

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 11 de Diciembre de 2023

IDA DE OLLA

Día 1: Se comienza con dos ideas a barajas: Tree dude.exe o un test de tipografía en línea. De momento se opta por el estimulador de reflejos mecánicos, veáse tree dude.exe.

Se comienza con la creación de una pantalla de inicio. con el siguiente árbol de directorios y ficheros: un html llamado index.html, un fichero css llamado style.css y un fichero .js llamado action.js

Como primera idea de introducción a este proyecto, comenzaremos colocando un mensaje de "PRESIONE CUALQUIER TECLA" en medio de la pantalla que parpadee en medio de la pantalla del usuario.

Tras media se consigue colocar el texto en la posición querida tras trastocar el css alineando el texto en el centro e introduciendo un padding top de 45vh, vh implica la altura relativa de la altura del navegador.

Con el texto ya colocado se añade un color de fondo apagado un tono de azul cercano al negro, y se ilumina el h1 con un text shadow amarillo... No obstante queda algo corto, por lo que se decide a alumbrarlo de una manera adicional.

Se duplica el h1 original con las mismas propiedas, pero con una clase adicional .lightning con las propiedades position absolute y un color amarillo en vez del blanco de la fuente original, con todo esto se consigue colocar verticalmente alineado con el h1 original, pero no horizontalmente, tras 15 minutos de roturas de cabeza se decide dar el estilo de lightning a un div donde meter el h1 duplicado, colocamos el div por encima del h1 original y se le aplica al div un 100% de ancho (weight) tras esto se consigue el éxito... Si no fuera porque los estilos de texto de bienvenidor sobreescriben los de lightning, esto obliga a la creación de dos nuevas clases.

original para el h1 original y sombra para el h1 duplicado, donde se le dará el color blanco y el text shadow amarillo a original y la fuente amarilla junto el difuminado de 20px. Para darlo más sustancia se le da un font-weight bold

Para terminar de "adornar" la pantalla de inicio se crea un último div encima de los dos primeros con un position absolute y un left de 10vw siendo vw en relativo al ancho de la ventana y se hacen visualizar los bordes izquierdo y derecho en color blanco para destacar en el fondo oscuro, así se da comienzo a la animación de las letras de introducción

Para realizar la animación de las letras primero se obtiene la longitud de uno de los h1 ya que ambos tienene la misma longitud, una vez que se tiene la longitud creamos un array donde guardaremos los intervalos de desaparición de cada letra, para esto se intentó hacer uso de una función anónima, no se pudo.

Se cambia el enfoque a realizarlo con un bucle que se reproducirá el nº de veces que i marque y con cada iteración se colocará en la posición marcada por j que irá aumentando hasta alcanzar el valor de i un número aleatorio entre 5000 y 1000, este será el intervalo con el cual se activará el parpadeo.

En un principio la idea era la de tratar de acceder a cada letra del string y cambiarle el color de manera directa... Cosa que era imposible, por lo que para hacerlo se tuvo que dar una clase a cada letra del 0 al la longitud de la cadena y ahí alterar sus propiedades.

Para probar la el setInterval, primero se probó a desde la propiedad de setInterval, llamar a una función con parámetros, ya que algunas propiedades de llamada de funciones parecen tener comportamientos equivocados y erróneos a la hora de ser llamadas.

Como en otros casos esto resultó dar error. Para solucionarlo se tuvo que hacer uso de una función anónima que llamaría a la función deseada con los parámetros especificados y en ese momento ya teníamos un setInterval con parámetros.

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 for in 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 de la cadena sin espacios ya que los mismos no eran contados en la animación. Para sacarlo se añadió al h1 de clase original, el id modelo, la cosa se complicó al usar etiquetas spam 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 texto h1 poniendo el 3º en la misma posición pero oculto, 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.

Para no rizar el rizo más de lo necesario se concluyó eliminar el text shadow al no tenerlo ambos podría causar problemas a la hora de modificarlo, por lo que temporalmente se elimina.

Cambiamos 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, dejaremos esto aparcado hasta saber obtener el padre de un elemento en ese entonces en función de si su padre es original u sombra alternará entre dos colores distintos

Para la versión 1 ambos cambiarán al mismo.

Una vez se dió 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 menu. La tarea de ese momento consistirá en mediante js al pulsar cualquier tecla te envie a la pagina siguiente. Primero agregamos el event listener al body

document.body.addEventListener("keydown", function () {

}.

A continuación tocaría averiguar como acceder a la nueva página usando javascript, con la siguiente línea se cambia la ruta de ejecución:

window.location.replace("menu/index.html");

Con esto se dió por finalizada la versión 1 de la pantalla de inicio.

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. En la ~~v1 solo habrán 3 opciones: /.. Para volver atrás, treedude.html para acceder al juego al primer juego y typingTest.html para acceder al 2º.~~

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 como modelo

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, dividiendolo 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 de abajo se decide reducir la altura del div padre a 90vh y generar otro div de la misma anchura pero de 7vh sin margénes 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. Para poder visualizar temporalmente el nuevo div le se le da el fondo white.

RESULTADO:

Se le aplica el borde blanco igual que a los demás y se continúa.

Se quitan los colores de fondo y se le 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>

Lugo 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.  
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 lo haremos 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

~~Ya en el segundo script igualamos el valor interno del html del id p time a valor pasado por el intervalo de la función timePasses cada segundo o 1000 milisegundos.~~

Desde el script de actions creamos 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 la función timePasses, es decir, la hora concatenada con los minutos.

El siguiente apartado que atacamos fue el de la selección, y para ello creamos 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, veáse pegados

De manera temporal en actions.js se comienza a crear ~~tres~~ 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 I elementos donde se juntarán, el de clases y el de selección en el array bidimensional globalSelector en la posición de I.

Una vez ya fue creado el array global de modelo preparamos 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 cual es la opción que está activada vease la casilla globalSelector[i][1] == true.

Una vez se obtiene la opción activada deberemos desactivarla, esto lo haremos con 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 tenemos la posición nueva que deberá ser activada y aún tenemos el modelo desactualizado, eso lo solucionamos con un bucle.

Recorremos el array global desde 0 a la longitud del mini array focus, si i==j entonces pondremos a true el modelo, de lo contrario lo pondremos falso la posición [j][1]. Por último usando el modelo volvemos a buscar el valor activado y mediante jquery le añadimos la clase selected a los elementos con la clase de la posición [i][0], nos llevamos las funciones al directorio de additionalScripts y creamos el fichero Iselectores, desde el action llamaremos a optionUp y le deberemos 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

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 tendríamos que incrementarlo, y cuando el valor sea 1 menos que la longitud del array (vease el último del array), lo convertiremos 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 deberemos 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 prebio en la 1ª ejecución desde action igualaremos el valor interno de html del 3er div por el valor de /.. que es por el momento el valor por defecto al inciar 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 incluimos 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 como 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 respectivamente

a continuación modificamos 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.

Crearemos 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 nos vemos obligados 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 llendo directamente al directorio importante 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 lo copiamos 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

Se descubre en Github que alguien hizó 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 iddle 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 si ponemos más espacios en la derecha se aleja más, por lo que se decide hacer más anchos los iddle, tampoco funciona volvemas a la idea original en el cutingleft añadir más espacios, solo que colocar el muñeco en el flex-strart, 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, ambo sprites estén alineados.

Se reintenta con el de la izquierda hasta conseguir el resultado deseado.

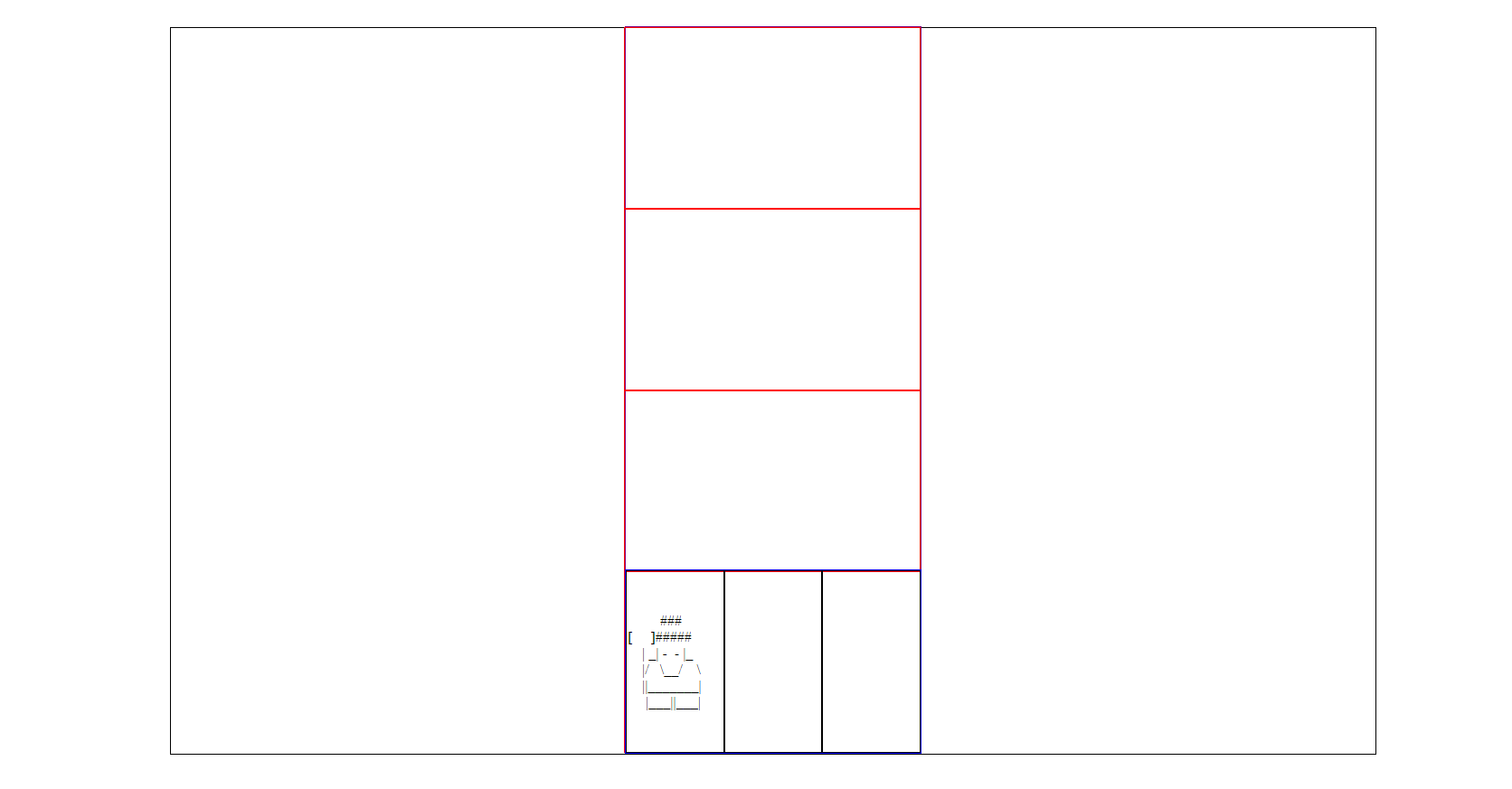
Antes de continuar con los sprites vamos a diseñar la ventana donde se mostrará todo.

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

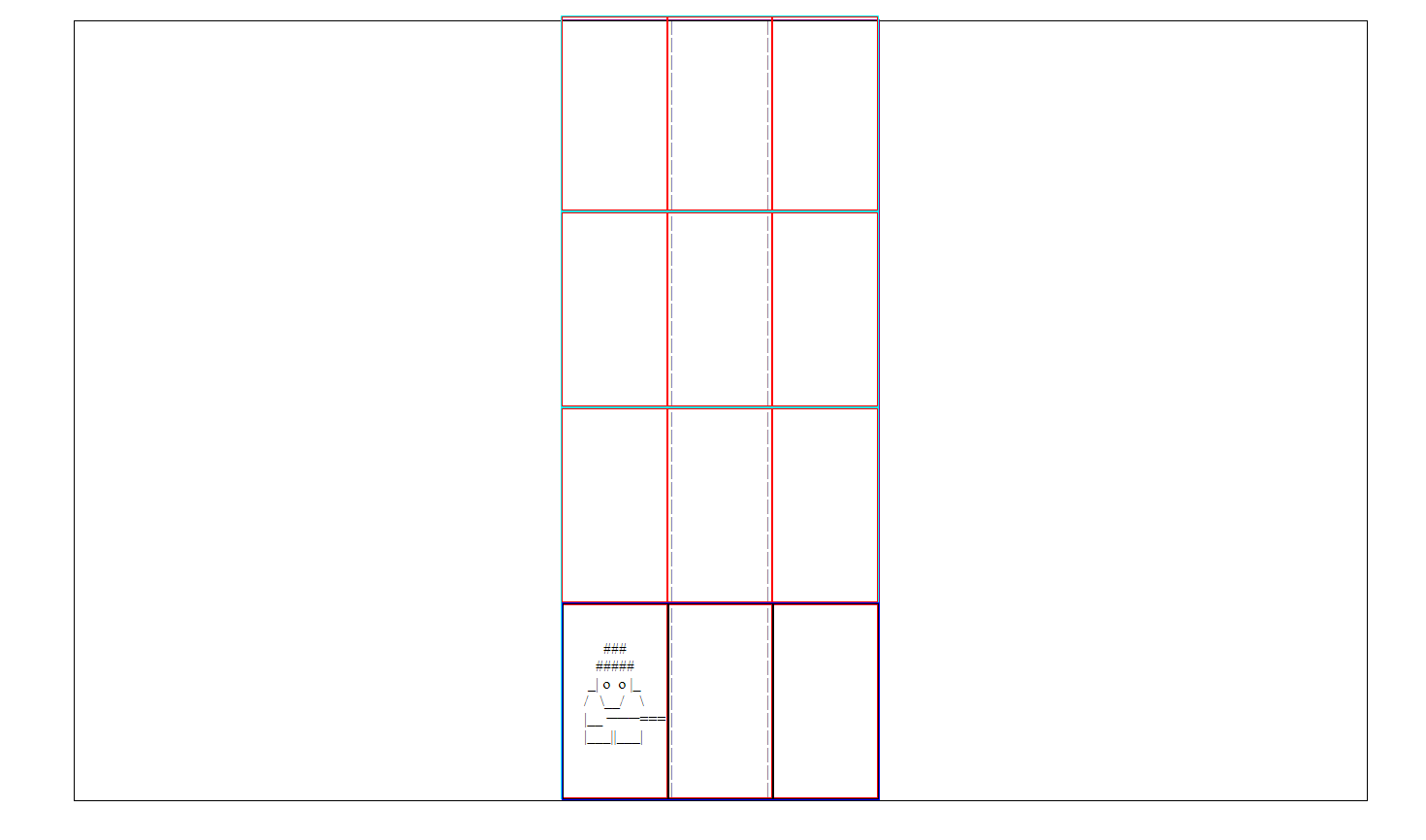
Primero se coloca el containerDude quien lleva la posición del leñador, lo colocamos abajo del todo en medio, con position absolute para que no ocupe un espacio y ese mismo pueda ser ocupado simultáneamente por otro.

Ahora toca colocar el containerTree, una vez tenemos una colocación temporal se continúa con los sprites



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



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.

Ya está terminada la animación del leñador hecha ahora toca generar la "copa" del árbol aleatoriamente, vease 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 donde 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 :D.

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 haremos un fichero extra.

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

Vamos ha hacer 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(. Se comprueba y funciona) después se llama a la función de pintar árbol para que teniendo en cuenta la cadena bidimensional pinte el árbol actualizado.

Lo siguiente que realizaremos 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" revisamos si el array del modelo del árbol[longitud array][n] y si la posición del leñador[n] en ambos se encuentra un 1 simúltaneamente 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 la colocaremos en la posición del leñador y borraremos la rama.

Primero usando de base el código usado para dibujar al leñador en la izquierda o la derecha lo usamos 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 sobreescribimos 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 conseguimos el resultado deseado por lo que probamos 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 venana no se destroce la estructura del juego

-------

TAREAS:

1. Creación estética de la pantalla de inicio (hecho)
2. Generación de números aleatorios (hecho)
3. Implementación de la animación de las letras que desaparecen en intervalos aleatorios (hecho)
4. Creación estético del menú principal (hecho)
5. Crear interacción menú principal con teclado para quitar el ratón (hecho)
6. Creación de los gráficos v1 del leñador (mediante ascii) (hecho)
7. Implementar los gráficos dentro de la ventana (hecho)
8. Implementar movimiento del leñador (hecho)
9. Implementar animación de corte (hecho)
10. Generación de las ramas de manera aleatoria (hecho)
11. Implementar muerte (hecho)
12. Mostrar puntos y crear puntos
13. Poner textos de la ventana
14. salir, (salir se puede en cualquier momento)
15. Aplicar estética al juego
16. forzar tamaño fijo para que no se desborde o se contraiga afectando a la vista del juego
17. Opción de volver a jugar
18. guardar puntos y mostrar ranking al final del juego, si la puntuación está entre los 10 primeros se guarda
19. Implementar pantalla de muerte

----------

1. creación visual de la intro
2. Implementar cuenta atrás de la vida y la recuperación de la misma
3. Implementar aumento de velocidad, así como el mayor gasto de vida
4. Aplicar efectos de sonido (al navegar por las opciones)
5. Implementar música y efectos de sonido (juego)

Para dudas se utilizó:

<https://www.w3schools.com/>

<https://www.geeksforgeeks.org/>

<https://es.stackoverflow.com/>

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

<https://trellat.es/texto-parpadeante-con-javascript/>

arrays multidimensionales

<https://www.freecodecamp.org/news/javascript-2d-arrays/>

para los ascii

<https://theasciicode.com.ar/>

funcionamiento del flex

<https://flexboxfroggy.com/>

---

<https://github.com/fr0zn/TreeDude>

—-

definiciones más concretas para gente que no entiende conceptos:

<https://es.wikipedia.org/>

Y

<https://www.tokioschool.com/noticias/sprite-videojuegos/>

Y

LIBRO DE BUGS

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.