HTML

Referenciar un css desde el html:

```
<link href="css/estilos.css" rel="stylesheet" type="text/css"/>
```

Poner una imagen:

```
<img src="imgs/change.png" alt="" width="150">
```

Crear una clase:

- Dentro del div, h1, etc:
- Dentro del css: .claseEjemplo{}

Crear un atributo:

- Dentro del div, h1, etc:
- Dentro del css: #idEjemplo{}

Cambiar el formato desde HTML:

Inline As an attribute of a particular HTML tag:

```
<h2 style="color:red; background:green;"> This is a red text with a green bg</h2>
```

Embedding As an <style> tag within the HTML document:

```
<style type="text/css">
  h2 {
  color:red;
  background:green;
   }
</style>
```

Linking By indicating which CSS file contains the style rules (esta es la forma normal de linkearlo):

```
<link rel="stylesheet" type="text/css" href="mystyles.css">
```

We can include a CSS file from within another: @import "newstyles.css";

CSS

Fuente normal: `font-weight: normal;`
Fuente en negrita: `font-weight: bold;`

```
Subrayados:
      text-decoration: none; /* Sin decoración */ text-decoration:
      underline red; /* Subrayado rojo */ text-decoration:
      underline wavy red; /* Subrayado rojo ondulado */
```

Texto tachado: `text-decoration:line-through;`
Texto en cursiva: `font-style: normal // font-style: italic // font-style: oblique;`

Margen izquierdo y borde rojo:

```
p {
  margin-left: 20px;
  border: red 2px solid; }
```

Margen interno y borde rojo:

```
p {
  border: red 2px solid;
```

```
        padding: 10px; }
```

Otras funciones:

```
#auto{     padding: 10px;  background: orange;   width: 95px;
cursor: pointer;      margin-top: 10px;     margin-bottom: 10px;
box-shadow: 0px 0px 1px #000;    display: inline-block; }
#auto:hover{     opacity: .8; } #div-mostrar{     width: 100%;
margin: auto;    height: 0px;     background: #000;     box-shadow:
10px 10px 3px #D8D8D8;     transition: height .4s; color:white;
text-align: center;
```

Ocultar etiqueta div:

```
Con el estilo "display: none" puedes ocultarla
<div class="auto" id="auto" style="display: none">
     Oculta
</div>
 <div class="auto" id="auto">
     No Oculta
</div>
```

Para que al pulsar un botón pase una cosa u otra:

```
var toggle_id = ["Music", "Sound", "Text"];
var toggle_state = [];
toggle_state["Music"] = false;
toggle_state["Sound"] = true;
toggle_state["Text"] = false;
```

We include one of such toggles like this: Note that upon user clicking on the toggle (i.e on the canvas) we call two functions:

• toggleCanvas(), which inverts the boolean value of toggle_state["Text"] and draws the corresponding graphics on the canvas; and

• toggleDisplayText(), which shows or hides a text we have in the HTML document, depending on the value of toggle_state["Text"].

Para importar una fuente:

```
  <link href="https://fonts.googleapis.com/
css?family=Gochi+Hand" rel="stylesheet">
```

o

```
<style>
@import url('https://fonts.googleapis.com
/css?family=Gochi+Hand');
</style>
```

Como implementar una slider bar que cambia la opacidad de un objeto del canvas:

Creas el canvas y un objeto input de tipo range y class slider

```
<input type="range" min="0" max="100" value="50" class="slider" id="myRange"
onchange="setOpacity()">
<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#d3d3d3;">
Your browser does not support the HTML5 canvas tag.
</canvas>
```

Después abrimos la etiqueta <script> para programar con Javascript:

```
<script>
    let c = document.getElementById("myCanvas");
    let ctx = c.getContext("2d");
    let slider; //variable que va a guardar el valor en el que se encuentra el slidebar
    setOpacity(); //funcion para cambiar la opacidad
    function setOpacity() {
        slider = document.getElementById("myRange"); //coge el valor de la barra, que en nuestro caso
        //va de 0 a 100
        ctx.clearRect(0, 0, c.width, c.height); //antes de dibujarlo, BORRAMOS lo que había antes
        redraw(); //dibujamps el circulo otra vez con la nueva opacidad, vamos a ver la función
    }
    function redraw() {
        ctx.globalAlpha = slider.value * 0.01 //cambio la opacidad por el valor del SLIDER entre 100
        //porque la opacidad va desde 0 a 1
        ctx.beginPath();
        let radio;
        radio = 0.4 * Math.min(c.width, c.height);
        ctx.arc(95, 50, radio, 0, 2 * Math.PI);
        ctx.fillStyle = "pink";
        ctx.fill();
        ctx.stroke();
    }
</script>
```

Ejercicio de programar un botón para cambiar elementos del div de invisible a visible (Examen junio):

```
function hideText()
{
  var pantallaInicial = document.getElementById("begin");
  var pantallaFinal = document.getElementById("end");

  if (window.getComputedStyle(pantallaInicial).display === "block")
  {
     pantallaInicial.style.display = "none";
     pantallaFinal.style.display = "block";
  }
  else if(window.getComputedStyle(pantallaInicial).display === "none")
  {
     pantallaInicial.style.display = "block";
```

```
    pantallaFinal.style.display = "none";
}
```

Animaciones:

Animations (CSS file):

```
/* Animation code */
@keyframes navigate {
    0%  {left: 0%; transform:rotate(0deg); background-color: red;}
    20% {left: 30%; transform:rotate(30deg); background-color: yellow;}
    40% {left: 60%; transform:rotate(60deg); background-color: green;}
    50% {left: 80%; transform:rotate(90deg); background-color: darkblue;}
    60% {left: 60%; transform:rotate(60deg); background-color: green;}
    80% {left: 30%; transform:rotate(30deg); background-color: yellow;}
    100% {left: 0%; transform:rotate(0deg); background-color: red;}
}

#boat {
    margin-top: 5em;
    margin-left: 3em;
    width: 120px;
    height: 120px;
    position: absolute;
    animation-name: navigate;
    animation-delay: 2s;
    animation-duration: 6s;
    animation-iteration-count: 3;
    animation-timing-function: linear;
}
```

HTML

```
<!-- Learn about this code on MDN:
https://developer.mozilla.org/es/docs/Web/CSS/CSS_Animations/Usando_animaciones_CSS -->

<h1 id="watchme">Watch me move</h1>
<p>
  This example shows how to use CSS animations to make <code>H1</code>
  elements move across the page.</p>
<p>
  In addition, we output some text each time an animation event fires,
  so you can see them in action.
</p>
<ul id="output">
</ul>
```

CSS

```
.slidein {
  animation-duration: 3s;
  animation-name: slidein;
  animation-iteration-count: 3;
  animation-direction: alternate;
}

@keyframes slidein {
  from {
    margin-left:100%;
    width:300%
  }

  to {
   margin-left:0%;
   width:100%;
 }
}
```

```
@keyframes spin {
    100% {
        transform:rotate(360deg);
    }
}

@keyframes leds-off {
    100% {
        opacity:0.3;
    }
}

@keyframes helix-movement {
    100% {
        background: linear-gradient(
          to bottom,
          lighten($color2, 15%) 50%,
          lighten($color2, 15%),
          $color2,
          $color2 50%
        );
        background-size: 100% 20px;
    }
}


@keyframes diving {
    0% {
         margin-top:5px;
    }
    50% {
         margin-top:15px;
    }
```

Abrir con

HTML5/CSS3

1. (20 %) Abre en NetBeans el proyecto AsignaturasVideoJuegos. En él puedes encontrar una sencilla página HTML que contiene listas de algunas asignaturas de primero y de segundo del grado de videojuegos. Edita convenientemente solo la hoja de estilos (CSS), solo el fichero HTML (HTML) o ambos (HTML+CSS), según se indica en cada caso, para que:

   (a) CSS El color de los elementos `<h1>` sea `darkgreen`.

   (b) HTML+CSS Las asignaturas de informática estén en negrita y en color marrón (`brown`). Utiliza para ello el nombre de clase `cs` (de *"computer science"*) en los correspondientes `<li>`.

   (c) HTML El color de fondo del bloque `<div>` de las asignaturas de primero sea `aquamarine`, y el del de las de segundo sea `beige`. Utiliza para ello el *atributo* `style` de los correspondientes `<div>`.

   (d) HTML El ancho de los elementos `<div>` sea del 20 % del ancho de la ventana del navegador y su valor de relleno (`padding`) sea de `2em`. Utiliza para ello un *elemento* `<style>`.

   (e) CSS Cuando un elemento `<div>` tenga el puntero del ratón encima (i.e. selector `hover`):

```css
div:hover {
    border: brown 3px solid;

    animation-name: divAnim;
    animation-duration: 1s;

}

@keyframes divAnim
{
    50%{ transform:scale(1.2) }
    80% {transform:scale(0.8)}
    /* a completar */
}
#h1{
    color: darkgreen;

}
.cs{
    color: brown;
    font-weight: bold;
}
```

PHASER:

En un js, decir que va a tener las funciones de inicializar, crear y actualizar

```
let initState = {
    preload: preloadInit,
    create: createInit
     update: updateInit };
```

Cargar una imagen en la función de preload:

```
game.load.image('craft', 'assets/imgs/craft.png');
```

Añadir texto en pantalla: (coordenada x, coordenada y, el texto, el estilo del texto)

```
let instructions = game.add.text(TEXT_OFFSET_HOR,
TEXT_OFFSET_VER, textI, styleI);
```

Cambiar el centro de una imagen/texto:

```
variable.anchor.setTo(0, 1);
```

Añadir un botón (coordenada x, coordenada y, imagen del botón y función a la que se llama cuando se pulsa):

```
btnStart = game.add.button(posX, posY, 'craft', clickStart1);
```

Activar la detección de colisión con los límites de la pantalla para 1 objeto (en este caso el botón):

```
    btnStart.checkWorldBounds = true;
```

Llamar a una función cuando el elemento que tiene activada la detección de colisiones con los límites del mundo, salga de estos:

```
    btnStart.events.onOutOfBounds.add(startPlay1, this);
```

Escalar un objeto:

```
btnStart2.scale.setTo(2.0);
```

cambiar la bounding box:

```
objeto.body.setSize(60, 60, 35, 35);
```

iniciar un nuevo estado (un nuevo js):

```
game.state.start('play');
```

llamar a una función cada x tiempo:

```
game.time.events.loop(FREQUENCY, moveButtonAndImage, this);
```

modificar la posición de objetos (tanto x en un objeto como y en el otro objeto):

```
btnStart.y -= DECREASE_Y;

btnStart2.x -= DECREASE_X;
```

En un js, decir que va a tener 3 funciones, la de carga, la de creación y la de actualizado:

```
let playState = {
    preload: preloadPlay,
    create: createPlay,
    update: updatePlay
};
```

declarar una constante de ritmo (en este caso es una décima de segundo):

```
const TIMER_RHYTHM_ = 0.1 * Phaser.Timer.SECOND;
```

constantes de vectores de probabilidad y velocidad; sirven para aumentar la dificultad a lo largo del juego:

```
const LEVEL_UFO_PROBABILITY =
        [0.2, 0.4, 0.6, 0.8, 1.0];
const LEVEL_UFO_VELOCITY =
        [50, 100, 150, 200, 250];
```

precargar sonido en el preload:

```
game.load.audio('sndlaser', 'assets/snds/laser.wav');
```

precargar spritesheet en el preload:

```
game.load.spritesheet('blast', 'assets/imgs/blast.png', 128,
128);
```

llamar a una función de creación (en este caso de lasers) desde el create, pasándole el número de objetos que se quieren crear como argumento:

```
createLasers(LASERS_GROUP_SIZE);
estilo de un texto(en este caso es el tamaño y el color):
let styleHUD = {fontSize: '18px', fill: '#FFFFFF'};
```

función que crea un grupo de objetos (en este caso de blasts) pasándole ese número como argumento; además de crear el grupo y añadir **number** blasts, para cada uno, llama a la función de setupBlast:

```
function createBlasts(number) {
    blasts = game.add.group();
    blasts.createMultiple(number, 'blast');
    blasts.forEach(setupBlast, this);
}
```

la función de setupBlast establece el centro del objeto en el centro de la imagen y le añade una animación:

```
function setupBlast(blast) {
    blast.anchor.x = 0.5;
    blast.anchor.y = 0.5;
    blast.animations.add('blast');
}
```

en el caso de que haya algún elemento en el grupo blasts, coge el primero y lo pone en la variable blast:

```
let blast = blasts.getFirstExists(false);
```

establece 2 posiciones, x e y en base a un elemento dado (ship):

```
    let x = ship.body.center.x;
```

```
        let y = ship.body.center.y;
```

resetea la posición del blast a esas coordenadas que hemos calculado anteriormente:

```
        blast.reset(x, y);
```

reproduce una animación:

```
        blast.play('blast', 30, false, true);
```

activa la detección de colisiones de un objeto:

```
    objeto.enableBody = true;
```

llama a una función para todos los elementos que forman parte de un grupo (en este caso dice que para todos los elementos que forman parte del grupo ufos, si salen de los límites del mundo, se llame a la función resetMember):

```
    ufos.callAll('events.onOutOfBounds.add',

                'events.onOutOfBounds', resetMember);
```

activa para todos los elementos del grupo la detección de colisiones con los límites de la pantalla:

```
    ufos.setAll('checkWorldBounds', true);
```

función que activa una roca en función de una determinada probabilidad fija. NOTA: Math.random es una función propia de Phaser que devuelve un número entre 0 y 1. NOTA 2: con el reset se establece cuál será su posición de spawn y la velocidad es la que marca su movimiento:

```
    function activateRoca() {
        if (Math.random() < currentRocaProbability) {
            let meteorito = roca.getFirstExists(false);
            if (meteorito) {
                let gw = game.world.width;
                let uw = meteorito.body.width;
                let w = gw - uw;
                let x = Math.floor(Math.random() * w);
                let z = meteorito.body.width / 2 + x;
                meteorito.reset(z, 0);
                meteorito.body.velocity.x = 0;
                meteorito.body.velocity.y = +currentRocaVelocity;
                meteorito.angle = Math.random() * 360; }
        }
    }
```

función que crea sonidos:

```
function createSounds() {
    soundLaser = game.add.audio('sndlaser');
    soundBlast = game.add.audio('sndblast'); }
```

función que elimina un objeto:

```
function resetMember(item) {
    item.kill();}
```

función que crea los controles del teclado (en este caso son las flechas y el espacio):

```
function createKeyControls() {
    cursors = game.input.keyboard.createCursorKeys();
    fireButton =
game.input.keyboard.addKey(Phaser.Keyboard.SPACEBAR);
}
```

el update es la función que se ejecuta constantemente; en este caso comprueba todas las colisiones (mediante la función de Phaser **overlap**), detecta las pulsaciones de las teclas (movimientos de la nave y disparos de la misma) y también cambia la posición de las estrellas:

```
function updatePlay() {
    game.physics.arcade.overlap(
            lasers, ufos, laserHitsUfo, null, this);
    game.physics.arcade.overlap(
            craft, ufos, ufoHitsCraft, null, this);
    game.physics.arcade.overlap(
            lasers, roca, laserHitsMeteorito, null, null);
    game.physics.arcade.overlap(
            craft, roca, MeteoritoHitsCraft, null, null);
    manageCraftMovements();
    stars.tilePosition.y += 1;
    manageCraftShots();
}
```

esta función permite disparar, es decir, cada vez que se presiona el botón de disparar (en nuestro caso el espacio), llama a la función de disparo (el parámetro de justPressed(30) es un pequeño delay que se le aplica a la pulsación del botón para que los disparos no sean demasiado consecutivos):

```
function manageCraftShots() {
    if (fireButton.justDown ||
            game.input.mousePointer.
            leftButton.justPressed(30))
        fireLasers();
}
```

función que controla el movimiento de la nave mediante la pulsación de las flechitas:

```
function manageCraftMovements() {
    craft.body.velocity.x = 0;
    craft.body.velocity.y = 0;
```

```
          if (cursors.left.isDown
                  || game.input.speed.x < 0)
              craft.body.velocity.x = -CRAFT_VELOCITY;
          else if (cursors.right.isDown
                  || game.input.speed.x > 0)
              craft.body.velocity.x = CRAFT_VELOCITY;
          if (cursors.up.isDown || game.input.speed.y < 0)
              craft.body.velocity.y = -CRAFT_VELOCITY;
          else if ((cursors.down.isDown
                  || game.input.speed.y > 0) &&
(craft.body.position.y < GAME_STAGE_HEIGHT - HUD_HEIGHT -
craft.body.height))
              craft.body.velocity.y = CRAFT_VELOCITY;
      }
```

TWEENS.

Los tweens son unas funciones que puedes hacer con phaser que modifican el aspecto de imágenes. Con esto se puede hacer lo que pedía el examen de este año de hacer una animación de la nave que recorriera la pantalla de esquina a esquina.

Ejemplo de Tween que se mueve:

welcome.js, and in the game itself, play.js. The code of welcome.js shows the potential of tweens:

```
let mainTween, downTween1, downTween2;
// ...
function displayScreen() {
  // ...
  mainTween = game.add.tween(hero3)
    .to({y: 32}, 2000, Phaser.Easing.Linear.None)
    .to({angle: -90}, 500,
        Phaser.Easing.Linear.None)
    .to({x: game.world.width - (32 * 2)}, 4000,
        Phaser.Easing.Linear.None);
  mainTween.delay(3000);
  mainTween.loop(true);
  mainTween.start();

  downTween1 = game.add.tween(hero1.scale)
    .to({x: 5, y: 5}, 1500,
        Phaser.Easing.Cubic.Out)
    .to({x: 1, y: 1}, 1500,
        Phaser.Easing.Cubic.Out);
  downTween1.onComplete.add(
      onDownTweenCompleted, this);
  downTween2 = game.add.tween(hero2)
    .to({alpha: 0.05}, 2500,
        Phaser.Easing.Linear.None)
    .to({alpha: 1.0}, 2500,
        Phaser.Easing.Linear.None);
  downTween2.onComplete.add(
      onDownTweenCompleted, this)
  downTween1.start();
  // ...
}

function onDownTweenCompleted(object,tween) {
  if (tween === downTween1)
      downTween2.start();
  else
      downTween1.start();
}
```

Más ejemplos de tweens:

```
game.add.tween(startButton).to( { alpha: 1 }, 1000, Phaser.Easing.Linear.None, true, 0, 1000, true);
```

Este cambiaba la transparencia del objeto que le pasemos(en este caso startButton) de 0 a 1.

```
game.add.tween(name.scale).to({x:1.5, y:1.5}, 1000, Phaser.Easing.Linear.Nono, true, 0, 1000, true);
```

Este cambia el tamaño de name a 1.5

EJERCICIO PHASER EXAMEN 2019

1.0

```
function createPlasmas(n){
    gplasma= game.add.group();
    gplasma.enableBody=true;
    gplasma.createMultiple(n, 'plasma');
    gplasma.callAll('events.onOutOfBounds.add', 'events.onOutOfBounds', resetMember);
    gplasma.callAll('anchor.setTo', 'anchor', 0.5, 0.5);
    gplasma.setAll('checkWorldBounds', true);
    game.time.events.loop(TIMER_RHYTHM, activatePlasma, this);
}
function activatePlasma(){
    if (Math.random() < 0.1) {
        let plasma = gplasma.getFirstExists(false);
        if (plasma) {
            let gw = game.world.width;
            let uw = game.world.height;
            let franja = gw + uw;
            let x = Math.floor(Math.random() * franja) - uw/2;
            let z = plasma.body.width / 2 + x;
            plasma.reset(z, 0);
            let d= Phaser.Math.distance(plasma.x, plasma.y, craft.x, craft.y)
            plasma.body.velocity.x = ((craft.x-plasma.x)/d)*100;
            plasma.body.velocity.y = ((craft.y-plasma.y)/d)*100
        }
    }

```

3.

```
let ship
let shiptween;
function preloadEnd(){
    game.load.image('craft', '/assets/imgs/craft.png');
}

function createEnd(){
ship= game.add.sprite(20,20,'craft');
ship.anchor.setTo(0.5, 0.5);
ship.angle = 90;


shiptween = game.add.tween(ship)
.to({x:780}, 5000, Phaser.Easing.Linear.None)
.to({angle:180}, 500)
.to({y:570}, 3000, Phaser.Easing.Linear.None)
.to({angle:270}, 500)
.to({x:20}, 3000, Phaser.Easing.Linear.None)
.to({angle:360}, 500)
.to({y:20}, 3000, Phaser.Easing.Linear.None)
.to({angle:360+90}, 500)



shiptween.loop(true);
shiptween.start();
```

Esto es la primera parte del ejercicio 3, creamos un nuevo estado End. En este nuevo estado End, creamos un tween para la animación que nos pide: que la nave dé una vuelta entera en bucle.
Los números son por el tamaño de la pantalla-el tamaño de la nave. El segundo número el tiempo que tardará en hacerlo (cuanto más alto, más lento irá la nave).
Mostrar la puntuación en pantalla:

```
let styleEnd = {font:'30px Arial', fill:'#FFFFFF'};
let textEnd = 'Last score: ' + score;
let textito= game.add.text((game.world.width/2), (game.world.height/2),
textEnd, styleEnd);
textito.anchor.setTo(0.5, 0.5); //para centrarlo
```

Que el fondo sea verde: `game.stage.backgroundColor='#008f39';`

Ejercicio HTML EXAMEN HECHO POR FRAN

index.html × # estilos.css JS toggle.js

```
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link href="css/estilos.css" rel="stylesheet" type="text/css"/>
    <title>Examen Junio - Ejercicio 1</title>

</head>

<body>
    <div id="begin">
        <h3 >PANTALLA INICIAL</h3>
            <div class="forExam">Este es un ejercicio de examen diseñado para que demuestres tus
                conocimientos de HTML, CSS y (un poquito, muy poquito) de JavaScript.
            </div>
            <div class="forExam">Tienes que conseguir que tanto este párrafo como el anterior y el
                siguiente se muestren de manera idéntica, tal como se indica en el
                enunciado del examen.
            </div>
            <div class="forExam">Para conseguirlo, fíjate en los estilos que hay definidos en el
                fichero <span class="negrita">estilos.css</span> que encontrarás dentro de la
                carpeta <span class="negrita">css</span>.
            </div>
    </div>
    <div id="end">
        <h3>PANTALLA FINAL</h3>
            <div class="forExam">Tienes que aplicar el estilo anterior igualmente tanto a este párrafo
                como al siguiente dentro de lo que sería esta sección.
            </div>
            <div class="forExam">Este es el párrafo final. Es el último. Observa bien el resultado
                al cargar la página.
            </div>
        <!-- Inserta a continuación la imagen del botón que hay en la carpeta imgs -->
    </div>
    <img  onclick="Cambiar()" src="/HTML-CSS/imgs/change.png" alt="replay"
    style="width:100px;
           height:110px;
           margin-top:3em;
           margin-left: 50%;" >
    <script type="text/javascript" src="js/toggle.js"> </script>


</body>
```

index.html | # estilos.css | JS toggle.js

```css
@font-face {
    font-family: 'CoelacanthRegular';
    src: url('../fonts/Coelacanth.otf');
}

body {
    background-color: slategrey;
    margin: auto;
    margin-top: 2em;
    width: 75%;
}

.forExam {
    background: #01395e;
    border-color: #b9b5b5 #b9b5b5 #272525 #272525;
    border-radius: 15px;
    box-shadow: 10px 10px 15px 5px #000000;
    border-width: 3px;
    padding: 1em 1em 1em 1em;
    font-family: 'CoelacanthRegular', normal;
    font-size: 18pt;
    text-align: center;
    color: #74c925;
    margin-bottom: 1em;
}
.negrita{
    font-weight: bold;
    text-decoration: underline;
}

#begin{
    display:inline;
}
#end{
    display: none;
}
```

index.html | # estilos.css | JS toggle.js

```javascript
let pantallaprincipal= true;
function Cambiar(){
    let pagina1= document.getElementById("begin");
    let pagina2= document.getElementById("end");

    if (pantallaprincipal ==true){
       pagina1.style.display='none';
       pagina2.style.display='inline';
       pantallaprincipal=false;
    }
    else{
       pagina2.style.display='none';
       pagina1.style.display='inline';
       pantallaprincipal=true;
    }
```

Examen del año pasado:

Preguntas sobre Phaser:

- La ventana inicial debe ser como mostramo en la siguiente imagen: 'xxx'
    - Al pulsar tres veces sobre ese botón se pasa a la ventana con el botón Play again!
    - Al pulsar un botón se vuelve a la pantalla inicial
    - El comportamiento debe repetirse infinitamente

Se define así en el main:

```
// Entry point
window.onload = startGame;  //IMPORTANTE ESTO PARA INICIAR

function startGame() {
   game.state.add('main', mainState);
   game.state.add('end', endState);
   game.state.start('main');
}
```

El window.onload es la función esencial para que se inicialice el juego, sin esta el estado saldrá como no definido.

El estado que queramos definir (main, end, etc.) deberemos definirlo en el main, como vemos en el ejemplo de arriba, en el html, y dentro del propio estado, deberemos de definir sus funciones propias, es decir, pensando en el end, por ejemplo:

- Preload
- Create
- Update

Estas funciones deberán no sólo definirse arriba sino poner la función entera abajo.

Otro fallo de este ejercicio es que faltaba cargar la imagen 'click me' que debemos cogerla de la carpeta correspondiente y cargarla en el end.js, también se lo debemos poner al botón con el título de la imagen, al botón le faltaba el nombre, tanto su declaración arriba como inicializarlo, por lo que el

```
btnEnd.anchor.SetTo(0.5, 0.5)
```

debe tener al objeto primero (btnEnd), y esta variable debe estar declarada arriba del todo.

Quedaría tal que así:

```
let btnEnd;
btnEnd = game.add.button(game.stage.width / 2,
game.stage.height, 'playAgain', onEndClick)
btnEnd.anchor.SetTo(0.5, 0.5)
```

Tenemos el mismo problema tanto en el main como en el end, en los dos javaScripts hay que cambiarlos.

Faltaba una letra en la sentencia de load de la imagen a cargar ¡Cuidado!

El último fallo es que al volver al principio, ya no se daban 3 clicks sino uno, esto ocurre porque el num_clicks de las veces que se ha clickado el botón, está a más de 3, y el valor está inicializado y declarado arriba del todo con lo que nunca se vuelve a poner a 0, lo que hay que hacer es declararlo al principio:

```
let numclicks;
```

y dentro de create lo inicializamos:

```
numclicks = 0;
```

- Al cargar la página, se debe reproducir el fichero de audio 'xxx', repetidamente, una vez tras otra

Hay que tener en cuenta que el juego no se llama game sino audioJuego por lo que las funciones como game.load...., ahora son audioJuego.load...

```
var audioJuego = new Phaser.Game(200, 200, Phaser.Canvas, "myGame",
{preload: loadAudio, create: playAudio});

//Cargamos el audio
function loadAudio() {
   audioJuego.load.audio('pewpew', 'audio/pewPew.mp3');
}

//Hacemos sonar el audio
function playAudio() {
   let audio = audioJuego.add.audio('pewpew');
      //hacemos que el audio suene en bucle
   audio.loopFull();
}
```

- El audio se debe parar si se pulsa la tecla P

Para este ejercicio debemos tener añadir un update arriba del todo junto a create y preload.

```
      var audioJuego = new Phaser.Game(200, 200, Phaser.Canvas,
"myGame", {preload: loadAudio, create: playAudio, update:
updateAudio});

      let audio;
      function loadAudio() {
         audioJuego.load.audio('pewpew', 'audio/pewPew.mp3');
      }
      function playAudio() {
```

```
        audio = audioJuego.add.audio('pewpew');
        audio.loopFull();
        pauseAudio =
audioJuego.input.keyboard.addKey(Phaser.Keyboard.P);
    }

    function updateAudio() {
        if(pauseAudio.isDown){
            audio.pause();
        }
    }
```

- Cambia en el css el color de los elementos <h1> a darkgreen
- Cada vez que se pulse una tecla numérica (del 0 al 9) debe reproducirse la canción 'xxx' en números pares y 'yyy' en números impares.
    - No modificar ni eliminar líneas, sólo añadir
    - No utilizar la función update, ni ninguna sentencia condicional
- Utilizar, entre otra posible información, lo que se desprende de cada uno de estos dos hechos:
    - Los códigos de las teclas son números enteros
    - En JS, la expresión 7 % 2 se evalúa a 1

Preguntas sobre HTML/CSS:

- El borde debe pasar a tener 3 píxeles de grosor y de color verde
- El fondo del canvas debe ser de color violeta oscuro
- Esa misma regla debe servir para todos los canvas que la página contenta o pueda contener

El HTML:

```
<html>
<body>
   <p><b>Dos</b> canvas </p>
   <canvas id="canvasLong" width="200" height="100">
       Your browser does not support the HTML5 canvas tag.
   </canvas>
```

Le tenemos que poner la etiqueta del canvasLong a todo

```
   <canvas id="canvasLong" width="100" height="200">
       Your browser does not support the HTML5 canvas tag.
   </canvas>
</body>
</html>
```

El css:

```
#canvasLong
```

Aquí le ponemos más borde al canvas y en solid lo cambiamos de color, en background le cambiamos el color al background, se puede poner con el color o en hexadecimal.

```
{  border:3px solid #22a501;
   background:purple; }
```

- Meter una imagen mediante una etiqueta img

- Hacer una animación de 2.5 segundos que primero gire a la izquierda y se escale hasta llegar respectivamente, a una orientación de -60º y un factor de escala de 1.5 a alcanzar y el 40% de la duración de la animación
    - Al 80% de la duración de la animación debe llegar a su orientación y tamaños iniciales

En HTML:

```
<html>
<body>
   <img src="img/new.jpeg" alt="new" id="new">
   <p><b>Dos</b> canvas </p>

   <canvas id="canvasLong" width="200" height="100">
       Your browser does not support the HTML5 canvas tag.
   </canvas>

   <canvas id="canvasLong" width="100" height="200">
       Your browser does not support the HTML5 canvas tag.
   </canvas>
</body>
</html>
```

En CSS:

```
#canvasLong
{  border:3px solid #22a501;
   background:purple; }
#new
{  position: relative;
   left: 50%;
   animation-name: gira;
```

```
   animation-duration: 2.5s; }

@keyframes gira
{  40% {transform:rotate(-60deg) scale(1.5); }
   80% {transform:scale(0.5); } }
```

\- . -

- En HTML + CSS cambia a que las asignaturas de informática estén en negrita y en color marrón (brown). Utiliza para ello el nombre de clase cs en los correspondientes
- En HTML cambia el color de fondo del bloque de las asignaturas de primero sea aquamarine y el de las de segundo sea beige. Utiliza para ello el atributo style de los correspondientes
- El ancho de los elementos sea del 20% del ancho de la ventana del navegador y su valor de relleno (padding) sea de 2em. Utiliza para ello el elemento <style>
- En CSS cuando un elemento tenga el puntero del ratón encima (i.e. selector hover)
    - Su borde sea de color marrón (brown) con trozo sólido y de 3 píxeles de grosor
    - Se aplique una animación de 1 segundo de duración en el que su tamaño aumente hasta un 20% a las décimas de segundo, y luego se reduzca al 70% de su tamaño original a las 8 décimas de segundo

Preguntas sobre JavaScript:

- Introducir los elementos necesarios para que las últimas cinco sentencias del fichero 'xxx', realmente las primeras cinco que deben ejecutarse, demoren su ejecución hasta que esté disponible la mayor cantidad de recursos posible en el navegador y puedan ejecutarse de la manera más segura posible

Coges las 5 líneas de abajo e inicializamos el window.onload de esta forma

```
window.onload = startGame;
```

Creas la función StartGame y lo metes dentro:

```
function startGame() {

   fightcounter.toTitle();
   fight.timer = setInterval(fight.animate,T);
   fight.initWeapons();
   initBackground();
   var T = 1000 / 50;
```

```
}
```

- Declara una función constructora de objetos, llamada FightCounter, capaz de construir un nuevo objeto fightcounter completamente equivalente al actual. Seguidamente, sustituye en el código la declaración y creación del objeto actual fightcounter por la declaración de la nueva función constructora y la creación del nuevo objeto fightcounter, que0 se construye mediante FightCounter

```
var fightcounter = new FightCounter(1);
function FightCounter(count)
{
   this.count = count;
}
```

- Introduce los elementos necesarios para que el título 'xxx' acepte clics de ratón en todo momento y los gestione de dos formas distintas. Procede incrementalmente según se indica a continuación:
    - Mientras se prolonga la animación de las armas, cada clic debe alternar entre pausarla y continuarla (7a)

Se le pone un addEventListener al título (mytitle) y llama al atributo advancing y lo cambia de valor

```
mytitle.addEventListener("mousedown", stopAndResume);

function stopAndResume(){
   fight.advancing = !fight.advancing;
}
```

    - Cuando la animación de las armas haya finalizado, debe cambiarse la gestión de clics para que en este momento un sólo clic reinicie la animación y la gestión anterior de los clics, es decir (7b)
        - Vuelva a colocar las imágenes en sus posiciones iniciales
        - desactive la actual gestión de clics en el título y
        - reactive la gestión de clics desarrollada en 7a

Dentro de advancedWeapons en la condición de colisión quitamos y ponemos los eventlistener correspondientes. SIN COMPLETAR

```
fight.advanceWeapons = function () {
   fight.gauntlet.clear();
   fight.stormbreaker.clear();
   fight.gauntlet.move(-1);
   fight.stormbreaker.move(2);
```

```
    var sb = fight.stormbreaker, gl = fight.gauntlet;
    if (sb.left + sb.width > gl.left + gl.width / 3) {
        clearInterval(fight.timer);
        mytitle.removeEventListener("mousedown", stopAndResume);
        mytitle.addEventListener("mousedown", reset);
    }
};


function reset()
{
    fight.gauntlet.clear();
    fight.stormbreaker.clear();
    fight.gauntlet.reset();
    fight.stormbreaker.reset();
    mytitle.removeEventListener("mousedown", reset);
    mytitle.addEventListener("mousedown", stopAndResume);
    fight.advancing = true;
}
```

- Amplia la gestión de (7b) para que, con cada reinicio de la animación de las armas, se incremente el objeto fightcounter y se traslade su valor actualizado al título, lo que da sentido a la existencia de este objeto en el código

- (13 %) Introduce el código necesario para que, en cualquier momento, un clic sobre la imágen Play cambie de pantalla, desactivando la actual y activando la segunda e iniciando su propio código.

Con esto se cambia de pantalla, se coge el id que se ha puesto en html, y se pone “none” para quitar y “inline” para poner

```
document.getElementById("IDENHTML").style.display = "none";
document.getElementById("IDENHTML").style.display = "inline";
```

Entonces simplemente habríaque hacer un addveentlistener del botón play (como con el de mytitle) para que llame una función que haga lo anterior.

Cosas de Javascript (Kendra):

Sustituir el rectángulo por una imagen dependiendo de su color:

```
   render(ctx) {
       if (this.color===SQUARE_COLOR){
           let img = new Image();
           img.src= "rocket.png";
           ctx.drawImage(img, this.x, this.y);
         }
       else{
       ctx.fillStyle = this.color;
       ctx.fillRect(this.x, this.y, this.width, this.height);}
   }
```

IMPORTANTE: cómo hacer en javascript que no puedas moverte si mueres:

```
      window.document.removeEventListener("keydown", handlerOne);

      window.document.removeEventListener("keyup", handlerTwo);
```

Funciones para hacer que el juego en JAVASCRIPT tenga una pantalla de muerte:

```
function ending () {        //Change to the end screen

   var imgEnd = document.getElementById("ending"); → tienes q cargar la
imagen en html con el nombre ending o lo que quieras

   gameArea.context.drawImage(imgEnd,0,0);

   endGame();

}

function endGame() {        //SHUT DOWN
   continueGame = false;
   obstacles = [];
   delete theSquare;
   clearInterval(gameArea.interval);
   window.document.removeEventListener("keydown", handlerOne);
   window.document.removeEventListener("keyup", handlerTwo);
```

```
   var canvas = document.getElementsByClassName("gamelayer");
   canvas.display='none';
   gameArea.context.font = "30px Arial";
   gameArea.context.fillStyle = "yellow";
   gameArea.context.fillText("TIME ALIVE: " + pad(minutes, 2) + ":" +
pad(seconds, 2) ,250,100);
```

EJERCICIO EXAMEN JAVASCRIPT 2019 JUNIO

8. (0,5 ptos.) Al ejecutar el juego verás que no funciona, ya que es necesario asignar los valores adecuados a las variables `canvas`, `canvasWidth` y `canvasHeight` en las líneas 111, 112 y 113 del fichero juego.js. Estos valores deben obtenerse a partir del elemento `canvas`, identificado como `canvasGame`, en el fichero index.html. Usa las funciones adecuadas del DOM para ello. Una vez resuelto, el juego ya debería de funcionar: usa las teclas de los cursores para mover el rectángulo granate (la nave) y esquivar los rectángulos oscuros (ovnis). **Si no sabes resolver este ejercicio**, y con el fin de que puedas continuar con el resto, dile al profesor que está cuidando el examen que te proporcione la respuesta.

9. (1,0 ptos.) Modifica la clase `Ship`:

8.

```
1. let canvas = document.getElementById("canvasGame");
2. let canvasWidth = canvas.width;
3. let canvasHeight = canvas.height;
```

9. (1,0 ptos.) Modifica la clase `Ship`:

- Modifica el constructor para que admita un parámetro más, `img`, que será *opcional*, con valor por defecto `null`. Su valor se almacenará en un atributo de la clase con el mismo nombre.

- Añádele dos métodos: `setX` y `setY` que permitirán cambiar la posición de cualquier instancia de la clase en los ejes X e Y respectivamente.
- Finalmente, modifica el método `render` para que, si el valor almacenado en el atributo `img` es distinto de `null`, dibuje la imagen correspondiente en el canvas. Si fuese `null` el método debe dibujar el rectángulo correspondiente tal como venía haciendo hasta ahora.

Usa la imagen del fichero spaceship.png que hay dentro de la carpeta imgs para representar la nave en lugar del rectángulo granate, considerando las modificaciones a la clase `Ship` que acabas de realizar.

Ejercicio 9.

```
class Ship {
    constructor(x, y, width, height, color, img=null) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
        this.color = color;
        this.speedX = 0;
        this.speedY = 0;
        this.img=img;

    }

    setSpeedX(speedX) {
        this.speedX = speedX;
    }

    setSpeedY(speedY) {
        this.speedY = speedY;
    }

    setX(x){
        this.x=x;
    }
    setY(y){
        this.y=y;
    }
```

pasamos la imagen por parámetro y lo ponemos como null
creamos las funciones setX y setY como nos pedían en el enunciado.
Luego, en la función render ponemos esto:

```
    render(ctx) {
        if(this.img!=null){
        ctx.drawImage(this.img, this.x, this.y);
        }
        else{
        ctx.fillStyle = this.color;
        ctx.fillRect(this.x, this.y, this.width, this.height);}
    }
```

Y antes de llamar a la clase para hacer un objeto de tipo ship, inicializamos la imagen:

```
let img = new Image();
img.src= "imgs/spaceship.png";
let theShip = new Ship(0, canvasHeight / 2, SHIP_WIDTH, SHIP_HEIGHT,
DEFAULT_COLOR_SHIP, img);
```

y se la pasamos como parámetro. Si existe la dibujará, y si no, dibujará un cuadrado.

Ejercicio 10.

10. (1,2 ptos.) Añade un nuevo tipo de ovni al juego: se dibujarán con la imagen nave_alien.png que encontrarás en la carpeta imgs, aparecerán cada 3 segundos con una probabilidad del 60 %, tendrán una posición aleatoria en el eje Y comprendida entre *una quinta* y *cuatro quintas partes* de la altura del canvas, su velocidad en X y en Y estará comprendida entre 6 y 15 y deben *rebotar* cuando alcancen cualquiera de los límites superior e inferior del canvas. La velocidad en Y podrá ser positiva o negativa con la misma probabilidad (50 %). El resto de características de este nuevo tipo de ovnis serán las mismas que tienen los originales. Almacénalos en el mismo vector en el que se almacenan los originales.

Añades estas constantes para marcar el tiempo y la probabilidad:

```
const SECONDS_ALIEN = 3;
const PROBABILITY_ALIEN = 0.6;
```

En el constructor de la clase GAMECANVAS, después de _ufos, añadimos _aliens(o el nombre que le queramos poner)

```
class GameCanvas {
    constructor(_canvas, _ship, _ufos, _aliens) {
        this.canvas = _canvas;
        this.ship = _ship;
        this.ufos = _ufos;
        this.aliens=_aliens;
        this.context = null;
        this.interval = null;
    }
```

Y cuando creemos un objeto de tipo GameCanvas, al final añadimos un [] para indicar que aliens es un Array también al igual que Ufos

```
let gameArea = new GameCanvas(canvas, theShip, [], []);
```

```
    render() {
        for (const ufo of this.ufos) {
            ufo.render(this.context);
        }
        for (const alien of this.aliens) {
            alien.render(this.context);
        }
        this.ship.render(this.context);
    }


    addAlien(alien){
        this.aliens.push(alien);
    }
    removeAlien(j){
        this.aliens.splice(j, 1);
    }

    addUfo(ufo) {
        this.ufos.push(ufo);
    }

    removeUfo(i) {
        this.ufos.splice(i, 1);
    }
```

Ahora básicamente hay que mirar todo lo que hace con los ufos para copiarlo con los aliens. En la función render renderiza los ufos, pues ponemos lo mismo pero con los aliens, y además creamos addAlien y RemoveAlien

creamos una funcion para crear los aliens

```
function crearAliens(){
    if (seconds% SECONDS_ALIEN===0){
        let chance2 = Math.random();
        if(chance2< PROBABILITY_ALIEN){
            let posY = Math.floor(Math.random() * (canvasHeight-(canvasHeight/5)*2)+canvasHeight/5);
            let imgalien = new Image();
                imgalien.src= "/imgs/nave_alien.png";
            let alien = new Ship(canvasWidth, posY, UFO_WIDTH, UFO_HEIGHT, DEFAULT_COLOR_UFOS, imgalien);
            let speed = Math.floor(Math.random()*(15-6)+6);

        alien.setSpeedX(-speed);
        if(Math.random()<0.5) speed*=-1;
        alien.setSpeedY(speed);
        gameArea.addUfo(alien);
        }

    }
}
```

Es igual que la de crearUfos, aun no he modificado las características que faltan como que aparezcan en tal sitio y tengan tal velocidad.

Se crea una imagen antes de crear un objeto tipo alien(un ship) y se le pasa por parámetro

En chrono que es donde crea los ufos ponemos la funcion crearAliens();

Ahora falta lo último, que tengan velocidad.

Nos vamos a la función updateGame() que es la que utiliza para que los ufos se muevan, y copiamos y pegamos el mismo código de ufos pero cambiando los nombres por el de Aliens:

```
function updateGame() {
    // Check collision for ending game
    let collision = false;
    for (let i = 0; i < gameArea.ufos.length; i++) {
        if (theShip.crashWith(gameArea.ufos[i])) {
            collision = true;
            break;
        }
    }
    if (collision) {
        endGame();
    } else {
        // Move ufos and delete the ones that goes outside the canvas
        for (let i = gameArea.ufos.length - 1; i >= 0; i--) {
            gameArea.ufos[i].move();

            if(gameArea.ufos[i].y <= gameArea.ufos[i].height/2 || gameArea.ufos[i].y > (canvasHeight-gameArea.ufos[i].hei
                gameArea.ufos[i].speedY*=-1;}
            if (gameArea.ufos[i].x + UFO_WIDTH <= 0) {
                gameArea.removeUfo(i);
            }
        }

        // Move our hero
        theShip.move();
        // Our hero can't go outside the canvas
        theShip.setIntoArea(gameArea.canvas.width, gameArea.canvas.height);
        gameArea.clear();
        gameArea.render();
    }
}
```

Para que esté entre ⅛ y 5/5 en el eje y:

```
let posY = Math.floor(Math.random() *
(canvasHeight-(canvasHeight/5)*2)+canvasHeight/5);
```

Para que sea una velocidad aleatoria entre 6 y 15:

```
let speed = Math.floor(Math.random()*(15-6)+6);
```

Vale esto es importante: al final del ejercicio te dice que se tienen que almacenar en el mismo array, y yo no lo había visto y he creado dos arrays diferentes. Voy a modificar el ejercicio para hacer que reboten y que esten en el mismo array, lo pondré a continuación:

Vale pues así es mucho más fácil, porque no hay que copiar todo el rato lo mismo como hemos hecho arriba, pero hay que pensar un poco más:

```
function crearAliens(){
   if (seconds% SECONDS_ALIEN===0){
       let chance2 = Math.random();
       if(chance2< PROBABILITY_ALIEN){
```

```
            let posY = Math.floor(Math.random() *
(canvasHeight-(canvasHeight/5)*2)+canvasHeight/5);
            let imgalien = new Image();
                imgalien.src= "/imgs/nave_alien.png";
            let alien = new Ship(canvasWidth, posY, UFO_WIDTH, UFO_HEIGHT,
DEFAULT_COLOR_UFOS, imgalien);
            let speed = Math.floor(Math.random()*(15-6)+6);
        alien.setSpeedX(-speed);
        if(Math.random()<0.5) speed*=-1;
        alien.setSpeedY(speed);
        gameArea.addUfo(alien);
        }
```

Es hacer esta función básicamente, y con el addUfo añadirlos en el mismo array que los ufos (hay que borrar las demás arrays y cosas que hemos hecho antes).

Eso si nos piden que sea en el mismo array, si no hay que hacerlo de la primera forma.

Para que reboten:

```
238     } else {
239       // Move ufos and delete the ones that goes outside the canvas
240       for (let i = gameArea.ufos.length - 1; i >= 0; i--) {
241           gameArea.ufos[i].move();
242
243           if(gameArea.ufos[i].y <= gameArea.ufos[i].height/2 || gameArea.ufos[i].y > (canvasHeight-gameArea.ufos[i].heigh
244               gameArea.ufos[i].speedY*=-1;}
```

en crearAliens() tenemos esto para que la velocidad en Y sea negativa o positiva aleatoriamente como nos dice en el enunciado:

```
if(Math.random()<0.5) speed*=-1;
```

Ejercicio 11.

11. (2,3 ptos.) Utiliza el elemento `div` identificado mediante la etiqueta `livesleft` para llevar un contador de las vidas del jugador. Tienes que modificar el juego para que permita *tres* vidas en total. Además, tras cada "muerte" de la nave —excepto la última, ya que el juego se acabará— el juego se pausará durante 4 segundos y se reanudará con la nave situada en su posición original y todos los ovnis que pudieran haber en pantalla eliminados. Para ello será necesario, *entre otras cosas*, que realices las siguientes acciones:

    - En primer lugar añade las reglas de estilo necesarias en el fichero ejercicio3.css que hay dentro de la carpeta css para que este elemento tenga un margen izquierdo de 2 espacios (em), un margen superior de un espacio, un espacio de relleno para sus elementos (*pad*) de 0,2 espacios, alineación vertical *middle* y modo de visualización *inline-block*.
    - Ya en el fichero juego.js, añade una constante para el número máximo de vidas. También tendrás que almacenar en una variable el elemento `livesleft` del DOM, tal como hiciste al principio de este bloque con el canvas y pasárselo en el constructor a la clase `GameCanvas` para que lo guarde (tal como ocurre con el canvas).
    - Añade un método a la clase `GameCanvas`, `renderLives`, que se encargará de dibujar las vidas que le quedan al jugador: para ello, utilizará tantas imágenes de la nave (spaceship.png) como sean necesarias para indicar las vidas restantes. Estas imágenes tendrán aquí unas dimensiones de 82 de ancho por 39 de alto —define constantes para almacenar estos valores—. Este método se invocará tanto al inicio del juego como cuando la nave "muera", es decir, colisione con un ovni.
    - Añade otro método a esta misma clase al que llamarás `removeAllUfos`. Este método simplemente dejará vacío el vector `ufos` de la clase.

1.

```
#livesleft{
margin-left: 2em;
margin-top: 2em;
padding: 0.2em; //esto me lo he inventao no sé realmente lo que es el PAD. Si alguien lo sabe que lo ponga bien.
vertical-align: middle;
display: inline-block;
}
```

2.

```
const MAX_LIVES = 3;
let livesleft = document.getElementById("livesleft");
```

Lo ponemos en el constructor pasándolo como parámetro.

Cosas del fútbol (práctica 3):

```
function init() {
    // set interval to call gameloop logic in 30 FPS
    tableFootballData.timer = setInterval(gameloop, 1000 / 30);

    // view rendering
    tableFootballData.request = window.requestAnimationFrame(render);
    tableFootballData.isRendering = true;
    tableFootballData.isPaused = false;

    // inputs
    handleMouseInputs();
```

```
// To handle mouse's enter, move, and leave events
function handleMouseInputs() {
    // get the playground element
    pGround = document.querySelector("#playground");

    // run the game when mouse moves in the playground.
    pGround.addEventListener("mouseover", function (event) {
        tableFootballData.isPaused = false;
    });

    // pause the game when mouse moves out the playground.
    pGround.addEventListener("mouseout", function (event) {
        tableFootballData.isPaused = true;
    });

    // calculate the paddle position by using the mouse position.
    pGround.addEventListener("mousemove", function (event) {
        tableFootballData.paddleB.y = event.pageY - tableFootballData.playground.offsetTop;
    });
```

movimiento de la bola:

```
function moveBall() {
    let ball = tableFootballData.ball;

    // check playground top/bottom boundary
    if (ballHitsTopBottom()) {
        // reverse direction
        ball.directionY *= -1;
    }
    // check right
    if (ballHitsRightWall()) {
        playerAWin();
    }
    // check left
    if (ballHitsLeftWall()) {
        playerBWin();
    }

    // Variables for checking paddles
    let ballX = ball.x + ball.speed * ball.directionX;
    let ballY = ball.y + ball.speed * ball.directionY;
    // check moving paddle here, later check right paddle
    if (ballX >= tableFootballData.paddleA.x && ballX < tableFootballData.paddleA.x +
        tableFootballData.paddleA.width) {
        if (ballY <= tableFootballData.paddleA.y + tableFootballData.paddleA.height &&
            ballY >= tableFootballData.paddleA.y) {
            ball.directionX = 1;
        }
    }

    // check right paddle
    if (ballX >= tableFootballData.paddleB.x && ballX < tableFootballData.paddleB.x +
        tableFootballData.paddleB.width) {
        if (ballY <= tableFootballData.paddleB.y + tableFootballData.paddleB.height &&
            ballY >= tableFootballData.paddleB.y) {
            ball.directionX = -1;
        }
    }

    // update the ball position data
    ball.x += ball.speed * ball.directionX;
    ball.y += ball.speed * ball.directionY;
```

el choque que de la bola con la pared de la derecha:

```
function ballHitsRightWall() {
    return tableFootballData.ball.x + tableFootballData.ball.speed *
        tableFootballData.ball.directionX > tableFootballData.playground.width;
```

cosas de la práctica 2 (la del martillo):

crear las armas:

```
class Weapon {
    constructor(c2d, filename, l, t) {
        this.canvas2d = c2d;
        this.image = new Image();
        this.image.myparent = this;
        this.image.src = filename;
        this.image.onload = function () {
            let m = this.myparent;
            m.canvas2d.drawImage(this, l, t);
            m.width = this.naturalWidth;
        };
        this.left = l;
        this.top = t;
        this.width;
    }
    ;
            clear() {
        let l = this.left, t = this.top;
        let w = this.image.naturalWidth;
        let h = this.image.naturalHeight;
        this.canvas2d.clearRect(l, t, w, h);
    }
    ;
            move(n) {
        this.left += n;
        let i = this.image;
        let l = this.left;
        let t = this.top;
        this.canvas2d.drawImage(i, l, t);
    }
```

cambiar el color del fondo con sus inicializaciones:

```
let colors = ["cyan", "green", "gray", "blue"];
let cIndex = 0;
let clash = {shield: undefined, mjolnir: undefined};
clash.timer = undefined;

clash.changeBackgroungColor = function () {
    const LEAP = 7;
    const LONG_PERIOD = 20;
    const SHORT_PERIOD = 1;
    let period;
    let sh = clash.shield, mj = clash.mjolnir;
    //pregunta si estan a una distancia corta
    if (sh.left + 2 * sh.width > mj.left)
        period = SHORT_PERIOD;
    else
        period = LONG_PERIOD;
    if (sh.left % period === period - 1) {
        //Coge colores aleatorios
        cIndex = (cIndex + LEAP) % colors.length;
        let bg = document.getElementById("battlefiel
        bg.style.backgroundColor = colors[cIndex];
    }
};

clash.advanceWeapons = function () {
    const SHIELD_MOVE = 1;
    const MJOLNIR_MOVE = -2;
    clash.shield.clear();
    clash.mjolnir.clear();
    clash.shield.move(SHIELD_MOVE);
    clash.mjolnir.move(MJOLNIR_MOVE);
    let sh = clash.shield, mj = clash.mjolnir;
    if (sh.left + sh.width / 2 > mj.left)
```

animación:

```
clash.advancing = false;

clash.animate = function () {
    if (clash.advancing)
        clash.advanceWeapons();
    clash.changeBackgroungColor();
};

clash.initWeapons = function () {
    clash.cv = document.getElementById("scene");
    let ctx = clash.cv.getContext("2d");
    let sh = "CaptainAmericaShield.png";
    clash.shield = new Weapon(ctx, sh, 0, 260);
    let mj = "ThorMjolnir.png";
    clash.mjolnir = new Weapon(ctx, mj, 651, 250);

};

function initBackground() {
    for (let i = 0; i < colors.length; i += 3) {
        colors.splice(i, 0, "light" + colors[i]);
        colors.splice(i + 2, 0, "dark" + colors[i + 1]);
    } // colors === ["lightcyan","cyan","darkcyan","ligh
    //let decIndex = Math.random() * colors.length;
    let decIndex = randomArrayIndex(colors);
    //console.log(decIndex);
    cIndex = Math.floor(decIndex);
    let bg = document.getElementById("battlefield");
    bg.style.backgroundColor = colors[cIndex];
}
;
```

ratón fuera del canvas y entryPoint:

```
clash.mouseOverOutCanvas = function () {
    clash.cv.onmouseover = function () {
        clash.advancing = true;
    };
    clash.cv.addEventListener("mouseout",
            function () {
                clash.advancing = false;
            });
};



//La funcion de generar un numero random
function randomArrayIndex(array) {
    let n = Math.random() * (array.length - 0) + 0;
    return n;
}

function entryPoint() {
    initBackground();
    clash.initWeapons();
    clash.mouseOverOutCanvas();
    const T = 1000 / 50; // 20 ms
    clash.timer = setInterval(clash.animate, T);
}
;

// Entry point
window.onload = entryPoint;
```

```
function initMyDOM() {
   let background = document.GetElementById("battlefield");
   let div1 = document.createElement("div");
   background.appendChild(div1);
}
```