Práctica 5

Juego del solitario I

Integración de Servicios Telemáticos
Máster Universitario en Ingeniería de Telecomunicación
ETSIT-UPV

Ismael de Fez Lava
Marisol García Valls
F. J. Martínez Zaldívar

Índice

1. Introducción y objetivos

En la presente práctica vamos a empezar a utilizar JavaScript junto con HTML y CSS para diseñar un sencillo juego. Aprovecharemos el periódico digital diseñado hasta ahora, como marco de trabajo. Así pues, este juego se mostrará como una parte del contenido de la sección `Ocio` del periódico digital, manteniendo por tanto el resto de características del periódico realizado hasta ahora. Emplearemos varias librerías `JavaScript` con las que pretendemos aumentar al cantidad de recursos y herramientas a utilizar en las prácticas. Concretamente, emplearemos la librería `Bootstrap` y otra necesaria por esta: `jQuery`. `Bootstrap` tiene varias funcionalidades, entre ellas, mostrar botones con un aspecto estético mejorado respecto a la representación estándar.

En esta serie de prácticas vamos a seguir utilizando lo estudiado en clases de teoría. Aunque las especificaciones pueden ser incompletas, deberían ser suficientes para conseguir un aspecto bastante similar a las referencias gráficas que aquí puedan aparecer.

Durante el transcurso de la práctica, el alumno podrá apreciar que una misma idea puede implementarse de múltiples formas; por ello se pretende que se reflexione sobre estas alternativas y elija la opción que le parezca más conveniente, tratando de justificarla adecuadamente en forma de comentarios dentro del código HTML, CSS o JavaScript.

Asimismo, es probable que deba consultar algunos detalles de implementación todavía no impartidos en la asignatura; la intención de ello es ejercitar y valorar oportunamente la competencia transversal de *aprendizaje permanente*.

1.1. Organización de la práctica

Los grupos de prácticas pueden ser de uno o de dos alumnos. Se sugiere que estos grupos sean estables a lo largo de todas las prácticas, y esta especialmente, ya que aprovecha los resultados de la práctica anterior y además tendrá continuidad y reaprovechamiento de todo el trabajo realizado para la siguiente. Se dedicará una sesión de dos horas para la realización de la presente práctica.

Se seguirá trabajando con el mismo repositorio GitHub que la práctica anterior (`Peri-digi-`), acumulando nuevos *commits* con los que se va a ir dotando al periódico de un mayor contenido. Antes de finalizar la sesión de prácticas en el laboratorio, debe realizarse un *commit/push* con etiqueta `pr5SesionFin` con el contenido de la práctica, esté en el estado que esté. Si quedan detalles pendientes de finalizar, recuérdese que se disponen de 72 h

adicionales para finalizarlos, en cuyo caso, el *commit* que se considere que representa el estado final de esta práctica deberá tener la etiqueta `pr5FinDef`.

Algunos ficheros (imágenes de la baraja de cartas, alguna plantilla a utilizar, ficheros con estilos CSS, ...), son proporcionados como complemento al enunciado de la práctica para mantener en los resultados un aspecto homogéneo y para que la dificultad y extensión esté contenida, estando ubicados en el subdirectorio hermano `html`.

1.2. Material necesario

El material necesario para llevar a cabo la práctica será un ordenador de sobremesa o un portátil, con cualquier sistema operativo convencional, un editor de texto plano y un navegador (preferiblemente alguno que posea y del que se conozcan y manejen capacidades de depuración).

Muy probablemente será necesario consultar informción de HTML, CSS y JavaScript. Recuérdese que un sitio bastante útil es
`http://www.w3schools.com`, entre otros.

2. Explicación del juego

Entre la práctica actual y la siguiente se implementará una versión elemental del juego del solitario. En esta práctica abordaremos la parte relativa a la organización de las cartas en los mazos y algún detalle más, y en la siguiente, la mecánica del juego.

Respecto a las reglas del juego, tomemos en cuenta las siguientes consideraciones:

- La baraja de cartas será la de póker con los cuatro palos (corazones, rombos, tréboles y picas), con el as, cartas del dos al diez, la jota *(jack)*, la reina y el rey.

  Importante: inicialmente se debe trabajar con un subconjunto de cartas, para no hacer el juego demasiado denso o largo y poder hacer pruebas de forma rápida y ágil, como por ejemplo: la carta del diez, la de la jota, la de la reina y la del rey de cada palo.

  Una vez que el juego esté mínimamente validado, debería ser algo extremadamente fácil conmutar al juego completo de cartas, por lo que todo el software debe preconcebirse para este cambio de circunstancias.

- Existirán 6 tapetes (superficies que albergarán mazos de cartas): el tapete *inicial*, el tapete de *sobrantes* y cuatro tapetes *receptores*, tal y

como puede observarse en la figura 3, que albergarán sendos *mazos de cartas* a lo largo de la partida.

- El juego consiste en coger la carta de arriba, bien del mazo ubicado en el tapete *inicial*, bien del tapete de *sobrantes*, e intentar depositarla en alguno de los mazos de los cuatro tapetes *receptores*, siempre y cuando se cumplan ciertas condiciones.

- El uso de las cartas del mazo de cada tapete es el siguiente:

  - Tapete *inicial*: conteniendo al principio el mazo completo y barajado aleatoriamente. Las cartas se cogerán por arriba, de una en una. La intención es intentar colocar la carta cogida en alguno de los tapetes *receptores* siempre y cuando se cumplan ciertas condiciones.
  - Tapete de *sobrantes*: empleado para depositar temporalmente en su mazo las cartas provinientes del mazo del tapete *inicial* que no pueden ponerse sobre ninguno de los mazos correspondientes a los denominados tapetes *receptores*. Además, la última carta depositada en el mazo de este tapete sirve de fuente alternativa de cartas al mazo del tapete *inicial*, para intentar depositarla sobre algún mazo de los tapetes *receptores* en cualquier momento, siempre y cuando se cumplan ciertas reglas tal y como ocurriría si proviniera del mazo del tapete *inicial*.
  - Cuatro tapetes *receptores*: donde se irán depositando sobre los mazos respectivos, las cartas en orden decreciente (comenzará obligatoriamente con el rey como primera carta y finalizará con la última) y alternando los colores (rojo y negro) en la secuencia de cartas que se va depositando en cada mazo.

- Cuando se agoten las cartas del mazo del tapete *inicial*, las que queden en el tapete de *sobrantes* serán automáticamente barajadas y dispuestas de nuevo en el tapete *inicial*, volviendo otra vez a comenzar con estas cartas restantes.

- El juego finaliza cuando no queda ninguna carta ni en el tapete *inicial* ni en el tapete de *sobrantes*. Cuando ocurra tal situación, se mostrará una ventana de aviso indicando el fin del juego y proporcionando información sobre la partida como por ejemplo tiempo de juego o cantidad de movimientos realizados.

En la figura 3 puede observarse el juego en un supuesto estadio inicial.

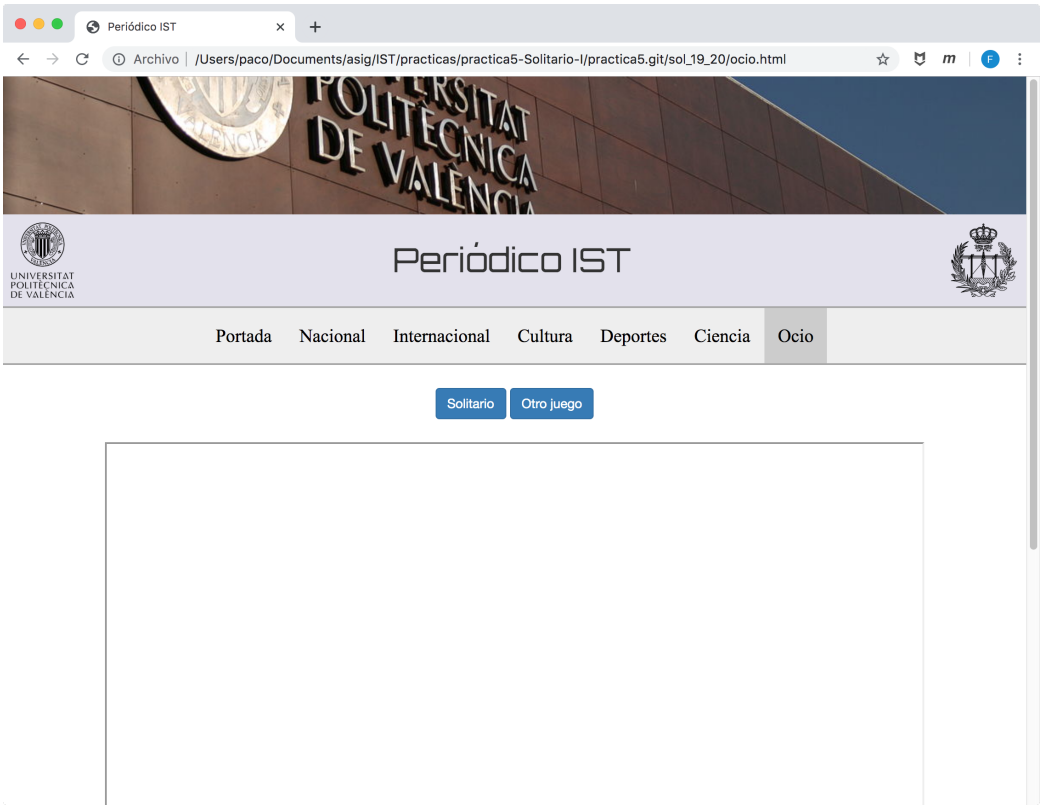

Figura 1: Situación inicial tras seleccionar `Ocio`

Figura 2: Situación inicial tras seleccionar el juego

Figura 3: Situación inicial tras seleccionar el juego (parte inferior detallada)

3. Desarrollo y sugerencias

Hay plena libertad en cómo afrontar el diseño y la implementación del juego propuesto. En cualquier caso, puede ser útil emplear las siguientes ayudas y atender a alguna que otra sugerencia:

- En esta práctica se completará la página web correspondiente a la sección de `Ocio` del menú de navegación del periódico, cuyo nombre será `ocio.html` y cuyo contenido será comentado a continuación.
- El juego estará realmente contenido de manera individual y aislada en un fichero denominado `solitario.html`, proporcionado con cierto contenido inicial (y que requeriá ser rellenado durante la práctica), el cuál será *incluido* mediante un `iframe` en la página web `ocio.html`.
- Detalles del fichero `ocio.html`
  - Empléese, por ejemplo, el fichero ya realizado en la práctica anterior, `index.html`, duplicándolo y renombrándolo como `ocio.html` como fichero de partida.
  - Elimínese el contenido de `ocio.html` que no tenga sentido ahora (artículos periodísticos) en esta página dedicada al ocio, manteniendo la parte de la cabecera, menú de navegación, pie, etc. Fíjese en las figuras 1, 2 y 3 para deducir qué se necesitará y qué no, como punto de partida.
  - Recuérdese enlazar este nuevo fichero `ocio.html` en el resto de páginas web del periódico para poder navegar hacia ella o volver de ella.
  - `ocio.html` contendrá dos partes significativas a desarrollar mínimamente:
    - Varios botones centrados horizontalmente y separados verticalmente unos 50 px del menú de navegación, seleccionando sendos juegos a programar: el primero de ellos tendrá como texto `Solitario`; el otro o los otros, pueden nombrarse como se desee provisionalmente, pues previsiblemente se utilizará para una posible futura práctica ampliando la cantidad de juegos. Se emplearán botones con cierta estética *especial* y distinta a la estándar, lo cual se detallará más adelante.
    - Un `iframe` que incluirá el documento `solitario.html` cuyas características se detallarán a continuación

- Detalles del fichero `solitario.html`

  - Este fichero no contendrá ninguna información relativa al sitio web ni al menú de navegación, por lo que es algo en principio completamente independiente de la estructura del sitio web.
  - Se sugiere que el fichero `solitario_plantilla.html` se renombre como `solitario.html` y que sea un punto de partida.
    - **¡¡¡Muy importante!!!** Se requerirá rellenar este fichero con los elementos `html`, `head` y `body` oportunos
    - Se podrá comprobar que este fichero se corresponde con una implementación *parcial* de la solución. Debería completarse correctamente, y para ello podemos acudir a las zonas del código resaltadas con el comentario:

      ```
      /*** !!!!!!!!!!!!!!!!!!! CODIGO !!!!!!!!!!!!!!!!!!!! **/
      ```

      como guía de por dónde realizar el completado del código.
  - Se adjunta el fichero de estilo `html/css/solitario.css`, el cual debe ser enlazado oportunamente en `solitario.html`. Este fichero contiene todas las declaraciones de estilo necesarias para que mínimamente se pueda visualizar algo parecido a lo que se muestra en la figura 3. Estúdiese su contenido e intente comprenderse su funcionalidad.
  - Síganse las indicaciones que aparecen como comentarios tanto en el fichero `solitario.css` como en el nuevo y pendiente de completar `solitario.html` para primeramente comprender el contexto del juego y a continuación rellenar el código JavaScript necesario para poner en marcha los primeros pasos del mismo.
  - Los ficheros gráficos, que en formato `png` representan las cartas de la baraja, podrán ser encontrados en el directorio `html/imagenes/baraja`. Obsérvese detenidamente de qué manera están formados los nombres de los ficheros y cómo identifican unívocamente cada carta.
  - Tareas a realizar en `solitario.html`:
    - Crear el mazo de la baraja con las cartas oportunas (inicialmente serán todas, pero después, en sucesivas iteraciones, no tiene porqué ser así).
    - Barajar el mazo de cartas y ponerlo sobre el tapete denotado en el código como `tapete_inicial`. Según la plantilla, un tapete es un `div`, por lo que dejar una carta sobre un tapete

será simplemente añadir el elemento `img` oportuno como hijo del citado `div`.
Se sugiere que se tengan en cuenta detalles como los valores respectivos de la propiedad `position` que debe tener en los elementos *contenedores* y en los elementos *contenidos*, así como los valores para las propiedades `top` y `left` para indicar cierto desplazamiento vertical y horizontal a cada carta respecto a la anterior y así ofrecer una disposición *en escalera* del mazo sobre el tapete.

- Indicar el número de cartas que hay en el mazo de este *tapete inicial*. Inicialmente, convendría hacer esta indicación del número de cartas de todos los mazos de todos los tapetes. Respecto a la ubicación de esta indicación, la sugerencia es hacerlo en la parte inferior derecha, tal y como puede observarse en la figura 3.
- Mostrar cómo va corriendo el tiempo que va transcurriendo con el formato `hh:mm:ss` en el sitio indicado a tal efecto.

Los resultados esperados pueden apreciarse gráficamente en la figura 3. En los ficheros proporcionados podrán encontarse ayudas y sugerencias en los comentarios para alcanzar fácilmente estos objetivos.

3.1. Botones: utilización de `Bootstrap`

La librería `BootStrap` tiene numerosos usos: permite concebir un comportamiento responsivo de la página web, adaptándose a las dimensiones de la ventana del navegador; también permite utilizar re-diseños de elementos HTML como botones, cajetines, etc., con una estética muy particular y mejorada normalmente, respecto al aspecto estándar. Su uso requiere la importación de tanto un fichero `css` de estilo como un fichero JavaScript.

`BootStrap` requiere adicionalmente la librería `jQuery` para su funcionamiento. Por cuestiones de eficiencia se sugiere que se importen ambas librerías desde ciertas CDN. Una posibilidad es la que se muestra a continuación:

```
<link rel="stylesheet"
      href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>
```

Se sugiere también que esta importación se realice antes de la de nuestros estilos propios para que en el caso de tener la misma *especificidad*, nuestros

estilos sean los que se impongan y no los de `BootStrap`, en el supuesto caso de que esas sean nuestras intenciones.

En cualquier caso observaremos muy probablemente que el aspecto de nuestra página ha cambiado en algunos detalles cuyo motivo será muy probablemente que `BootStrap` reescribe algunas declaraciones de estilo que no son controladas por nuestros ficheros de estilo.

Se debería indagar sobre cuáles son estas propiedades, pero con el ánimo de que el alumno no emplee demasiado tiempo en ello, se sugiere que se compruebe que si incluimos las siguientes propiedades de estilo en los elementos

```
seleccion-css ... li {
  font-family: 'Times New Roman', Times, serif;
  box-sizing: content-box;
}
```

muy probablemente volvamos a tener el mismo aspecto que antes de la inclusión de la librería de `BootStrap`

Búsquese información sobre tipos de botones en `BootStrap`. Una sugerencia de uso puede ser la que se muestra a continuación:

```
<button type="button" class="btn btn-primary" id="aSolitario">
  Solitario
</button>
```

3.2. Carga del `iframe`

Hay que asociar al evento *click* del botón comentado anteriormente, la carga del `iframe` que incluya a `solitario.html`. Podríamos plantear varias alternativas: especificación del atributo `onclick`, utilización de un objeto JavaScript que represente al botón, ajustando la propiedad `onclick` a cierta rutina que carge en el `iframe` el citado fichero html, etc.

La alternativa que se propone es sutilmente distinta: sustituir el texto interior `"Solitario"` por un ancla cuyo atributo `target` sea el nombre del iframe (atributo `name`), es decir:

```
<button type="button" class="btn btn-primary" id="aSolitario">
  <a href="solitario.html" target="mi_iframe">Solitario</a>
</button>

<iframe width="80%" name="mi_iframe"></iframe>
```

Si así lo hacemos, observaremos cómo ha quedado muy probablemente *desvirtuado* el aspecto del texto del botón debido a que se ha aplicado el estilo

estándar del ancla. Una posible solución consiste en modificar las propiedades CSS de esa ancla, eliminando cualquier *decoración* (`text-decoration:none`) y provocando que *herede* el color del texto (`color:inherit`).

Compruébese, comentando y descomentando la carga de *BootStrap/jQuery* que la estética de los botones es mejor, actuando acorde con el criterio personal de cada uno.

3.2.1. Dimensiones del `iframe`

De las porciones de código anteriores relativas al `iframe` puede observarse que la anchura elegida ha sido del 80 %. Estímese de forma aproximada qué altura en pixeles sería suficiente para que todo el solitario pueda aparecer en la ventana del `iframe` sin necesidad de *scrolling* y ajústese el atributo oportuno de dicho `iframe`.

Asimismo, determínese un valor aproximado para la propiedad de estilo `min-width` de forma que aparezcan barras de desplazamiento o de *scroll* cuando la anchura que queda para el `iframe` sea inferior a la anchura indicada.

Por último céntrese horizontalmente. Hay varias posibilidades: como los `iframe` tienen la propiedad de estilo `display:inline` por defecto, una posibilidad es cambiarla a `block` y reajustar la propiedad `margin:auto`.

4. Resultados a entregar

Actualícese el repositorio de GitHub con los ficheros que cumplan las especificaciones requeridas a lo largo del enunciado de la presente práctica. Recuérdese las condiciones de la entrega para evitar una merma en la calificación de la práctica.