TRABAJO PARA EL PROYECTO FINAL DEL CURSO DE DESARROLLO DE APLICACIONES WEB

ESTIMULADOR DE REFLEJOS MECÁNICOS

AUTOR: Moyano Fajardo, Daniel

DNI: 05964807X

En Madrid, a día 13 de Junio de 2024

RESUMEN

El proyecto “SUPER MENU” se trata de una oda a la interfaz de usuario del primer Super Hot y mostrar la cantidad de contenido que queda eclipsado por el juego que el usuario suele perderse al jugar el título, adicionalmente en la 2ª evaluación una vez terminado el subproyecto de treedude, un minijuego presente dentro del menú de Super Hot, se realizará un test de tipografía para ayudar a mejorar la mecanografía del usuario. Todos los apartados del menú y los propios juegos se desarrollaron usando el editor de código: Visual Studio Code con los lenguajes html, css y javascript. En un primer momento se desarrolló el proyecto para ser utilizado en ordenadores y portátiles.

Con todo esto, la meta final del proyecto es mostrar todo el contenido adicional del Super Hot original y que pasen un buen rato practicando mecanografía y jugando el minijuego de treedude. La idea final es imitar el comportamiento del treedude original con una barra de vida que vaya bajando más rápido según el jugador vaya obteniendo más puntos aumentando así la presión sobre el jugador, así mismo con el test de tipografía su meta será ayudar a mejorar la mecanografía del usuario.

Las puntuaciones de treedude serán guardadas en las cookies y si el usuario decide pagar podrá guardarlas en una bbdd mediante blockchain y una API externa, mientras que las puntuaciones del test de tipografía serán guardadas en el localStorage.

En resumen, el proyecto de SUPER MENU busca enseñar el contenido adicional presente en el juego de SUPER HOT, mejorar los reflejos del usuario mediante el minijuego de treedude y la mecanografía mediante el test de tipografía.

TABLA DE CONTENIDOS

TABLA DE IMÁGENES

ESTADO DEL ARTE

Para el correcto entendimiento de la documentación del proyecto es recomendable tener conocimientos básicos de programación ya que aunque se trató evitar en algunas partes se pueden encontrar etiquetas html y/o pedazos de código, no obstante en este apartado se presentan definiciones importantes por conocer para entender el proyecto

Las definiciones las podemos agrupar en los siguientes grupos:

- Etiquetas de html
- Conceptos de programación
- Conceptos de videojuegos
- Otros

I. ETIQUETAS HTML

Divs: Capas creadas por un programador para dividir la página en secciones.

Etiqueta span: Suele ser usada para delimitar zonas de textos y aplicarlos diferentes estilos o contextos.

h1, h2, h3…: Etiquetas de html cuya función es el equivalente de los estilos Título 1, Título 2, Título 3… del MicrosoftWord.

P: Etiqueta de html para la inserción de texto, al cerrarse hace un salto de línea

Span: Etiqueta de html genérica para la inserción de texto

II. CONCEPTOS DE PROGRAMACIÓN

{}: Dependiendo del contexto puede significar una cosa u otra:

- En pseudocódigo: implica que lo que hay entre los símbolos sería un conjunto de diferentes elementos
- En código: Se denominan llaves, utilizadas para definir el comienzo y final de un bloque de código

<>: Usado en pseudocódigo lo que se encuentre entre los menor y mayor que es obligatorio introducirlo

Array: Conjunto de elementos del mismo tipo que son agrupados.

Bucle: Bloque de código que se repite un número X de veces

Clase: En css una clase es la manera de referirse a un conjunto de elementos y aplicarles simultáneamente a todos el mismo estilo.

Código ASCII: O American Standard Code for Information Interchange, es un sistema codificado que asigna un valor numérico a diferentes caracteres utilizados en la comunicación electrónica. Fue desarrollado en la década de 1960 como un estándar para la transferencia de datos entre diferentes dispositivos informáticos.

Formato de color rgb(x,y,z); Red, **G**reen, **B**lue: Cada valor indica la cantidad del color que se aplica sobre el color final, al contrario que con los colores primarios de pigmento la suma de los 3 colores primarios de luz (rojo, verde y azul) da el blanco

Función: Conjunto de acciones/ bloque de código

Función anónima: Conjunto de acciones que no están enlazadas a un identificador.

Función trim: función que quita los espacios presentes al inicio de una cadena de caracteres (también conocida como string) y al final.

Github: plataforma donde uno puede crear repositorios de código y compartir proyectos de desarrollo con la comunidad al mismo tiempo que se posee un historial de versiones y modificaciones.

Interfaz de usuario: medio que permite la comunicación entre el usuario y la máquina, equipo o dispositivo usado

Mecanografía: Habilidad para introducir escribir mediante un teclado, los tests de mecanografía sirven para hacer que aumente la precisión y la velocidad en la que el sujeto pueda ser capaz de introducir información mediante el teclado.

Paleta de colores: Conjunto de colores usados en este caso en la estética de una página web

Parámetro: Valor que es pasado a una función

setInterval: Función que se activa en intervalos marcados por una cantidad de tiempo indicada por el programador y es usada para reproducir código cada X tiempo.

Trim: Función que sirve para quitar los espacios presentes al principio y final de una cadena

TypingTest: En español test de tipografía: Prueba para medir y aumentar la velocidad de escritura de un usuario con un teclado.

III. CONCEPTOS DE VIDEOJUEGOS

Discord: Red social donde se pueden crear servidores con usuarios, organización mediante roles, posibilidad de pasar archivos, entre otras funciones.

Hacer un DOOM: el DOOM original a pesar que parece estar hecho en 3D el mismo fue hecho completamente en 2D aumentando el tamaño de los sprites según el jugador se "acercaba") y replicar todo mediante código y manipulación de sprites.

Mapa de bits: Estructura o fichero de datos que representa una imagen de forma digital, en el que las dimensiones están determinados por la cantidad de píxeles horizontales y verticales que componen a cada imagen

Shooter: Refiriéndonos al ámbito de los videojuegos un shooter es aquel cuya misión principal es empuñar un arma de fuego como por ejemplo una pistola y normalmente acabar con todos los enemigos en pantalla

Sprites: Pequeño mapa de bits que es dibujado en la pantalla a partir de un dibujo inicial, en el ámbito de los videojuegos se define Sprite como un conjunto de imágenes que representa un personaje, objeto o una parte de ellos de manera gráfica y que se utiliza para poder crear cualquier efecto de movimiento o para cambiar su estado o posición en la escena

SUPER HOT: El segundo Shooter independiente más innovador que he jugado en años, siendo el 1º ULTRAKILL (y DOOM ETERNAL dentro de los triples A), la mecánica principal es que el mundo se mueve a una velocidad directamente proporcional al movimiento del jugador, si el jugador se queda quieto el juego irá a una cámara lenta tan alta que parecerá que no se mueve y si se mueve rápido el juego se verá a velocidad normal

Triple A: Se refiere a juegos desarrollados por grandes empresas de videojuegos como pueden ser Nintendo o SEGA. Siendo el caso opuesto a los juegos independientes, los cuales son creados por equipos más reducidos de personas, ejemplos de este último tipo son juegos como Super Hot y/o Undertale

IV. OTROS

Acorde: Conjunto de tres o más sonidos diferentes combinados armónicamente.

INTRODUCCIÓN

La idea inicial era la de crear un selector de juegos entre dos proyectos, el Estimulador de reflejos mecánicos también conocido como el minijuego treedude dentro del grandísimo juego SUPER HOT. Y el entregable de la primera evaluación y el test de mecanografía conocido como typingtest para la 2ª. Dicha idea evolucionó a replicar el menú de SUPER HOT para dar importancia a la increíble Interfaz del usuario del juego que a menudo queda eclipsada por la genialidad del propio título.

Debido a esto el uso del ratón quedará limitado en un principio solo al typingtest, e incluso se verá si también es eliminado de dicha zona, por otro lado, sí estará disponible un ratón propio en ciertos simuladores de otros apartados del menú. En el treedude la dificultad principal estará ubicada en la animación del juego que deberá ser hecha con puro código ASCII.

Luego la parte de la mecánica del juego será algo más sencilla: por un lado, se tendrá la vida que irá bajando poco a poco a no ser que el leñador treedude tale el árbol, cada vez que el árbol sea talado se bajará una posición. El árbol tendrá dos lados en los que se puede generar una rama, si en algún momento rama y leñador colisionan será Game Over para el leñador, parte del desafío será generar las ramas aleatoriamente en un lado u el otro. Por otro lado, se encuentra la parte del Typing test: consistirá en lo siguiente.

Mediante ya se verá si un array o un fichero json externo habrá una larga lista de palabras. Se generarán unas primeras 6*6 números aleatorios del 0 - al nº de palabras del fichero para seleccionar las palabras de manera aleatoria. La idea es calcular las palabras * minuto y pulsaciones * por minuto del usuario en un tiempo de 1 minuto. cuando el usuario complete una línea de palabras se generarán abajo del todo una ristra nueva de seis palabras, la cuenta atrás iniciará con la primera pulsación del usuario sea correcta o incorrecta.

Para finalizar como oda a una buena interfaz de usuario se intentará replicar el menú del superhot, en un principio el único directorio que no tendrá funcionalidades será el de settings, no obstante, será incluido y representado también el resto se tratará de replicar cuanto más parecido posible mediante código evitando el uso de videos siempre que sea posible, como por ejemplo VR donde se intentará hacer un DOOM.

Debido a las posibles dificultades que este aumento puede suponer se irá avanzando en la parte del menú paulatinamente según se vaya avanzando en los proyectos originales, véase treedude y typingTest. En un primer momento durante las versiones primeras se usará la fuente por defecto del navegador a la de programar, será una vez alcanzado una serie de mínimos entregables antes de la fecha de entrega que

se barajará la posibilidad de cambiar la fuente por una más similar a la vista en el juego de SUPER HOT.

A pesar de intentar replicar la interfaz de usuario presente en el superhot, se optará por paletas de colores semejantes, más no las mismas que las vistas en ese juego, como ejemplo tenemos el fondo que en vez de ser completamente negro es un azul marino muy apagado.

Todas las desventuras sufridas a lo largo de la realización del proyecto serán anotadas en este mismo documento y les será dado un lavado de cara previo a la entrega de la documentación, márgenes, estilos, limitación de apartados, cambios a un léxico más profesional, evitar repeticiones, introducir un orden estricto cronológico

El proyecto va dirigido a personas y jugadores de todas las edades, aunque seguramente sean los de edades más avanzadas los que apreciarán de mejor manera el aspecto escogido para la implementación de la interfaz del proyecto, otro detalle que tendrá el público destino del proyecto será una escasa sociabilidad y ganas de autosuperación

DESARROLLO

I. PANTALLA DE INICIO

Como cualquier juego se necesitaba una página donde comenzase la acción, se crea el árbol de directorios básicos un index.html, style.css y action.js. Se comienza introduciendo en medio de la pantalla un mensaje con el siguiente texto:

PRESIONE CUALQUIER TECLA

Con el texto ya colocado se añade el color de fondo: un apagado tono de azul cercano al negro. En primer momento se decide iluminar el texto con un sombreado amarillo, pero se decide que queda algo corto, por lo que se decide alumbrar de una manera alternativa y poco convencional. Se duplica el mensaje con las mismas propiedades, pero se le añade una clase adicional. `Lightning`

Con la propiedad de position absolute que hace que no ocupe ningún lugar en el esquema de la pantalla y por lo tanto no empuje al su original y se le da una fuente amarilla en contraposición a la fuente blanca de su original, luego se difuminó el duplicado y se le dio algo más de sustancia mediante hacer la fuente en negrita.

1. Laanimación de lasletras del inicio

Para aportar un efecto sorprende y fresco a la pantalla de inicio se decidió implementar animaciones a las distintas letras del mensaje de inicio, primero fue necesario obtener la longitud del mensaje, una vez que se disponía de dicho dato se creó un array donde serán guardados los distintos intervalos de desaparición de cada letra, para esta funcionalidad se trató hacer uso de una función anónima, pero no pudo usarse.

Llegados a este punto se cambió el enfoque para realizarlo con un bucle que sería reproducido el número de veces especificado por una variable `i` y con cada iteración del bucle se colocará en la posición marcada por otra variable "`j`" que irá aumentando hasta alcanzar el valor de la variable `i`: un número aleatorio entre 1000 y 500.Este será el intervalo con el cual se activará el parpadeo. Al principio ingenuamente se trató de acceder a cada letra del mensaje y cambiarla el color de manera directa… Cosa que era imposible.

Para realizar las animaciones se tuvo que dar una clase individual a cada letra del 0 a la longitud del mensaje y en dicha clase sí. Alterar sus propiedades. Para probar el `setInterval`, primero se probó el llamar a una función con parámetros a modo de prueba ya que no sería la primera vez que funciones similares tienen problemas al llamar a otras funciones y estas poseen parámetros. Como en otros casos

esto resultó en error, tras horas de investigación se dio con la solución, para poder realizar esta tarea sería necesario una `función anónima`.

Desde el `setInterval` llamaría a la función anónima la cual no contaría con parámetros. Significando que el `setInterval` funcionaría a la perfección y la función anónima sería la cual llamaría a la función que necesitaba los parámetros especificados. Fue así como se solucionó el problema

Disponiendo ya de los parámetros en el `setInterval` ahora se necesitaba poder afectar a varios elementos al mismo tiempo con la función. De una no se podía por lo que se tuvo que hacer con un bucle que recorriera los distintos elementos con la misma clase

De primeras se probó con un bucle `forin` de los elementos de la clase especificada, el problema era que ejecutaba el código una vez más de la necesaria, cambiando el enfoque a algo más simple se saca la longitud del resultado de sacar todos los elementos de la clase especificada y se recorre desde el 0 dejando de ejecutar el código al llegar a la longitud especificada.

Una vez se resolvió el tema de afectar a dos o más elementos a la vez tocaba cambiar la propiedad que era necesaria, comenzando por el color de la fuente. Para ello se tuvo que descubrir el formato en el que se guardaba el color, siendo este `rgb(x,y,z)`; Ahora tocaba adaptar el código con lo nuevo cambiado y aplicado teniendo que llamar a las clases individuales una a una.

Primero se tuvo que sacar el tamaño del mensaje sin espacios ya que los mismos no eran contados en la animación. Para sacarlo se añadió al mensaje original, el identificador "`modelo`", la cosa se complicó al usar etiquetas `span` para cada una de las letras la longitud dada por la cadena era mayor a lo que realmente sería, por lo que se tuvo que clonar por tercera vez el mensaje poniendo el 3º en la misma posición, pero oculto y `positionabsolute`, el mismo solo tendría la cadena tal cual, sin `span` ni nada, con un css similar al de la clase `lightning` pero sumándole `visibility`: `hidden` para que no se muestre nunca. Luego se tuvieron que quitar los espacios internos por lo que `trim` no nos valía.

Se encontró la función `replace` con el siguiente parámetro `/\s/g,` ". Con esto ya obtuvimos el tamaño de la cadena sin espacios. El siguiente paso era el de llamar a la función que nos daba los tiempos del intervalo por cada letra. Tocaba un bucle.

Se cambió en el css el formato de los colores para encajar con el de JavaScript. Ahora comenzamos con una condición para diferenciar si se trata del `h1` original o del `h1` sombra.Al estar ahora modificando cada clase y su duplicado en concreto, el modificar el color de fuentes a distinto en función, de si su padre es original u sombra alternará entre dos colores distintos, blanco para el original cuando estaba encendido o amarillo para la sombra cuando estaba encendido y negro ambos cuando estaba

apagado. Sin embargo, en distintas pruebas se vio que quedaba mejor estéticamente si ambos se les daba el color blanco cuando estaban encendidos por lo que finalmente se fue por dicho camino.

2. LA INTERACCIÓN DEL USUARIO

Una vez se dio por concluida la parte visual de la pantalla de inicio se procedió a la interacción. Se comenzó creando un directorio con los ficheros index.html, style.css y action.js en el directorio menú. La tarea de ese momento consistirá en mediante js al pulsar cualquier tecla se envíe al usuario a la página siguiente. Primero se agregó el `event listener` al body. A continuación, tocaría averiguar cómo acceder a la nueva página usando javascript, con la siguiente línea se cambia la ruta de ejecución: Y con esto se dio por finalizada la versión 1 de la pantalla de inicio.

II. COMENZANDO CON EL MENÚ

Comenzando con el menú se establece el mismo color de fondo que la pantalla de inicio y la página fue dividida en 3 divs: uno padre y 2 hijos dependientes del 1º. La idea surgida fue la de no poder navegar con el ratón en esta pantalla y obligar al usuario a usar el teclado para navegar entre las distintas opciones.De manera temporal al `div` padre se le da un width del 70% y una altura de 97vh. El menú que trataremos de replicar en la V1 será el inglés original, comenzando con la siguiente pantalla

Ilustración 1 Vista del modelo a replicar

La parte superior: LEVELS, ENDLESS, CHALLENGES, README, REPLAYS, RECRUIT, CREDITS y more en un principio serán eliminadas, mientras que la parte de quit.exe, será sustituida por /.., la ubicación marcada abajo será C:\MENU\, siendo la pantalla anterior la de inicio la supuesta C:\ . Como primer

intento se dispone el método de montaje de la página a grid, dividiéndolo en 3 columnas iguales.

Al primer div se le da un `grid -column: 1 / span 2`, al segundo no se le especifica nada quedándose en la posición 3, con un `span` de 1. Y por último al 3º le damos un `grid-column: 4 / span 4`. Para diferenciarlos se les da fondo: azul al primer div (el de más a la izquierda), transparente al 2º y rojo al 3º.

Para poner la parte de abajo se decide reducir la altura del `div padre` a 90vh y generar otro `div` de la misma anchura, pero de 7vh sin márgenes justo debajo que será el que lleve la parte de la ruta. Para quitar y aplicar los nulos márgenes arriba y abajo, pero mantenerlos a la izda y la derecha se modifica el `margin a: 0 auto`; aplicando 0 márgenes arriba y abajo, pero los automáticos a la derecha y la izquierda y así se coloque en medio. Para poder visualizar temporalmente el nuevo `div` le se le da el fondo white.

Se le aplica el borde blanco igual que a los demás y se continúa. Se quitan los colores de fondo y se les da a los hijos del container mediante `#container > div`. El borde blanco presente en los padres para delimitarlos. Primero se rellenarán los `div` estáticos de este menú siendo los dos de la izquierda y el de abajo del todo.

El primero que ponemos el /.., por el momento se creará en una etiqueta p con clase: "`goBack`". En el `div` de en medio como en el juego introducimos lo siguiente en otra etiqueta p: <UP-FOL>, se le dará la misma clase que el `p` del primer `div`, pero para que no confunda los símbolos de menorque y mayorque con una etiqueta se tendrán que introducir mediante su código: `&lt` y `&gt`, como queda muy pegado se le aplica a la etiqueta `p` en general temporalmente un `margin-left` de `10px;`

A continuación, solo quedarían los directorios: SETTINGS, APPS, DEMOS, CELLULAR, WIRES, GAMES, VR, ART y VIDEOS. Junto a sus respectivos >FOLDER< en el 2º div, cada una con su clase go<`Directorio`> Luego se introdujo el C:\ en el div inferior. Lo siguiente que se tuvo que hacer fue introducir los elementos posicionados de forma absoluta en las esquinas situadas a la derecha la de arriba con la hora y la de abajo con el "sistema operativo" comenzamos con la sencilla, la de abajo.

Se creó una clase general para posicionar elementos de manera absoluta "abs", y para posicionar el ID SO y el background color el mismo que el del body para ocultar la línea del borde que ocupa y que quede igual que en el menú original. Para la hora se creó un segundo p con clase `abs` e id `time` que se colocó en la parte de arriba y en la misma posición de izquierda/ derecha que el p de SO. Una vez se terminó de colocar los elementos estéticamente llegaba la hora de darle funcionalidad comenzando por la hora.

1. Haciendo un reloj

Primero será necesaria una constante con la fecha y horas del día en el que se esté ejecutando el programa. Esto se obtiene con const <nombre variable> = new Date(), que por defecto sin parámetros añadidos coge la fecha del sistema.

Para aumentar la limpieza del código la función del reloj será alojada en un subdirectorio aparte con el script time, se obtiene la hora y los minutos mediante la propiedad getHours() y getMinutes(), luego para darle un formato más arreglado se quiere que cuando los minutos sean de un solo dígito, le añada un 0 adicional a la parte delantera del número, esto se hará mediante la función addCero(n) la cual recibirá los minutos y en caso de que sea < 10, le añadirá como una cadena un 0 delante. Luego la cadena restante será devuelta para ser utilizada donde se la llame

Desde el script de actions se crea la función de update que será activada en un intervalo de 1000 milisegundos la cual igualará el valor del elemento con ID time el valor devuelto por nuestra función timePasses, es decir, la hora concatenada con los minutos.

2. Nuestro puntero para seleccionar opciones

El siguiente apartado que fue atacado fue el de la selección, y para ello se creó otra clase en el css llamada selected, la cual mediante jQuery añadiremos o quitaremos según este seleccionada esa opción o no.

Por defecto al iniciar la página la opción seleccionada será la que esté situada la 1ª pero en versiones posteriores la opción seleccionada será aquella en la de la que anteriormente volviste, es decir, si se accede al directorio de APPS, pero luego vuelves, te debería seleccionar por defecto a APPS.

Mediante addClass(.<clase>) se agrega la clase de selected a los elementos, pero al ver el resultado se decide cambiar los elementos absolutos por la etiqueta span, lo cual obliga a modificar sus posicionamientos. Una vez recolocados se ajusta el margin-bottom de los p para que la selección se vea igual que en el producto original, véase pegados

De manera temporal en actions.js se comienza a crear dos arrays, uno con cada nombre de las distintas clases seleccionables y otro con si está seleccionado o no. Ambos serán metidos dentro de un 3er array "globalSelector" y mediante un bucle de sus elementos donde se juntarán, el de clases y el de selección en el array bidimensional globalSelector en la posición de i.

2.1 OptionUP

Una vez ya fue creado el array global de modelo se prepararon las funciones de avanzar una opción hacia arriba o hacia abajo que serán usadas cuando se pulse una tecla arriba o abajo. Comenzando con la función de avanzar hacia arriba lo primero será saber cuál es la opción que está activada véase la casillaglobalSelector[i][1] == true. Una vez se obtiene la opción activada deberemos desactivarla, esto lo se hizo mediante el removeClass

Si la opción activada era la 1ª se situará en la última opción, de lo contrario se restará a la posición activada 1. Ya se sabe la posición nueva que deberá ser activada pero aún tenemos el modelo desactualizado, eso habrá que solucionarlo con un bucle.

Recorremos el array global desde 0 a la longitud del mini array focus, si i==j entonces se igualará a true el modelo, de lo contrario se igualará a falso la posición [j][1]. Por último, usando el modelo se vuelve a buscar el valor activado y mediante jquery se le añade la clase selected a los elementos con la clase de la posición [i][0], nos llevamos las funciones aldirectorio de additionalScripts y creamos el fichero Iselectores, desde el action llamaremos a optionUp y se le deberá pasar el array de las clases del fichero.

Ya finalizadas las funciones, llamamos a la función cuando se pulse la tecla de arriba mediante event listener al body. Al solo pasar las clases solo se efectuará la función una vez por lo que pasamos las clases y sus valores de verdadero o falso. No se consigue el resultado deseado por lo que deberemos hacer el array global en actions.js y mandarlo como parámetro junto clases. Una vez hechos los cambios debidos se consigue la acción deseada

2.2 OptionDown

Ahora tocaría con el código usado para optionUp hacer el código de optionDown, a tener en cuenta sobre los cambios necesarios estaría el en vez de reducir el valor de la posición del seleccionado se tendrá que incrementar, y cuando el valor sea 1 menos que la longitud del array (véase el último del array), se igualará a -1 para luego aumentarlo en 1 como a todos los demás y que vaya al primer elemento del array usado de modelo para las opciones.

Una vez terminado el código añadimos en el if del eventlistener del teclado un else if si la tecla pulsada es la flecha hacia abajo. Lo siguiente de lo que se ocupó el equipo fue el actualizar el tercer div, desgraciadamente en el momento en el que se pusieron a ello no tenía el material original a mano, por lo que de manera temporal se dispusieron a actualizarlo de una manera básica: para los diferentes directorios:

```
directory:<nombre de directorio> (tabulación) |>FOLDER<

Y para "/.." directory: GO UP
```

Para esto se tuvo de aumentar el array bidimensional a Tetradimensional, nombre de la clase, estado, tipo de "archivo": si es /.., fichero o archivo, para esta primera pantalla, si es /.. o fichero, en caso de ser fichero, estaría el último campo, nombre del fichero o directorio.

En los eventkey añadiremos un igualador del interior del html, debajo del valor de globalSelector actualizado. Con el valor devuelto por la función Update3(tetracadena,clases.length). Aunque previo en la 1ª ejecución desde action se igualará el valor interno de html del 3er div por el valor de /.. que es por el momento el valor por defecto al iniciar la página. Esto se realizó mediante jquery

```
     <variable>= $(#summary);

<variable>.html("<span>directory: GO UP</span>");
```

Se le aplica un padding al div summary (el 3º) para que no quede tan junto y se ajusta #SO para que quede algo mejor colocado. Ya que el actualizar el div 3º también tiene que ver con la selección también se incluye la función de Update3 en Iselectores.

Primero Update3 tiene que saber cuál es elemento seleccionado, por lo que se invoca a la función previamente hecha que coincidentemente cuenta con los mismos parámetros, el valor que devuelve lo guardamos en una variable. Si el campo elegido tiene el valor /.. en el tipo devolverá:

```
     “<span>directory: GO UP</span>”
```

Y cómo de momento no se tienen ficheros y solo directorios solo se tomará como otra posibilidad que sea un directorio de forma temporal else devolverá:

```
     "<span>directory:" + cadenaTetradimensional[elegido][3] +

" |FOLDER</span>";
```

Al no estar del todo de acuerdo como se termina representando se crean dos clases adicionales flleft y flright con las propiedades de float left y right respectivamentea continuación se modifica el código devuelto en caso de no ser /..

```
      "<span class='flleft'>directory:" +

cadenaTetramensional[n][3] + "</span><span

class='flright'>|FOLDER</span>"
```

Una vez ya se tiene el código aplicado en la tecla de pulsación arriba lo llamamos también al pulsar abajo. Y con esto se comienza la codificación de la pulsación de la tecla enter. Se creará una función más en Iselectores navegation(cadenaTetradimensional, longitud de clases)

Comenzando como la anterior función lo primero es averiguar cuál es la opición seleccionada, para eso se usa findTrue, junto los mismos parámetros que recibió la función. Si el campo elegido tiene el valor /.. en el tipo devolverá:

```
      ".."
```

de lo contrario devolverá cadenaTetramensional[n][3]. Esto será completado en action que estará esperando el resultado de navegation para concatenar al final del valor

```
      "/index.html"
```

Ahora antes de proseguir con las pruebas la situación obliga a crear la primera tanda de árbol de directorios todos ellos respetando la estructura index.html, style.css y action.js. Se procede a mejorar la interfaz del menú mediante el uso del flex. Se continúa yendo directamente al directorio importante GAMES.

III. CREANDO EL MENÚ DE GAMES

Se empieza fusilando la base del menú para adaptarlo al menú de GAMES, aquí se encuentran los elementos FILES, por lo que se actualiza la función de Iselectores Update3, haciendo ahora la distinción entre es /.., directorio u otra cosa, en este caso ficheros/archivos. En el último caso temporalmente solo devolverá:

```
       "<span class='flleft'>app:" + cadenaTetramensional[n][3]

+ "</span><span class='flright'>|--FILE->#</span>".
```

En el producto original el "#" era un código numérico, pero al en este punto no tener identificadores para los archivos lo dejaremos como una incógnita y a tener en cuenta para futuras versiones.

En el modelo original se tenía a treedude.exe y shrl.exe, en nuestra versión shrl.exe será sustituido por nuestro "typingtest.exe", se redirigen las rutas de los scripts y para finalizar, se actualiza la capa de route a la ubicación de MENU\GAMES. Y al igual que se hizo con el html fusilamos también el css y js para ser usado de base para el código js, el css es copiado tal cual

Comenzando ahora sí con los cambios se modifica el array de clases, ahora solo teniendo 3 elementos: goBack, goTreeDude y goTypingTest, debido a que focus es construido dependiendo de clases, el mismo ya se ve modificado automáticamente.

Otros que sufren modificaciones son los arrays de tipo y nombre tipo ahora solo teniendo 3 elementos /.., archivo y archivo y nombre solo teniendo: "","treeDude" y "typingTest" todo lo demás se queda igual. Se prepara el árbol de directorios de games y se comienzan las pruebas.

IV. DESARROLLO DE POR FIN, TREEDUDE.”EXE”

1. Los sprites del leñador

Se descubre en GitHub que alguien hizo el minijuego de treeDude en C, es usado como referencia de momento para realizar los sprites. Se crea en aditionalScripts draw.js que será quien tenga las funciones de dibujar los sprites.

Para hacer las funciones que pasarán las cadenas con los dibujos se hace uso de las funciones con flecha sin parámetros al ser html los espacios serán sustituidos   y debajo del código mediante comentarios se muestra el resultado final

Primero se realizan los sprites del talador quieto tanto posicionado a la izquierda como la derecha mediante un array constante que se devuelve en una función seguimos haciendo el sprite del talador cortando por la izquierda, este se realiza usando de base el talador quieto situado a la derecha ya que es el que más se asemeja

Una vez se terminó el sprite tocaba alinearlo y colocarlo un poco más arriba que el quieto y hacer la sensación del saltito del juego original. Para la sensación de salto simplemente se agregó una fila adicional vacía y así dar la impresión de que está más alto.

Primero para alinearlo se prueba poniendo más espacios en lado izquierdo, no funciona y de poner más espacios en la derecha se aleja más, por lo que se decide

hacer más anchos los sprites del talador quieto, tampoco funciona, se vuelve a la idea original de en el cutingleft añadir más espacios, solo que colocar el muñeco en el flex-strat, esta acción da resultado. Se prosigue con el fellercutingRight se coge como base el iddleLeft y se añade la celda extra para la sensación de salto.

Tras añadir el suficiente nº de espacios en el iddle right se consigue que apesar de estar colocadas en el justify-content: center, que ambos sprites estén alineados. Se reintenta con el de la izquierda hasta conseguir el resultado deseado.

1.1 Diseñando el escenario

Antes de continuar con los sprites se procede a diseñar la ventana donde se mostrará todo. Para hacerlo se coge el código de container y "supahContainer" de páginas anteriores. Luego para visualizarlo le damos un borde visible a container. Una vez ya tenemos el marco comenzamos a colocar los elementos de posición

Primero se coloca el "containerDude" quien lleva la posición del leñador, lo colocamos abajo del todo en medio, con position absolute para que no ocupe un espacio y ese mismo pueda ser ocupado simultáneamente por otro. Ahora toca colocar el "containerTree", una vez tenemos una colocación temporal se continúa con los sprites

Vista:

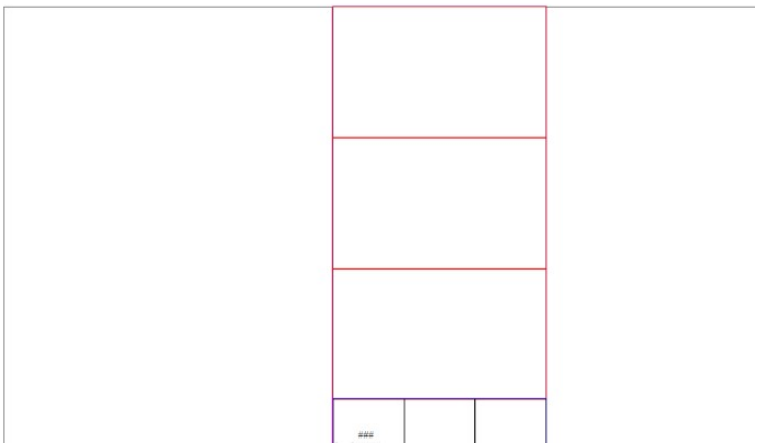


Ilustración 2 Vista de containerDude y containerTree

En un principio el tronco se realizará sin adornos, siendo más similar a una tubería, en futuras versiones es muy probable que se mejore. Una vez se coloca el primer tronco y queda ajustado, dividimos las celdas superiores también y se coloca un tronco en todas las celdas del medio

Vista:

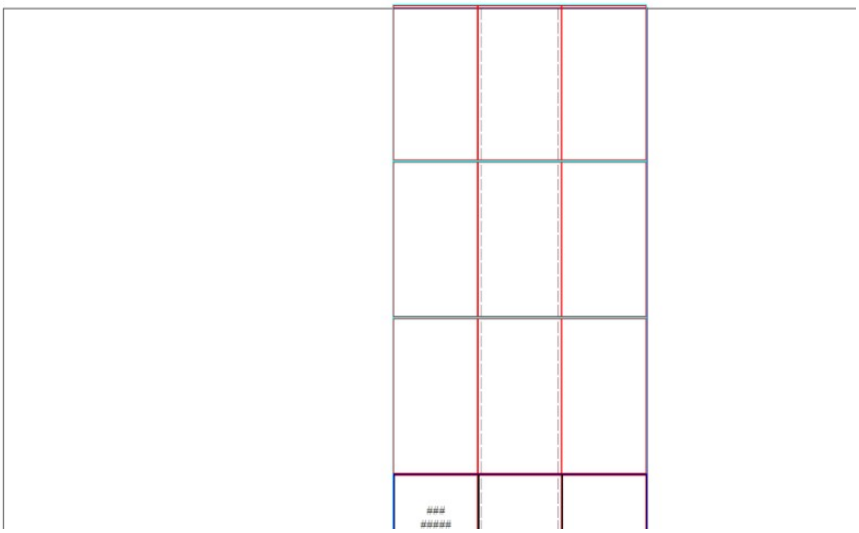


Ilustración 3 Vista de containerTree con el propio árbol

Se prosigue con la creación de los sprites de las ramas y se prosigue con la animación del movimiento del leñador para que pase del de cutting a iddle cuando suelte la tecla.

2. La generación aleatoria

Ya está terminada la animación del leñador hecha ahora toca generar la "copa" del árbol aleatoriamente, véase la celda más superior. Primero generamos dos nº aleatorios entre 0 y 1. Si el primero sale 1 no podemos permitir que el segundo también salga 1 por lo que manualmente lo ponemos a cero.

Ahora se crea otro array este unidimensional de dos elementos para saber dónde está nuestro leñador. Si la última tecla que se pulsó fue la izda el primer campo será 1 y el 2º será 0, si fue la derecha el 1º será 0 y el 2º 1. Como por defecto el leñador siempre comienza en la izda. Creamos el array directamente con 1 y 0.

Ahora toca representar lo indicado por el modelo en el árbol primero debemos de recorrer el array bidimensional y tener en cuenta de si estamos en la celda de la izquierda o la derecha, así como si tiene o una rama en el modelo. Mediante la concatenación de los valores de J e I + las etiquetas del padre e hijo se puede seleccionar la celda correspondiente.

El siguiente paso que realizamos es llevar la función de pintarArbol a el fichero de draw.js. La de generarnuevo0 al no tener relación con la ilustración de modelos, la dejamos dentro de action.js, aunque cuando vayamos teniendo más funciones de mecánicas se terminará creando un fichero extra donde guardar dichas funciones aparte. De momento toca agregar la función de talar el modelo para luego replicarlo en

la ilustración, para nuestra función talar le pasaremos las dos cadenas la del árbol y la de la posición del talador.

Primero, se recorre el array del árbol desde la longitud de árbol -1 al 0 pasando el valor a de [i] a [i-1] tras ello nos disponemos a hacer uso de generarNuevo0 para sustituir el primer valor, pero nos da error, teniendo que revisar cómo solucionarlo.

Se configura para que no se le pase ningún parámetro pero que devuelva un array de dos elementos el cual será guardado en la posición del array del árbol[0] que sería lo mismo que la posición [0] del array devuelto se guarde en el árbol[0][0] y la posición [1] en el [1][1].

Tras aplicar estos cambios al primer modelo igualamos el valor de su posición [0] a generarNuevo0. Después se llama a la función de pintar árbol para que teniendo en cuenta la cadena bidimensional pinte el árbol actualizado.

3. UNNUS ANNUS MEMENTO MORI

Lo siguiente que se realizará será comprobar cuando el leñador y una rama "colisionan". Como se sabe es el último bloque de arrays el que contiene las ramas que están a la altura del leñador.

Por lo que con un "if" se revisa si el array del modelo del árbol[longitud array][n] y si la posición del leñador[n] en ambos se encuentra un 1 simultáneamente de ser así de momento, que nos avise por la consola, donde N será la posición del array a comprobar

Una vez conseguimos detectar correctamente las colisiones toca crear desde cero al no tenerlo en proyecto de GitHub el sprite de la tumba. Una vez hecha ya la tumba, tenemos que colocarla o en la posición en la que estaba el leñador o la rama, para facilitarnos las cosas se colocará la tumba en la posición del leñador y se borrará la rama.

Primero usando de base el código usado para dibujar al leñador en la izquierda o la derecha es usado para pintar la tumba en función de si el modelo de la posición del leñador estaba situado a la izquierda o la derecha.

Y luego se sobrescribe el valor 1 de la posición[i] a 0,0 y volvemos a invocar la función de pintar el árbol para que borre la rama sobrante. Lo siguiente es quitar todos los eventListener existentes por el momento siendo estos pulsar tecla izquierda, pulsar tecla derecha y soltar las respectivas teclas.

Al hacerlo con JavaScript mediante el removeEventlisterner no se consigue el resultado deseado por lo que se prueba mediante jQuery el método.unbind en una función. Al crear el eventListener con JavaScript y quitarlo con jQuery no parece funcionar. Se intenta realizar el eventListener también desde jQuery

Una vez lo cambiamos funciona a la perfección, lo siguiente que es realizado es dar unas proporciones fijas y mínimas a la ventana del juego y las propiedades del div para que al cambiar el tamaño de la ventana no se destroce la estructura del juego

4. Retomando el proyecto

Dichas dimensiones no parecen funcionar del todo, por lo que se dejan a un lado temporalmente, pasamos a la creación de puntos, para empezar creamos dos divs adicionales con posicionamiento absoluto, uno que llevará la cantidad de vida/energía restante, que irá descendiendo según vaya pasando el tiempo, mientras que el 2º tendrá los puntos introducidos en una p, la manera en la que se irá actualizando la puntuación será cada vez que el jugador tale el árbol, si por dicha pulsación el jugador acabó colisionando con una rama se le bajará un punto ya que de lo contrario sumaría a pesar de haber perdido:

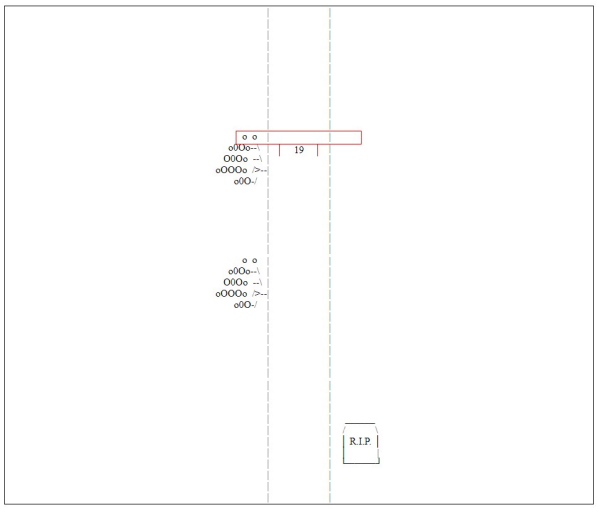


Ilustración 4: Vista del no aumento de puntuación

5. Colocando los textos

Los textos que vamos a colocar serán los siguientes, colocado en la parte superior izquierda: TREE DUDE TREE DUDE DUDE by piotr y la parte inferior derecha con una fuente más apagada se colocará: press ESC to quit.

Ilustración 5: modelo de lo que se quiere conseguir

Se crean dos etiquetas `span` con clase `abs` (para colocarlas de manera absoluta) con los ids `presentation` and `hint`, de manera temporal se les dan las propiedades de los antiguos `time` y `SO`, al tener posiciones similares. Se consigue el que se muevan al mismo son que el recuadro al poner el estilo de `container` su `position` a `relative`. Se colocan `presentation` y `hint` en `-1vh`, en `top` y `bottom` respectivamente, así como `3vwleft` y `right` respectivamente.

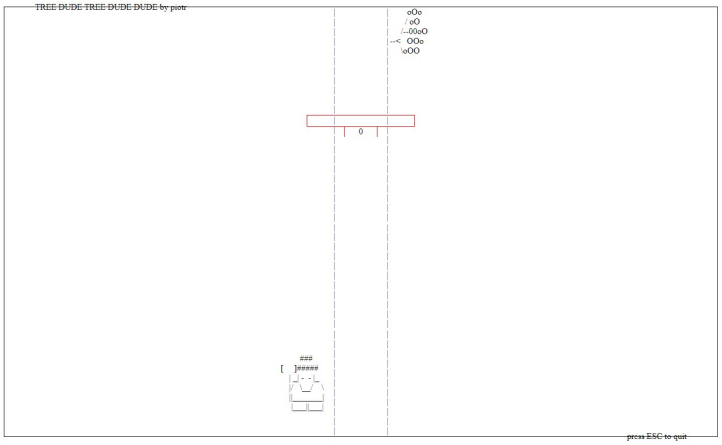


Ilustración 6: resultado final

6. Salida y preparamiento de la vida

Para programar y habilitar la habilidad del usuario de poder salir cuando quiera del juego, se acción a la parte del código donde se detectaba que una tecla se había pulsado y se añadió la condición de SI se ha pulsado la tecla de esc mediante el siguiente código

```
else if (tecla == "Escape")
```

Una vez se haya detectado dicha pulsación, se hizo uso de la función de `window.location.replace`junto el direccionamiento relativo para acceder al

anterior index: `("../index.html");` Luego dentro del `containerlife` se crearon otros `divs: lifeBG (life background)` y `Contenedorlife`, ambos se les otorga el position `absolute`, `flex-start`, un `width` de `98%, 2px`.

7. Aplicando el esquema de colores y la estética de la página

Comenzando con el color de fondo del body y el color de la fuente, aplicamos ambos el color de fondo oscuro y el color blanco de fuente. Gracias a Internet, Reddit y dafont obtenemos la fuente de Pios del juego pudiendo hacer aún más similar nuestra aplicación, nos descargamos la fuente y la ubicamos en una nueva carpeta de ASSETS, ahora en el css para usar la fuente nueva escribimos las siguientes líneas.

```
@font-face {
    font-family:<nombre que recibirá la fuente>;
    src: url(<ruta de la fuente>);
}
```

Al aplicar la nueva fuente se descubre que por lo menos temporalmente, quizá no sea buena idea aplicarla en todos elementos de todas las ventanas

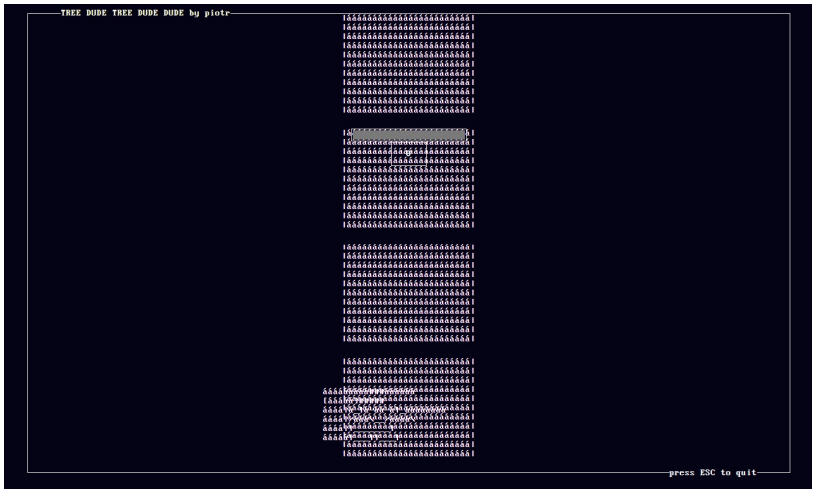


Ilustración 7: vista del error al aplicar la fuente

Tras ver el error somos obligados a solo introducir la fuente del super hot en los elementos que no sean estéticos, siendo aplicados solo en los textos y puntuación, luego para mejorar la exactitud con la que se mostrará la página a pesar de cambiar de entorno aplicamos el css de nomalize.css

8. Mejorando el tronco

Lo que detiene el minijuego de ser más responsive es el tronco actual, el cual actualmente está formado por un número X de “|”, obligando a que siempre cuente con las mismas dimensiones independientemente de la pantalla. Primero se comenzó comentando todas las líneas de código que se usaron para dibujar tronco sin ramas, para ver cómo quedaba el resultado final sin el tronco de caracteres.

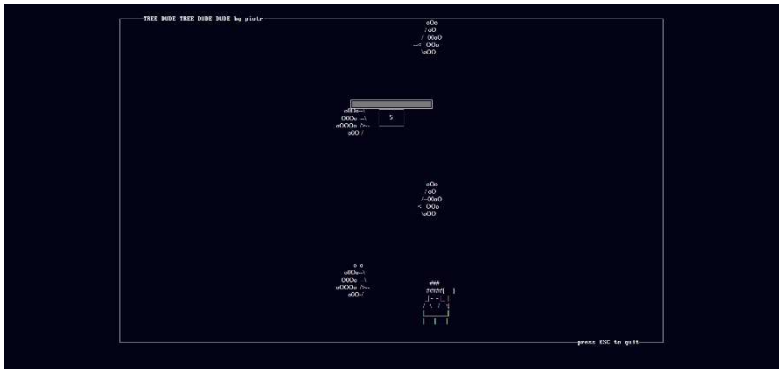


Ilustración 8: Juego sin tronco

Cómo se puede observar en la imagen la colocación de las ramas por los menos sigue siendo correcta, lo siguiente que se hizo fue devolverle temporalmente el color de los bordes del containerTree y sus "hijos" para poder ver como estaba segmentado y poder tomar una ruta de acción. Con todas las divisiones por fin visibles nos pusimos a limpiar:

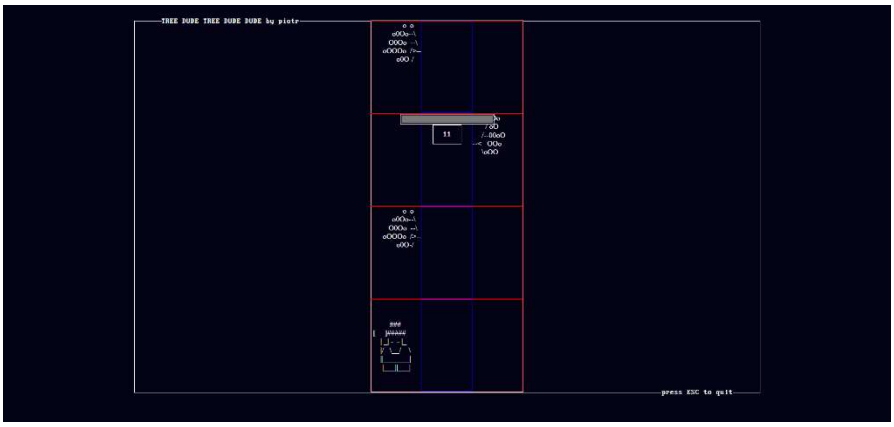


Ilustración 9: los bordes del containerTree, sus "hijos" y "nietos"
**Leyenda: borde blanco (no apreciable) = bordes del propio containerTree*
borde rojo = bordes de los div “hijos” del containerTree
borde azul = bordes de los div “nietos” del containerTree

Se comenzó quitando los bordes del containerTree y sus "hijos", dejando solo los de los "nietos", después para filtrar más el resultado solo dejamos los "nietos" con

la clase `col1`, siendo estos solo los de la columna de en medio. Se pudo hacer referencia a los mismos mediante el siguiente código:

```
#containerTree > div > .col1 {
   border:1px solid blue;
}
```

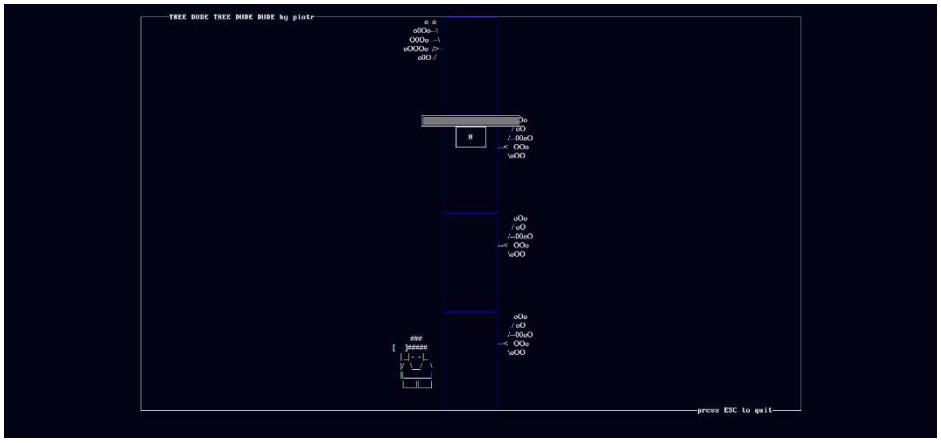


Ilustración 10: Bordes de los nietos col1

Para finalizar el nuevo tronco ajustable se estableció el `border-style` a `none` y `dashed` para no tener los bordes superiores e inferiores y tener los bordes izquierdo y derecho con un patrón de rayas, el color lo estableceremos al color del leñador, letras, etc... Véase blanco, además se le dotó de un ancho de `2px` para diferenciar el tronco del leñador. Y para que las ramas no desentonasen con el tronco se las dio un `font-sizelarge` a través de los `div`*hijos* del containerTree.

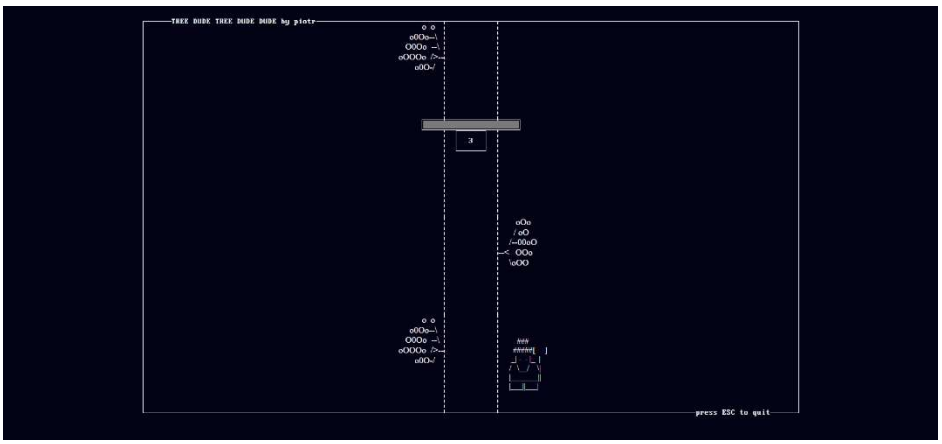


Ilustración 11: Nuestro tronco responsive

9. Do it again!

Hasta este momento la única manera de volver a jugar es recargando la página usando el ratón o pulsando F5, en contraste el recurso original al morir que salía un mensaje parpadeante a la altura del leñador "[PRESS ENTER TO RESTART!]" sobre la mitad (horizontal) de la pantalla. La posición más cercana es el div b, ahí se creó un **P**árrafo con clase abs (absoluta).

Para el parpadeo volveremos a nuestro action.js original, donde ya surgió la necesidad de crear un objeto que parpadease, así fue como se usó la ya existente función "`intervalos`" como base para hacer una segunda función de intervalos y la misma ser colocada en aditionalScripts. En el index de treedude se agregó la línea para obtener el js de intervalos antes de action.js y así se pudiera hacer referencia a la misma desde actions.js

Al **P**árrafo se le dio el id `retryHint` y así desde su css se igualó el `display` de `retryHint` a `none`. Tras la modificación anterior se comenzó a adaptar la nueva función de parpadeo para empezar se configuró para que hiciese referencia a un identificador en vez de una clase y se suprimió el bucle al ser ahora `marcador` un único elemento. La condición pasó de usar como referencia el color al `display`, si el `display` era`none`, entonces se cambiará a `block`, de lo contrario se invertirá el valor.

En el action de treedude se creó la variable `ded` con valor `false` predeterminado, ahora donde se chequeaba una tecla pulsada ahora tendremos en cuenta también el caso en el que se pulse el `Enter`, para esta versión solamente haremos una recarga de la página automática solamente si la variable ded está true, de lo contrario no ocurrirá nada.

Lo siguiente que se hizo fue modificar la función de talar para: cuando el jugador muriese se habilitase el parpadeo del texto y se asignase el valor `true` a la variable ded, tras dar el valor de true a ded se llamaría a intervalos pasándole como parámetros el id del **p** que parpadeará y el tiempo de 2000 entre cada animación milisegundos (2 segundos)

Debido a este nuevo ajuste nos vemos obligados a comentar la función de `removeEventlistener` y su respectiva llamada, para que el leñador no pueda moverse muerto, pondremos la condición para moverse que además que se pulse la tecla correspondiente ded tenga el valor false y de esa manera el usuario podrá regresar cuando quiera, reintentar cuando este muerto y moverse cuando esté vivo

V. CREANDO SONIDO

De lo siguiente que nos encargaríamos sería de la música y efectos de sonido, preparamos dos ficheros js nuevos en `aditionalScripts` sounds.js y songs.js. Comenzaremos con sounds.js a hacer las pruebas y luego será donde serán programados los distintos efectos de sonido, una vez conseguido los efectos de sonido necesarios, se programará la música

Por el momento en el index de treedude lo enlazamos con el aditional scripts de sounds.js. Nuestra primera prueba fue que al pulsar la tecla s/S sonase un pitido, al volver a pulsar la tecla s/S dicho pitido tendría que parar de sonar. En esta versión del proyecto se creó temporalmente una variable: `soundStatus`, dicha variable se inicializaría en `false` en el archivo de action.js (el mismo archivo que es donde se reciben las pulsaciones del jugador), si la variable de `soundStatus`está en `false` se llamará a `demoPitido1ON`(), cuando este en true se llamará `demoPitido1OFF`() ambas funciones ubicadas en sounds.js.

En sounds.js tendremos la variable contexto inicializada con el valor `new AudioContext()` y las variables `oscilador` y `volumen` ambas inicializadas con valor null, ahora en `demoPitido1ON()` como siguiente paso igualamos los valores de `oscilador` y`volumen` con los valores `contexto.createOscillator()` y `contexto.createGain()` respectivamente, luego se agregarán las siguientes líneas

```
oscilador.connect(volumen);
volumen.connect(context.destination);
oscilador.start(0);
```

Mientras `demoPitido1OFF`(), para parar el sonido primero probaremos con la línea que ponen en la página:

```
volumen.gain.exponentialRampToValueAtTime(
  0.00001, contexto.currentTime + 0.04
)
```

Tras realizar las comprobaciones vemos que no se reproduce el sonido, revisando la ejecución del código parece que el navegador no permite la reproducción de sonido de forma "automática" a no ser que sea llamada por el click del usuario y esto parece suceder con todos los navegadores, para poder activar la reproducción del sonido tendremos que o entrar en la configuración del navegador en el que se ejecute la página o buscar la manera de "engañar" al navegador para que piense que el usuario ha hecho click, en este momento queremos que al pulsar una tecla se reciba la pulsación, pero después también se intentará reproducir el sonido sin que el usuario pulse nada, para la música.

Gracias a stackoverflow y la propiedad click() de javascript podemos emular dicha pulsación sobre un elemento, y así por fin conseguimos que funcione el sonido. Como siguiente acción implementaremos las variables filtro y frecuencia. Filtro en principio solo hemos encontrado como posibles valores los siguientes:

- `Sine`
- `Square`
- `Triangle`
- `Sawtooth`

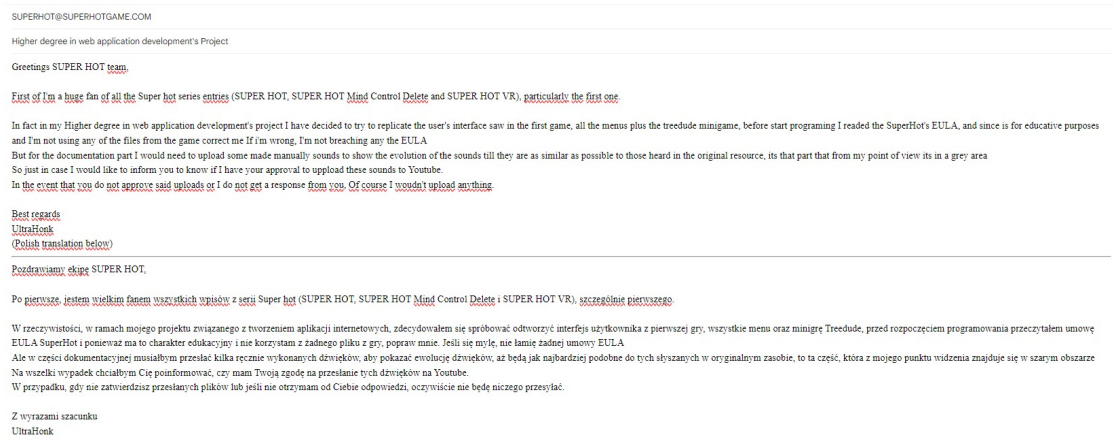
Mientras que la frecuencia simplemente es el tono que se reproducirá. De momentos se crean las variables filtro y frecuencia con valor null. Luego para facilitar futuras pruebas creamos las funciones: `setFiltre(tipo)` y `setFrequency(note)`:En `setFiltre` chequearemos el valor que es pasado a la función haciendo distinción entre `square`, `triangle` o `sawtooth` (¡IMPORTANTE! EL TEXTO DEBE ESTAR ESCRITO TAL CUAL EN MINÚSCULAS), en principio si recibe cualquier otra cosa se tomará como que si se introdujo Sine para aplicar el filtro primero igualaremos nuestra variable filtro al valor pasado para luego aplicar al oscilador el filtro mediante `oscilador.type`

Después se procedería a llamar la función de `setFiltre` justo antes de hacer el `oscilador.connect(volumen)`, para comprobar el éxito primero se grabó el pitido sin incluir `filtro` ni `frecuencia`, para después probar aplicar el filtro de `square` (el por defecto es el `sine`) y una frecuencia de `261.6` (el por defecto es alrededor de `440.0`).

Como última prueba se procedió a hacer sonar dos sonidos distintos al pulsar la tecla s, un Sine continuo de frecuencia `261.6` y un `square` que sonará un segundo y luego se apagará cada 5 segundos hasta que el usuario vuelva a pulsar la tecla que entonces ambos sonidos deberán apagarse. Al contrario que con un intervalo constante si queremos detener a uno tendremos que crearlo dentro de una variable, la prueba termino en un éxito. Dichas pruebas se podrán encontrar como mínimo en la presentación.

1. Anexando

Al no poder poner archivos de sonido anexos en la documentación nos vemos forzados a recurrir a Youtube, creando un canal donde subir los anexos de sonidos y pruebas, pero antes de eso necesitaremos que los dueños originales del juego véase (SUPER HOT team) es por eso que les notificaremos vía email a su correo de contacto ubicado en su página web



SUPERHOT@SUPERHOTGAME.COM

Higher degree in web application development's Project

Greetings SUPER HOT team.

First of I'm a huge fan of all the Super hot series entries (SUPER HOT, SUPER HOT Mind Control Delete and SUPER HOT VR), particularly the first one.

In fact in my Higher degree in web application development's project I have decided to try to replicate the user's interface saw in the first game, all the menus plus the treedude minigame, before start programing I readed the SuperHot's EULA, and since is for educative purposes and I'm not using any of the files from the game correct me If i'm wrong, I'm not breaching any the EULA
But for the documentation part I would need to upload some made manually sounds to show the evolution of the sounds till they are as similar as possible to those heard in the original resource, its that part that from my point of view its in a grey area
So just in case I would like to inform you to know if I have your approval to uppload these sounds to Youtube.
In the event that you do not approve said uploads or I do not get a response from you. Of course I wouldn't upload anything.

Best regards
UltraHonk
(Polish translation below)

Pozdrawiamy ekipę SUPER HOT,

Po pierwsze, jestem wielkim fanem wszystkich wpisów z serii Super hot (SUPER HOT, SUPER HOT Mind Control Delete i SUPER HOT VR), szczególnie pierwszego.

W rzeczywistości, w ramach mojego projektu związanego z tworzeniem aplikacji internetowych, zdecydowałem się spróbować odtworzyć interfejs użytkownika z pierwszej gry, wszystkie menu oraz minigrę Treedude, przed rozpoczęciem programowania przeczytałem umowę EULA SuperHot i ponieważ ma to charakter edukacyjny i nie korzystam z żadnego pliku z gry, popraw mnie. Jeśli się mylę, nie łamię żadnej umowy EULA
Ale w części dokumentacyjnej musiałbym przesłać kilka ręcznie wykonanych dźwięków, aby pokazać ewolucję dźwięków, aż będą jak najbardziej podobne do tych słyszanych w oryginalnym zasobie, to ta część, która z mojego punktu widzenia znajduje się w szarym obszarze
Na wszelki wypadek chciałbym Cię poinformować, czy mam Twoją zgodę na przesłanie tych dźwięków na Youtube.
W przypadku, gdy nie zatwierdzisz przesłanych plików lub jeśli nie otrzymam od Ciebie odpowiedzi, oczywiście nie będę niczego przesyłać.

Z wyrazami szacunku
UltraHonk

Ilustración 12: Imagen del correo enviado al equipo de SUPER HOT

2. Mejorando los setters:

Se mejoran los setters de filtro y frecuencia para que funcionen independiente del oscilador que se pase agregando un parámetro para el mismo

```
function setFiltre(o,tipo) {
    if (tipo == "square") {
        filtro = tipo;
    }
    else if (tipo == "triangle") {
        filtro = tipo;
    }
    else if (tipo == "sawtooth") {
        filtro = tipo;
    }
    else {
        filtro = "sine";
    }
    o.type = filtro;
    //oscilador.type = filtro;
}
function setFrequency(o,note) {
    frecuencia = note;
    o.frequency.value = frecuencia;
}
```

3. Creando la clase Sonido:

Para aumentar la velocidad de las pruebas se procede a crear la clase sonido a la que se le pasará por el momento `frecuencia` y `filtro` (este último previo a comprobación que sea correcta) al constructor. A la hora de crear la clase sonido, nos vemos obligados a convertir las variables en propiedades y las funciones en métodos.

Lo cual nos obliga entre otras cosas a agregar “this.” Previo al uso de cualquiera de las propiedades para que los métodos funcionen correctamente.

Tras conseguir crear la clase se descubre que se podía usar una misma instancia para reproducir múltiples sonidos simultáneamente… No dejando que esto acabe con la moral del equipo se decide dejar en el estado actual para una mayor claridad e implementación en el código. Sin perder más el tiempo se procede con la búsqueda de un sonido similar al del recurso original.

Una vez encontrado creamos un subdirectorio adicional en `additionalscripts` “`class`” ahí creamos `Sound.js` que tendrá guardada la clase de `Sonido`, luego será en `sounds.js` donde se crea la función `hit`, que será la responsable de reproducir el sonido mediante el uso de la clase de `Sonido`, desde una función que creará una instancia de la misma que será reproducida gracias al método sonar presente en la clase a la cual se llamará 10 veces por el bucle pasando como parámetros “`sawtooth`” para usar de `filtro` y el `valor` de `i` que con cada iteración irá aumentando en 1. Una vez creada dicha función se añade en el `index.html` la referencia al script de `Sound` de dentro de `class` y se coloca antes de la referencia del `sonds.js` para que todo funcione correctamente.

4. Pruebas del sonido espacial

Se procede a modificar la clase de `Sonido` para implementar el sonido espacial, para realizar dicha acción creamos una nueva propiedad: `canal` y para las pruebas creamos el método `newSonido`, el comienzo del método es igual que el de su antecesor `sonar`:

```
This.oscilador = this.contexto.createOscillator();
This.volumen = this.contexto.createGain();
```

Es aquí donde añadimos la línea que le asignará el valor a la propiedad canal

```
This.canal = this.contexto.createPanner();
```

Gracias a esta página: <buscar página en casa> descubrimos como cambiar el canal:

```
this.canal.positionX.setValueAtTime(-
1,this.contexto.currentTime);
this.canal.positionX.setValueAtTime(1,
this.contexto.currentTime + 1);
this.canal.positionX.setValueAtTime(0,
this.contexto.currentTime + 2);
```

Dependiendo del valor de la izquierda se reproducirá por un canal distinto: -1 es el canal izquierdo, de seguir disminuyendo el valor pej: -20, el sonido irá "alejándose". 0 representa ambos canales y 1 representa el canal derecho, al igual que con el canal izquierdo, (aunque a la inversa) a más se aumente el valor más se irá alejando el sonido, aunque en este caso por la derecha. Una vez lanzadas las instrucciones deberemos conectar el oscilador al volumen, el canal y la propiedad de destino de contexto, esta acción la realizamos mediante la siguiente línea

```
this.oscilador.connect(this.volumen).connect(this.canal).
connect(this.contexto.destination);
```

Una vez conectadas las instrucciones, se deberá iniciar el oscilador mediante `start(0)` y pararlo una vez se haya reproducido esto último por el momento se calcula manualmente y hardcodeado con el valor que debería tener para no cortar el audio a la mitad.

Para esta primera versión solo tendremos en cuenta si el sonido se reproducirá por el canal izquierdo, el derecho o ambos. Por lo que crearemos el método de `setCanal` en base a 3 posibilidades:

- pasamos la cadena left como parámetro se devolverá el valor -1 (canal izquierdo)
- pasamos la cadena right como parámetro se devolverá el valor 1 (canal derecho)
- pasamos cualquier otra cadena como parámetro se devolverá el valor 0 (ambos canales)

5. Aplicando el nuevo sonido

Una vez ya tenemos el `newSonido` y el método de `setCanal` funcionando volvemos a `sonar` y se comienza con el proceso de adaptación, para empezar, le agregamos un nuevo parámetro: `c` = `canal`, que asignaremos el valor de la misma manera que lo hicimos en el método de `newSonido`.Como `sonar` solo durará 1 segundo usamos la propiedad de `positionX` de canal, aplicando como parámetros el valor devuelto por `setCanal`(con el parámetro pasado al método de sonar véase: "`c`"), 2º parámetro this.contexto.currentTime para que se aplique desde el principio. Llegados a este punto llega el momento de conectar el oscilador, y usamos la misma línea que en newSonido:

```
this.oscilador.connect(this.volumen).connect(this.canal).
connect(this.contexto.destination);
```

Luego se aplica como antes el valor de la frecuencia usando el `setFrecuency`, y el resto se deja igual, se inicia mediante start, se reduce el sonido hasta silenciarlo, etc...

5.1 Mejorando la parte de setFrecuency

Como el método de `setFrecuency`, no tiene en cuenta la estupidez o la mala gana del usuario y solo asigna el valor sin hacer comprobaciones, borramos el método y la parte de establecer el valor de la frecuencia: (`this.oscilador.frequency.value`) la metemos en un `try Catch` donde se intentará asignar el valor del parámetro pasado a sonar siendo este "`n`". Si no puede porque se haya pasado un valor no válido (pej: una letra), en el catch se le asignará el valor del pitido por defecto: `440.0`

VI. RETOCANDO EL PROYECTO

Aquí se comenzó a realizar cambios varios sobre diferentes páginas para entregar un producto más acabado en la fecha de entrega, entre las tareas realizadas se cambió las fuentes de las páginas, se finalizó el audio y una vez silenciado se terminase de ejecutar (ya que previamente solo se le baja el volumen hasta un punto que no fuera perceptible), se modificó el método `newSonido` para usar de modelo de referencia para la siguiente actualización del próximo trimestre donde mejoraremos la producción de sonidos aplicando distancias también, esto lo hacemos metiendo en un bucle `for` las instrucciones de asignación de `positionX` de `canal` y la de conexión de `oscilador`, el bucle irá desde el valor `-20` hasta el `20` saliendo del mismo en el `21` con cada iteración en el bucle el canal aumentará en 1, comenzando desde el `-20` y se reproducirá `0.5` segundos después que el anterior, el tiempo siendo marcado por la variable inicializada j que comienza en `0` y como se indicó antes se incrementará en cada iteración en `0.5` e `i` que marcará el canal y es la que es usada de referencia para finalizar el bucle. Una vez salido del bucle y los sonidos ya "en cola" se inicializa el `oscilador` en el segundo `0`, marcado por el valor que se pasa en

`this.oscilador.start(0)` y calculando se establece la parada del `oscilador` en `22`.

También se modificó la función `hit`() de `sounds.js` para que recibiera el parámetro de en qué canal se reproducirá para que dicho parámetro se envíe al método de sonar el cual convertirá la cadena en su respectivo valor.

Se quita del `action.js` de `treedude.exe` el tener en cuenta la tecla s y se aplican a los `keyDown` de la tecla izquierda y derecha la función `hit`(). En los que se pasará `left` en caso de haber pulsado la flecha izquierda y `right` en caso de haber pulsado la tecla de la derecha.

1. **Creación de créditos**

Se comienza a crear el espacio, adaptando los scripts y añadiendo al árbol de directorios el apartado de créditos/CREDITS apartado donde se indicarán de donde se sacaron las ideas y los recursos. Como por ejemplo la fuente de SUPER HOT

2. **Optimización de puntos**

Se optimiza la función de talar de actions.js de treedude.exe para que cuando mueras en vez de sumarle un punto y restarle uno simplemente no se le sume punto a su puntuación

3. **Relocalizando las funciones**

Para intentar lograr una mayor claridad de código y una distribución mayor del código se crea en aditionalScripts el fichero treedude.js, tras asegurarnos de referenciar correctamente a dicho js desde el index.html de treedude y procedemos a mover las funciones: `talar(bidimensional,pos)` y `generarnuevo0()` .

4. **Creación de las páginas de error**

Dejando apartado temporalmente el talador se procede a crear la función `generateErrorHtml`() en `aditionalscripts` en el fichero recién creado: `pageNoFound.js`, además se crea en `assets createFont.css` donde ahorraremos en líneas de código creando la fuente ahí y haciendo que los htmls que hagan uso de la fuente: `Pios`referencien a dicho fichero. Tras comprobar que se aplican correctamente las fuentes en los elementos que antes tenían la fuente se prosiguecon la creación del script para las páginas de "error".

Al ser una página temporal usada solo para las páginas que serán desarrolladas en el futuro mantendremos un diseño simple:Mediante una variable crearemos un `h1` con el texto "PÁGINA AÚN NO DESARROLLADA", en otra variable crearemos un `h2` con "disculpen las molestias", también se crea la variable `draw` inicializada con la etiqueta p y `dibujo` en un principio inicializada vacía, pero la idea será iniciarla con un Sprite generado en `draw.js`, terminando con las variables se creó a `indicador` que se inicializará con 0.

Una vez tenemos todas las variables las adjuntamos al html a luego inicializaremos el intervalo que llamará a la función `anímate`, que llevará el dibujo que cambiará según el valor de la variable indicador: 0,1 o 2, todos estos elementos al final mediante `$("body").append(<{variables con las etiquetas html}>,` se "pegarán" al body.

En `draw` creamos los tres nuevos sprites de la animación. Una vez terminados los sprites se inicializa la variable, tras esto aplicamos la fuente pios en los `hs` (`h1`, `h2` y `h3`) ya que gracias a experiencias anteriores experimentando con la fuente no ejecuta bien nuestros sprites generados mediante ASCII, sin embargo, es aquí cuando se aplica la fuente donde se descubre que tampoco funciona correctamente los acentos, por lo que nos vemos obligados a quitarlos si queremos usar la fuente.

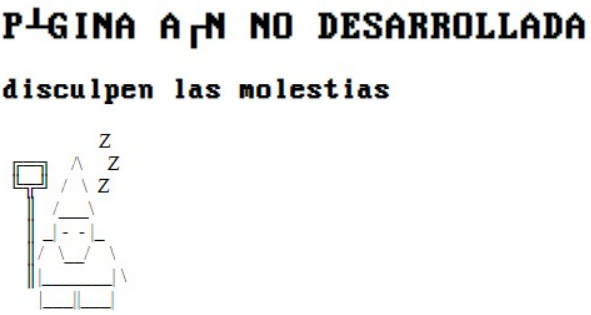


Ilustración 13: Muestra de cómo se ven representados los acentos,

Lo siguiente que se realiza es alinear los elementos de texto al centro, siguiendo el modelo del `index.html` original colocamos el `h1` en medio de la pantalla, por otro lado el talador, mediante un `position absolute`, será colocado en la esquina inferior izquierda al introducir `bottom: 1px`. Y para finalizar con la página de error se le aplican los colores del fondo y fuente del esquema de toda la página y retocamos el intervalo para que cambie más rápido y cogemos el código de detección de tecla e independientemente de la tecla que pulse volverá a atrás. Esto se lo pondremos con una variable que contendrá la etiquetade `h3`.Una vez que aseguramos el correcto funcionamiento del códigose aplica el mismo a todas las demás páginas que no han sido desarrolladas aún.

Ilustración 14: Resultado final de nuestra pantalla de "error"

VII. GESTIONANDO LA ENERGÍA

Fue a partir de este punto donde se decidió tirarse a la piscina y se comenzó a hacer un gasto de energía básico, el comportamiento no sería tan complicado como el de la energía del juego original, pero se quiso ir comenzando para que en la siguiente entrega ya se mejore, pero al mismo tiempo ya tener una base.

1. GASTO DE LA ENERGÍA

En el minijuego de treedude el jugador no comienza con toda la vida rellena, al contrario, comienza solo con la mitad de la energía, para luego ir recuperándola cortando el árbol lo suficientemente rápido para ganar más rápido de lo que la pierde, es por eso que establecemos en el css el ancho del elemento que representa la cantidad de energía restante del jugador al 50%. Temporalmente se crea la variable `interruptor`con valor por defecto `false,` Será en el evento de `keydown` en el caso de que cualquiera de las dos teclas de corte (flechas: izquierda y/o derecha) donde se revise el valor de dicha variable, si la misma tiene el valor `false` la función de `startTime` será llamada y la variable de interruptor será igualada a `true`, de lo contrario al no llamar la función de startTime la energía no se bajará.

6.1 startTime

La función de startTime tendrá un parámetro, el tiempo, volviendo a usar la función de `intervalos` como base creamos la variable intervalo la cual será inicializada vacía y en `startTime` será igualada con un window.setInterval que dentro contenga la función que se ocupará de ir restando energía y el tiempo que pasará entre cada llamada a la función.

1.2 loseEnergy

La función que se ocupaba de ir restando la energía se denominó `loseEnergy`, el funcionamiento de la misma consiste en mediante jquery y la creación de una variable interna obtener el elemento con el id de `Contenedorlife`, la representación gráfica de la energía restante del jugador, en ese momento al tener el elemento se abría la posibilidad de obtener y modificar las propiedades de dicho elemento, por lo que la acción que se llevo a cabo fue la de igualar el ancho del

elemento a su ancho – el 5% de su ancho en caso de estar al 98% (el motivo de no ser el 100% es debido a que se destroza estéticamente sobresaliendo de su contenedor), hasta ahora en todas las pruebas el ancho que daba en caso de estar en el valor máximo que le permitimos (98%) el ancho que devuelve siempre era 196, pero una teoría del equipo de programadores que ha surgido es que si se cambia la pantalla en la que es ejecutada la página este valor podría variar causando errores y/o comportamientos inesperados, no obstante por falta de tiempo se mantuvo de esta manera por el momento.

Haciendo los cálculos el 5% de 196 es 9.8, por lo que será este el valor que se irá restando a la energía total, finalmente se aplicó el comportamiento y se pasó al siguiente apartado

1.3 Muriendo

Tras conseguir que la energía fuera bajando era el momento de manejar el momento en el que la misma llegase a 0 o incluso valores negativos… O eso pensaba el equipo de desarrollo pero recibieron la agradable sorpresa de que al sacar el ancho del elemento, el mismo solo llegaba hasta el valor 0 como mínimo, por lo que ese fue un caso que no tuvieron que tener en cuenta.

Para morir por la falta de energía se creó la función de die, la idea original era la de pasar a die posición del talador, sin embargo, esto terminó dando problemas por lo que se tuvo que encontrar una alternativa

La solución que se optó fue la de crear la posición del talador una variable global para poder acceder a ella directamente, lo siguiente que se hizo fue finalizar el intervalo de reducción de energía una vez que el talador moría, (independientemente del motivo, ya sea por una rama o por la falta de energía), esto se hizo mediante `clearInterval(intervalo)` en la función de die(), esto se pudo hacer ya que intervalo es una variable global y por lo tanto startTime y die() pueden acceder sin problemas a dicha variable.

2. OBTENIENDO ENERGÍA

Usando de base el código de `loseEnergy` se crea la función `getEnergy`(), se comienza creando la variable energía al elemento con el id Contenedorlife, en este punto es la condición la que cambia: si el ancho + 6% de 196 (11.76) es mayor 196 se igualará el ancho del elemento a 196, de lo contrario se sumaria al ancho del elemento 11.76.

La función de getEnergy() será llamada cuando se sume un punto a la puntuación del jugador ya que significará que el corte fue ejecutado con éxito y el talador sobrevivió

VIII. GUARDANDO LA PUNTUACIÓN DEL JUGADOR

Cuando se movieron las funciones de lugar por algún motivo la funcionalidad de talar fue alterada, ejecutándose la parte de obtención de puntos 2 veces en lugar de una. Se consigue arreglar el bug cambiando el else donde se ejecuta la suma de puntuación por un else if para que solo se ejecute una vez.

1. GUARDADO DE DATOS MEDIANTE COOKIES

Una vez se arregló el bug de la puntuación doble se procede con el guardado de los puntos mediante el uso de cookies, comenzando simples creamos un `alert` del tipo confirmar y cancelar que comunique al usuario que la página usa cookies para guardar, este mensaje en futuras versiones será mostrado con un mensaje personalizado diferente a la de los `alerts`que vienen por defecto.

Para la gestión de las cookies se creó en `aditionalScripts` el fichero `galletas.js`, siguiendo los apuntes de la asignatura cogemos como base las funciones: `getCookie` y `setCookie`, para modificarlas y hacer que sean más fáciles de entender: getCookie fue convertida en checkCookie y setCookie en do Cookie, es en este momento donde las funciones adaptadas no realizan las acciones que tendrían que realizar, por ello procedemos a volver a dejar las funciones como estaban antes, sin embargo, ni por estas funciona. Siguiendo una página de documentación para desarrolladores publicada por Mozilla Firefox probamos de otra manera, sin embargo, no se detecta el elemento browser.

2. GUARDADO DE DATOS MEDIANTE SESSIONSTORAGE

Hasta que se descubra el motivo de por qué las cookies no se crean se opta por un método alternativo de guardado, tras probar la instrucción de guardar un item en el "`sessionStorage`": `sessionStorage.setItem("<nombre del item>",<valor del item>);` Después mediante `sessionStorage.getItem("<nombre del item>");` y comentar la línea de creación del item recargamos la página y comprobamos que en efecto se guardó correctamente el item.

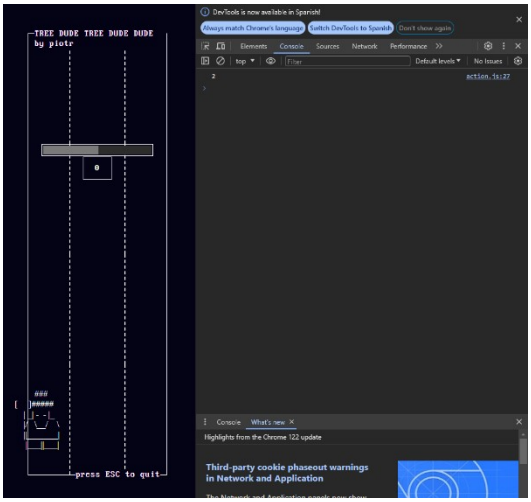


Ilustración 15: Página sin recargar la página (previo a la creación del item)

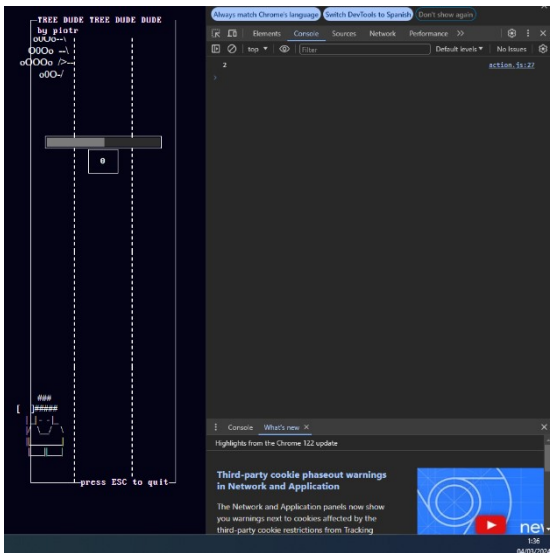


Ilustración 16: Página post recarga (después de la creación del item)

Llegados a este punto por fin se consigue implementar (aunque solo mientras la ventana este abierta) una manera de guardar la información se procede a adaptar las instrucciones al código presente en treedude desde el fichero de `galletas.js` (ya que el objetivo de hacer el guardo de datos mediante cookies aún no se ha dejado), se comienza creando un `alert` de confirmar, cancelar similar al de las cookies modificando el mensaje que muestra para reflejar la naturaleza del guardado de datos actual, no obstante antes de realizar la pregunta se comprobará si el usuario tiene alguna puntuación guardad, si ya tiene alguna no se le hará la pregunta y se asignará el valor `true` a `permission` ya que eso implicará que el usuario ya aceptó el guardado de datos. En caso de no tener ninguna se realizará la pregunta y se asignará el valor de la respuesta del usuario a la variable `permission` (si pulsa confirmar a `true`, si pulsa cancelar `false`).

Una vez que el jugador muera si `permission` es `true`se comprobará si la puntuación obtenida es mayor que la almacenada en el `sessionStorage,` si es mayor se guardará la nueva, si es menor o igual no haremos nada, esta acción será realizada mediante la función `comparePoints(puntos)`. 1º revisaremos si el usuario nos dio el permiso de guardar la puntuación, de ser ese el caso compararemos la puntuación obtenida y si es mayor la guardaremos como nuevo valor en item "puntuacion", de lo contrario el minijuego se ejecutará como hacía antes de guardar la puntuación: Juegas, Mueres, Recargas, Juegas, Mueres, Recargas...

Para esta versión al morir saltará un `alert` comentando la información al usuario (¿su puntuación es un record? ¿Cuál es su record?), en próximas versiones será cambiado para ser mostrado en la pantalla del juego de una manera más agradable. Al usar el `alert` desgraciadamente sale antes de que se actualice la pantalla, para arreglarlo metemos la llamada a `comparePoints(puntos)` en un

```
setTimeout(() =><función>,50);
```

Gracias a esto también lo usamos en el mensaje inicial para que no salgan partes que no se hayan cargado aún cuando salte el primer mensaje.

IX. PASADO A LIMPIO

Finalmente se comenzó a preparar la entrega del código para el trabajo y la optimización del mismo, se comenzó modificando el rango de las variables tratando de usar la menor cantidad de variables globales a no ser que se tratase de una que sí o sí de una variable que debe de accederse de forma global. Luego se borraron las líneas de código que repetían código innecesariamente. En action.js de menu se usa jquery para al cargar la página el elemento con el id time ya tenga la hora insertada y no muestre la hora por defecto de 00:00. En los bloques de código donde se detecta la pulsación de una tecla se procede a borrar la variable de tecla, usada para guardar el valor de event.key, el valor de la tecla pulsada y se usa directamente event.key en las condiciones.

Mientras revisamos todos los ficheros aprovechamos para establecer el title de los html a cabeceras para las pestañas algo más representativas de la función de cada página, se unifican los bloques de estilos en el style.css de treedude.exe de #a, #b, #c y #containerDude > div, ya que ambas formas se refieren a los mismos elementos. Se realiza lo mismo con el grupo de: #f0, #f1, #f2 y #f3, y #containerTree > div.

Se realiza una agrupación del contenido de los ficheros css agrupando los bloques de código en función de si son estilos a etiquetas, clases o identificadores, dentro de cada grupo también se agrupan en función de su nivel de profundidad de la jerarquía: Cuanto más cercano este al nivel a la etiqueta body, más arriba será colocado, respecto etiquetas del mismo nivel fueron ordenadas en orden alfabético, exceptuando el caso de los estilos aplicados a etiquetas, a ellos se les ordenó en función de lo que engloban: pej: html, body, div, p, etc… Además de tratar de presentar el código lo más estéticamente agradable posible.

Se deja listo para el próximo trimestre, (aunque comentado) la media query para time y SO de las páginas de los menus (la del menú principal y la del directorio de games)

Los js fueron ordenados de la siguiente manera:

1. Variables globales
2. Variables locales
3. Funciones
4. Código suelto

Tras los arreglos, se comprueba que todo continúa funcionando y se guarda el proyecto hasta recibir nuevas noticias de los testers.

X. PRUEBAS EN PRODUCCIÓN

1. ORGANIZACIÓN

Como último paso se procede al testeo, a lo largo del desarrollo por supuesto se hicieron diferentes pruebas de ejecuciones, sin embargo, aunque sea de manera inconsciente los programadores evitarán ejecutar su código de manera errónea, por ello se pone en contacto con usuarios ajenos al proyecto para que hagan de testers, conseguimos los siguientes voluntarios:

- UltraHonk (Nombre del usuario del equipo de desarrollo para realizar los testeos)
- DAXON
- Hikaru
- Onlin

Toda la gestión del testeo ajeno se realizo mediante la herramienta de Discord

En el servidor creado se crearon los siguientes canales:

- General: Canal por defecto, usado como puerta de entrada y/o cuestiones que no se ajusten a alguno de los otros canales.
- ZIP: Canal usado para pasar a los testers versiones limpias y actualizadas del proyecto
- Sugerencias: Canal donde los testers podrán aportar sugerencias sobre el proyecto
- Bugs_fallos: Canal donde los testers podrán comunicar bugs y/o fallos encontrados a la hora de ejecutar el proyecto
- Opiniones_finales: Canal que se borrará antes de entregar la documentación de este trimestre, será en el mismo donde pondrán una opinión básica sobre el funcionamiento general del proyecto
- Reclutamientos: Al igual que el equipo de desarrollo pudo encontrar una serie de testers, se invita a los testers que inviten a sus contactos para tener más opiniones y ejecuciones bajo diferentes entornos y condiciones
- Normas: Donde están enumeradas las 5 reglas del servidor

- Dudas: Canal donde los testers podrán preguntar sus dudas sobre los distintos apartados del servidor

2. APORTACIONES

- Aportado por Hikaru, dicho error ya fue solucionado:
    - El fallo de audio actual presente en treedude sigue un patrón inicial bastante consistente:
        - Cuando el jugador va a una velocidad media-alta (la barra de "vida" se mantiene al máximo la mayor parte del tiempo)
            - El sonido se desactiva en los siguientes tramos de puntuación de puntuación (números aproximados, ya que varían según la partida)

              > [50 – 130] con el punto de partida siendo el más consistente, y el punto de salida teniendo una variación de hasta +10 a +30

              > [150 – 200] con ambos teniendo una variación de aproximadamente +10

              > A medida que se va subiendo de puntuación los rangos son más inestables
        - Cuando el jugador va a una velocidad lenta (la barra de vida permanece a un nivel cercano al centro)
            - > No se detectan fallos en el sonido
    - De todas formas, a modo de prueba, se aportan dos vídeos a dos velocidades distintas para que se compruebe el punto de corte más consistente [50] es completamente evitable si se va lo suficientemente lento. (Cabe añadir que, si se empieza rápido hasta llegar al 50, y una vez allí se espera a que casi se acabe la barra de vida antes de proseguir el sonido no sufre interrupción).
    - Esto lleva conclusión a la siguiente hipótesis: Es posible que la razón por la que se produzcan esos silencios a intervalos determinados (y que se asume que varían según el jugador) sea por "enmascaramiento de silencios". Esto se debería a qué quizás el audio no tenga un corte perfecto en su final, sino que quizás se prolongue el silencio

ligeramente, y cuando se produzca una acumulación de estos, estos solapen a los efectos de sonido nuevos.

- Otra hipótesis es que el fallo simplemente se deba a un problema con la cadencia de reproducción, pero ahí no se puede probar nada

El error ocurría debido a la creación masiva sin querer de new AudioContext(), al estar el mismo en la clase en vez de estar fuera. El navegador llegaba a un punto donde no podía procesar nuevos AudioContext() y por ello no sonaban hasta que se liberaban y podía ejecutar otra tanta, al ubicarlo fuera de la clase el error parece a ver sido resuelto

XI. APLICANDO LAS SUGERENCIAS

En el trimestre pasado nuestro equipo de testers aportó una serie de sugerencias y se comenzaron a incluir aquellas que se pudiesen adaptar rápidamente.

1. Mensaje de guardado de datos:

"Este es un problema mucho menor que tengo con el aviso inicial de guardado de puntuación con treedude, y es solamente un detalle con el formato del texto de la ventana emergente que lo haría cambiar tal que así"

Añadiendo una línea en blanco adicional entre párrafos, además de puntos al final, y cambiando el "permites" por un "permite" (al estar tratando al usuario de "usted". Todo esto con el fin de mejorar la legibilidad.

Se realizan los cambios sugeridos por el tester

Ilustración 17: Nuevo mensaje

2. Efectos de sonido adicionales:

"Para hacer la navegación del proyecto más amena no estaría mal incorporar efectos de sonido más sutiles al navegar por el menú del proyecto"

Llegados a este punto se volvió *a revisar el recurso original ya que efectivamente cuenta con un audio al navegar entre las opciones, la idea* fue *al igual que con el* efecto de sonido del *golpe*, *replicarlo de la manera más similar posible*

Lo que se replicó **NO** fue el sonido de fondo, si no el sonido que se reproducía al cambiar de opción

2.1 Filtro

Lo primero que se hizo fue buscar el filtro de sonido más similar o el posible que hayan utilizado, para ello se reprodujo los sonidos por defecto de todos los filtros. Temporalmente se usó el action.js de settings para pruebas. Se preparó el terreno adjuntando en el index.html de settings sounds.js y sounds.js, y fue ahí donde se desarrolló la función: optCh() que se encargaría de reproducir el sonido, dicha función sería llamada desde el action. Para ahorrar en líneas de código se propuso la idea de importar la clase de Sound.js en sounds.js, esto se hizo de la siguiente manera.

2.1.1 *Importación y exportación…. Hardcodeado*

En Sound.js se introducirá “export” delante de class y en sounds.js será añadido al principio del script el siguiente bloque de código:

```
import {
<elemento importado: en este caso, la clase>
…
} from ‘<ruta del js que importa>’
```

Para comprobar su correcto funcionamiento se volvió al talador, y se quitó la referencia del index.html hacia Sound.js y para comprobar si seguía funcionando, por el contrario, al ser ejecutado saltaron entre otros el siguiente error:

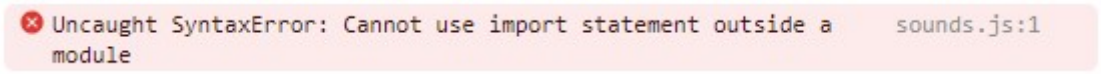


Ilustración 18: Error de MÓDULO

Para solucionarlo en los index.html que hagan referencia a sounds.js habrá que agregarles el atributo type=”module”. Al realizarlo se puede observar un error diferente:

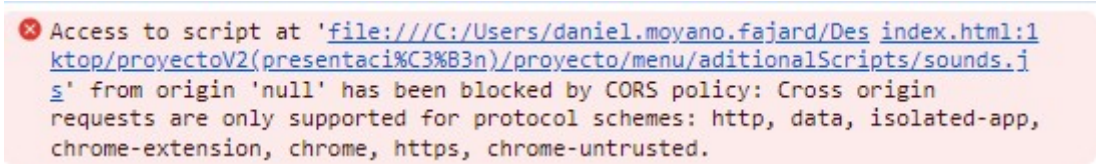


Ilustración 19: Error de servidor

Esto ocurre por las directivas de seguridad del navegador, las cuales impiden el acceso de archivos locales a las páginas. Para poder solventar el problema fue necesario instalar la siguiente extensión:

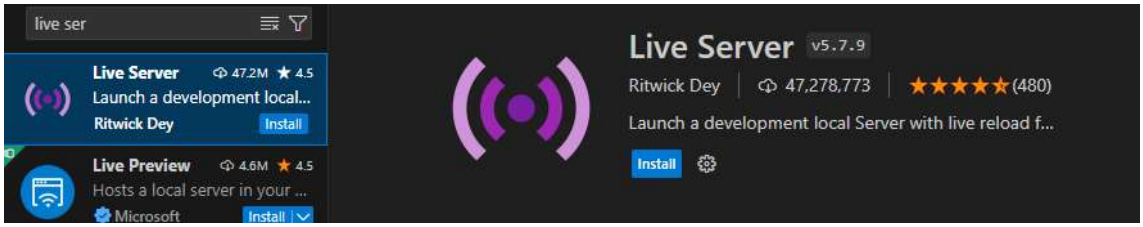


Ilustración 20: Extensión Live Server

Tras múltiples intentos de diferentes maneras que acabaron en fracaso, se procedió a realizar cierta trampa mediante el uso de jquery, se crea el script adicional de soundInitializer.js, el cual adjuntará al body los scripts de la clase de Sound y sounds en el orden correcto para poder evitar repetir el código, en ese momento se

encontraba hardcodeado pero más adelante se modificó para que se hiciese mediante la obtención de rutas relativas de los archivos y pudiera ser usado desde diferentes ficheros.

2.1.2 DIRECCIONAMIENTO RELATIVO

Para crear la función y que funcione independiente de la localización del HTML donde sea anexado, lo primero que sería necesario era el diferenciar el nivel donde se encuentra y en función de ello retroceder o avanzar dentro del árbol de directorios. Para obtener la ubicación actual se usó la línea:

```
window.location.pathname+window.location.search
```

Buscando en internet, se descubrió que en Node.js existe el método path.relative() para sacar rutas relativas de elementos, no obstante, se quiso evitar el uso frameworks para que el código fuese de mano propia y no hubiesen problemas con servidores externos, librerías faltantes a la hora de la entrega, etc... . Usando solo javascript se encontró una página que nos indicaba cómo saber si la carpeta indicada existe donde es marcada, pero dicha manera ya no es soportada en navegadores actuales por lo que se tuvo que realizar de otra manera.

Se procedió a obtener la ruta en la que se encuentra el html mediante contar las barras "/" de la URL, a ese número se le restará 2 ya que es en el apartado de menú y no en el de la raíz donde se encuentra el directorio de aditionalScripts, en caso de que el resultado sea -1, eso implicará que se encuentra en el html de la raíz (la portada) por lo que para ese caso en concreto prepararemos la variable que llevará la ruta negativa para acceder a /menú/.

De lo contrario solo quedará una opción: el resultado de contar las barras y restarle 2 es igual o mayor que 0. Es ahora cuando entraríamos en un bucle que se repetiría menguando el valor de la variable que llevaba el resultado de la operación de las barras – 2 y cada vez que menguara su valor agregaríamos un: "../" subiendo un directorio en el src, estas acciones serían repetidas hasta que el valor de la variable sea 0.

Finalmente, independientemente del caso que se diera le agregaremos a la variable que llevaría la dirección relativa de los scripts el directorio "aditionalScripts/". Y por último ejecutaríamos las líneas para adjuntar al body los scripts de la clase Sonido y el script sound:

```
        $("body").append("<script src = "+ <variable de la ruta> +"class/Sound.js
defer></script>");


        $("body").append("<script src = "+ <variable de la ruta> +"sounds.js
defer></script>");
```

En este momento se llegó a la tesitura en que algunos sonidos debían tener volúmenes diferentes, como es en este caso donde se requería que los sonidos del menú fuesen más bajos. Es por ello que se necesitó añadir la posibilidad de invocar sonidos con diferentes volúmenes. El método sonar hasta ese momento tenía los parámetros f: filtro, n: valor/ frecuencia, c: canal. Se creó el método de setVolumen en la clase Sonido.js, al cual se le pasará el parámetro numérico vol: donde se juzgará en un try catch. Si el valor es válido se asignará dicho valor al volumen, de lo contrario se usará el valor por defecto: 1.

XII. 2º ROUND DE GALLETAS

Al ahora realizar la ejecución desde un servidor se volvió a realizar una cookie de prueba para ver si ahora sí eran creadas, el resultado fue positivo. Por lo que se procedió a comentar la parte de comparePoints y sustituirla por su equivalente usando cookies, las galletas al funcionar de manera diferente fue obligatorio crear una función específica para la eliminación de las cookies guardadas.

Se comenzó con la función de checkCookie, a la cual se la pasaría como parámetro el nombre de la cookie a consultar, sin embargo, en caso de que no se le pasase un valor, por defecto tendrá en cuenta el valor “puntuacion", valor que era usado en la antigua función: checkSession.

Dentro de la función será donde se “cocinaría” la cookie a buscar, se creará la variable nombre la cual llevará el valor del nombre de la cookie concatenado con “=”, por otro lado, estarían las “migas” que cogerá la cookie del documento y la dividirá tomando como referencia el carácter “;”.

Es en este punto donde se accede en un bucle que recorrerá el contenido guardado en migas, en cada iteración tendremos la variable caracter que obtendrá el valor de la posición de i relativa a migas. Mientras que el primer carácter de dentro de la variable caracter sea “ “, se cogerá la subcadena de la variable caracter, comenzando desde el siguiente carácter hasta el final.

Si la posición de la variable nombre coincide con la 1ª posición de la cadena de la variable ya preparada de caracter se devolverá la subcadena resultante de coger los caracteres desde la longitud de la variable de nombre y la longitud de carácter.

De lo contrario si i recorre toda la longitud de migas y no se cumple la condición de anteriormente nombrada la función devolverá null, indicando que no se encontró la cookie.

Una vez que ya disponíamos de la función de checkCookie() la incorporamos al código: en un if se comprobará si el valor devuelto por checkCookie es nulo/falso,

para ejecutarlo y precisar menos líneas de código invertiremos el valor obtenido y así si devuelve un valor no realizará nada, pero si devuelve nulo se invertirá y se realizarán las acciones pertinentes.

Se prosiguió avanzando con la adaptación de la función usada para borrar las cookies como valor por defecto use el de puntuación, como siguiente paso se recordó que la función de checkSeession era llamada desde el action.js y desde la función ya se preparaba todo. En vez de borrarlo todo se optó a adaptar la función de checkSession para usar checkCookie y su resultado devuelto para la evaluación de la condición. Como siguiente paso se alteró el mensaje del alert para reflejar el método de guardado de datos actual.

Tras esto se modificó la función de doCookie para que como parámetros por defecto tuviese: valorCookie/puntuación: 0, nombreCookie: puntuacion y Dias: 7.

Por último, se modificó la función de comparePoints para que use checkCookie y doCookie, ya la función de doCookie sirve tanto para crear una nueva galleta como para rescribir una ya existente si pasamos como nombre de la cookie el nombre de la cookie ya creada. Una vez terminado de adaptar el código tocó probarlo.

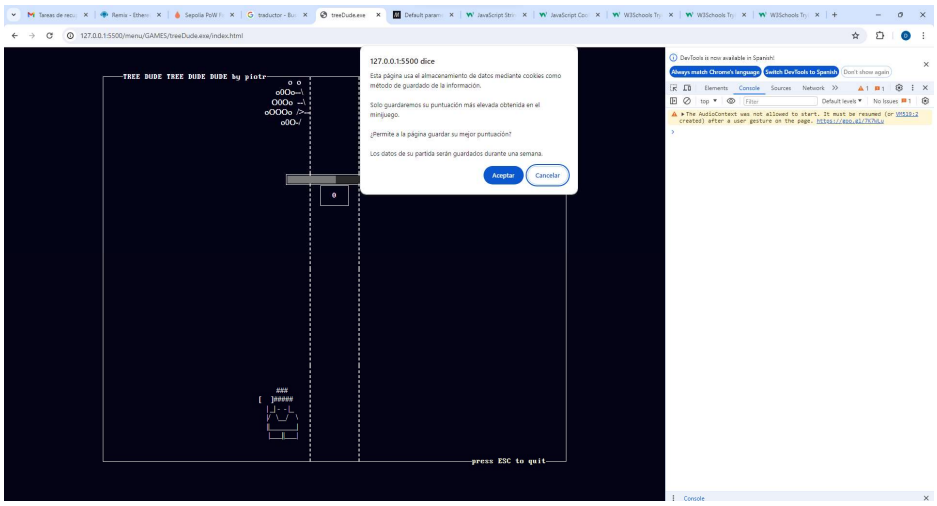


Ilustración 21: Prueba 1 no guardamos los datos

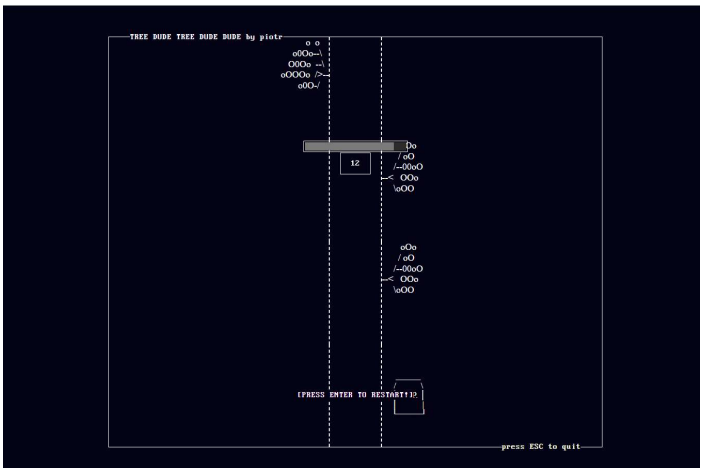


Ilustración 22: Sacamos 12 ptos

Ilustración 23: Resultado

Como podemos observar no se guardó la puntuación

Ilustración 24: Guardamos puntos

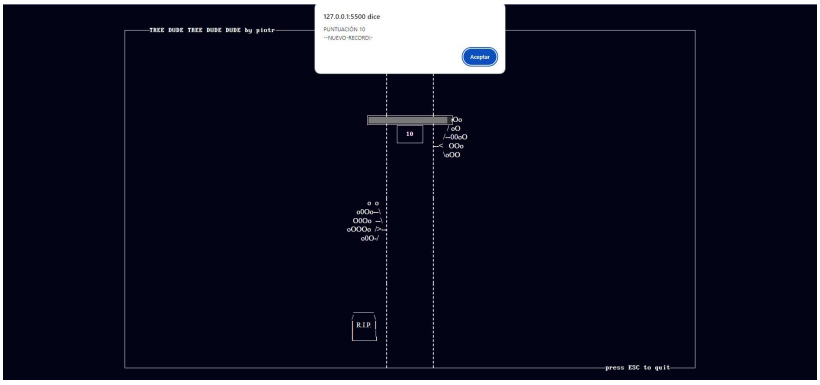


Ilustración 25: Resultado 2

XIII. IDEANDO

Cogiendo de los ejercicios de clase obtuvimos el ejemplo del uso de una promesa: (Tema 3)

La cual consistía en tirar 6 veces un dado imaginario y si salía un 6 daba error. Se comprobó su correcta funcionalidad y comportamiento para un mayor entendimiento sobre la misma puesto que nos podría ser de utilidad a la hora de manejar varios elementos de forma simultánea.

XIV. MÚSICA

Para comenzar se fue testeando el cómo se escuchaban las “acciones” del talador al mismo tiempo que reproducimos un sonido que luego terminó evolucionando a la música del juego, primero se obtuvo el código que ya habíamos confeccionado referente a la ejecución de código simultáneo en el action.js de settings con las pruebas que se realizaron con las promesas y por el momento fue movido a songs.js por el momento. Luego en el html de treedude se agregó la referencia al songs.js.

Tras comprobar la ejecución correcta del código antiguo se procedió a que se repitiera indefinidamente, esto se consiguió encapsulando el código en una función que será llamada en un intervalo definido en action.js, como siguiente paso se volvió a comentar el código simultáneo para comenzar a adaptar el “esqueleto” a las necesidades surgidas, en un principio no queríamos que se repitiese varias veces por lo que la parte de iteraciones fue comentada, junto la constante que llevaba los números: “constnumbers = []” y en su lugar se creó la constante timbre = new Sonido. Al no queret repetir el código varias veces también se comentó la parte del bucle y el if fue sustituido por un try catch.

Es en este momento donde se aprovechó para retocar la clase Sound, en concreto el método sonar para que todos los valores tengan un valor por defecto y así

facilitar la ejecución de sonidos de prueba no teniendo que poner ningún parámetro. Es así como por defecto se utilizó el filtro a sine, el valor de la frecuencia 440.0, el canal 0 (siendo este el de ambos canales con su valor más “cercano”) y el volumen 1.

Volviendo a songs en caso de que saltase un error en el catch se haría uso de un reject donde el error será guardado en fail y tras ello se igualaría error a true y “message” a fail, si todo se ejecuta correctamente se llegaría a la parte del resolve donde se igualaría error a false y como no disponíamos de ningún valor que queramos saber, se comentaró la parte de value. Por último se comentó la parte interior de la invocación a doTask (esto debido al no repetir la invocación ninguna vez), también se comentó la parte del then para no llenar la consola de mensajes pero la parte del catch se dejó tal cual ya que en caso de algún error será necesario saber que es lo que ocurrió, tras esto la hora de volver a probar la ejecución del código llegó. Con esto nuestra prueba finalizó con éxito.

Ahora era necesario generar la melodía, es decir una sucesión de notas por lo tocó descomentar la parte de las iteraciones, pero sin tocar todo el rato la misma nota, en su lugar tocó diseñar la manera de tocar la nota correspondiente. Por lo que en un principio hubiera sido este el momento donde buscar la frecuencia de cada nota si no fuera por en Reddit parece que alguien ya nos hizo parte del trabajo, no obstante se tuvo que esperar para comprobar los descubrimientos del usuario a que se pudiese acceder a la página al estar la misma capada por el proxy de la escuela:

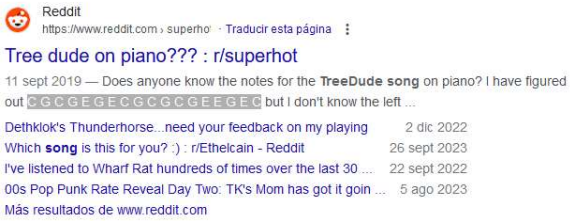


Ilustración 26: Resultado web Reddit

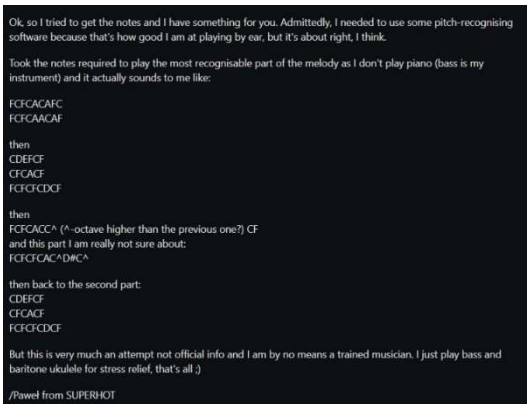


Ilustración 27: Acordes usados de referencia

Para aquel que no tenga conocimientos musicales habremos incluido la explicación de lo que es un acorde. Esto más la página que nos indicó en un principio a la creación de sonidos ya contábamos con todo el material necesario para sacar las frecuencias necesarias ya que la misma cuenta con una tabla de frecuencias de los distintos acordes según su tono. De momento para la música se usará el mismo filtro usado que para los efectos de sonido, véase sawtooth.

Para esta versión era preciso que la reproducción de sonido se repitiese 18, (aunque con diferentes valores dependiendo de la iteración), es en este momento donde se accederá a la clase Sound de nuevo para preparar el método setSound que usaremos para establecer el valor de la frecuencia, de momento ya tenemos un tono… Aceptable, pero la duración de las notas en la canción original es más corta que en la nuestra por lo que se creó un nuevo método y en la construcción de la clase Sound se añadirá otro parámetro nuevo: t de tiempo o time, por defecto se le puso el valor que tenía al principio 0.5. Luego el método será setDuration el cual validará el tiempo pasado como parámetro. Tras esto se adaptó al nuevo tiempo de espera para volver a "ejecutar" la canción.

Con esto se creó el intervalo para reproducir la música, cuando el jugador muere, se finaliza el intervalo de la música para inicializar otro con la música que se reproducirá al morir, ahora bien, el problema que puede surgir es que el jugador muera antes que se termine un "bucle" de la canción que se reproduce al estar vivo por ello se deberá chequear el valor de la variable ded antes de reproducir cada nota, si la variable esta en false se reproducirá la nota que toque en dicha iteración, de lo contrario para finalizar la ejecución de la canción se lanzará un error. Que será recogido en el catch, con esto se pasó a retocar un poco más las diferentes canciones que se reproducirán y se dará por finalizado el apartado.

XV. MUTE

Ahora que se consiguió la reproducción y música, era necesario dar la opción de poder quitar el sonido al usuario si así lo desea. Primero gracias a Bootstrap se obtuvo los iconos de volumen:

Ilustración 28: Iconos de bootstrap usados

Por el momento se comentaron las líneas del icono que usaremos para cuando el sonido esté desactivado, como siguiente paso se les aumentó el tamaño a 32 y se colocaron al lado derecho de la pantalla dejando un margen de 20px, como siguiente paso se registró cuando se la "M" fuese pulsada, desde action y se creó la **var**iable mute con valor false por defecto, una vez que ya se registraba correctamente la pulsación se procedió a crear en treedude.js la función chgVol() donde si el valor de mute es verdadero se cambiará a falso y viceversa. Una vez se comprueba el correcto

funcionamiento del cambio del valor, queda sustituir el icono en función del nuevo valor de mute.

Tras esto en esta versión lo que se hará será continuar con la ejecución de las melodías solo que con el volumen 0. Para ello se modificó la función de hit para que reciba el valor de mute si es verdadero se reproducirá el golpe con volumen 0, de lo contrario no se pasará el valor del volumen reproduciéndolo por defecto a valor 1.

XVI. AUMENTANDO LA DIFICULTAD DE LA GENERACIÓN DE LAS RAMAS

Hikaru: *"Al igual que en el juego original de "treedude", no estaría mal ver una rampa en la dificultad a medida que el jugador va progresando y acumulando más y más puntuación, ya que en mi experiencia, el usuario promedio tiene más que suficiente tiempo para cortar como para hacer cuanta puntuación quiera (siempre y cuando no se confíe o intente ir demasiado rápido). Esto se podría implementar por medio de una reducción en el tiempo entre cortes mientras la puntuación del jugador más aumente (que en mi experiencia no debe ser difícil de implementar), o incorporando patrones de ramas más complejos (ya que asumo que su generación actual depende enteramente de RNG), aunque esto sería mucho más difícil."*

Honk: "*Ah y en cuanto los patrones de ramas también mi idea es alterarlos a partir de cierta puntuación: Actualmente la manera en la que es generada una rama es la siguiente: Se lanza una "moneda" sale cara, creas rama izquierda, ¿Se creó antes una rama? Sí, FIN Siguiente rama, se lanza "moneda" sale cruz, no se crea rama, ¿Se creó antes una rama? No, se lanza otra "moneda" sale cara, se crea rama derecha Siguiente rama, se lanza "moneda" sale cruz, no se crea rama, ¿Se creó antes una rama? No, se lanza otra "moneda" sale cruz, no se crea rama Resumiendo, el jugador tiene 50% de probabilidades de que le salga una rama en la izquierda, 25% de posibilidades que le salga en la derecha y 25% de probabilidades de que no le salga ninguna... O eso creo, hace mucho que no estudio estadística*"

Honk: "*El comportamiento que se implementaría al obtener una puntuación > X, aún no la he decidido es el quitar el 25% de posibilidades de que no le salga rama y que sea un 50% de probabilidades que le salga una rama en la izquierda y otro 50 en la derecha por lo que no tendría movimientos de "descanso" como si tiene ahora (refiriéndonos a las ocasiones donde no hay rama en ninguno de los dos lados)*"

Hikaru: "*yo no creo que los espacios sin rama deban desaparecer, ya que son un buen espacio de respiro para el jugador mientras avanza entre secciones, pero quizás una distribución más parecida a: 45% IZQ 45% DER 10% NaN evitaría que un lado fuera más favorable que el otro*"

Para crear la nueva generación de ramas se usará como base la generación de ramas original. En el código original generábamos 2 números aleatorios, uno para la izquierda y otro para la derecha, si el valor de la izquierda generaba una rama ya se

dejaba el valor de la derecha a 0, por el contrario se generaba otro número aleatorio entre 0 y 1 dando como posibilidades que se generase una rama en la derecha o no, de esta manera como se explicó en las capturas previas la manera de hacer "trampa" en el juego es quedándose en la derecha ya que la posibilidad que te salga una rama en menor a que te salga en la izda.

Para solucionar esto se generará a partir de ahora un único número aleatorio del 1 al 100, si el número es menor de 45 se creará una rama en la izquierda, por otro lado, si el número es mayor a 45 y menor a 91 se creará a la derecha, por último, si es mayor de 90, no se generará rama. Ahora que ya contábamos con el enunciado tocaba codificarlo, para empezar se eliminarán las líneas de código que nos sobran como el interruptor SW y el bucle. Ahora se modificará la propiedad de Math.floor(Math.random()…) de *2 a *100) +1:

```
let rama = Math.floor(Math.random() * 100 + 1);
```

Y por último tocó modificar el condicional, a continuación se muestra el código del condicional con cada condición a una línea, en este caso por lo tanto el poner las llaves quedaría como voluntario, no obstante es preferible siempre usar las llaves para macar de manera más clara el fin de los bloques de código:

```
*
if (rama < 45) cadena = [1,0];

elseif (rama < 91) cadena = [0,1];

else cadena = [0,0];
```

Se comprueba la correcta ejecución del código

XVII. AUMENTANDO LA DIFICULTAD SUBIENDO DE NIVEL

Como en esos momentos no podíamos contar con los medios de revisar el patrón que sigue treedude para subir de nivel vamos a ir avanzando en los aspectos más básicos que sí podíamos hacer siendo estos: el gráfico y el sonoro, ya que cuando el jugador sube de nivel salta un indicativo sonoro que siempre es el mismo y un letrero aleatorio en la parte superior de la pantalla ocultando brevemente los puntos:

1. GRÁFICOS

Los posibles letreros que le pueden salir al jugador son los siguientes:

- FASTER
- LEVEL UP
- TICK TOCK

- CHOP CHOP
- SPEED UP

Y nosotros adicionalmente añadiremos uno extra: HONK, primero: mediante ASCII trataremos de reproducir manualmente el resultado final de uno de los letreros: como letrero de ejemplo se usó FASTER.

Nota: Los espacios en blanco para no descentrar los caracteres se realizaron mediante el mismo caracter pero con fuente blanca.

El siguiente paso es conseguir que se vea igual en el navegador, primero lo haremos con fondo blanco y luego en negro para esto se volverá a usar de forma temporal SETTINGS, para esto en draw crearemos una nueva función: para la primera pruebacraremosinvokeSquare que creará un array bidimensional 5*5 poniendo todos los valores a true, después crearemos la función writeChar a la que se le pasará el valor devuelto por invokeSquare, la cual irá recorriendo el array bidimensional, si el valor es verdadero agregará a una cadena █, de lo contrario agregará █, una vez que termine con la primera agrupación de subcadenas además agregará un: “<br>”. Para finalmente devolver la cadena total, después esa cadena la agregaremos al body. Una vez que se compruebe que funciona correctamente dando como resultado, para conseguirlo antes se accederá al css de settings para agregar las diferentes clases, en este caso al tener el fondo del documento por defecto blanco y la fuente negra solo deberemos crear el estilo de spc que se le dará una opacidad de 0 para que sea transparente.

Ilustración 29: Resultado en web

Para corregir las líneas será necesario asignarlas un margin de -1px

Tras conseguir que funcione correctamente crearemosinvokeLetter la cual recibirá por parámetro la letra que queremos invocar y en función de la letra que introduzcamos se creará su array bidimensional correspondiente.

Una vez ya tenemos el esquema correcto de todas las letras de la palabra de prueba (faster) movemos las funciones de invokeLetter y writeChar a letters.js

A continuación gracias al siguiente vídeo conseguimos la invocación a una función con N parámetros.

https://www.youtube.com/watch?v=1Q0npMK0Odw

Primero se creó una cadena con los esqueletos de cada una de las letras necesarias.

Después para juntarlas en una palabra primero se deberá sacar la longitud de la palabra en sí, así mismo cuantos "cuadrados" ocupa cada letra, para conseguir realizarlo a tiempo y facilitarnos la tarea haremos que todas las letras ocupen los mismos cuadrados máximos

Y tras mucho tiempo se consiguió que se muestre la palabra faster, con ciertas trampas pero al no quedar demasiado tiempo se da por correcto burro como animal de compañía. Por algún motivo al escribir el contenido mediante javascript siempre parece recortar parte del final del mensaje de manera provisional para solucionarlo lo que se ha hecho es introducir espacios al final del mensaje y así que los caracteres que se "coma" sean los espacios.

Una vez que contemos ya con todas las letras necesarias para los mensajes tocará el seleccionar el mensaje una vez llegada a la subida de nivel de manera aleatoria, mostrarlo un segundo, hacerlo desaparecer, volver a hacer que aparezca para por fin ocultarlo definitivamente.

Para esto se hizo la prueba cuando se alcancen los 10 puntos, al llegar a dicha puntuación llamaremos a la función levelup que generará un número aleatorio del 0 al 5, mediante un switch case se pintará la palabra equivalente.

Es en esta página donde encontramos el código js para hacer que el texto parpadee, de manera similar a la que codificamos nosotros al principio, solo que añadiendo temporizador y permitiéndonos una mayor flexibilidad en la adaptación del código:

https://codedamn.com/news/html/how-to-create-flashing-blinking-text-using-html

```
let<variable> = document.getElementById(<idElemento');
```

```
setInterval(function() {

blinking_text.style.display =
(blinking_text.style.display == 'none' ? '' : 'none');

        }, <tiempo>);
```

Explicado brevemente el intervalo nuevo revisa el display del elemento marcado, si es none, lo quita dejando que el display por defecto se aplique, de lo contrario lo iguala a none y así todo el rato. Con esto vamos al script de draw para crear la función blinking usando la anterior marcada como base. A la función la pasaremos un mínimo de un valor y máximo de tres valores: el id del elemento que parpadeará, el tiempo de cada intervalo y cuantas veces lo hará. Dentro de la función se creó la variable: metronomo (para llevar el intervalo) y contador igualado a 0, mientras que los parámetros serán: identifcador que siempre deberá pasársele el id del elemento que queremos que parpadee, time con valor por defecto 250 y repeticiones que será recibida por parámetro con valor por defecto 8. Una vez creada la variable de metronomo y entendiendo el funcionamiento de los parámetros se prosiguió a crear la variable objeto la cual se igualará al valor del documento de dentro del html con el id pasado por el parámetro identificador.

Tras esto se establecerá el intervalo explicado previamente añadiendo dentro del mismo la condición de si contador==repeticiones finalizaremos el intervalo. Ya preparada la función en draw.js ahora proseguimos a rellenar msg con el mensaje que toque cada vez para esta versión y hacerlo más sencillo estableceremos el valor del letrero desde treedude.js justo antes de llamar a blinking, una vez que todo está implementado solo queda probar el código.

2. SONORO

Al ir justos de tiempo no nos enrollaremos con tratar de obtener el mismo sonido, se aprovechó para modificar el método de setCanal para además de aceptar left y right como valores válidos también aceptar valores numéricos para poder jugar con las distancias, esto se implementa mediante la propiedad `<variable>.isNan`: devuelve false en caso de ser un número el valor de la variable por lo que al ponerlo en un if hará que se salte la igualación a 0 en caso de ser un número.Luego realizamos la comprobación de si está o no en mute la página, para que si está en mute lance un “error” y no suene nada. De lo contrario creamos una promesa asíncrona que se repetirá desde el -4 al 6 con una pasa de 500 milisegundos entre cada iteración y cada

3. FUNCIONALIDAD

El mensaje ya se encuentra implementado también el sonido solo falta implementar que de verdad aumente la velocidad, lo que se codificará es: (TEMP) en el `if puntos==10` finalizar el intervalo de bajada de energía y volverlo a crear pero con un intervalo menor. Para facilitarnos a futuro haremos que el tiempo del intervalo de energía se guarde en una variable y a esa misma variable irla restando su valor hasta llegar a un mínimo X.

Tras investigar el comportamiento del juego deducimos que en realidad, sube de nivel siempre al llegar a 11, y a partir de ahí para saber la siguiente puntuación requerida antes de subir de nivel solo hay que aumentar en uno las unidades y sumarle a las decenas el nuevo valor de las unidades quedando de la siguiente manera:

```
11 --> 32 --> 63 --> 104 --> 155…
```

Tras sacar el patrón lo programamos para aplicarlo y mezclamos las diferentes partes de la subida de nivel: (sonora, funcional y visual)

CONCLUSIONES

El trabajo de recuperación de web3 quizá no dejó al equipo de programación enfocarse todo lo que les hubiera gustado al proyecto pero al final decidieron centrarse en la parte de treedude y dar un producto decente que no tratar de abarcar otras áreas desde 0 y entregar un producto más deficiente pero más grande. Por otro lado el tester hizo un gran trabajo y no se le podría pedir más, demostrando conocimientos e interés por el producto así como el leerse la letra pequeña del contrato entregado.

Aunque hubiera estado bien haber recibido noticias del equipo de super hot…

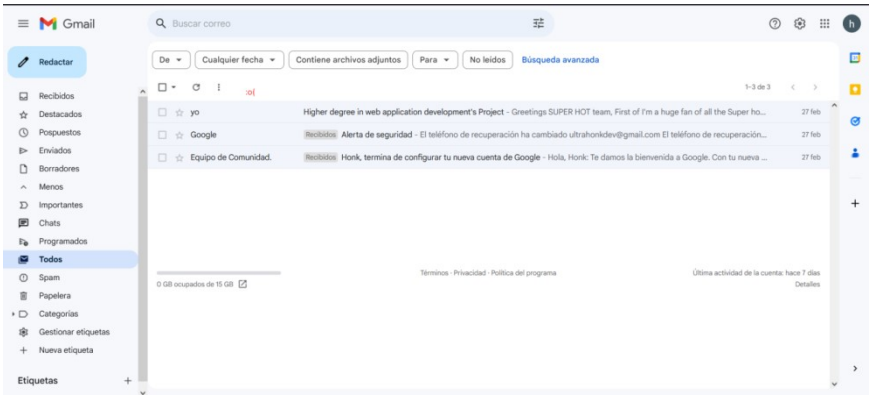


Ilustración 30: :(

I. CONCLUSIONES DEL TESTER

Hikaru: “h:o(nk

nah está muy guay, muy chula la implementación de la música y los sonidos de navegación del menú (por no hablar de los mensajes que aparecen durante el juego y el increíble nuevo nivel de dificultad)

primeras impresiones de esta nueva versión: 10/10”

BIBLIOGRAFÍA

Para dudas generales se utilizaron:

W3Schools online web tutorials. (s. f.). https://www.w3schools.com/

GeeksforGeeks. (s. f.). *GeeksforGeeks | a computer science portal for geeks*. https://www.geeksforgeeks.org/

Stack Overflow en español. (s. f.). Stack Overflow en español. https://es.stackoverflow.com/

Para obtener la base del script para hacer que las letras desaparecieran se usó:

Trellat. (2023, 26 octubre). Texto parpadeante con Javascript. Trellat. https://trellat.es/texto-parpadeante-con-javascript/

Para repasar las bases de los arrays multidimensionales se usó la siguiente página

Olawanle, J. (2023, 20 enero). JavaScript 2D Array – Two dimensional arrays in JS. freeCodeCamp.org. https://www.freecodecamp.org/news/javascript-2d-arrays/

Para visualizar todos los ASCII para representar los distintos sprites se usó:

Peter. (s. f.). The complete table of ASCII characters, codes, symbols and signs, American Standard Code for Information Interchange, The complete ASCII table, characters,letters, vowels with accents, consonants, signs, symbols, numbers ascii, ascii art, ascii table, code ascii, ascii character, ascii text, ascii chart, ascii characters, ascii codes, characters, codes, tables, symbols, list, alt, keys, keyboard, spelling, control, printable, extended, letters, epistles, handwriting, scripts, lettering, majuscules, capitals, minuscules, lower, case, small, acute, accent, sharp, engrave, diaresis, circumflex, tilde, cedilla, anillo, circlet, eñe, enie, arroba, pound, sterling, cent, type, write, spell, spanish, english, notebooks, laptops, ascii, asci, asccii, asqui, askii, aski, aschi, aschii,20240124. The complete table of ASCII characters, codes, symbols and signs. https://theasciicode.com.ar/

Para obtener y repasar el funcionamiento y propiedades del posicionamiento flex se usó:

Flexbox Froggy. (s. f.). A game for learning CSS flexbox.

https://flexboxfroggy.com/

Como ayuda en la creación de algunos sprites se utilizó el descubierto proyecto de github:

https://github.com/fr0zn/TreeDude

Para ayuda en la documentación y explicación de conceptos se usaron las siguientes páginas:

Google. (s. f.). https://www.google.com/

colaboradores de Wikipedia. (s. f.). *Wikipedia, la Enciclopedia libre*. https://es.wikipedia.org/

School, T. (2023, 28 junio). Sprite videojuegos: ¿qué son y para qué sirven? Tokio School. https://www.tokioschool.com/noticias/sprite-videojuegos/

Antes de continuar. (s. f.). https://translate.google.es/

Machuca, F. (2022, 1 abril). *Aprende qué es un mapa de bits y haz que tus trabajos resalten por su calidad*. https://www.crehana.com.

https://www.crehana.com/blog/estilo-vida/que-es-mapa-bits/

Para la generación de sonidos mediante código se consiguió realizar en gran parte a la base explicada en la siguiente página, así como el propio código fuente de la misma:

Gauthier, M. G. (2016, 1 noviembre). *Generate sounds programmatically with Javascript*.

https://marcgg.com/blog/2016/11/01/javascript-audio/

Para intentar realizar el guardado de datos mediante cookies

Cookies.set() - Mozilla | MDN. (2024, 22 enero). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/cookies/set

Sign in - Google Accounts. (s. f.). https://classroom.google.com/w/NTg5NTg5MjM1OTAx/t/all

Para realizar el guardado de datos en el sessionStorage:

Sign in - Google Accounts. (s. f.). https://classroom.google.com/w/NTg5NTg5MjM1OTAx/t/all

Para la obtención de los acordes de tree dude

Acordes Tree Dude super HOT. (s. f.). Reddit. Recuperado 13 de junio de 2024, de https://www.reddit.com/r/superhot/comments/d2ioku/tree_dude_on_piano/?rdt=63212

Para el formato APA de la bibliografía

Scribbr. (2022, 28 noviembre). *Formato APA con el Generador APA de Scribbr*. https://www.scribbr.es/citar/generador/apa/

Para los iconos de volumen:

Contributors, M. o. J. T. A. B. (s. f.). *Bootstrap*. https://getbootstrap.com/

APÉNDICES

De dónde se sacaron las ideas del proyecto

Save 70% on SUPERHOT on Steam. (s. f.). https://store.steampowered.com/app/322500/SUPERHOT/

Typing Speed Test - Online typing test. (s. f.). https://typing-speed-test.aoeu.eu/