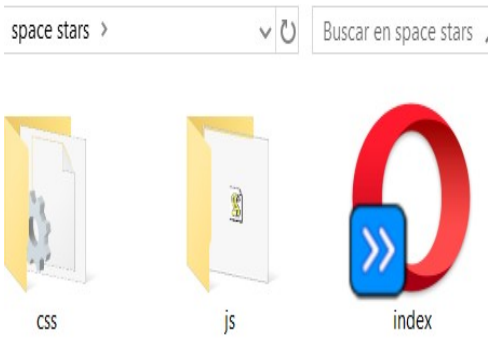
Memoria SpaceStars

Explicacion de archivos utilizados , código ,funciones y funcionalidades extras

Distribucion y archivos utilizados :

Este juego esta hecho con html , css y javascript ,y su distribucion es el archivo html (index.html) en el directorio principal que es el esqueleto del juego donde se van a presentar todas las funcionalidades css y javascript cuyos archivos estan enlazados a este . El archivo css (style.css) esta en la carpeta css y son los estilos y el aspecto de los elementos html de index.html . El archivo javascript (script.js) esta en la carpeta js y se encarga de ejecutar las funcionalidades y hacer funcionar la logica del juego . No hay archivos adicionales descargados .



Explicacion del codigo html :

En el archivo index.html estan los enlaces con el css y javascript y tambien los elementos que seran modificados por el css y el javascript , existen bastantes tipos de elementos y hay que saber que es cada uno , la mayoria son muy basicos pero se destaca que algunas tienen funciones que se activas al pulsar sus campos con onclick , hay muchas maneras de activar una funcion , tambien esta el elemento canvas que es la zona de juego , tambien cuando carga la pagina empieza el juego con onload en el body y tambien hay un script que es una api de geolocalizacion con sus propios metodos , clases y funciones que habria que aprenderse , en este caso solo se esta usando lo mas basico que es mostrar la latitud y la longitud que se mostrara si todo esta bien , sino mostrara el error correspondiente y si no funciona la api manda un mensaje personalizado cambiable . Su funcion getLocation es la que se encarga de decidir que pasa , la funcion showPosition escribe la posicion y la funcion showError escribe los errores .

```
    <link rel="stylesheet" type="text/css" href="css/style.css">
    <script type="text/javascript" src="js/script.js"></script>
</head>
<body onload="canvasStars()">
  <div>
      <h1>Space Stars</h1>
      <span id="mensaje" onclick="reiniciar()">Intenta llegar a l
      Puntuación: <span id="puntuacion">40</span><br>
      Tiempo: <span id="tiempo">15</span>
  </div>
  <canvas id="miCanvas" width="600" height="600"></canvas>
```

```
<script>
const x = document.getElementById("demo");

function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition,showError
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
    x.innerHTML="Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}

function showError(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      x.innerHTML = "User denied the request for Geolocation."
      break;
```

Explicacion del código css :

En el archivo style.css se puede ver el codigo que cambia el aspecto de los elementos html , obvio con sus reglas ortograficas . Se pone el elemento con las llaves detras y con dentro la cosa que quieres cambiar , despues dos puntos , despues el valor y por ultimo un punto y coma . Puedes poner todos los que quieras dentro de las llaves , para cambiar el aspecto de los elementos con clase antes de las llaves hay que poner un punto y el nombre de la clase junto, para cambiar el aspecto de los elementos con id , antes de las llaves hay que poner almohadilla y el nombre de la id junto y puedes cambiar el aspecto de muchos elementos a la vez poniendo una coma entre ellos . Y bueno para poder trabajar con ello hay que saber los cambios que se pueden realizar y sus nombres y ponerles valores validos .

```css
h1{
    margin: 0;
}
#mensaje{
    font-size: 18px;
    font-weight: bold;
    cursor: pointer;
}
#puntuacion,#tiempo{
```

Explicacion del codigo javascript :

En el el archivo script.js estan todas las funciones que hacen que el juego sea jugable y funcional , al principio esta la declaracion de las variables publicas que seran utilizadas dentro de las funciones , las funciones tambien tiene sus variables privadas que no sirven fuera de ellas , la estructura de este codigo javascript es una funcion principal canvasStars que ejecuta a las demas funciones al cargar la pagina porque tiene un onload en el html , hasta que se acaba el juego .

```javascript
//Declaración de variables

var canvas, ctx;

var naveX = 0; //Posición original en x de la nave

var naveY = 0; //Posición original en y de la nave

var nave = new Image(); //Imagen para capturar la nave

var fondoNave = new Image(); //Imagen para capturar el fondo

var contador= 40; //Contador de movimientos

var tiempo = new Date(15000); //Para tener solo 15 segundos en milisegundos

var stop; //Para parar el temporizador
```

```javascript
//Inicio el juego

function canvasStars(){

    //Obtengo el elemento canvas
    canvas = document.getElementById("miCanvas");

    //Especifico el contexto 2D
    ctx = canvas.getContext("2d");

    //Llamo a la función que pinta el fondo con las estrellas

    pintarFondo();


    //Llamo a la función que pinta la nave

    pintarNave();

    //Llamo a la función que pinta la base

    pintarBase();


    //Llamo a la función que pinta los asteroides

    pintarAsteroides();
```

Explicacion de las funciones javascript :

Voy a explicar que hace cada funcion de una en una .

La funcion canvasStars hace que el juego empiece y funcione ejecutando las demas funciones .

La funcion pintarFondo usa el ctx que es el fondo y con fillStyle cambia el color de fondo a negro y con rect elige el tamaño de lo que se va a pintar y con fill lo añade , dentro de esta funcion tambien esta el codigo para pintar estrellas que es un bucle for y la condicion del centro es el numero de estrellas , dentro se declaran las variables x e y que tendran una posicion random con Math.random *600 para que no se salgan y despues vuelve a usar el ctx.fillStyle para que sean blancas y arc porque son circulos pero igual que rect y fill para añadirlo y por ultimo se define la variable fondoNave para que guarde el fondo y no se pierda cuando pase la nave .

```
function pintarFondo(){

    //Pinto el fondo negro

    ctx.fillStyle = "black";

    ctx.beginPath();

        ctx.rect(0, 0, 600, 600); //posición x, posición y, ancho, alto

    ctx.closePath();

    ctx.fill();


    //Pinto 100 estrellas

    for (i=0; i<100; i++){

        //Posiciones x e y aleatorias

        var x = Math.random() * 600;

        var y = Math.random() * 600;
```

La funcion pintarBase y pintarNave son lo mismo que en la imagen anterior entre las tres primeras anotaciones solo que en pintarNave se a definido la imagen de la nave para guardarla y poder moverla

```
    ctx.fill();

    //Guardo la nave como imagen

    nave = ctx.getImageData(0, 0, 30, 30);
```

La funcion pintarAsteroides tambien es casi igual solo que como las estrellas estan en lugares random como ya he explicado y tambien comprueba que no esten en la casilla de la nave y la casilla de la base con if y si estan en ellas se mueven 30 pixeles para atrás para tener el camino libre

```
function pintarAsteroides(){

    for (i=0; i<30; i++){

        var x = Math.random() * 570;
        var y = Math.random() * 570;

        //Compruebo que no hay asteroides en la nave

        if (x < 30 && y < 30){

            x = x + 30;
            y = y + 30;
        }

        //Compruebo que no hay asteroides en la base
```

Para que la funcion para moverl la nave funcione hay que añadir esta linea de codigo dentro de la funcion canvasStars .

```
//Añado el escuchador del teclado

window.addEventListener('keydown', moverNave, true);
```

La funcion moverNave hace que al pulsar los botones con el codigo correcto se mueva , esto se hace por un switch , dentro de cada caso se actualiza el contador llamando su funcion , tambien se cancela el movimiento si se va a salir del canvas con un if y break y tambien hace uso de lo que vimos antes de fondoNave con un ctx.putImageData para quitar la nave despues para guardar el fondo que seva a tapar y despues para mover la nave a su siguiente posicion y tambien se detecta la colision llamando su funcion .

```
actualizarContador();

//Compruebo si se va a salir por la izquierda

if (naveX == 0){
    break;
}

//Borro la nave (pintando fondoNave encima)
ctx.putImageData(fondoNave, naveX, naveY);

//Actualizo la x

naveX = naveX - 30;

//Capturo el fondo que voy a tapar

fondoNave = ctx.getImageData(naveX, naveY, 30, 30);

//Muevo la nave

ctx.putImageData(nave, naveX, naveY);

//Compruebo colisión

detectarColision();

break;
```

En la funcion detectarColision se crean las variables pixels y elementos que son todos los pixeleles del interior del juego y con un bucle for se va a recorrer cada pixel del juego para ver si he colisionado con algo con los if que hay dentro y el fondoNave.data[i] , [i+1] y [i+2] es el codigo del color rgb y cada color tiene uno y cada objeto es de un color entonces se puede saber con que has colisionado y depende pasan cosas diferentes .Si chocas con un asteroide te manda un mensaje de que has perdido y se acaba el juego llamando a la funcion finalizar , si llegas a la base lo mismo pero cambia el mensaje

```
function detectarColision(){

    var pixels = 900; //Porque la imagen es de 30x30 pixels

    var elementos = pixels*4; //Porque cada pixel tiene 4 bytes (RGBA)



    //Recorro en busca del rojo (asteroide) o del azul (base)

    for (var i = 0; i < elementos; i += 4){



        //Asteroide (255, 0, 0)

        if (fondoNave.data[i] == 255 && fondoNave.data[i+1] == 0 && fondoNave.data[i+2] == 0){

            var mensaje = "¡Lo siento! Has chocado con un asteroide.<br>Pincha AQUÍ para volver a intentarlo.";

            finalizar(mensaje);

            break;

        }
```

La funcion actualizarContador baja el valor que se añadio inicialmente en uno cada vez que se le llama y eso ocurre cuando haces algun movimiento porque es desde la funcion de mover , que se accede a esta funcion y se escribe en un elemento del html y tambien dentro tiene que llegando a diferentes numeros con if se cambia de color y si te quedas sin puntos aparece un mensaje de perder y se acaba el juego .

```
//Decremento en cada movimiento

contador--;

//Capturo el elemento en el que escribir la puntuación

var spanPuntuacion = document.getElementById("puntuacion");

//Actualizo el contador

spanPuntuacion.innerHTML = contador;

//Compruebo el valor para cambiar el color del texto

if (contador < 6){

    spanPuntuacion.style.color = "red";

}

else if (contador < 11){

    spanPuntuacion.style.color = "orange";
```

La funcion temporizador hace que baje el valor inicial de la variable tiempo en 500 ms todo el tiempo , tambien dentro se usa la funcion rellenar ceros en la parte donde se escribe el nuevo valor para que solo se escribar segundos enteros y despues tambien cambia de color como la funcion anterior

```
var ms = tiempo.getMilliseconds() - 500;

tiempo.setMilliseconds(ms);

//Muestro la nueva fecha

var texto = rellenaCeros(tiempo.getMinutes()) + ":" + rellenaCeros(tiempo.getSeconds());

var spanTiempo = document.getElementById("tiempo");

spanTiempo.innerHTML = texto;



//Compruebo el valor para cambiar el color del texto

if (tiempo.getSeconds() < 6){

    spanTiempo.style.color = "red";

}
```

La funcion rellenaCeros solo es un if que hace que siempre se devuelvan numeros enteros

```
if (numero < 10){
    return "0" + numero;
}
else{
    return numero;
}
```

La funcion finalizarJuego lo que hace es solo escribir el mensaje correspondiente a cada ocasión que sea llamado , una linea para bloquear el movimiento y una linea para que la pagina deje de cargar .

```
function finalizar(mensaje){

    //Capturo el elemento en el que voy a escribir

    var spanMensaje = document.getElementById("mensaje");

    //Escribo el mensaje en ese elemento
    spanMensaje.innerHTML = mensaje;

    //Bloqueo el movimiento del teclado
    window.removeEventListener("keydown", moverNave, true);

    clearTimeout(stop);

}
```

La funcion reiniciar solo es un reload que se activa al pulsar el elemento html con un onclick que lo llama

```
//Reinicio el juego
function reiniciar(){
    window.location.reload();
}
```

Funcionalidades extras :

He añadido una funcion pintarCombustible que es igual a la funcion pintarAsteroides pero de color naranja y solo genera dos

```
//Compruebo que no hay combustible en la base

if (x > 540 && y > 540){

    x = x - 30;

    y = y - 30;

}


//Pinto el combustible

ctx.fillStyle = "orange";

ctx.beginPath();

    ctx.rect(x, y, 20, 20);

ctx.closePath();

ctx.fill();
```

y he añadido su if de su color en la funcion detectarColision para que al colisionar con el haga cosas . Lo que hace cuando colisionas con el es que suma movimientos al contador , suma tiempo al temporizador y borrarse cuando pasas por encima , y eso es lo que mas me gusta de esta funcionalidad extra porque solo aparecen dos pero depende de tu suerte tener mas usos , porque puede que aparezca justo para que lo cojas entero o puede que como te mueves de 30 en 30 pixeles puedas usar mas de una vez uno .

```
//Combustible (255, 165, 8)

if (fondoNave.data[i] == 255 && fondoNave.data[i+1] == 165 && fondoNave.data[i+2] == 0){

    contador = contador+5;
    tiempo.setMilliseconds(tiempo.getMilliseconds()+5000);
    this.fondoNave=ctx.getImageData(0, 0, 30, 30);
    ctx.putImageData(fondoNave,20,20);
    break;

}
```