



# GRADO EN DISEÑO Y DESARROLLO DE VIDEOJUEGOS

## VJ1217: DISEÑO Y DESARROLLO DE JUEGOS WEB

Prueba de programación

2 de junio de 2022

Usuario examen: **usuario-de-examen**

Contraseña: **la-del-examen**

Nombre y apellidos: \_\_\_\_\_

DNI: **dni-de-examen**

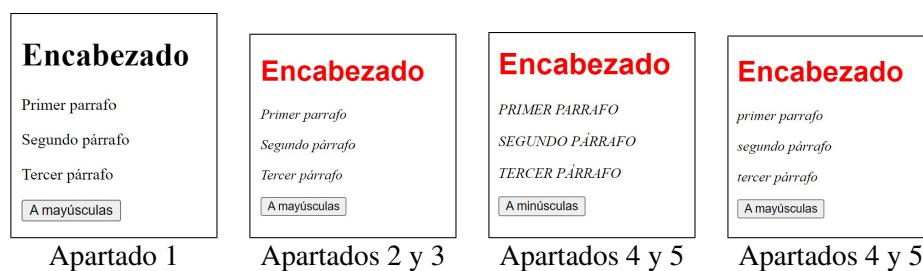
**Previo.** Descarga en el Escritorio de tu ordenador el paquete **ExamenJunio.zip** de *Aula Virtual*. Está en la sección **Exams**. Descomprímelo y desempaquéalo para obtener la carpeta **ExamenJunio**, que contiene tres subcarpetas dedicadas a cada una de las siguientes secciones.

**ATENCIÓN:** A la hora de probar el resultado de los diferentes ejercicios en el navegador debes emplear la opción **Open with Live Server** de **VS Code** tal como hemos visto en las clases de prácticas.

## HTML5/CSS3 (2,5 ptos.)

Abre en **VS Code** la carpeta **Bloque I - HTML-CSS** (da igual que esté vacía). A continuación, crea el fichero **ex1.html** teniendo en cuenta las siguientes indicaciones y los resultados mostrados en la Figura 1.

1. Crea un encabezado, tres párrafos y un botón con los textos que se muestran en la Figura 1.
2. Mediante una regla de estilo CSS, cambia la fuente del encabezado a *sans-serif*, su color a *rojo*, y el tamaño a *xx-large*.
3. Mediante una regla de estilo CSS, cambia el texto de todos los párrafos a cursiva.



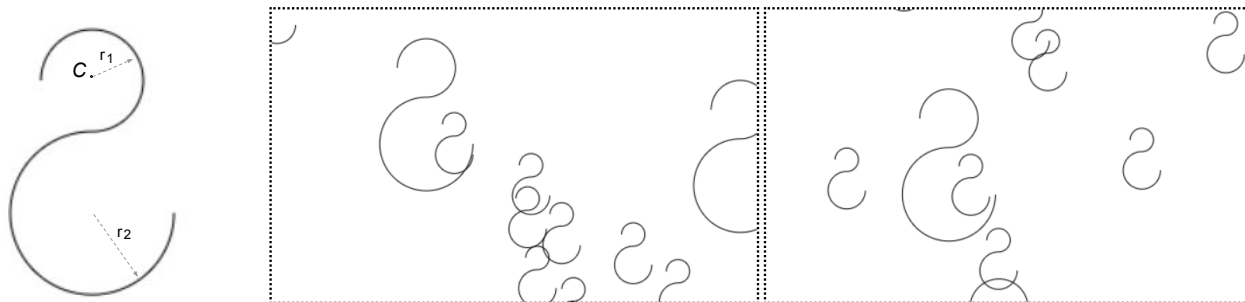
**Figura 1:** Resultados intermedios en el bloque de HTML5/CSS3

4. Se desea poder modificar el texto de los párrafos cuando se presiona el botón. Para empezar:
  - a) Asocia la función `cambia()` al atributo adecuado del elemento botón.
  - b) Escribe la función para que al presionar el botón se modifique su etiqueta “A minúsculas” si está en “A mayúsculas” y viceversa.
5. Crea una segunda función `procesaTexto(tipo)` que cambie el contenido de todos los párrafos a mayúsculas o a minúsculas, según el parámetro de entrada. Esta función se deberá invocar desde `cambia()`.

## JavaScript (4,0 ptos.)

Abre en VS Code la carpeta Bloque II - JavaScript. Al abrirla, encontrarás un fichero HTML, denominado `index.html`, y una carpeta, `js`. Dentro de dicha carpeta hay un fichero llamado `hooks.js`.

Completa el fichero `hooks.js` con las siguientes indicaciones. Dicho fichero ya incluye las constantes que **deberás utilizar** adecuadamente en tu código. Se necesita consultar, pero **no se debe modificar**, el contenido de `index.html`.



(a) Un hook:  $C$ ,  $r_1$  y  $r_2$

(b) Varios hooks (tres de ellos *grandes*) en dos canvases de una página

**Figura 2:** Hooks: (a) parámetros que definen un hook y (b) ejemplos de hooks dibujados en canvases

6. Para representar y manipular *hooks* (Figura 2(a)), completa la clase `Hook` tal que:

- El constructor reciba las coordenadas  $(x, y)$  de su posición  $C$ , el tamaño (`'small'` o `'big'`) y un contexto de dibujo de un canvas. El radio  $r_1$  es 12 para los hooks *pequeños* y 30 para los *grandes*. En ambos casos el radio  $r_2$  es un 60 % mayor que  $r_1$ .
- El método `draw()` dibuje el hook usando el contexto indicado al instanciar el objeto.
- El método `move()` cambie la posición del hook tanto en  $x$  como en  $y$  según sendos valores aleatorios en el rango continuo  $[-5, 6[$ , generados cada vez que se invoque al método y que deben sumarse a la posición actual. El efecto visual al repetir este movimiento con cierta rapidez es como un «temblor».

7. Para crear, mover y dibujar hooks, completa las siguientes funciones:

- `createHook(event, canvas)` para crear un hook en la posición del ratón, dada por las variables locales  $x$  e  $y$ , ya calculadas, y asociarlo al contexto `'2d'` del canvas `canvas`. El tamaño del hook será `'big'` con probabilidad 0,4. El nuevo hook se añadirá al array `hooks`, que es una variable global ya inicializada.
- `moveHooks()`, que mueva todos los hooks del array `hooks`, cada uno según (6c).
- `cleanCanvas(cv)` para «limpiar» el canvas `cv`, es decir eliminar todo lo que se haya dibujado.
- `drawHooks(canvases)` para limpiar todos los canvases del array `canvases` y después dibujar todos los hooks de `hooks`. Apóyate en `cleanCanvas()` de (7c)

8. Para gestionar hooks (Figura 2(b)) y asociarlos a cualquier canvas cuya clase sea “hooks” y que exista en el momento en que se cargue la página, completa la función `start()` de modo que:

- La variable `canvases` sea un array que contenga todos esos canvases.
- Asocie, a cada uno de dichos canvases, un *listener* del evento de pulsación del botón del ratón, que implique la ejecución de `createHook()`. Para poder pasar a `createHook(e, cv)` tanto el evento `e` como un canvas `cv`, usa una función anónima. Por ejemplo, la función anónima

```
function(event) { f(event, 3, 'Hi!'); }
```

permitiría la llamada a la función  $f()$  que, además de recibir `event` (y poder así obtener la información asociada al evento), recibiría más argumentos (aquí, el 3 y 'Hi!') de relevancia para esa hipotética  $f()$ .

- c) Se inicie un temporizador para mover y redibujar todos los hooks cinco veces por segundo. Usa oportunamente la función `moveAndDrawHooks(c canvases)` ya definida.

9. Añade la oportuna función para que, con la pulsación del botón “< Undo”, se elimine de `hooks` el último hook creado.

## Phaser (3,5 ptos.)

Abre en VS Code la carpeta Bloque III - Phaser. Al abrirla, encontrarás una estructura de carpetas y ficheros que conforman un juego similar al *Shoot 'em up* que estudiamos en la práctica 5 —es exactamente dicho juego, al que se le ha eliminado el primer estado y se le ha modificado el tercero, el que mostraba el *Hall of Fame* al final, por uno muy simple—.

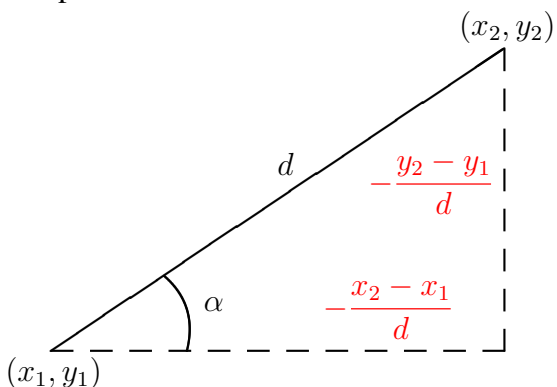
10. Supón que incorporamos a este juego una gran nave enemiga que se sitúa ajustada al borde superior. Esta nave se mueve solo horizontalmente, de izquierda a derecha y viceversa, rebotando en los bordes cuando los alcanza. Es indestructible, el impacto de los láseres no le afecta. No obstante, contesta cada impacto de láser con un disparo de plasma dirigido a la nave del jugador. Cuando dicho plasma alcanza a la nave del jugador se origina el mismo proceso que cuando un UFO estándar colisiona con ella.

Desarrolla los siguientes ejercicios para concretar este planteamiento recién esbozado. Si tienes dudas acerca del funcionamiento de este juego, puedes consultar el vídeo que hay disponible en *Aula Virtual* y que muestra cómo quedaría el juego tras las modificaciones pedidas. **Nota:** el vídeo no tiene sonido.

- a) Incorpora la gran nave alienígena, `assets/imgs/bigufo.png`, a la escena. Sitúala inicialmente en el centro en horizontal y ajustada al borde superior. Dóta la del sistema físico *Arcade*, provócale un movimiento horizontal muy lento y haz que rebote en los bordes izquierdo y derecho, para mantenerse siempre en movimiento.
- b) Crea la infraestructura necesaria para producir y gestionar los disparos de plasma: un grupo en *Phaser* cuyos 40 miembros usen la imagen `assets/imgs/plasma.png`, tengan el sistema físico *Arcade* habilitado, comprueben los bordes del escenario y se desactiven al salir del mismo.
- c) Detecta los impactos de láseres sobre la gran nave. Por cada uno, elimina el láser impactante y prepara y ejecuta un disparo de plasma dirigido a la posición de la nave del jugador desde el centro de la gran nave. Para ello, ten en cuenta las siguientes fórmulas para calcular las velocidades necesarias.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$v_x = -\cos \alpha \cdot v = -\frac{x_2 - x_1}{d} \cdot v$$

$$v_y = -\sin \alpha \cdot v = -\frac{y_2 - y_1}{d} \cdot v$$


El diagrama muestra un triángulo rectángulo con vértices en  $(x_1, y_1)$  y  $(x_2, y_2)$ . La hipotenusa, que representa la distancia  $d$ , está etiquetada como  $d$ . El ángulo  $\alpha$  se mide en el vértice  $(x_1, y_1)$ . Los catetos están etiquetados como  $-\frac{y_2 - y_1}{d}$  (vertical) y  $-\frac{x_2 - x_1}{d}$  (horizontal).

La velocidad  $v$ , fija también, establécela en 200.

- d) Implementa la detección del impacto de un disparo de plasma en la nave del jugador y haz que se gestione igual que cuando un UFO estándar impacta contra ella.

11. Crea un estado inicial que comience la ejecución del juego y dé paso al actual estado `playState`.

Implementa una animación en el estado inicial que evolucione la nave del jugador y una nave alienígena según las coordenadas y el nivel de transparencia (atributo **a** de las tripletas) anotadas en el fichero **assets/anim/anim.json**. Concretamente, para cada nave, partiendo de los valores de la tripleta central, continúa alcanzando los valores de la primera tripleta, volviendo a la tripleta central para alcanzar esta vez los valores de la tercera tripleta, y finalmente volver a la tripleta central para acabar en ella. Especifica 2 segundos de duración para cada una de estas cuatro fases de la animación. Y fíjate en que en los movimientos enfrentados de ambas naves, en las esquinas alcanzan la máxima transparencia y en el centro la máxima opacidad.

Al finalizar la animación, no antes, debe aparecer el botón **assets/imgs/startbutton.png**, cuya pulsación provocará la transición a **playState**. El fichero **assets/anim/anim.json** determina, también, la posición de este botón.

Configura las tres imágenes para tener su ancla en el centro, y las dos naves para mostrarse al doble de su tamaño original. Recuerda que puedes consultar el vídeo en *Aula Virtual* si tienes dudas al respecto del funcionamiento de esta pantalla.

**Entrega:** Una vez hayas concluido, abre la carpeta **Espai** en disc personal que hay en el Escritorio de Windows y copia, dentro de ella, la carpeta **ExamenJunio**, que debe tener todos los contenidos de los ejercicios que has estado desarrollando. Si no copias esta carpeta en **Espai** en disc personal será como si no hubieses efectuado el examen.