad de los participantes: Es una gran oportunidad tanto para la universidad como para los estudiantes que hemos participado en este proyecto, colaborar en la difusión de *Alice* y de *Rebeca* ya que se trata de un producto respaldado por muchos colaboradores tanto técnicamente a lo largo de los años de desarrollo como colaboradores de promoción.

2.2 TAREAS A REALIZAR

El trabajo realizado en este proyecto ha sido muy extenso. Para simplificar podemos resumirlo en los diferentes hitos que se han ido completando.

- Internacionalización del código:

  Se modifica el código fuente de *Alice* de manera que el resultado pueda traducirse fácilmente a cualquier idioma y región, es decir que *Rebeca* sea un software adaptable.

- Adaptar *Alice* al español:

  Una vez completada la internacionalización se adapta el software al castellano para que la aplicación sea difundida a través de la Universidad Rey Juan Carlos a cualquier instituto o universidad de España. En este punto, se transforman también las imágenes que contienen textos.

- Proporcionar soporte para acentos y caracteres internacionales:

  El formato necesario para poder visualizar acentos y caracteres del lenguaje español es UTF-8. El objetivo es que *Rebeca* soporte este formato.

- Escribir, diseñar y maquetar una guía didáctica:

  Se elabora una guía que sirva como referencia de aprendizaje a los usuarios de *Rebeca*. La distribución de esta guía ha de ser efectiva y asequible, utilizando varios medios de acceso.

- Corregir errores conocidos:

  El código fuente de partida, versión del código *Alice* 2.0 que es el que proporciona la Universidad *Carnegie Mellon* contiene algunos bugs, se trata de solucionar estos errores.

2.3 VIABILIDAD DEL PRODUCTO

Una vez planteados los objetivos del proyecto debemos analizar si estos son o no viables.

- **Modificación del código fuente:**

El código fuente de *Alice* está disponible en la página oficial www.alice.org, se trata de código fuente libre y por tanto puede descargarse gratuitamente. Una vez ha sido modificado el código, para poder presentar la aplicación bajo el nombre de *Alice* o *Alicia* y poder utilizar las imágenes de *Alice* es necesario que se conceda un permiso por escrito por parte de la Unversidad, este permiso nos ha sido denegado.

Por este motivo una vez finalizada la adaptación del producto, ha sido renombrado a *Rebeca a través del espejo*. Inicialmente se eligió el nombre Alicia pero por motivos de copyright no nos fue permitido utilizar éste como nombre final por ser demasiado parecido al original *Alice*. Una vez elegido el nombre *Rebeca* es posible modificar el código para internacionalizarlo y adaptarlo sin infringir las licencias de copyright.

Para evitar los problemas de licencias de uso de los paquetes de imágenes fue necesario cambiar la imagen inicial del producto, está imagen de entrada es el logo de *Alice* y está protegido como marca registrada en el territorio de los EEUU. Aunque los derechos que les otorga la marca no se extienden fuera de sus fronteras, al no darnos permiso para poder utilizarlo hemos preferido utilizar una imagen alternativa que nos ha cedido amablemente una estudiante.

Por otra parte las imágenes utilizadas en la guía han sido autorizadas por sus autores bajo la licencia *Creative Commons* Reconocimiento-No-Comercial-CompartirIgual con la que dotamos tanto a *Rebeca* como a la guía elaborada. Esta licencia permite modificar el código resultante por otros interesados.

- **Soporte para acentos y caracteres:**

El código de caracteres empleado por Java es Unicode y recoge los caracteres de prácticamente todos los idiomas representativos del mundo (son unos 65.536). Los caracteres Unicode del alfabeto occidental corresponden a los primeros 256 enteros; es decir van desde [0, 255]. A cada carácter le corresponde unívocamente un número entero perteneciente al intervalo [0, 65536] o a [0, 255] si se trabaja sólo con el alfabeto occidental. Por ejemplo, la letra ñ es el entero 164. El formato de codificación de caracteres UTF-8 permite codificar cualquier carácter Unicode.

El código *Alice* está desarrollado en Java, por este motivo la visualización de caracteres propios de la codificación UTF-8 no supone un problema y nuestro propósito es viable.

- **Redacción de una guía didáctica:**

El problema del idioma original no sólo afectaba al propio software *Alice* sino a toda la documentación y libros escritos sobre él. Por este motivo decidimos que la realización de una guía explicativa del funcionamiento del producto era una idea asequible y necesaria.

Bajo el título de *Rebeca. Aprende a programar con gráficos 3D interactivos* hemos intentado hacer una guía que sirva de referencia a todos aquellos que quieran introducirse en el mundo de la programación utilizando *Rebeca*.

Esta guía puede obtenerse a través de Internet en formato digital e impreso. Bajo la licencia *Creative Commons* los autores decidimos prescindir de royalties para evitar problemas de licencias con las imágenes utilizadas y para que la guía sea accesible a todo aquel que desee utilizarla.

- **Eliminación de errores:**

Dado que es viable modificar el código fuente de *Alice* y que existe documentación sobre los errores que se encuentran en la versión Alice 2.0. El objetivo es asequible y disponemos de información para localizar y solventar los errores.

2.4 METODOLOGÍA UTILIZADA

2.4.1 INTERNACIONALIZACIÓN

Para realizar la internacionalización del código crearemos un archivo de recursos por cada archivo .java del código que contenga cadenas de texto.

Este archivo de recursos viene a ser un diccionario que contiene pares clave-valor. A cada clave le corresponde un valor concreto que será devuelto al código cuando este proporcione una clave a cambio de su correspondiente par.

El código de Alice es muy extenso por lo que este proceso es un trabajo largo y tedioso. Además hay que tener en cuenta el número de desarrolladores que han colaborado en el código y su independencia, lo que implica cambios muy fuertes en el estilo y arquitectura del software.

El entorno de trabajo con el que se realiza este proceso proporciona una herramienta que localiza los literales (valores fijos) en el código, esto facilita en gran medida la internacionalización.

2.4.2 ADAPTACIÓN

El proceso de adaptación consiste en asignar a cada clave su valor correspondiente. Es decir en este momento se realiza la tarea de traducción en nuestro caso la clave es la cadena a traducir y el valor su equivalente en castellano.

Al haber utilizado la herramienta de internacionalización automática que facilita el compilador el proceso de adaptación ha sido simultáneo al de internacionalización, es decir, a medida que se iba generando el archivo de recursos se iba asignando el valor correspondiente a la clave introducida.

Una vez se ha internacionalizado y adaptado un archivo .java concreto es necesario compilar y ejecutar el software para verificar que el proceso se ha

realizado correctamente y no se han generado errores derivados de la modificación del código, que los valores elegidos son coherentes y además encajan en el interfaz.

2.4.3. JUEGO DE CARACTERES

Ha sido necesario realizar una labor de ingeniería inversa para solventar el problema de los caracteres. A partir de la información encontrada identificamos las instrucciones que eran necesarias para que la máquina virtual de Java pudiera ejecutar la aplicación con el juego de caracteres deseado.

2.4.4. GUÍA DIDÁCTICA

El objetivo de la guía didáctica elaborada es que sea accesible a la mayor cantidad de público posible, para ello hemos optado por que el contenido esté expresado en lenguaje sencillo y vaya acompañado de imágenes de un gran atractivo visual, de esta manera la lectura resulta amena haciendo que el aprendizaje de la herramienta sea agradable. Con los ejemplos elegidos en la guía se permite ver con facilidad las posibilidades que ofrece *Rebeca* y familiarizarse con el entorno.

Con esta guía hemos intentado ofrecer un viaje a través de *Rebeca* y esperamos conseguir un acercamiento a la aplicación por parte de los lectores.

2.4.5. ERRORES CONOCIDOS

Para solucionar los errores del código era necesario hacer una depuración del código, una vez internacionalizado y adaptado *Alice* a *Rebeca* apareció una nueva versión *Alice* 2.2 en la que algunos de los de errores se habían corregido. Aunque nosotros ya habíamos solventado parte de los errores en *Alice* 2.0, decidimos migrar R*ebeca* a la versión 2.2.

2.5. MEDIOS HARDWARE Y SOFTWARE UTILIZADOS

La máquina virtual de Java necesaria para ejecutar *Rebeca* es Java SE Runtime Environment (JRE) 5 o superior en el sistema. En la carpeta del juego *Rebeca* se encuentra esta plataforma, por lo que no es necesario realizar ninguna instalación adicional.

Para realizar la tarea de internacionalización hemos abierto el proyecto *Rebeca* en el entorno de desarrollo Netbeans 6.7.1 junto con el JDK, Java SE Development Kit.

- Recurso hardware utilizado:

  Portátil Hacer Aspire 5520 (AMD Turion 64x2 2.00 GHZ, 3GB de RAM, nVidia GeForce 7000M)

- Recursos software

  Sistema operativo Microsoft Windows XP Professional Edition SP3 – 2009
  Alice 2.0, 2.2 (Windows) y su código fuente
  NetBeans IDE 6.7.1 (Windows)
  Java SE Runtime Environment 5 (JRE)
  Java SE Development Kit 6u 16 (SDK)
  Java Media Framework API 2.1 (JMF)
  Java 3D API 1.5.2 (J3D)
  Jython 2.1 (version de Python para Java)
  Adobe Photoshop CS4
  Adobe InDesign CS4
  Exe4j 4.3
  Google Chrome
  Winzip 12.1

3. DESCRIPCIÓN

INFORMÁTICA

3.1. ESTADO DEL ARTE

Cuándo se comienza un proyecto, sobre todo si es de gran envergadura, es deseable que contemple la internacionalización del mismo con el objetivo de poder asegurar la posterior adaptación a diferentes idiomas.

En el caso de *Alice* los desarrolladores no tuvieron en cuenta que posteriormente fuera a ser necesario adaptar el código, por lo que no partimos de un software internacionalizado. Además *Alice* es el producto de muchísimas colaboraciones, lo que implica que el código no sigue una metodología rigurosa. *Alice* no está apenas comentado lo que facilitaría la comprensión de lo que se está haciendo en cada momento. Tampoco se puede acceder a una documentación sobre el mismo que nos permita acercarnos un poco más a los métodos seguidos por los programadores. Con todo esto nos enfrentamos a más de 1900 archivos java que suman más de 170000 líneas de código y que debemos internacionalizar para que pueda ser adaptado a cualquier idioma.

Alice resultó tener gran éxito y aceptación en EEUU, su lugar de origen, por lo que la idea de trasladarlo a otros países ya se ha contemplado y por tanto internacionalizar el código de *Alice* es un reto que se plantearon antes otros desarrolladores, en ningún caso el proceso de internacionalización logró llevarse a cabo.

Como ejemplos de los intentos por internacionalizar el código al castellano podemos nombrar el caso de José Garrido, profesor adjunto en Ciencias de Computación en la Universidad Kennesaw State, en Georgia o John Alexis Guerra profesor de la Universidad Tecnológica de Pereira, en Colombia. Estos dos profesores intentaron llevar a cabo la traducción de *Alice* al castellano con la finalidad de poder utilizarlo en sus clases como herramienta de aprendizaje para sus alumnos. Ninguno de los dos logró alcanzar su cometido, debido a lo extenso del proyecto, la desinformación y la falta de colaboración por parte del equipo de desarrolladores de *Alice*. Sin embargo no han sido estos los únicos intentos de

traducir el software, en el foro de la página web de *Alice* hay colaboraciones de traducciones al alemán por parte de la Universidad de Bremen (Alemania). En Kenai, pagina de acceso libre de Sun *Microsystems* aparecen iniciativas de versiones de *Alice* en árabe y en chino, aunque sin completar. Incluso por parte de nuestra Universidad, anteriormente un grupo de alumnos de postgrado intentaron llevar a cabo este mismo proyecto sin éxito.

Por parte de los propios desarrolladores también se comenzó la labor de internacionalizar el código debido a la demanda de traducciones a diferentes idiomas que requerían los usuarios, finalmente no se terminó de llevar a cabo el propósito.

3.2. APORTACIONES

Para comenzar el análisis del proyecto debemos en primer lugar familiarizarnos con el material del que disponemos.

No conocíamos que entorno de desarrollo había elegido el equipo técnico para trabajar, ni que procesos debíamos seguir para compilar/ejecutar el código. Este fue nuestro primer reto. Nos enfrentábamos a un gran proyecto de ingeniería inversa.

Las aportaciones para cada etapa del proyecto se enumeran a continuación:

- Importación del código: Labor de investigación sobre cómo integrar los archivos Java en el IDE y como compilar/ejecutar estos archivos.
- Internacionalización y adaptación: Modificación del software para hacerlo adaptable a cualquier idioma y en nuestro caso adaptarlo al castellano.
- Codificación de los caracteres: permitir que la aplicación utilice los caracteres propios del castellano.
- Paquete de imágenes: modificación de las imágenes por sus equivalentes con los textos traducidos.

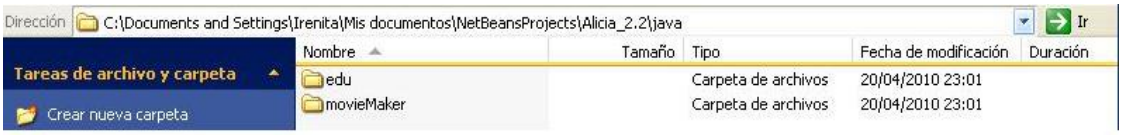
- Generación de ejecutables: creación de los ejecutables para el usuario final.
- Guía didáctica: elaboración de un libro de referencia con ejemplos destinado a facilitar el aprendizaje a los usuarios.

3.2.1. IMPORTACIÓN DEL PROYECTO

3.2.1.1. CÓDIGO FUENTE

Al descargar el código de la página oficial esperábamos encontrar a parte de todos los archivos .java que componen el software la información necesaria para comenzar a trabajar con él, (el entorno de trabajo recomendado, los requisitos necesarios para generar el proyecto y las instrucciones básicas para ser guiados en una primera ejecución del código), pero no es así. A continuación se describen los procedimientos seguidos para poder trabajar con el código descargado.

Creamos nuestra carpeta de trabajo donde descomprimimos los archivos descargados.

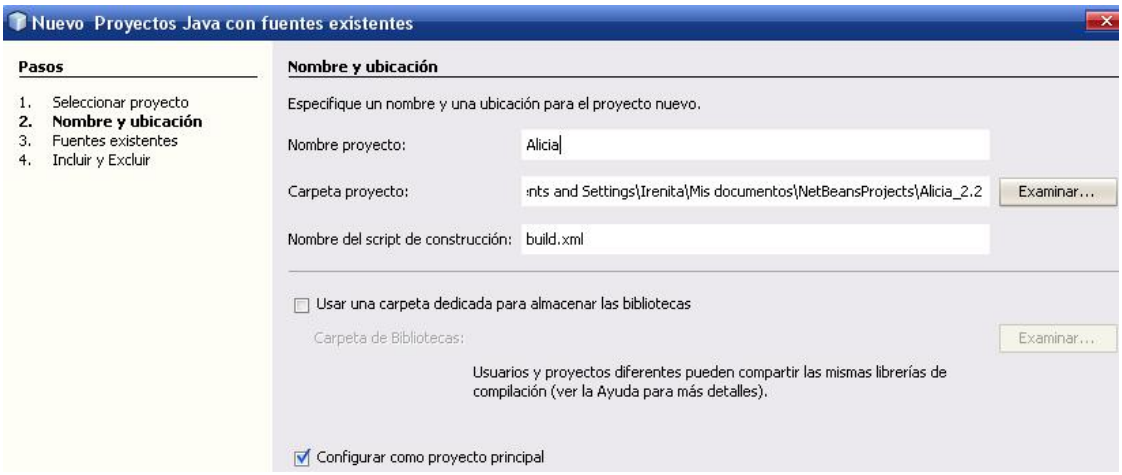


3.2.1.2. NETBEANS IDE

3.2.1.2.1. Importación del código

En NetBeans Abrimos el código eligiendo el método Proyecto java con fuentes existentes. Establecemos la ruta de trabajo antes definida, que contiene el código fuente descomprimido.

Seleccionamos la opción Configurar como Proyecto Principal y después la carpeta java de nuestro directorio en la sección fuentes existentes. Tenemos nuestro proyecto alicia (El cambio a Rebeca fue posterior, una vez había finalizado la internacionalización y se nos denegó el uso del nombre Alicia) listo para comenzar a trabajar.

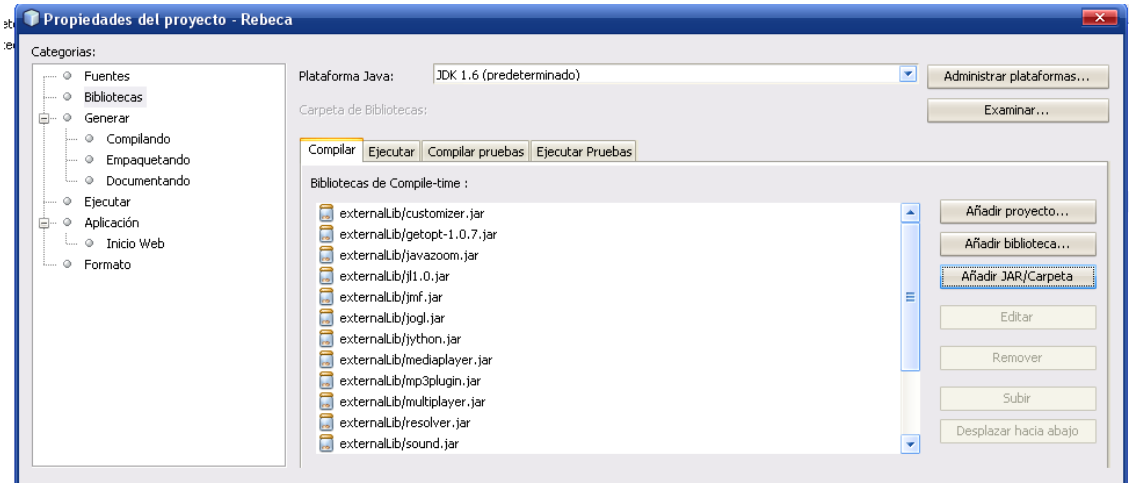


3.2.1.2.2. Compilación y Ejecución

Para compilar debemos utilizar la opción limpiar y generar MainProject de la barra de herramientas. El primer intento produjo una gran cantidad de errores indicados por el compilador, estos se producían porque faltaban paquetes a los que el código java hacía referencia.

Para solventar este problema recurrimos a el profesor Javier Sánchez quién nos ayudó en la búsqueda de estos paquetes necesarios, pero a medida que íbamos añadiendo los paquetes descargados de Internet nos dimos cuenta de que era imposible seguir este camino por resultar inabordable.

En el foro de Alice encontramos la solución, todos los paquetes necesarios para la compilación del código y que no aparecían en el código fuente podían encontrarse en el paquete del juego en la carpeta Required. Por lo tanto pudimos localizar las librerías, el JRE, y Jython. Copiamos la carpeta Required dentro de la carpeta del proyecto e indicamos al compilador donde están las librerías, (botón derecho sobre el proyecto, elegimos la opción propiedades y en bibliotecas categorías se selecciona añadir jar/carpeta).



Hemos añadido todas las librerías de la ruta: .\Alicia_2.2\Alicia\externalLib. Como se ve en la imagen elegimos el JDK 1.6.

Ahora si podemos Limpiar y generar Main Proyect, el proceso es correcto y de esta forma se genera el archivo Java(.jar)

Este archivo .jar es el que interpreta la máquina virtual de java en el momento de la ejecución. Para ejecutar el proyecto usamos la herramienta de Ejecutar *Main Project* de la barra de herrmientas.

Al intentar ejecutar la aplicación la consola de errores nos indica que faltan archivos. Encontramos en la carpeta del juego el archivo *IfAliceFailsTryThis*.config que contiene lo siguiente:

```
javapath jre/bin/java

# Initial heap size
vmparam -Xms32m

# Maximum Heap size
vmparam -Xmx512m

# Enable incremental garbage collection
vmparam -Xincgc

# vmparam -XX:+ForceTimeHighResolution

# python path
vmparam -Dpython.home=jython-2.1
vmparam -Dpython.path=jython-2.1/Lib/alice

# jars
addjars lib
```

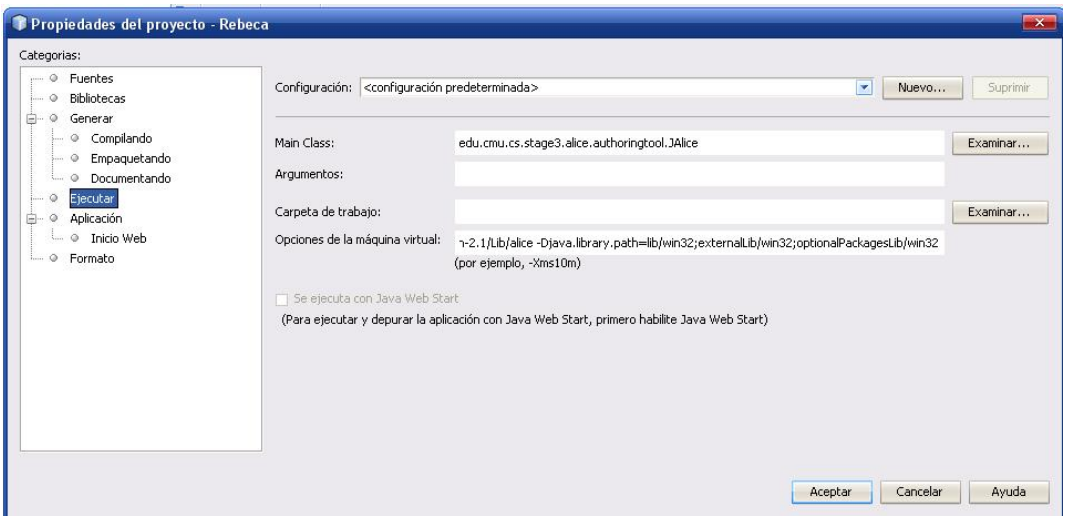
```
addjars externalLib

# jni libs
vmparam -Djava.library.path=lib/win32;externalLib/win32

mainclass edu.cmu.cs.stage3.alice.authoringtool.JAlice
```

Esto datos se corresponden con instrucciones para la máquina virtual de Java , la ruta donde se encuentra el JDk y la ubicación de las librerías externas del juego.

Dentro de Ejecutar -> Categorías en las propiedades de proyecto *Alicia* (después *Rebeca*) establecemos las opciones de la máquina virtual:



3.2.2. INTERNACIONALIZACIÓN Y ADAPTACIÓN DEL PROYECTO (I18N, L10N)

Éste es el punto clave del proyecto, como hemos detallado anteriormente la finalidad principal es conseguir un software adaptable a cualquier idioma.

3.2.2.2. HERRAMIENTA

La herramienta utilizada para realizar el proceso de internacionalización es el entorno de desarrollo NetBeans que proporciona herramientas que facilitan la tarea de la internacionalización.

3.2.2.3. MÉTODO SISTEMÁTICO

La funcionalidad de internacionalización de la que disponemos en NetBeans localiza las cadenas de texto y genera un archivo de recursos para cada archivo. Java internacionalizado.

3.2.2.3.1. Búsqueda de cadenas

Para cada archivo .java del código hay que realizar la operación de internacionalización que facilita NetBeans.

NetBeans es capaz de detectar todos los literales del código, se deja como tarea del desarrollador decidir si es una cadena que tiene que ser traducida o no. En un código tan extenso como el de *Alice* la internacionalización es un trabajo casi imposible, debido al gran número de archivos .java que hay que inspeccionar y a que cada uno de ellos lleva consigo una labor no solo de traducción si no que implica también ser capaz de entender el contenido lógico del código del archivo que se está tratando para comprender como funciona y si es posible y necesario internacionalizar las cadenas detectadas.

3.2.2.3.2. Archivo de recursos

Para cada archivo .java internacionalizado se crea un archivo de recursos asociado que se guarda en la misma ruta donde se encuentra el .java, este archivo tiene el mismo nombre que el .java pero con la extensión .properties.

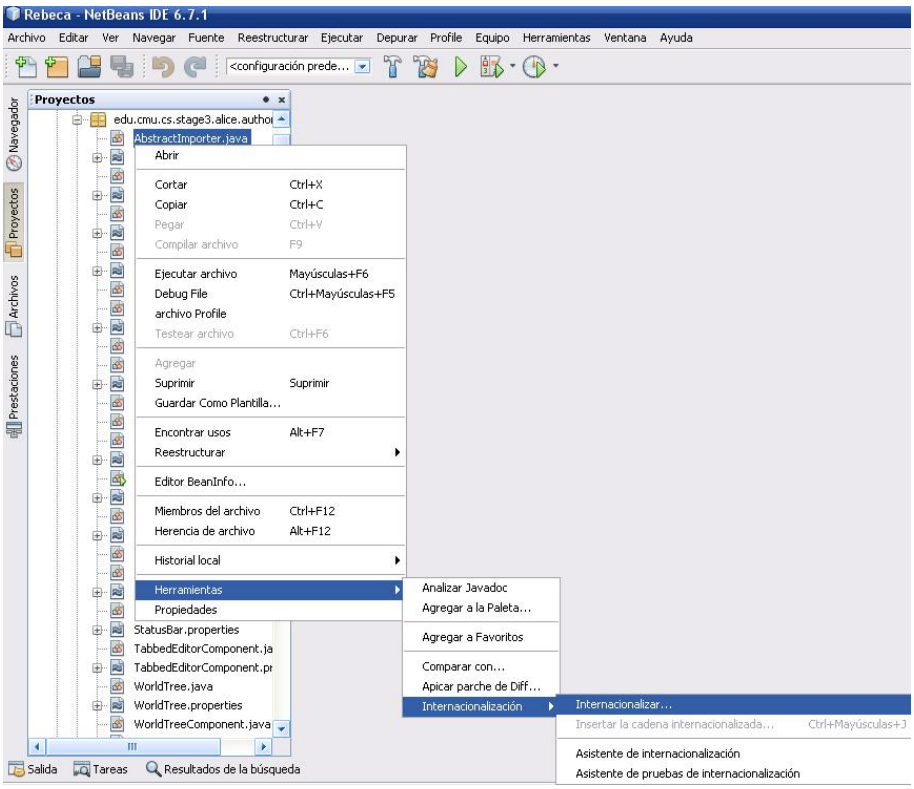
Una vez se ha localizado la cadena, se elige el texto que va a sustituirla, en el archivo de recursos se crea un nuevo par clave-valor donde la clave es la cadena original y el valor la nueva cadena por la que se va a sustituir la cadena original.

Cuando se haga la llamada al archivo de recursos se realizará con la clave como parámetro y se retornará el valor asociado.

3.2.2.3.3. Ejemplo i18n-L10n de un archivo java

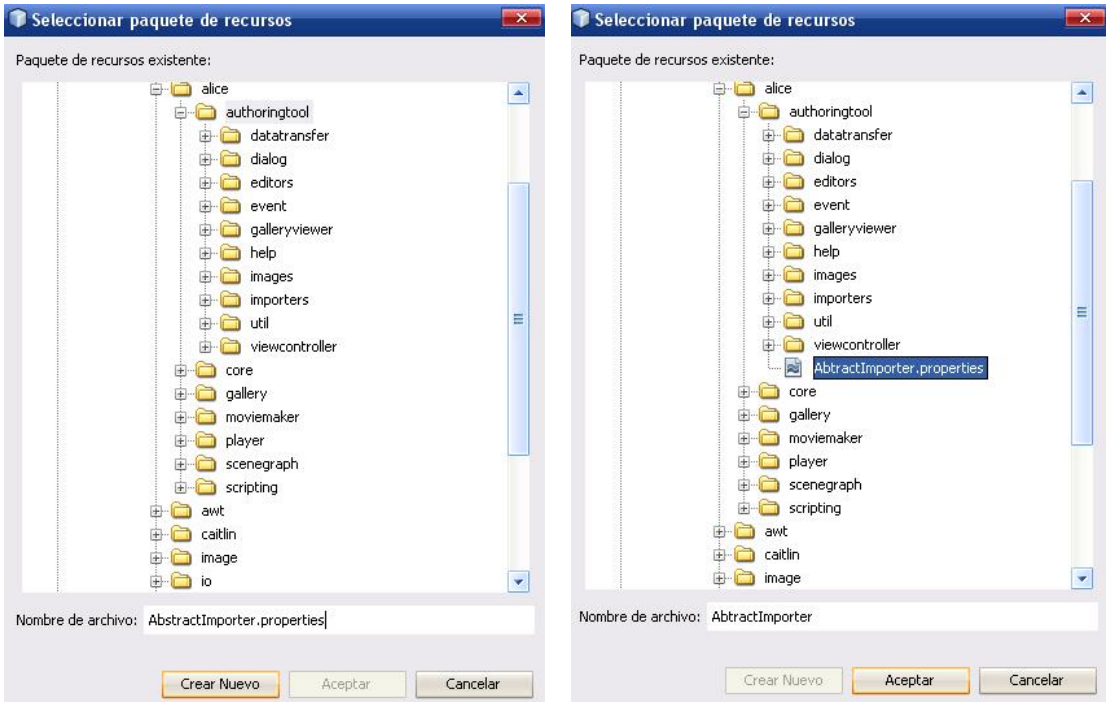
A continuación detallamos paso a paso como internacionalizar un archivo java con NetBeans.

Lo primero que hacemos es elegir el archivo java que queremos internacionalizar, sobre este hacemos click con el botón derecho y seleccionamos Herramientas Internacionalización, Internacionalizar:

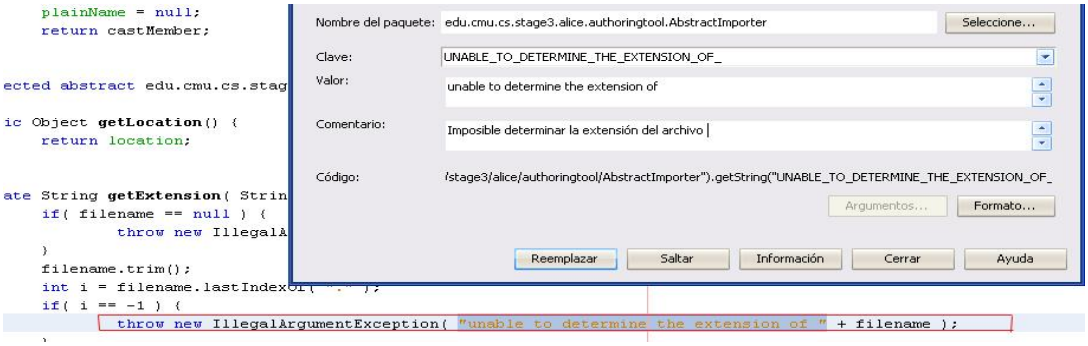


Al elegir Internacionalizar... la herramienta comenzará a buscar las cadenas de texto del archivo .java

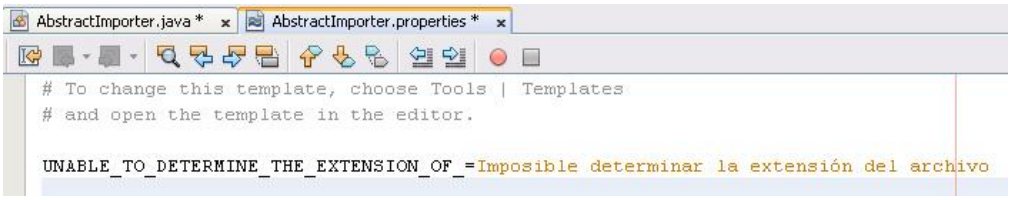
Por cada cadena detectada debemos determinar si se trata de un literal a traducir o no. En caso de que si lo sea, el procedimiento es sencillo se selecciona el archivo properties que va a estar asociado al archivo .java, si no existe lo creamos:



Solo resta indicar la traducción de la cadena, este será el valor asociado a la clave en el archivo de recursos.



Cuando le damos a Reemplazar se introduce una nueva entrada en el archivo de recursos, a la clave "UNABLE_TO_DETERMINE_THE_EXTENSION_OF" se le adjudica el valor "Imposible determinar la extensión del archivo" Este archivo es el que habrá que modificar en el caso de adaptar el código a otro idioma:



Por otra parte en el código se inserta una llamada al archivo de recursos, en este ejemplo, la línea señalada del código en la imagen anterior queda transformada de la siguiente manera:

throw new IllegalArgumentException("unable to determine the extension of " + filename);

→

throw new IllegalArgumentException(
java.util.resourceBundle.getBundle("edu/cmu/cs/stage3/alice/authoringtool/AbstractImporter").getString("UNABLE_TO_DETERMINE_THE_EXTENSION_OF_".) + filename);

Para internacionalizar el código de Alice ha sido necesario realizar este proceso en todos los archivos .java, exactamente 1992 archivos. Ha sido por tanto una enorme tarea que al principio parecía inabarcable, pero que conseguimos finalizar con mucho trabajo y organización.

3.2.2.3.4. Ejemplo i18n-L10n de un archivo de recursos

Una vez se ha internacionalizado el archivo .java y el archivo de recursos .properties ha sido confeccionado, la adaptación a otro idiomas es un proceso muy sencillo. Este era el objetivo, hacer de *Rebeca* un software fácilmente adaptable a cualquier idioma. Lo único que tendrá que hacer el interesado en adaptar *Rebeca* es asignar el valor correspondiente cada clave del archivo de recursos.

Ejemplo de adaptación al francés del archivo de recursos *Actions.properties*:

New_World= Nouveau Monde
Create_a_new_world= Créer un nouveau monde
Open_World...= Ouvrir un monde…
Open_an_existing_world= Ouvrir un monde existant
Save_World= Garder le Monde

Save_the_current_world= Garder le monde actuel

Save_World_As...= Garder le monde ...

Export_As_A_Web_Page...= Exporter en page Web ...

Export_as_a_web_page= Exporter en page Web

Import...= Importation…

3.2.2.4. PROBLEMAS DE LA HERRAMIENTA

Hay que tener en cuenta que la herramienta de NetBeans ha sido de gran ayuda pero no es perfecta.

En primer lugar no todas las cadenas que localiza la herramienta deben internacionalizarse pues muchas de ellas son literales propios del código y no de la interfaz. Al tratarse de un código muy extenso realizado por numerosos desarrolladores la lógica del mismo varía de unos archivos a otros por lo que es difícil establecer un patrón a la hora de identificar las cadenas que hay que internacionalizar. Aun tratando cuidadosamente cada archivo para no internacionalizar cadenas que no se pueden cambiar es necesario compilar y ejecutar el proyecto cada vez que se finaliza la internacionalización, de esta manera podemos asegurarnos de que no hemos cometido ningún error. Además es necesario realizar este control puesto que cuando estamos internacionalizando un archivo no sabemos dónde exactamente se encuentran las cadenas en la interfaz, por lo que puede que los textos no se adapten al tamaño que tienen disponible.

Otro problema es que no se localizan todas las cadenas, una vez se finalizó la internacionalización vimos que quedaban partes sin traducir en la interfaz, por lo que el proceso que se siguió fue localizar manualmente los literales sin traducir en los archivos .java, una vez ubicados insertar la llamada al archivo de recursos e incluir el par clave-valor correspondiente. En muchos casos no se trató de un proceso sencillo pues el literal en cuestión podía encontrarse en varios archivos y no en todos debía internacionalizarse.

3.2.2.5. DISPERSIÓN DE LOS TEXTOS

No todos los textos a traducir que contiene Alice se encuentran en los archivos .java. Después de internacionalizar los más de 1900 archivos y revisar todas las cadenas que quedaban sin traducir buscándolas en el código encontramos qué la política de desarrollo cambió en algún punto de la evolución del código diferente. Dado que Alice se ha ido elaborando por etapas, en alguna de ellas se decidió situar los literales de texto en archivos externos. En concreto se localizaron los textos en dos archivos: *Alice Style.py* y *StandardResources.py.* Ambos archivos se encuentran en la carpeta Resources del juego.

La extensión .py pertenece a Python por lo que creemos que vienen de la versión de Alice 1.0 que estaba desarrollada en Python. Estos archivos contienen numerosas cadenas y son interpretados desde java con jython, hemos traducido las cadenas de estos archivos para la adaptación del código al castellano.

Ejemplo extraído del archivo *Alice Style.py*:

```
formatMap = {
        edu.cmu.cs.stage3.alice.core.question.visualization.model.Item : "el valor de <<<subject>>>",
        edu.cmu.cs.stage3.alice.core.response.visualization.model.SetItem : "dejar <<<subject>>> = <item>",

        edu.cmu.cs.stage3.alice.core.question.visualization.array.ItemAtIndex : "el valor de <<<subject>>>[ <index> ]",
        edu.cmu.cs.stage3.alice.core.response.visualization.array.SetItemAtIndex : "dejar <<<subject>>>[ <index> ] = <item>",
        edu.cmu.cs.stage3.alice.core.question.visualization.array.Size : "tamaño de <<<subject>>>",

        edu.cmu.cs.stage3.alice.core.question.visualization.list.Size : "tamaño de <<<subject>>>",
        edu.cmu.cs.stage3.alice.core.question.visualization.list.Contains : "<<<subject>>> contiene <item>",
```

edu.cmu.cs.stage3.alice.core.question.visualization.list.IsEmpty : "<<<subject>>> esta vacío",

edu.cmu.cs.stage3.alice.core.question.visualization.list.FirstIndexOfItem : "primer índice de <item> de <<<subject>>>",

edu.cmu.cs.stage3.alice.core.question.visualization.list.LastIndexOfItem : "último índice de <item> de <<<subject>>>",

edu.cmu.cs.stage3.alice.core.question.visualization.list.ItemAtBeginning : "elemento al comienzo de <<<subject>>>",

edu.cmu.cs.stage3.alice.core.question.visualization.list.ItemAtEnd : "elemento al final de <<<subject>>>",

edu.cmu.cs.stage3.alice.core.question.visualization.list.ItemAtIndex : "elemento con índice <index> de <<<subject>>>",

[…]

}

3.2.2.6. EL IDE NO PERMITE DEPURAR ALICE

Una de las grandes dificultades al trabajar con el código de *Alice* ha sido no poder depurar. Con el IDE NetBeans tratamos de poner diferentes puntos de ruptura para trazar el código pero al ejecutar con la opción paso por paso el proyecto, se ejecuta sin parar en los puntos como si se tratara de una ejecución ordinaria.

A pesar de haber compilado y ejecutado el código en otros entornos como Eclipse no se pudo depurar en ningún caso.

Una correcta depuración del código nos hubiese sido muy útil para detectar los errores causados al internacionalizar los archivos. Determinar el funcionamiento de la aplicación a priori, sin poder trazar el código con las herramientas que contiene el IDE hace el trabajo de ingeniería inversa realizado muy complicado, ya que se trata de un proyecto extenso y complejo nada fácil de seguir.

3.2.2.7 PAQUETE DE IMÁGENES

3.2.2.7.1. Extensión de la adaptación

El proceso de adaptación no sólo afecta a las cadenas que se encuentran en el código, también es necesario modificar las imágenes de *Alice* que contienen textos o de lo contrario el software final *Rebeca* estaría parcialmente traducido. Para poder traducir los textos integrados en las imágenes es necesario editar y modificar todas aquellas imágenes que contengan cadenas.

3.2.2.7.2. Elección de la herramienta

Para modificar las imágenes contenidas en *Alice* se eligió la herramienta Adobe Photoshop CS4.

3.2.2.7.3. Realización

Para modificar los textos utilizamos la herramienta de capas de Photoshop, con esta herramienta es posible dividir la imagen en diferentes capas que en conjunto forman la imagen final.

En un primer momento se cambiaron las imágenes con el nombre *Alice* a *Aicia*, pero finalmente como ya hemos mencionado se transformaron a *Rebeca*, las imágenes del banner de inicio y el splash de *Rebeca* han sido cedidas por una estudiante que nos permitió modificar su dibujo original y utilizarlo en el juego.

3.2.2.7.4. Ejemplos de concordancia

3.2.3. CODIFICACIÓN DE LOS CARACTERES

3.2.3.1. PROBLEMA Y SOLUCIÓN

Una vez finalizado por completo el proceso i18n-L10n, lo que nos llevo más de un año de trabajo, es necesario crear los ejecutables para poder difundir nuestra aplicación *Rebeca* totalmente adaptada.

Mientras ejecutamos el juego desde NetBeans no detectamos este problema porque el IDE permite elegir la codificación de caracteres con la que se quiere ejecutar la aplicación, pero cuando creamos los ejecutables de manera que no fueran dependientes del IDE nos dimos cuenta de que no se visualizaban correctamente algunos caracteres típicos del castellano como la "ñ" y los acentos.

Se observó de donde podía venir el problema y nos dimos cuenta de que las cadenas que no se visualizaban correctamente se encontraban en los archivos.py por lo que pensamos que el problema estaba al interpretar los archivos Python.

Buscamos información en diferentes foros tratando de encontrar una solución para la correcta interpretación de los archivos .py.

La primera posible solución que probamos fue añadir a la cabecera de los archivos Python la línea: *(#-*- coding: utf-8 -*-)* y escribir delante de cada cadena codificada en utf-8 la expresión (u), pero el resultado fue el mismo, seguían sin visualizarse correctamente algunos caracteres. Este resultado no fue del todo sorprendente pues en todos los foros sobre Python consultados se afirmaba que desde una versión bastante anterior a la nuestra ya existía soporte para codificar caracteres en utf-8. Después de esta experiencia nos inclinamos por buscar otras posibles causas del problema.

Pensamos que la solución podríamos encontrarla en la configuración IDE puesto que esta permitía la correcta interpretación de los caracteres utf-8. Como no nos era posible ver las instrucciones exactas de ejecución de *Rebeca* nos llevamos la aplicación a Linux. Mediante comandos de Shell como *ps –ef | grep java* pudimos

ver la línea exacta de ejecución, los parámetros y opciones que usaba NetBeans. De esta manera encontramos la opción necesaria para la máquina virtual de Java:

Dfile.encoding = UTF-8

El problema se solucionó cuando incluimos este parámetro al generar los ejecutables.

3.2.3.2. EXTENSIÓN DEL PROBLEMA

En la barra de herramientas de *Rebeca* se encuentra la opción de exportar la animación creada en formato html. Al exportar el código se producía el mismo error en la visualización de caracteres utf-8.

Observamos que la página web html se construía a partir de una plantilla que se encontraba en dos ubicaciones, una en el código y otra en el juego:

AppletTemplate.html en la carpeta /etc/ del juego.

ExportCodeForPrintingContentPane.java en la carpeta edu/cmu/stage3/alice/authoringtool/dialog/ del código fuente

La solución en este caso fue más sencilla, bastó con añadir en la cabecera del código html de los archivos la siguiente etiqueta:

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

3.2.4. GENERACIÓN DE EJECUTABLES Y EMPAQUETADO

A continuación, una vez explicado cómo se solucionó el problema de los caracteres, vamos a detallar el proceso de generación de ejecutables y su empaquetado.

Es importante señalar que el éxito de una aplicación también depende de las facilidades de acceso a la misma que se proporcionen. Una vez generados los ejecutables se procede a empaquetar los archivos necesarios para que no se requiera por parte del usuario ningún tipo de instalación ni de la propia aplicación *Rebeca* ni de ningún otro software adicional.

3.2.4.1. EJECUTABLE .JAR MULTIPLATAFORMA

Un archivo .jar es un formato de fichero que contiene recursos necesarios (clases, imágenes, sonidos…) para ejecutar una aplicación Java y que se utiliza para distribuir esta aplicación. Para ejecutar un archivo .jar lo más importante es tener la máquina virtual de java en el equipo, de manera que el sistema operativo pueda llamarla y que ésta ejecute el fichero.

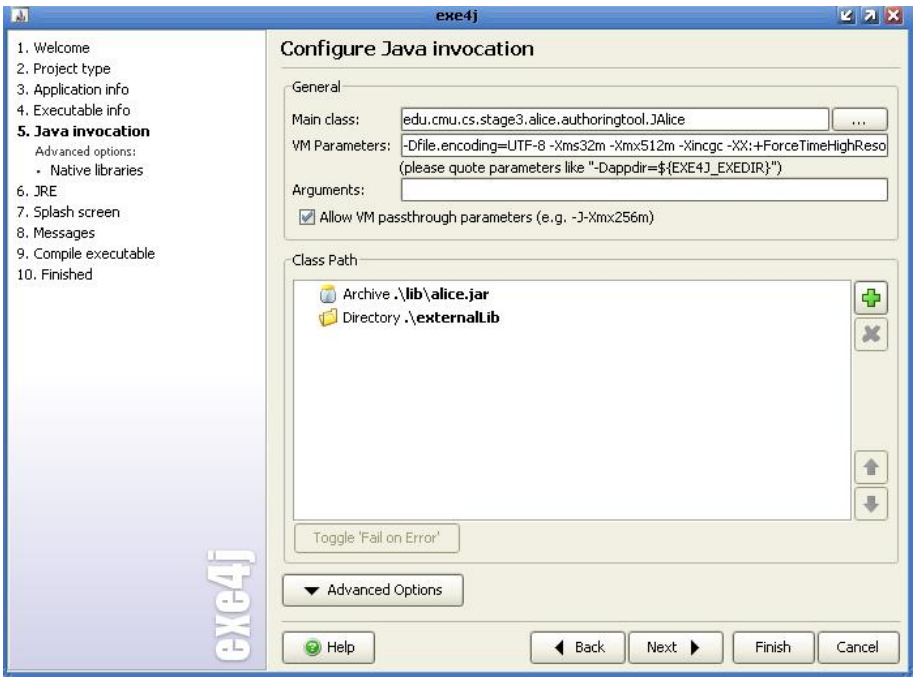
El problema en nuestro caso es que para ejecutar el archivo .jar tenemos que indicar una serie de parámetros a la máquina virtual ya sea desde NetBeans o desde consola, esta tarea debe ser transparente para los usuarios, que no tienen porque conocer NetBeans ni estar familiarizados con la ejecución de aplicaciones desde consola. Por tanto tenemos que crear ejecutables que realicen la llamada al archivo .jar enviándole también estos parámetros a la máquina virtual de Java.

Vamos a generar dos archivos .ejecutables uno para la plataforma Windows y otro para Macintosh.

3.2.4.2. EJECUTABLE .EXE WINDOWS

Para generar el archivo .exe de Windows hemos utilizado la herramienta exe4j. A través de esta herramienta podemos generar el ejecutable que contiene los parámetros necesarios para la máquina virtual de Java.

La generación del ejecutable .exe con la herramienta exe4j, sólo requiere seguir unas sencillas instrucciones de configuración, entre las que podemos destacar la ruta del fichero .jar de *Rebeca*, la imagen que queremos de icono en nuestro ejecutable y como puede verse en la siguiente imagen la cadena de parámetros indicados a la máquina virtual de Java.

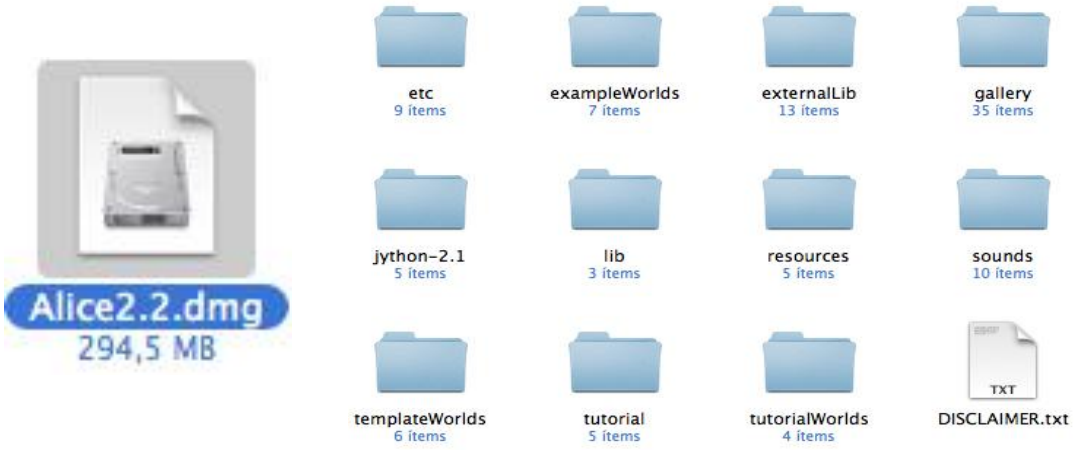


Una vez creado el archivo .exe se realiza el empaquetado. Seguimos el modelo de empaquetado de los creadores de Alice, que distribuyen una carpeta comprimida en formato zip. Esta carpeta incluye el JRE (Java Runtime Environment) necesario para la ejecución de la maquina virtual de Java, de esta manera se evita que el usuario tenga que descargarlo el mismo.

Creamos el archivo *Rebeca.zip*, sólo es necesario descomprimir este archivo en la carpeta que se desee y hacer doble clic sobre el ejecutable *Rebeca.exe* para lanzar el juego.

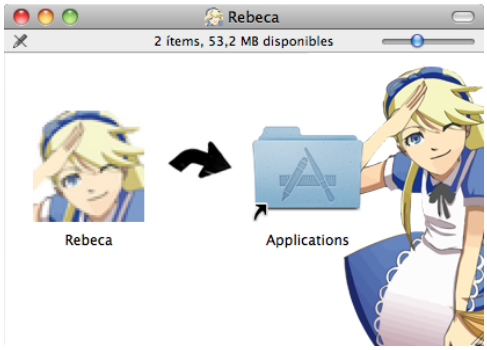
3.2.4.3. EJECUTABLE MACINTOSH

En este caso la solución es más sencilla, la página oficial de www.alice.org proporciona el archivo .dmg para ejecutar *Alice* en Macintosh, al extraer este .dmg podemos ver todos los archivos comprimidos que contiene por tanto solo tenemos que sustituir los archivos necesarios de *Rebeca,* como el archivo .jar generado al ejecutar *Rebeca* y los archivos e imágenes que hemos modificado al realizar la internacionalización del código. Una vez sustituidos estos archivos volvemos a generar, gracias a SimplyDisk, el .dmg de *Rebeca* para Macintosh.



El empaquetado se realiza con SimplyDisk, que permite incluir un menú autoejecutable que facilita la tarea de descomprimir Rebeca en la carpeta Aplicaciones que especificamos como predeterminada.

El resultado obtenido a partir de la herramienta SimplyDisk es el siguiente menú autoejecutable:



3.2.6. GUÍA DIDÁCTICA

Creemos que para que *Rebeca* tenga mayor aceptación entre los usuarios a los que va dirigido es conveniente que estos puedan documentarse de alguna manera, el fin de esta guía es que sirva como manual de referencia durante la iniciación en el manejo de la interfaz.

La guía contiene una breve introducción, dónde se explica el origen de *Alice* y el por qué de *Rebeca.* Después hay un desarrollo extenso que explica el funcionamiento de *Rebeca* a partir de un ejemplo práctico. Decidimos que la mejor manera de acercar al usuario a *Rebeca* era haciéndole participar de manera activa con la aplicación. Esperamos que la guía sirva a los estudiantes y a los profesores que quieran utilizar *Rebeca* como herramienta en sus clases.

La guía ha sido realizada utilizando las herramientas Adobe Photoshop para el tratamiento de imágenes y Adobe InDesing para la maquetación y edición gráfica.

El trabajo que ha requerido la realización de esta guía ha sido muy laborioso, en primer lugar la elaboración del ejemplo es una tarea que afrontamos teniendo en cuenta distintos factores, queríamos que resultase sencillo para los usuarios pero que a su vez fuera completo. Por otra parte nunca habíamos redactado textos que fueran a distribuirse para llegar a un gran número de lectores, teníamos que ser muy cuidadosos y conseguir que los textos fuesen lo más sencillos y atractivos posibles.

Las imágenes de la guía también han sido elegidas con mucho cuidado, en este punto cabe agradecer las colaboraciones de los artistas que han cedido sus imágenes para poder ser utilizadas en nuestra guía:

Hackett, Sabrina Elisabeth: Wood trail II.
Arrais, Elsio: Follow the rabbit.
Ilinca, Roman: Alice in Wonderland.

3.2.6.1. DISTRIBUCIÓN

Una vez finalizada la guía tuvimos que decidir cómo íbamos a distribuir la guía del juego *Rebeca.* Para conseguir llegar al mayor número de personas decidimos no cobrar ningún tipo de royalties de autor. Otros factores que tuvimos en cuenta fueron: que la guía pudiera distribuirse en forma electrónica y editorial, y poder publicar bajo una licencia de libre distribución

Después de informarnos, sopesamos las diferentes alternativas y decidimos inclinarnos por Lulu, (www.lulu.com). Lulu es un servicio de autopublicación, que imprime y edita bajo demanda, ofrece ISBN gratuito y además distribuye tanto en España como en Latinoamérica que son las dos áreas de mayor interés para nuestro software adaptado *Rebeca*. Además Lulu permite descargar una copia de la guía en PDF desde su página.

Puesto que hemos decidió prescindir de royalties, el único precio que tendrán que pagar aquellos que quieran adquirir la guía de *Rebeca* serán los gastos de envío y la parte que Lulú cobra por editar y distribuir el producto.

3.2.7 LICENCIAS

Nuestro próposito en todo momento es que el trabajo realizado pueda ser disfrutado por todos, es decir sea libre, por este motivo hemos elegido acogernos a las licencias CC (Creative Commons). Este tipo de licencia es bajo el que se distribuye tanto la guía de *Rebeca* como el código internacionalizado *Rebeca*.

Tanto la guía como el juego de Rebeca podrán ser descargados libremente desde la página www.gmrv.es/rebeca-es.

3.2.8 ERRORES CONOCIDOS

Como ya hemos comentado a lo largo de la memoria cuando comenzamos el proyecto lo hicimos a partir de la versión *Alice 2.0* pero durante el periodo de internacionalización la comunidad *Alice* saco una nueva versión *Alice 2.2* en la que se habían eliminado ciertos bugs conocidos de la versión anterior, esta versión también incluía algunas nuevas funcionalidades como poder exportar en video las animaciones creadas por los usuarios.

En este momento teníamos que tomar la decisión de seguir con la versión de *Alicia* que estábamos llevando a cabo y eliminar los errores o bien migrar el trabajo realizado a la nueva versión. Finalmente optamos por la segunda opción. Esto supuso volver a revisar todo el código de *Alice 2.2* realizando la internacionalización del código tal y como se ha descrito anteriormente con la singularidad de que esta vez los archivos de recursos propios de la adaptación ya los teníamos y sólo en algunos casos fue necesario añadir nuevos pares clave valor.

Después de haber migrado el código la Universidad Carnegie Mellon nos denegó el uso de las imágenes de *Alice* por tratarse de una marca registrada en EEUU esto supuso cambiar las imágenes de *Alice* a las nuevas imágenes de Rebeca tanto en el juego como en la guía.

El equipo de Alice también sugirió que el nombre de *Alicia* era demasiado parecido al de *Alice*, y que deseaban no ser confundidos con otros grupos de desarrollo, por lo que consideramos conveniente cambiar de nombre a la aplicación. Para esto tuvimos que modificar tanto las imágenes como el código para que todos los textos *Alicia* pasaran a ser *Rebeca.*

Tanto en la creación de *Alicia* como en la migración a la versión 2.2 y durante la transformación a *Rebeca* han surgido muchos y diferentes errores que hemos ido solventando poco a poco examinando el código en detalle.

4. CONCLUSIONES

4.1 RESULTADOS ALCANZADOS

Desde el momento en que se decidió afrontar el proyecto de internacionalización y adaptación al castellano de *Alice* el principal objetivo fue que el resultado del trabajo facilitara a los estudiantes el aprendizaje de la programación orientada a objetos.

Rebeca es un software completamente internacionalizado y perfectamente adaptado al castellano, que puede adaptarse fácilmente a cualquier idioma, por este motivo podemos afirmar que se trata de una herramienta educativa que puede utilizarse en cualquier país independientemente de su lengua.

No sólo la interfaz de Rebeca está completamente en castellano, además las sentencias típicas de programación (if, for, while…) también se han traducido para dar lugar a un lenguaje propio de *Rebeca* con el que los estudiantes de enseñanza media se sienten cómodos.

Se han realizado varios cursos y talleres piloto con Rebeca en Juvenalia y en el fesTICval, estas experiencias han tenido una gran acogida y han sido muy positivas. Los alumnos de educación secundaria que estuvieron "jugando" con Rebeca se mostraron muy receptivos con el programa y sorprendidos al comprobar que la programación es una disciplina realmente creativa y puede llegar a ser muy divertida.

Consideramos que la facilidad de uso de *Rebeca*, los positivos resultados obtenidos con los jóvenes estudiantes y el hecho de que sea un producto que se puede obtener de forma totalmente gratuita y accesible a cualquier público objetivo, hacen que *Rebeca* pueda llegar a ser una iniciativa con éxito, y esperamos que sea aprovechada por los profesores de enseñanza media para iniciar a sus alumnos en el mundo de la programación.

Hasta el momento vemos nuestros objetivos cumplidos, Rebeca cuenta con el apoyo de la Universidad Rey Juan Carlos que está trabajando en una página web

desde donde pueda descargarse tanto el juego como el código fuente, dando oportunidad también a que la aplicación pueda seguir creciendo siendo modificada por desarrolladores que estén interesados.

A través del foro de *Alice*, donde se conoce nuestro trabajo, hemos recibido varias peticiones desde diferentes partes del mundo interesándose por Rebeca y con el deseo de poder adaptar el software a su lengua, queremos destacar entre los interesados a Martina Rosenboom quién nos contacto desde la Universidad de Bremen con la intención de adaptar rebeca al alemán, también nos han contactado desde Francia y países de Ibéroamérica.

Los sistemas educativos occidentales a los que pertenecemos, lentamente, van cubriendo los objetivos planteados, pero nuevas dificultades salen al paso y requieren la búsqueda de caminos diferentes a los ya conocidos, en los que juegan un papel fundamental las TIC y los nuevos recursos como Rebeca.

Por todo lo anterior podemos afirmar que Rebeca es un nuevo recurso educativo que permite llevar adelante acciones encaminadas al logro de un aprendizaje más efectivo, a desarrollar aptitudes para la sociedad del conocimiento y, en definitiva, a mejorar la educación elevando la calidad del proceso de enseñanza–aprendizaje.

Además es una herramienta innovadora útil para desarrollar nuevas metodologías en el aula, a distancia (*on-line*) o semipresencial (*blended*), por no hablar de la educación no formal, que representan hoy un campo poco explorado donde la tecnología juega un papel determinante en el aprendizaje.

5. BIBLIOGRAFÍA

- **Página Web oficial del proyecto Alice.**

  http://alice.org/

- **Tutorial de internacionalización mediante NetBeans IDE.**

  http://netbeans.org/kb/docs/java/gui-automatic-i18n.html

- **Página oficial del entorno de desarrollo integrado NetBeans. Proporciona diversos documentos explicativos y tutoriales para proyectos en los que interviene el IDE.**

  http://netbeans.org/kb/index.html

- **Página para desarrolladores Java de Sun Microsystems.**

  http://java.sun.com/

- **Página oficial del proyecto educativo Scratch**.

  http://scratch.mit.edu/

- **P. DANN Wanda; COOPER Stephen; PAUSCH Randy: Learning to Program WITH**

  Alice.USA: PEARSON Prentice Hall, 2006. 319 p. ISBN 0-13-187289-3.

- **GARRIDO Jose: Alice: The Programming Language.**

  USA. Paperback, 2008. 60 p. ISBN 076375059X

- **Página de publicación y venta de material bajo derechos de autor.**

  www.lulu.com

- **Proyecto de traducción de Alice al árabe.**

  http://kenai.com/projects/arabic-alice/members

- **Proyecto de traducción de Alice al chino.**

  http://kenai.com/projects/alice/forums/forum/topics/1789-About-Translate-Alice-3-to-Chinese

- **Página del foro de la comunidad Alice dedicas a la internaciolización del programa.**

  http://www.alice.org/community/group.php?do=discuss&group=&discussionid=15

  http://www.alice.org/community/showthread.php?t=1061

  http://www.alice.org/community/showthread.php?t=171

  http://www.alice.org/community/showthread.php?t=155

  http://www.alice.org/community/showthread.php?t=32

- **Página dedicada a crear una comunidad de artistas que muestren sus obras.**

  http://www.deviantart.com/

  HACKETT Sabrina Elizabeth: Wood Trail II. Disponible en:

  http://hitomii.deviantart.com/art/Wood-Trail-II-87175896

  ARRAIS Elsio: Follow the rabbit. Disponible en:

  http://frostwake.deviantart.com/gallery/

  ILINCA Roman: Alice in Wonderland. Disponible en:

  http://twinkygreenpenguin.deviantart.com/art/Alice-in-Wonderland-109028071

  LANIER Alexandria: I am the mad hatter. Disponible en:

  http://heidicrimyoungblood.deviantart.com/art/i-am-the-mad-hatter-143332819

- **Página oficial del proyecto Alice adaptado al español (Alicia).**

  http://www.gmrv.es/alice-es

6. APÉNDICES

6.1. LICENCIA DE DISTRIBUCIÓN

Licencia a la que se acogen tanto el software Rebeca como la guía de Rebeca:

Consultada en Enero 2010 en la url:

http://creativecommons.org/licenses/by/3.0/es/legalcode.es

6.2. GUÍA DIDÁCTICA REBECA