

TRABAJO PARA EL PRIMER TRIMESTRE DE LA ASIGNATURA DE DESARROLLO EN ENTORNO CLIENTE

ESTIMULADOR DE REFLEJOS MECÁNICOS

AUTOR: Moyano Fajardo, Daniel

DNI: 05964807X

ASIGNATURA: DESARROLLO EN ENTORNO CLIENTE

En Madrid, a día 7 de diciembre de 2023

**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

[I. Etiquetas html 1](#_Toc160676234)

[II. Conceptos de programación 1](#_Toc160676235)

[III. Conceptos de videojuegos 3](#_Toc160676236)

[I. PANTALLA DE INICIO 6](#_Toc160676237)

[1. Laanimación de lasletras del inicio 6](#_Toc160676238)

[2. LA INTERACCIÓN DEL USUARIO 8](#_Toc160676239)

[II. Comenzando con el menú 8](#_Toc160676240)

[1. Haciendo un reloj 10](#_Toc160676241)

[2. Nuestro puntero para seleccionar opciones 10](#_Toc160676242)

[**2.1** OptionUP 11](#_Toc160676243)

[**2.2** OptionDown 11](#_Toc160676244)

[III. Creando el menú de GAMES 13](#_Toc160676245)

[IV. Desarrollo de por fin, TREEDUDE.”exe” 14](#_Toc160676246)

[1. Los sprites del leñador 14](#_Toc160676247)

[**1.1** Diseñando el escenario 15](#_Toc160676248)

[2. La generación aleatoria 16](#_Toc160676249)

[3. UNNUS ANNUS MEMENTO MORI 17](#_Toc160676250)

[4. Retomando el proyecto 18](#_Toc160676251)

[5. Colocando los textos 18](#_Toc160676252)

[6. Salida y preparamiento de la vida 19](#_Toc160676253)

[7. Aplicando el esquema de colores y la estética de la página 20](#_Toc160676254)

[8. Mejorando el tronco 21](#_Toc160676255)

[9. Do it again! 23](#_Toc160676256)

[V. CREANDO SONIDO 24](#_Toc160676257)

[1. Anexando 25](#_Toc160676258)

[2. Mejorando los setters: 26](#_Toc160676259)

[3. Creando la clase Sonido: 26](#_Toc160676260)

[4. Pruebas del sonido espacial 28](#_Toc160676261)

[5. Aplicando el nuevo sonido 29](#_Toc160676262)

[**5.1** Mejorando la parte de setFrecuency 29](#_Toc160676263)

[VI. RETOCANDO EL PROYECTO 29](#_Toc160676264)

[1. Creación de créditos 30](#_Toc160676265)

[2. Optimización de puntos 30](#_Toc160676266)

[3. Relocalizando las funciones 30](#_Toc160676267)

[4. Creación de las páginas de error 30](#_Toc160676268)

[VII. GESTIONANDO LA ENERGÍA 32](#_Toc160676269)

[1. GASTO DE LA ENERGÍA 32](#_Toc160676270)

[**1.1** startTime 32](#_Toc160676271)

[**1.2** loseEnergy 32](#_Toc160676272)

[**1.3** Muriendo 32](#_Toc160676273)

[2. OBTENIENDO ENERGÍA 33](#_Toc160676274)

[VIII. Guardando la puntuación del jugador 33](#_Toc160676275)

[1. GUARDADO DE DATOS MEDIANTE COOKIES 33](#_Toc160676276)

[2. GUARDADO DE DATOS MEDIANTE SESSIONSTORAGE 34](#_Toc160676277)

[IX. PASADO A LIMPIO 35](#_Toc160676278)

[X. PRUEBAS en producción 36](#_Toc160676279)

[1. ORGANIZACIÓN 36](#_Toc160676280)

[2. APORTACIONES 37](#_Toc160676281)

TABLA DE IMÁGENES

[Ilustración 1 Vista del modelo a replicar 8](#_Toc160676170)

[Ilustración 2 Vista de containerDude y containerTree 15](#_Toc160676171)

[Ilustración 3 Vista de containerTree con el propio árbol 16](#_Toc160676172)

[Ilustración 4: Vista del no aumento de puntuación 18](#_Toc160676173)

[Ilustración 5: modelo de lo que se quiere conseguir 18](#_Toc160676174)

[Ilustración 6: resultado final 19](#_Toc160676175)

[Ilustración 7: vista del error al aplicar la fuente 20](#_Toc160676176)

[Ilustración 8: Juego sin tronco 21](#_Toc160676177)

[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 21](#_Toc160676178)

[Ilustración 10: Bordes de los nietos col1 22](#_Toc160676179)

[Ilustración 11: Nuestro tronco responsive 22](#_Toc160676180)

[Ilustración 12: Imagen del correo enviado al equipo de SUPER HOT 26](#_Toc160676181)

[Ilustración 13: Muestra de cómo se ven representados los acentos, 31](#_Toc160676182)

[Ilustración 14: Resultado final de nuestra pantalla de "error" 31](#_Toc160676183)

[Ilustración 15: Página sin recargar la página (previo a la creación del item) 34](#_Toc160676184)

[Ilustración 16: Página post recarga (después de la creación del item) 34](#_Toc160676185)

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

# 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

# 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); R**ed, **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.

# 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

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

# 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.

## 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.

## 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.

# 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.

## 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.

## 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.

### 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

1. 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] + "&emsp;|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.

# 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.

# Desarrollo de por fin, TREEDUDE.”exe”

## 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 &nbsp; 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.

### 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:

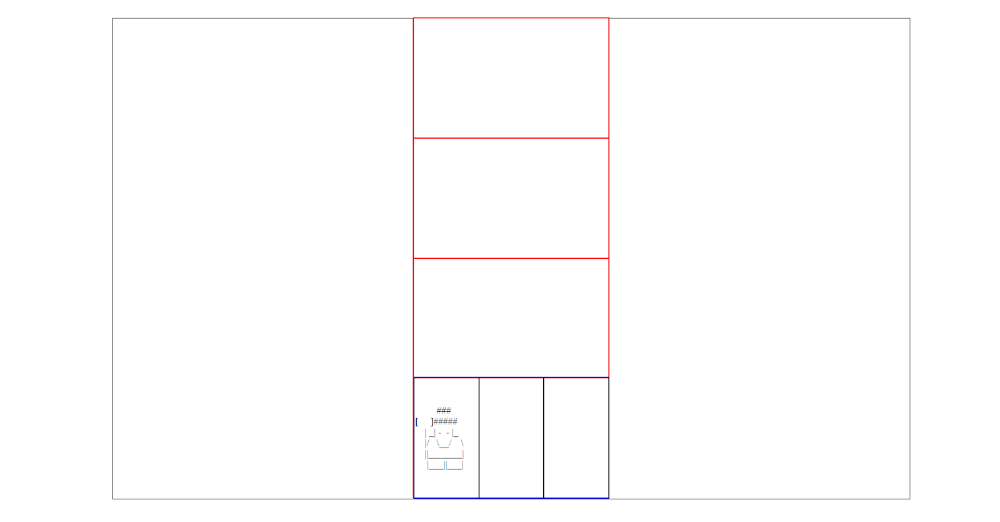


Ilustración 2 Vista de containerDude y containerTree

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:

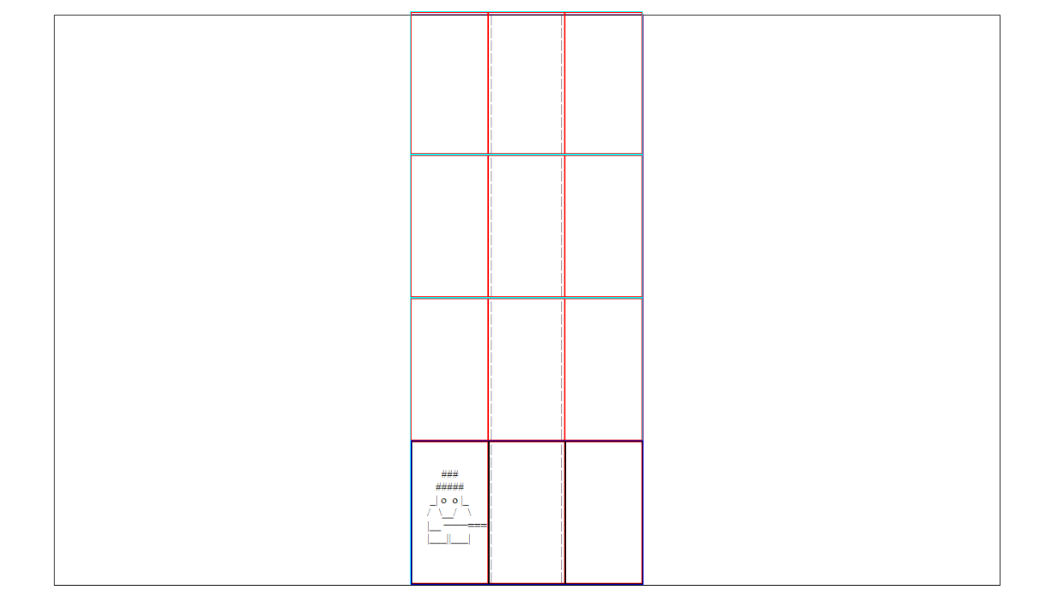


Ilustración 3 Vista de containerTree con el propio árbol

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.

## 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.

## 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

## 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:

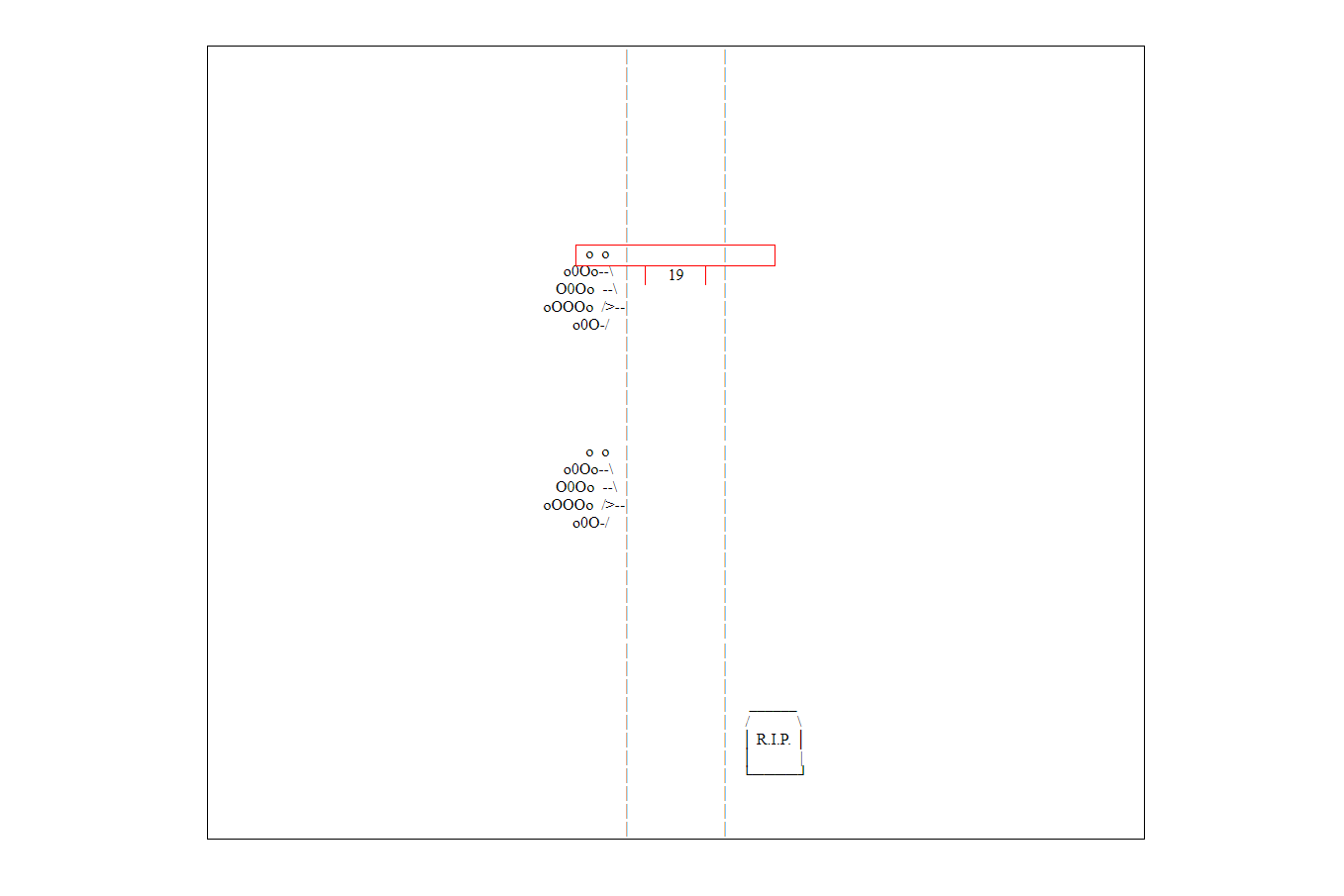


Ilustración 4: Vista del no aumento de puntuación

## 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.

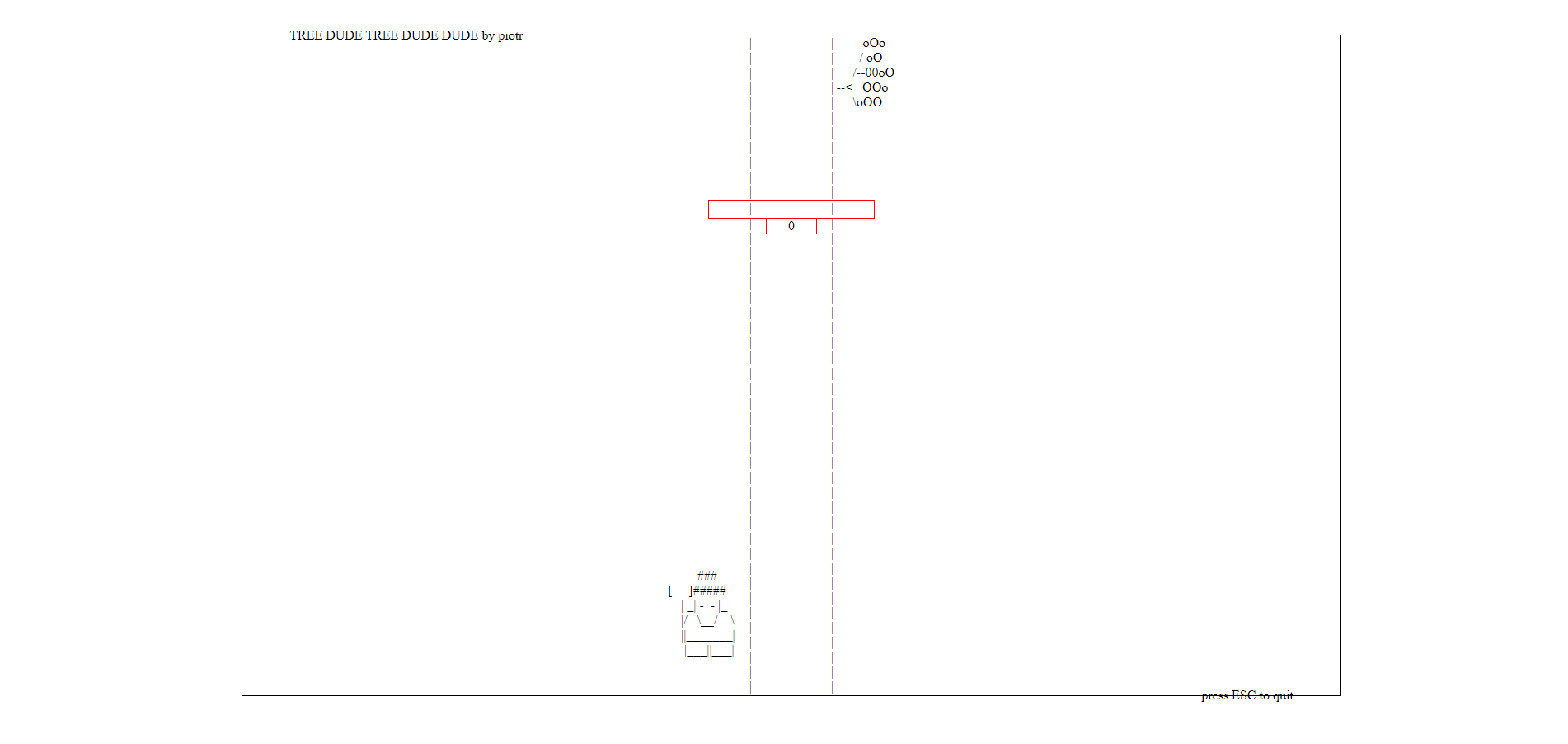


Ilustración 6: resultado final

## 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.replacejunto 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.

## 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

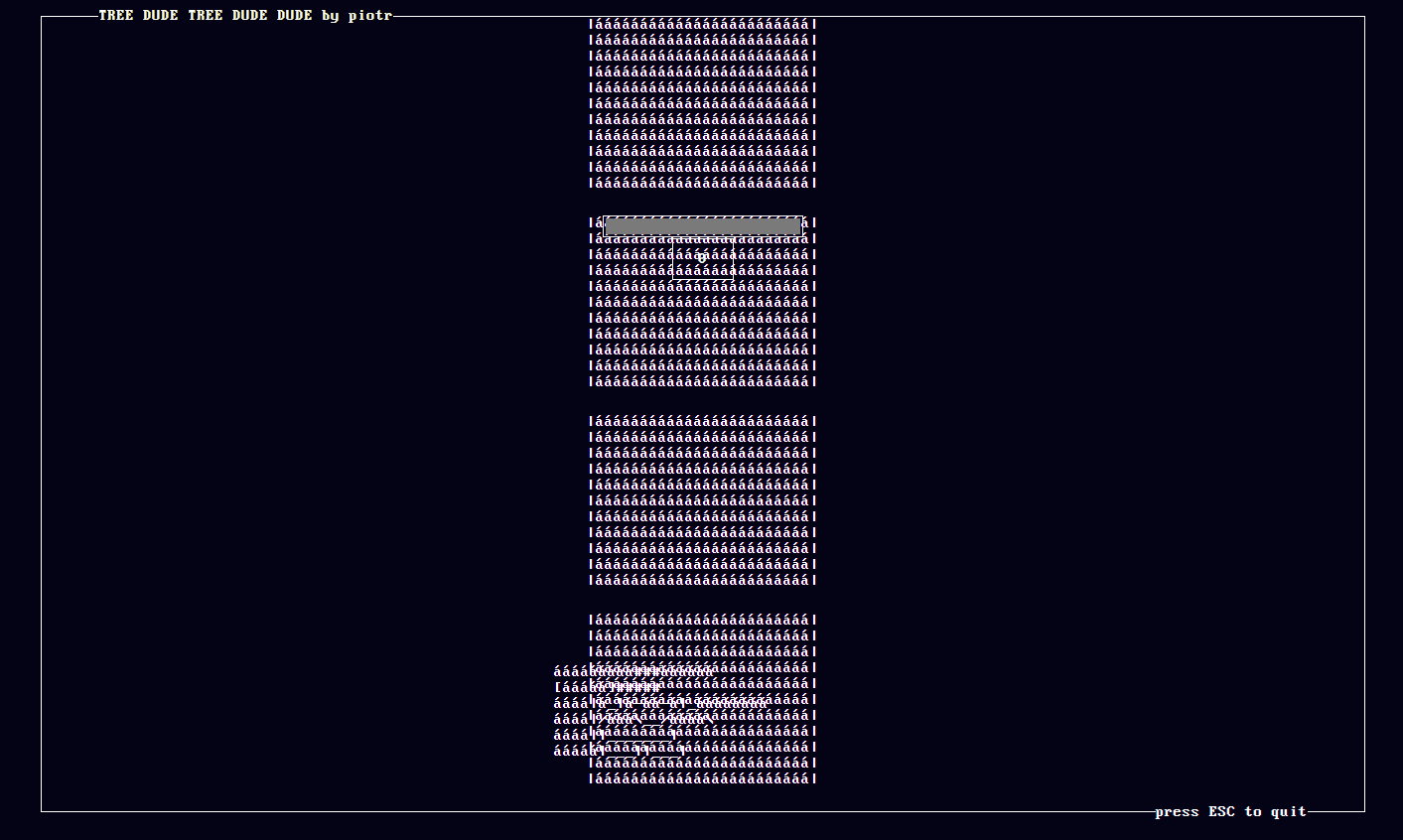


Ilustración 7: vista del error al aplicar la fuente

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

## 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.

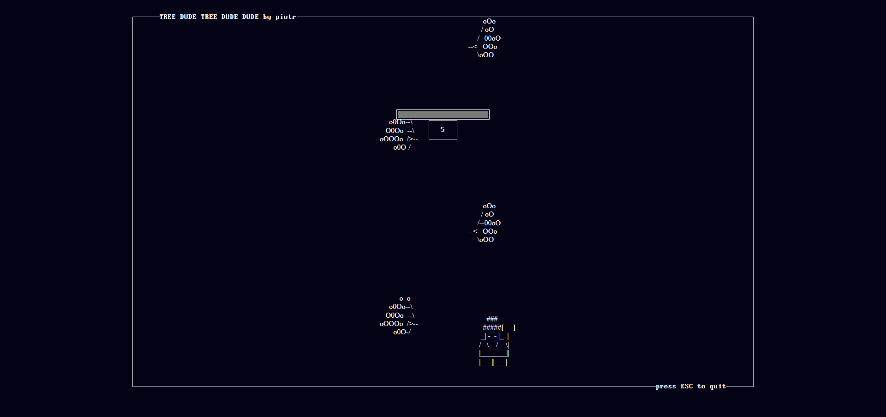


Ilustración 8: Juego sin tronco

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:

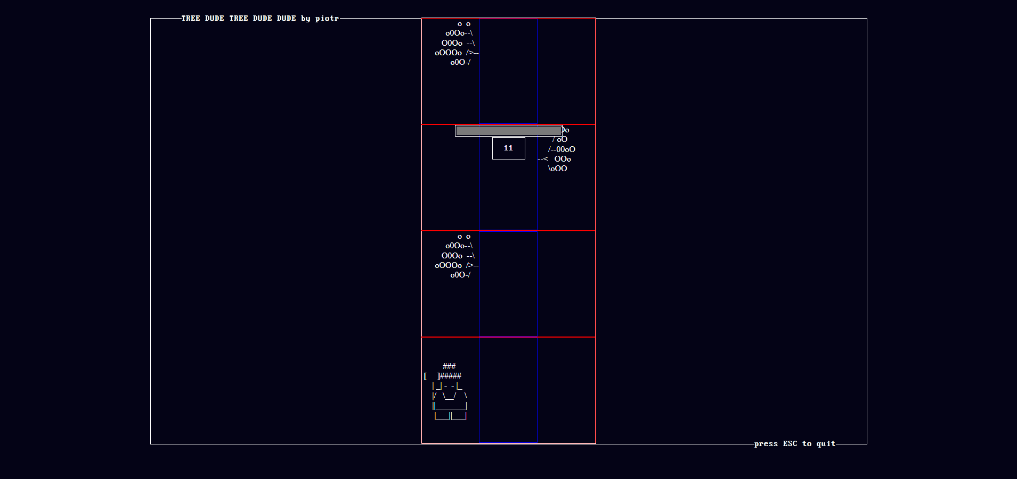


Ilustración 9: los bordes del containerTree, sus "hijos" y "nietos"  
\*Leyenda: borde blanco (no apreciable) = bordes del propio containerTree  
borde rojo = bordes de los div “hijos” del containerTree  
borde azul = bordes de los div “nietos” del containerTree

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;

}

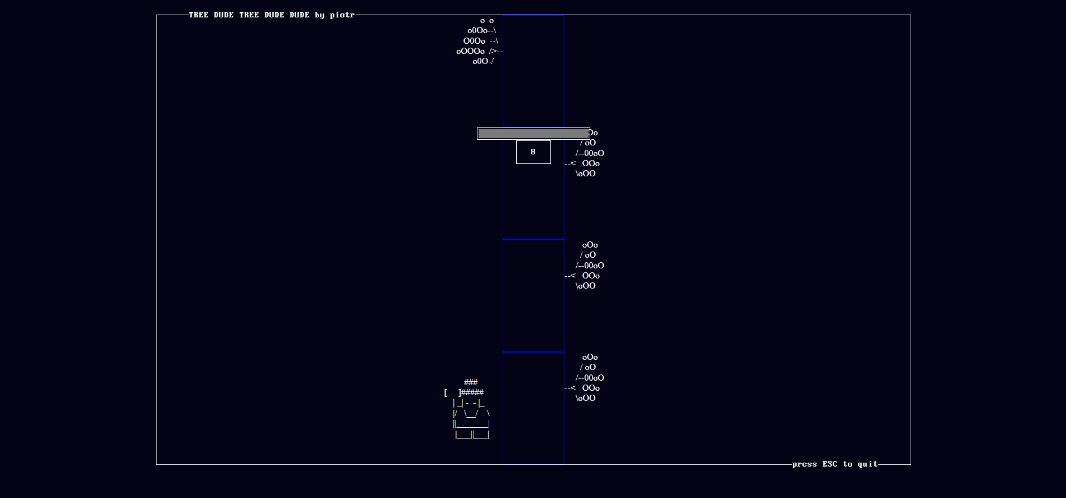


Ilustración 10: Bordes de los nietos col1

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.

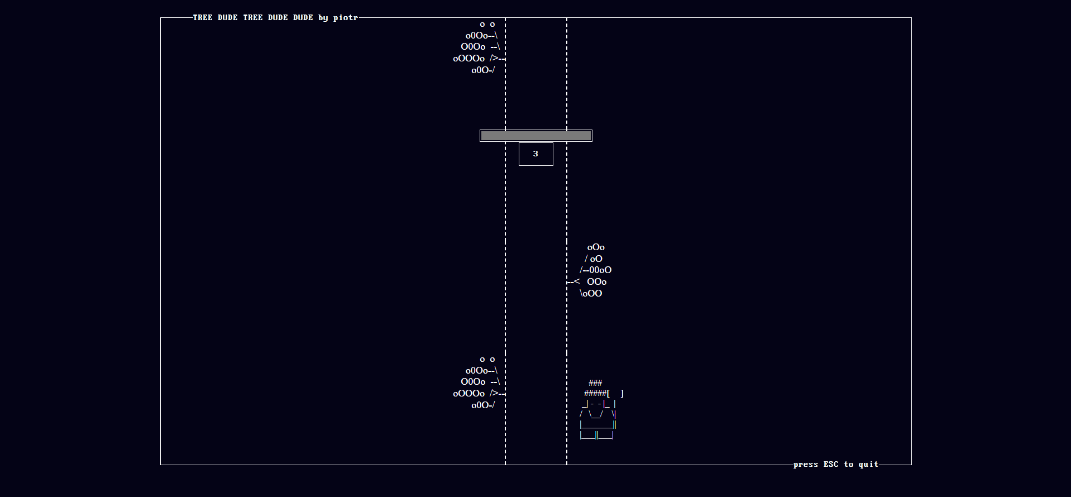


Ilustración 11: Nuestro tronco responsive

## 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 eranone, 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

# 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 soundStatusestá 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 yvolumen 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.

## 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

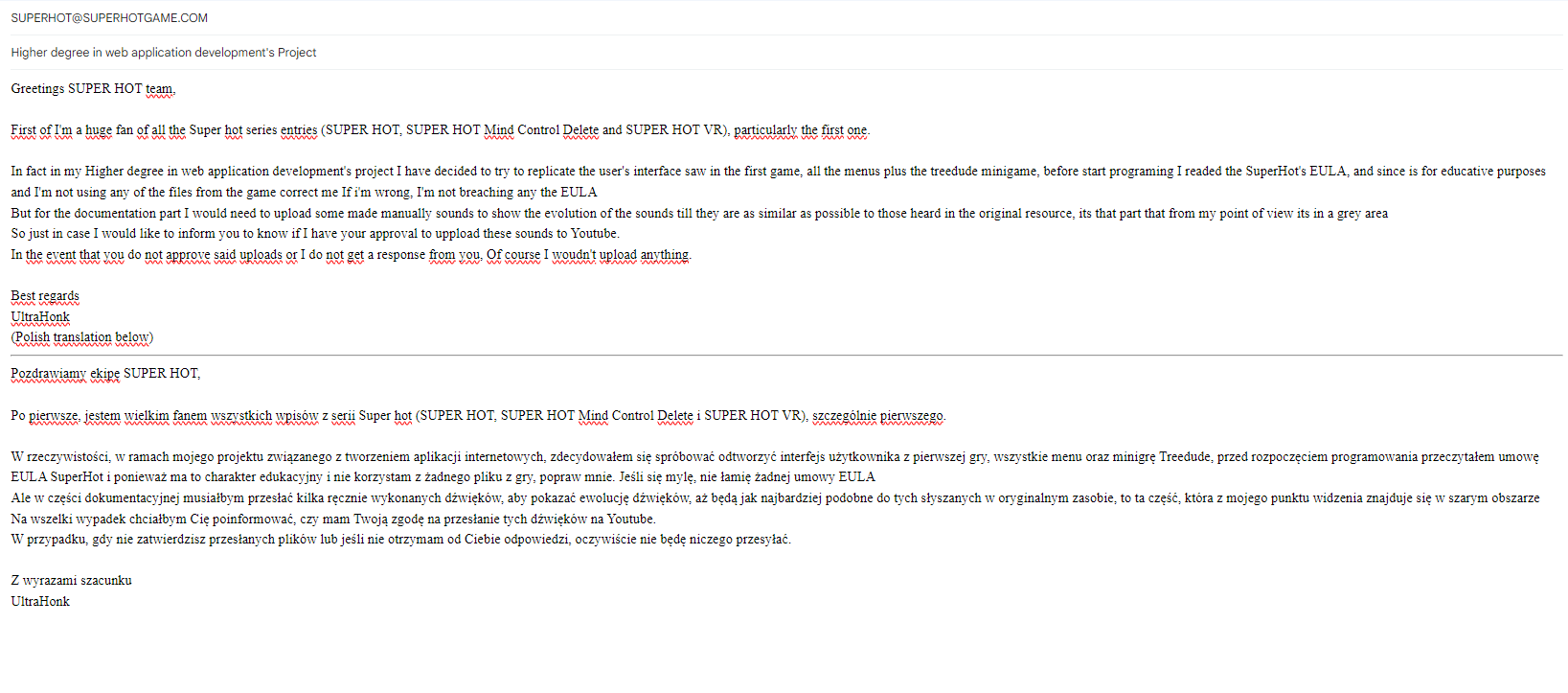


Ilustración 12: Imagen del correo enviado al equipo de SUPER HOT

## 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;

}

## 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.

## 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)

## 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...

### 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

# 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.

## 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

## 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

## 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() .

## 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: Piosreferencien 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.

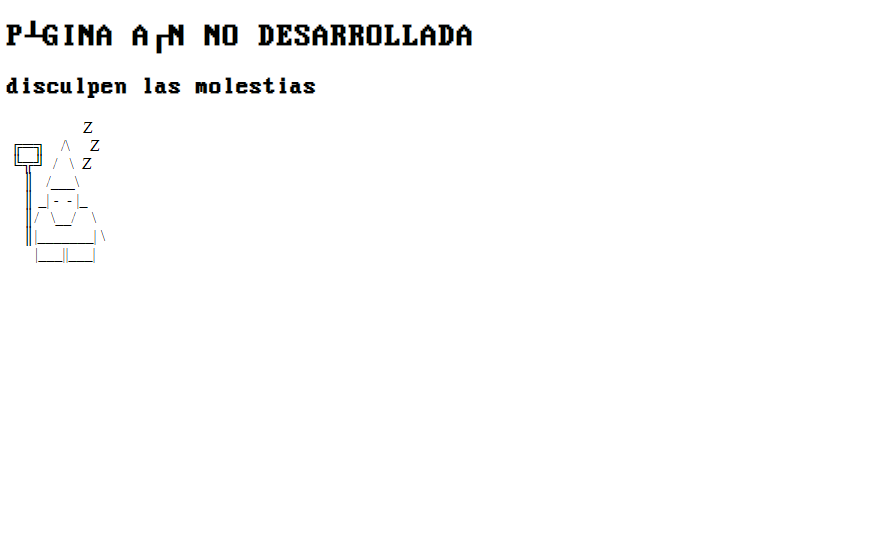


Ilustración 13: Muestra de cómo se ven representados los acentos,

Lo siguiente que se realiza es alinear los elementos de texto al centro, siguiendo el modelo del [index.html](#_PANTALLA_DE_INICIO) 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.

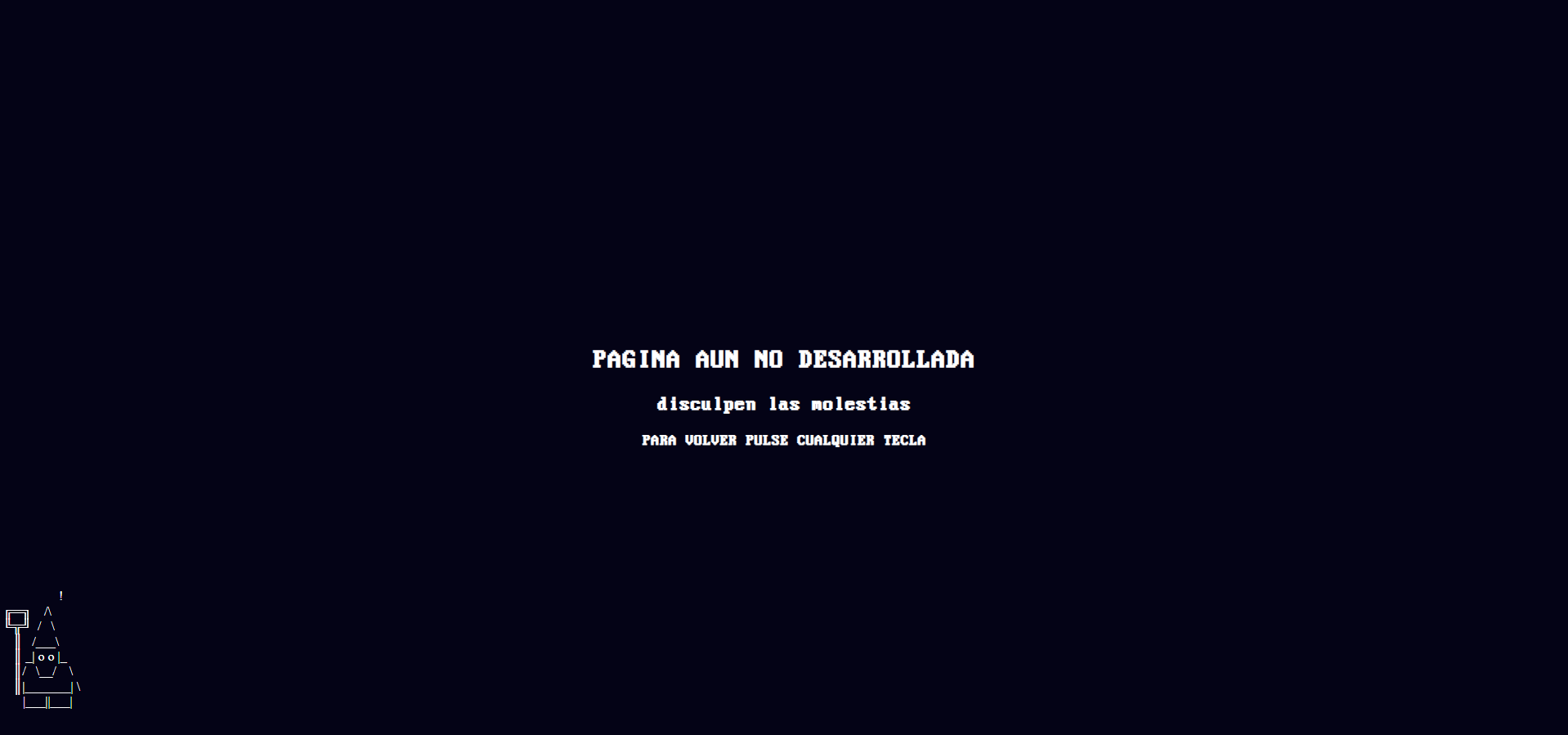


Ilustración 14: Resultado final de nuestra pantalla de "error"

# 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.

## 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 interruptorcon 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.

### startTime

La función de startTime tendrá un parámetro, el tiempo, volviendo a usar la función de [intervalos](#_*_You_called) 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. 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. 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.

## OBTENIENDO ENERGÍA

Usando de base el código de [loseEnergy](#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

# 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.

## 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 alertsque 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](#enlaceMozilla) probamos de otra manera, sin embargo, no se detecta el elemento browser.

## 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.

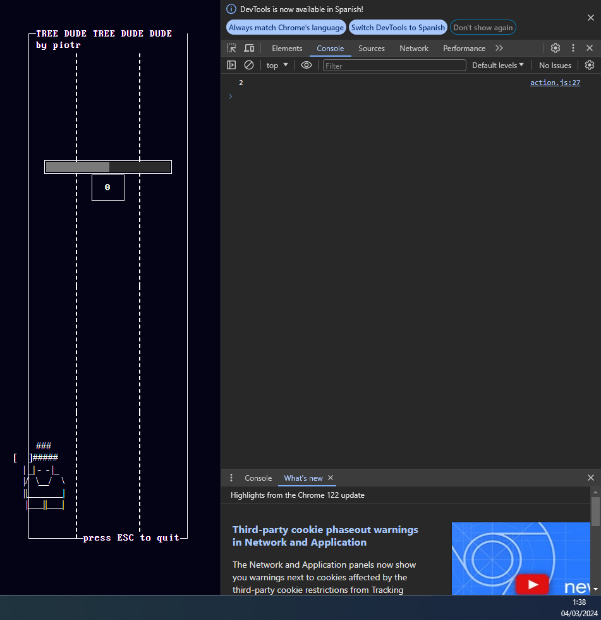


Ilustración 15: Página sin recargar la página (previo a la creación del item)

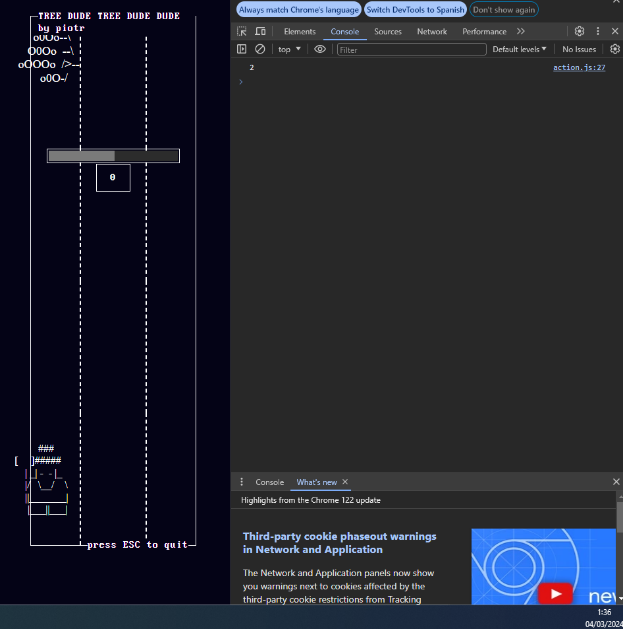


Ilustración 16: Página post recarga (después de la creación del item)

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 truese 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.

# 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.

# PRUEBAS en producción

## 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

## 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

CONCLUSIONES

En contraste con la 1ª evaluación no se pudo avanzar tanto, sin embargo, se consiguió llegar a realizar un mínimo jugable, por la parte del talador aún le faltarían ciertos elementos para mejorar su calidad: cambiar los alerts por defecto por mensajes personalizados, ir aumentando la velocidad de pérdida de vida progresivamente cuando “suba” de nivel y principalmente hacer más responsives las diferentes páginas sin que las mismas se destruyan. Para el próximo trimestre lo que se tendrá que lograr sería dar los últimos retoques al talador y comenzar con la codificación de typingTest por la parte desarrollo de entorno cliente, por otro lado, para el trabajo de fin de grado será necesario implementar las diferentes asignaturas del grado: Un pago por la parte de blockChain, APIS para obtención de datos de una bbdd por parte de desarrollo en backend y “diDoom” por parte de desarrollo de interfaces el cual, aunque ya se encuentra dentro de los archivos aún no es accesible desde el navegador. Nos quedaron en el tintero las siguientes tareas:

1. forzar tamaño fijo para que no se desborde o se contraiga afectando a la vista del juego
2. creación visual de la intro
3. Implementar aumento de velocidad, así como el mayor gasto de vida
4. Aplicar efectos de sonido en el menú
5. Implementar música y efectos de sonido (juego)

Debido a falta de tiempo y conocimientos hubo una serie de errores que aún no se pudieron solventar los siguientes errores

* Al redimensionar el menú y hacerlo más pequeño la barra de localización se mueve al tener como referencia, al contrario que las p el tamaño de la ventana.
  + Una posible forma de solucionarlo sería haciendo de su posicionamiento absoluto, pegarlo abajo del todo en la misma ubicación... arreglado más o menos
* Al cambiar el texto en el div summary el tamaño del div cambia, se debería de poner un width fijo y que no se expandiera más, pero de momento se hace la vista gorda.
* Si son pulsadas al mismo tiempo las teclas de izquierda y derecha en el minijuego de treedude es posible que el personaje se “bugueé” y esté debajo de una rama sin morir
* En el juego del talador al redimensionar la pantalla a anchos menores en el navegador de firefox los sprites (en concreto el del talador), textos y otros elementos se deforman y destrozan

Llegados a este punto el acabar el test de mecanografía y el menú al completo, no parece que sea posible, más conseguir finalizar treedude de una manera óptima y una base del test de mecanografía es una meta que aunque apurados quizás sea posible alcanzar

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

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/>