Solitario Web

David Tolosa Alarcón

Proyecto Final Asignatura IST
Máster Universitario en Ingeniería de Telecomunicación
Curso 2020-21

Índice general

1

Introducción

El proyecto que se ha elegido a realizar como trabajo final de esta asignatura, IST, es la implementación de un solitario web a partir del trabajo realizado en las prácticas. Se ha finalizado el solitario inicialmente implementado y se ha añadido nuevas variantes del solitario (Spider, Spider 2 colores) para así añadir nuevas funcionalidades al proyecto.

Figura 1.1: Snapshot de la página web del proyecto

Este documento tiene como fin explicar brevemente los elementos incluidos así como su relación. Teniendo el proyecto una estructura diferenciada en parte HTML (página web), CSS (estilo) y JavaScript (funcionalidad juegos) se va a proceder a explicar cada una de éstas, intentando solamente indicar como se ha procedido y qué cosas nuevas se han incluido.

2

HTML & CSS

La parte del diseño de la página web se ha basado en el periódico web que se ha implementado en las prácticas, habiendo modificado mínimamente ésta parte en el proyecto, solamente se ha variado ciertos elementos para hacerlos adecuados al sentido de la página y se han incluido nuevos elementos que se describirán a continuación.

2.1. HTML

En el proyecto existen cinco ficheros HTML que se corresponden a las páginas a visualizar en el proyecto. El fichero **index.html** es desde dónde se inicia la página web y tiene la distribución que se indicó y se dió en las prácticas para el periódico.

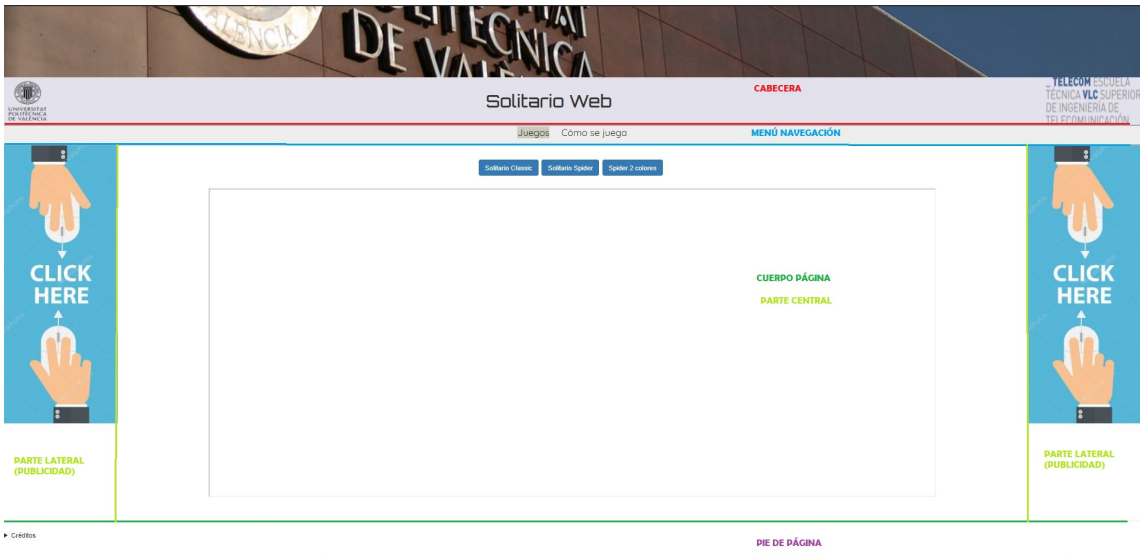

Figura 2.1: Elementos HTML presentes en la página web

- **Cabecera :** Muestra una imagen de fondo con un encabezado con el nombre de la página y los logos de la UPV y ETSIT. Éstos incluyen elementos ancla que direccionan a las respectivas páginas web de las entidades.
- **Elemento de Navegación :** Dispuesto horizontalmente debajo de la cabecera en el que se indican las distintas páginas relacionadas al proyecto. Este proyecto incluye la página principal que es dónde se incluyen los juegos y una página dónde se explican las normas.
- **Cuerpo de la página :**Se encuentra divido en tres partes (*tres divs*), la parte central dónde se incluirá el contenido de la página y los elementos laterales dónde se incluye la publicidad.
- **Pie de página :**Como pie de página se incluyen los créditos, siendo éstos mi nombre con la opción de enviarme un email, igual que en las prácticas.

La página principal incluye además en la parte central del cuerpo un elemento *iframe* que es dónde se carga las páginas html de los juegos así como unos elementos tipo *button* para seleccionar entre cada juego.

La otra página que se puede direccionar desde el menú de navegación, **instrucciones.html** tiene la misma distribución explicada antes e incluye una breve explicación de las normas de juego de cada variante del solitario.

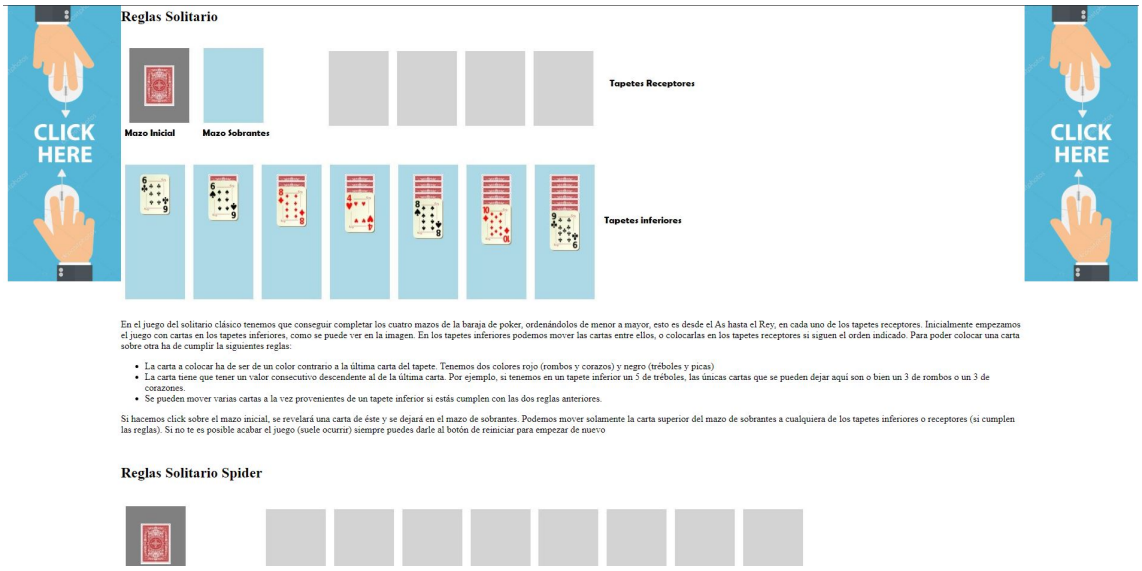

Figura 2.2: Snapshot página normas

Para ello, dentro de la parte central del cuerpo de la página se han incluido varios artículos (uno para cada variante del Solitario implementada) con una explicación acompañada de una imagen para hacer más sencilla su comprensión.

Los otros tres ficheros html , **solitario.html**, **solitario_spider.html** y **solitario_spider_2.html** son aquéllos donde se va a visualizar cada respectivo juego. Éstos incluyen, de manera similar a las prácticas, los elementos necesarios para la representación del juego:

- **Elementos título (*headers*) :** Con el título del juego así como un marcador del tiempo total de juego transcurrido y un marcador de movimientos. Para ello estos elementos incluyen un contador (*elemento span*) que se implementará en javascript.
- **Elemento mesa :** Un elemento tipo *div* dónde se incluirán los distintos tapetes (*también divs*) necesarios en función del juego utilizado. Los tapetes harán de elemento dónde se puedan colocar las cartas del juego. A diferencia de las prácticas, se han eliminado los elementos span dónde se incluirían los contadores de cartas de cada tapete.
- **Elemento *button* :** Un botón (color rojo) para poder reiniciar el juego cuando se quiera

Debido a que la variante Spider tiene una distribución distinta a la variante clásica del Solitario, se han creado dos distribuciones distintas para cada juego, aun así es necesario crear un archivo html para cada juego ya que cada uno tiene una funcionalidad distinta (fichero javascript distinto) y hay que indicarlo adecuadamente en cada caso.

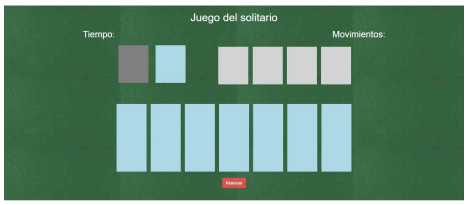

Figura 2.3: Fichero HTML para Solitario Clásico

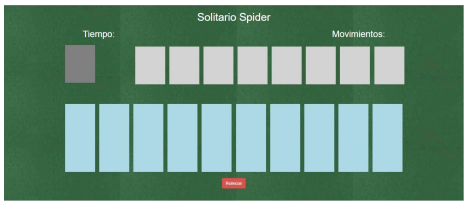

Figura 2.4: Fichero HTML para variantes Spider

Descrito brevemente, en el solitario clásico disponemos del tapete inicial o de robo (tapete gris oscuro), un tapete de sobrantes (al lado del incial), cuatro receptores (grises) y siete tapetes inferiores dónde se desarrolla el juego.

En la variante Spider no tenemos un tapete de sobrantes (no es necesario ya que cuando se roba se añaden las cartas a los tapetes inferiores) y se dispone de más tapetes receptores ya que se juega con ocho mazos en vez de cuatro. También en esta variante se dispone de más tapetes inferiores (diez).

2.2. CSS

Los ficheros CSS incluidos en este proyecto corresponden con los desarrollados durante las prácticas, las modificaciones en éstos han sido mínimas. Los ficheros utilizados son:

- **general.css :** Aquí se incluyen las especificaciones de estilo generales para las páginas index.html e instrucciones.html , indican el estilo de la cabecera y del menú de navegación así como la organización de los elementos en la parte central del cuerpo y la visualización *sticky* de los elementos de publicidad. También incluye el estilo del pie de página.
- **articulos.css :** Este archivo es para el estilo de los artículos creados en instrucciones.html, indica el tamaño y tipo de letra del título y del cuerpo del artículo así como la disposición y tamaño de las imágenes incluidas.
- **juegos.css :** Estilo para la visualización del elemento *iframe* dentro de index.html dónde se cargan las páginas de los juegos, también incluye el estilo de los botones dónde se selecciona el juego.
- **solitario.css :** Fichero para el estilo del fichero html del solitario, indica el tamaño y tipo de letra de los títulos, tamaño y distribución de los tapetes así como sus colores.
- **solitario_spider.css :** Variante del fichero solitario.css dónde se ha indicado una distribución distinta de los tapetes superiores para los ficheros html del Solitario Spider y Spider 2 Colores.

3

JavaScript

Esta parte del proyecto corresponde con la implementación de la funcionalidad del juego del Solitario así como de las variantes Spider y Spider 2 colores. Para ello se ha basado en el trabajo realizado durante las últimas prácticas, por lo que se van a obviar algunas explicaciones de funcionalidades como el contador de tiempo y movimientos, ya que éstas venían incluidas en los ficheros de las prácticas.

Se han creado tres ficheros: **solitario.js**, **solitario_spider.js** y **solitario_spider_2.js**, cada uno implementa la funcionalidad de cada tipo de juego. Estos ficheros son bastante parecidos ya que comparten la estructura y muchas funciones, algunas han sido modificadas o incluidas nuevas para completar la funcionalidad de cada variante. Por tanto se va a explicar el funcionamiento de manera general (esto es para el solitario clásico) y si es necesario se incluirá un inciso explicando la variación incluida en el solitario Spider o Spider 2 colores.

La manera de mostrar las funcionalidades implementas va a ser indicando y explicando las rutinas asociadas a los posibles eventos que se generen durante el juego así como las funciones principales desarrolladas que permiten el correcto funcionamiento del juego.

Inciso : Respecto a los archivos obtenidos de las prácticas para la implementación del juego, en concreto las imágenes de las cartas, se ha añadido la imagen de la cubierta/tapa/parte trasera de la carta para una mejor visualización del juego.

3.1. Rutinas asociadas a eventos *onClick*

En el juego el programa presta atención a dos posibles eventos *onClick*. Uno de ellos es al **hacer click sobre el botón de reiniciar** que lo que realiza es la llamada a la función **comenzar_juego()** y por tanto reinicia el tablero y juego en sí. Esta rutina ya se incluía en la práctica.

La otra rutina que se ha implementado está asociada a **hacer click sobre el tapete inicial**, lo que se ha querido implementar es que cuando hacemos click sobre él robemos carta. Para ello se ha implementado la función **pasar_carta()**, lo que realiza es mover una carta del mazo inicial al de sobrantes, si el mazo inicial está vacío hace una llamada a otra función, **mover_mazo(*tapete_procedencia*, *tapete_destino*)**, para que coloque todas las cartas del mazo de sobrantes de nuevo al inicial. Ésta es la manera de robar cartas desarrollada en el juego.

Variante Spider : Para las variantes Spider no se dispone de un mazo de descartes dónde van a parar las cartas robadas, si no que cada vez que se roba se coloca una carta en cada tapete inferior, para implementar esta funcionalidad distinta se ha creado la función **pasar_cartas()** asociada también al evento *onClick* en el tapete inicial.

3.2. Rutinas asociadas a eventos *onDragStart*

Los eventos *onDrag* están asociados a las cartas para que éstas se puedan coger y sean jugables. La función llamada durante el evento *onDragStart* es **al_mover()** y fue implementada durante la práctica, en esta función se recoge información de la carta jugada como la procedencia y el id de ésta, además se relaciona directamente el elemento completo de la carta a la variable **carta_jugada** para utilizarlo posteriormente en otras funciones.

Además se ha implementado unas funciones para **actualizar los parámetros *draggable* de las cartas**, éstas son :

- **configurar_carta(mazo)**: Esta función hace *draggable* solamente la carta superior (*top*) de un mazo, no permitiendo coger ninguna carta por debajo de ésta. Esta función es utilizada para actualizar los parámetros *draggable* de los mazos de los tapetes receptores y del de sobrantes.
- **configurar_carta_inferior(mazo)**: Esta función actualiza los parámetros *draggable* de las cartas de los mazos de los tapetes inferiores, permitiendo no sólo coger la carta top si no coger múltiples cartas si éstas están ordenadas de acuerdo a las normas del solitario. De manera que podemos mover grupos de cartas de acuerdo a las posibilidades del juego. Para cada variante del Solitario se ha implementado esta función de forma distinta ya que cada juego permite de manera distinta mover múltiples cartas de un tapete inferior a otro.

Estas funciones son llamadas cada vez que se coloca una carta/cartas en un tapete, de manera que siempre tenemos actualizado el parámetro *draggable* de las cartas del mazo, permitiendo solamente mover las cartas permitidas en función del juego. Además estas funciones actualizan la **visualización** de las cartas que se encuentran tapadas o boca abajo, cuando una de estas cartas se encuentra en la posición top de un mazo, ésta se hace visible y se muestra la imagen de la carta correspondiente.

3.3. Rutinas asociadas a eventos *onDrop*

Estos eventos están asociados a cuando se suelta la carta sobre un tapete, para configurar los tapetes y que escuchen a estos eventos y realicen la correspondiente llamada a la rutina indicada se ha creado la función **configurar_tapete(destino)** que establece la relación de las rutinas con el evento *onDrop*. Esta función es llamada cada vez que se suelta una carta para todos los tapetes.

La función llamada durante el evento es **soltar(e)**, ya creada en las prácticas. Esta función recoge información del tapete destino y con la información indicada durante el evento *onDragStart* sobre la carta seleccionada permite evaluar las diferentes posibilidades de dejar una carta sobre un tapete según la variante del juego.

Muy resumidamente, en el Solitario Clásico se permite colocar cartas sobre los tapetes receptores si cumplen con la normas del solitario (del mismo palo y en orden ascendente) y se permite colocar una carta o múltiples sobre un tapete inferior si éstas están ordenadas consecutivamente de manera descendente y en color inverso.

Variante Spider : Para las variantes Spider no se permite colocar cartas sobre los tapetes receptores y para colocar cartas en los tapetes inferiores éstas han de ser de valor consecutivo descendente sin importar el color (tanto en Spider como en Spider 2 colores).

3.4. Funciones para comprobar las normas del Juego

Para simplificar el código así como su visualización se han creado unas funciones que evalúan si se puede realizar el movimiento de colocar cierta carta en cierto tapete. Estas funciones simplemente evalúan la acción indicada según las normas del juego y devuelven *True* o *False* en función de si está permitido o no el movimiento.

Éstas funciones son :

- **se_puede_añadir(*carta*, *tapete_receptor*)**. Esta función evalúa si se puede dejar una carta en un tapete receptor. Según las normas del solitario clásico se tienen que dejar sobre cada tapete receptor las cartas en orden ascendente (*esto es empezando por el As*) y cada tapete solo tiene que tener cartas de un mismo palo.
- **se_puede_añadir_inferior(*carta*, *tapete_receptor*)**. Aquí se evalúa si se puede dejar una carta sobre un tapete inferior. Las reglas a seguir son: Si hay una carta sobre un tapete inferior se puede dejar solamente la carta con un valor consecutivo inferior al de ésta y de color contrario si es solitario clásico o sin importar el color para el Spider. Si no hay ninguna carta sobre un tapete inferior solamente se puede colocar un rey si es el solitario clásico o cualquier carta para el Spider.

También se han creado funciones para comprobar si se ha completado el juego que de manera similar a las anteriores devuelven *True* o *False* si el juego se ha completado o no. Esta función es **ha_acabado()** y lo que evalúa es si se han completado todos los mazos en cada tapete receptor.

Variante Spider : Para los solitarios Spider además se ha creado otra función, **hay_escalera(mazo)**, que evalúa si existe una escalera del Rey al As del mismo color en el mazo indicado (de los tapetes inferiores). Si esta función devuelve *True* se llama a la función **guardar_mazo(tapete)** para que deje el mazo completado (escalera Rey al As) sobre un tapete receptor disponible (que esté vacío).

3.5. Funciones para la colocación de cartas

Se han implementado una serie de funciones para colocar una carta de acuerdo a la representación que se le quiere dar según el tapete. También se ha creado una función para eliminar una carta del mazo y del tapete.

- **quitar_carta(*tapete*, *carta*)**. Quita la carta indicada del mazo del tapete así como elimina al nodo hijo de la carta asociado al tapete.
- **colocar_carta(*tapete*, *carta*)**. Añade la carta al mazo del tapete indicado y la relaciona con él (hace a la carta hijo del tapete). En esta función las cartas se colocan centradas en el tapete y una encima de otra. Es la forma de representación de cartas para los tapetes superiores (inicial, sobrantes, receptores).
- **colocar_carta_inferior(*tapete*, *carta*)**. Realiza la misma función que la anterior pero con una diferente representación. Aqui las cartas se muestran centradas horizontalmente pero cada una se muestra separada verticalmente de arriba a abajo. Esta es una forma tradicional de mostrar las cartas en el solitario para los tapetes inferiores.
- **colocar_multiples_cartas(*tapete*, *carta*, *index_carta*)**. Esta función es para colocar múltiples cartas en los tapetes inferiores y de acuerdo a la representación de las cartas para los tapetes inferiores.

3.6. Función para iniciar Juego

comenzar_juego(), ya implementada en las prácticas, es la función que realiza la disposición inicial de los elementos del juego. Se ha modificado para que deje cartas sobre los tapetes inferiores según la variante del solitario (antes solo dejaba las cartas sobre el mazo inicial) de manera que tenemos una distribución de las cartas igual a como se colocan al principio en el juego del Solitario y Solitario Spider.

4

Manual Usuario. Conclusión

La forma de poder utilizar este proyecto es bastante simple, si accedemos a él desde la página *index* podremos desde allí hacer click a cualquiera de cada una de las variantes del solitario. Si queremos previamente conocer las reglas podemos hacer click sobre *Cómo se juega* en el menú de navegación para una breve explicación del juego.

Una vez seleccionado el juego se mostrará este en la página principal. Si hacemos click sobre el mazo de robo/inicial (aquél con las cartas tapadas) robaremos cartas. Para mover las cartas entre los distintos tapetes nos sirve con hacer click sobre ella y arrastrarla, si vemos que la carta no aparece arrastrada (se ve semi-transparente) eso es porque el movimiento de esa carta no se nos está permitido. Para colocar una carta arrastrada sobre un tapete nos sirve con soltar el click izquierdo del ratón. Si se quiere en cualquier momento se puede pulsar el botón de reiniciar para empezar de nuevo (es bastante probable que lo tengas que utilizar ya que se pierden muchas partidas). Si se completa el juego se mostrará un mensaje a través del browser indicando que hemos ganado junto al tiempo total y la cantidad de movimientos realizados.

4.1. Conclusión

La implementación de este proyecto ha sido un trabajo interesante ya que relaciona varias partes, con distintos lenguajes de programación y que requiere de una estructura y organización previa para poder realizarlo correctamente y para evitar la dificultad de comprensión del trabajo o fallos que se vean arrastrados durante la realización de éste. Por mi parte, elegí la continuación del trabajo del solitario porque me pareció interesante y porque prefería la implementación de algo en javascript a utilizar servicios API o crear una página web con distinto estilo/configuración. Aún así y como he dicho, este trabajo relaciona varias partes y he podido aprender durante el transcurso de la asignatura (sobre todo a nivel práctico) como implementar las distintas partes que componen una página web, que es al fin y al cabo una herramienta más que me puede ser útil en un futuro a nivel profesional (y por qué no a nivel personal si quiero realizar algo parecido).