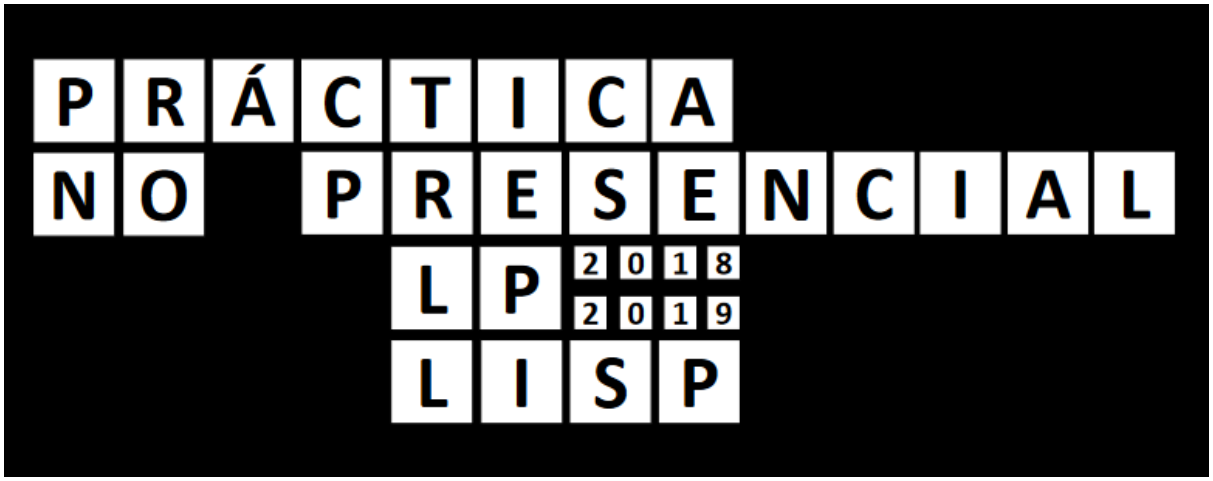


Miguel Albertí Pons
Juan Miguel Frau Ripoll

06/05/2019
Grupo x02 (jueves)

1. Enunciado

El objetivo de la práctica es llevar a cabo el desarrollo de una aplicación, mediante los conocimientos obtenidos del lenguaje de programación LISP, que simule el juego de las palabras ocultas.

El juego de las palabras ocultas consiste en ir descubriendo palabras mediante la introducción de letras que la componen. El usuario introducirá letras hasta que adivine la palabra o se equivoque 17 veces. Por cada letra fallida se visualizará de forma aleatoria una sección de las dieciséis que componen la imagen de la palabra correspondiente.

La puntuación por palabra será obtenida mediante una puntuación inicial de 255 puntos a la que se le irá restando 15 puntos por letra fallada.

En la parte superior de la interfaz gráfica se mostrará la puntuación total y el número de jugadas realizadas junto al nombre que deberá introducirse al inicio del programa.

Todas las palabras a adivinar estarán almacenadas en el archivo palabras.txt y tendrán una imagen correspondiente con el mismo nombre con extensión img y con unas dimensiones de 200x200 píxeles.

El usuario, una vez acaba una jugada, debe poder elegir si quiere seguir jugando o no. En caso de contestar afirmativamente se cogerá una nueva palabra del archivo de texto y se volverá a jugar. En caso de respuesta negativa o que no queden más palabras por adivinar en el archivo, se dará por finalizada la partida y se guardará la puntuación junto al nombre en el fichero resultados.txt.

2. Archivos proporcionados y ejecución

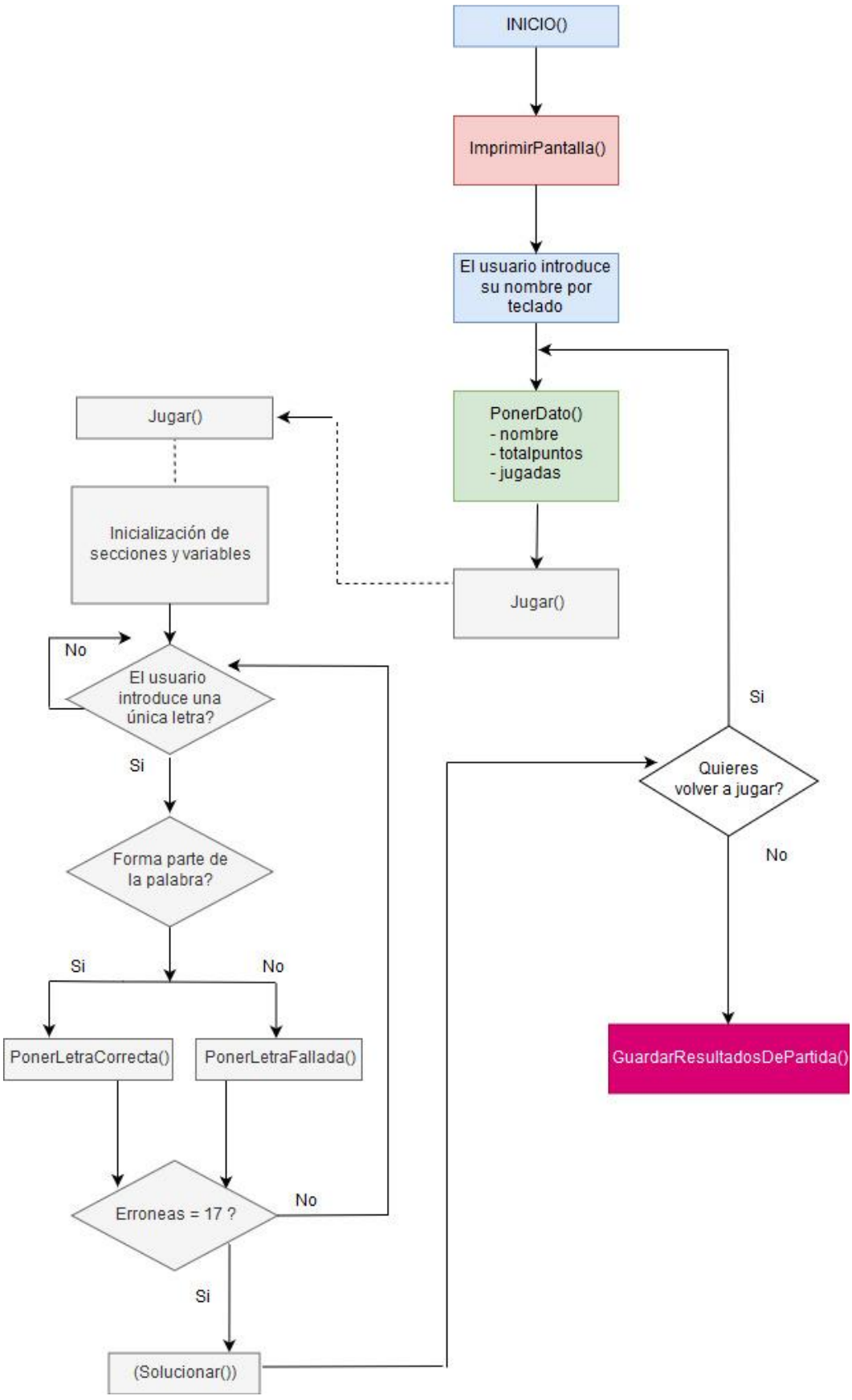
Los archivos necesarios para llevar a cabo la ejecución son los siguientes:

- Archivo **palabras.txt** encargado de almacenar las palabras que se adivinarán.
- Archivo **resultados.txt** encargado de almacenar los resultados de cada partida que se juegue.
- Archivo **main.lsp** donde se encuentra todo el desarrollo de la aplicación.
- Archivo **init.lsp** proporcionado por el profesor y que será necesario para la utilización de funciones como strcat.
- Carpeta **img** con todas las imágenes que se utilizarán.
- Programa de conversión (**Converter**) realizado en Java que convertirá imágenes de extensión bmp a imágenes de extensión img. Este programa está formado por un método principal de llamará al método convertir con el nombre la imagen por parámetro y el método convertir que de lo que se encarga es de guardar los bytes leídos en un nuevo archivo img pero quitando los primeros bytes correspondientes a la cabecera.

Para llevar a cabo la ejecución es necesario que todos los archivos y carpetas anteriormente mencionados estén ubicados en el mismo directorio que el intérprete de Lisp. Además, es necesario que no se le cambie ningún nombre a ningún archivo ni carpeta, sino el programa no responderá correctamente.

Para su ejecución se deberá introducir en el intérprete el comando (load 'main.lsp) y, posteriormente, la llamada a la función principal (inicio).

3. Descripción funcional



Al cargar el programa en el intérprete de LISP se llamará a la función inicio() donde están recogidas las principales funciones del programa.

En primer lugar, pintaremos la interfaz que verá el usuario mediante la función *imprimirPantalla* y por la cual nada más empezar tendrá que introducir su nombre y dicho nombre será pintado mediante la función *ponerDato*. Seguidamente, se accederá a la función *jugar* donde se empezará a adivinar palabras y donde el mensaje que aparece en el cuadro de diálogo cambiará y empezaremos a pedir letras.

En la función *jugar* se entrará en un bucle donde cada iteración será una jugada. En cada iteración se realizará una inicialización tanto de secciones como de variables que serán las siguientes:

- Se leerá la palabra a adivinar del fichero.
- Lista de las letras de la palabra una por una para saber cuándo se han adivinado todas las letras.
- Borrado de las secciones de letras falladas y de las letras de la palabra de la jugada anterior.
- Pintado de las secciones anteriores que han sido borradas y pintado de negro toda la sección de la imagen que corresponde a la palabra.
- Se pondrá a 0 la variable *erroneas* que contiene el número de veces que se han equivocado para calcular posteriormente la puntuación y para saber si han llegado a los 17 errores.
- Se pondrá la lista *listaletrasfalladas* sin ningún elemento para ir añadiendo las que se han fallado y evitar repeticiones de pintado de letras ya pintadas.
- Se realizará la lista *intentos* utilizada para pintar la sección elegida aleatoriamente.
- Inicialización de la imagen correspondiente a la palabra con la función *initImagenGrande* en la cual se guardará la información de cada pixel de la imagen en un array de 200x200 donde cada casilla tendrá los colores rgb de cada píxel.

En este punto, se entrará en un segundo loop en el cual se pedirá introducir una letra. Una vez se introduce la primera letra, la parte lógica del programa empezará a funcionar y se realizará un control de errores comprobando que se ha introducido una sola letra, si es así, se mirará si pertenece a la palabra, en caso negativo se llamará a la función *ponerLetraFallada* que incrementará el número de errores, pondrá la letra en el apartado de letras falladas de la interfaz y mostrará un fragmento de la imagen de 16 segmentos; en el caso de que la letra se encuentre en la palabra que el usuario debe encontrar, se añadirá dicha letra en la parte de la interfaz correspondiente mediante la función *ponerLetraCorrecta*.

Si el usuario ha introducido 16 letras erróneas se mostrará la imagen de 16 segmentos en su totalidad y el usuario tendrá una última oportunidad para acertar la palabra.

Finalmente, se saldrá del loop interno una vez haya acertado o fallado la palabra, de este modo, se le pedirá por el cuadro de diálogo si desea volver a jugar. En caso afirmativo, se realizará una nueva iteración del loop principal proporcionándole otra palabra e imagen nueva, en caso negativo, se saldrá del loop principal y se llamará antes de finalizar el programa a la función *guardarResultadosPartida* que escribirá su puntuación en un fichero.

4. Descripción de funciones

defun pintarColor. Pinta una imagen en color, de extensión .img y de cualquier tamaño en las coordenadas indicadas.

defun pintaCuadrado. Pinta un rectángulo mediante las coordenadas indicadas.

defun cuadroImagen. Pinta la parte de la interfaz de la imagen que corresponde a la palabra, iniciándola toda a negra.

defun titulo. Pinta en la interfaz gráfica la sección del título.

defun datosPartida. Inicializa pintando en la interfaz gráfica las secciones donde estarán los datos del jugador, puntos y jugadas realizadas.

defun palabrapar. Pinta los rectángulos de las palabras que tengan una longitud par para conseguir que estén centradas.

defun palabraimpar. Pinta los rectángulos de las palabras que tengan una longitud impar para conseguir que estén centradas.

defun cuadroPalabra. Pinta los rectángulos que forman una palabra en blanco.

defun ventanaComunicacion. Pinta la ventana de comunicación en la interfaz.

defun letrasFalladas. Pinta la sección de letras falladas en la interfaz.

defun imprimirPantalla. Función encargada de llamar a todas las funciones encargadas de pintar secciones de la interfaz.

defun ponerDato. Función que inserta un dato, ya sea el nombre de usuario, puntos o jugadas, en la interfaz colocando las imágenes de las letras o números correspondientes.

defun guardarResultadosPartida. Guarda los datos del usuario, nombre y puntuación, en el fichero "resultados.txt" cuando un jugador acaba la partida.

defun borrar. Borra la sección con los parámetros que se le indican. Se utiliza para borrar las secciones de la palabra y de las letras falladas cada vez que se acaba una jugada para posteriormente se vuelvan a pintar, el resto de secciones no se borrarán.

defun jugar. Función que tendrá prácticamente toda la lógica del programa. Se encargará entrar en un loop donde cada vez que se quiera realizar una jugada leerá la palabra e inicializará su correspondiente imagen, volverá a pintar las secciones de la interfaz gráficas necesarias para volver a

jugar y las variables que deben de restablecerse cada jugada. Entrará en un loop en el cual se comparar la letra introducida con las de la palabra, saldrá de este loop si adivina todas las letras o falla 17 veces. Finalmente se volverán a cargar los nuevos datos con la puntuación total y se pedirá si quiere volver a jugar, en caso negativo saldrá del primer loop.

defun solucionar. Completa la palabra al llegar a los 17 errores y completa la imagen correspondiente a la palabra tanto si la adivina como si no.

defun ponerletracorrecta. Pone la imagen de una letra al adivinarla.

defun initImagenGrande. Encargada de inicializar la imagen correspondiente a la imagen que tiene que adivinar el usuario. Lo que hace es leer el archivo con la imagen y guarda en un array bidimensional de 200 x 200 cada uno de los píxeles de la imagen y donde en cada casilla de los pixeles estará formado por un nuevo array de tres posiciones con los valores de los tres colores rgb.

defun ColocarSeccion. Encargada de ir completando la imagen de 16 segmentos. De forma aleatoria elegirá un número de 0 a 15 y lo pintará, este número será eliminado de una lista creada con todas las secciones para evitar repetir números con el random.

defun ponerletrafallada. Mirará si la letra errónea ha sido introducida anteriormente comparando con una lista que contendrá todas las letras falladas introducidas anteriormente. En caso de que no se haya introducido se pintará la imagen de dicha letra.

defun imprimirLinia. Mantiene la línea de comunicación en la posición de la ventana correspondiente evitando que el resto de elementos se muevan.

defun inicio. Función principal encargada de llamar al resto de funciones para que el programa se ejecute de forma correcta.