Miguel Ángel Gil Martín

Grupo:3.3

Memoria Web

Índice:

Contenido

Introducción:

Mi proyecto consiste en un mundo formado por una cuadricula bidimensional, los cuadrados pueden ser: tierra, agua, césped o mineral de clics.

El juego consiste en minar los clics del mundo, para ello puedes usar los picos que puedes comprar en la tienda.

Controles:

Movimiento:

W,A,S,D o Arriba, Abajo, Derecha, Izquierda.

Tienda: E.

Mutear o desmutera la música: M

Seleccionar ítem del inventario:

1,2,3,4,5,6

Minar: clic izquierdo del ratón (mantener pulsado y dentro del área de influencia del jugador).

Principales clases y scripts del proyecto:

Todo el juego se pinta en un canvas web, ese canvas se localiza en el archivo index.html.

En este documento también encontramos el importe de todos los demás documentos que conforman el proyecto.

Code.js:

En este documento se establecen algunas de las variables globales mas importantes del juego, como puede ser el dentaTime o el contexto(ctx). También es donde se lleva el control del loop principal del juego y por lo tanto el que se encarga de llamar al resto de clases y funciones.

Assest.js:

En este script hay unas funciones que se encarga de cargar todos los archivos visuales del juego de forma asíncrona al proceso principal y almacena los archivos en variables publicas.

(aunque el sistema cargue de forma asíncrona los documentos, el loop principal de code no se inicia hasta que se comprueba que todos los documentos están cargados).

Animation.js:

En este escript se encuentra la clase Animation, esta clase sirve para que ella sola sea capaz de seleccionar el frame de una animación o el fragmento de un atlas. Esto se hace mediante las funciones: print_a_frame y anímate.

```
//esta funcion la creé para pintar items que no tenían animacion como las tools
print_a_frame = function(ctx,world_position,sprite_position,space_between_frames)
```

```
animate = function(ctx,position,animation_number,statico){
```

Para poder iniciar esta clase y poder usarla correctamente hay que llamar a su Start() y ajustar los parámetros de: imagen(tiene que ser un atlas), el numero de animaciones(columnas), el numero de frames(filas), las dimensiones del Sprite y los fps a los que quieres que se reproduzca la animación.

```
//para inicializar la clase se necesita la imagen, la cantidad de animaciones
Start = function(img, n_animatios, n_frames_per_animation, dimensions,fps){
```

Player.js:

En este script se encuentra la clase player, desde aquí se controla al player: movimiento, minar, abrir el inventario.

Esta clase contiene referencias al inventario, a la camera y a los controles.

Esta clase no tiene funciones aparte de las principales de Start(), Draw() y Update().

Invenory.js:

En este script se encuentra la clase Inventory, esta clase se encarga de administra el inventario y la tienda, tiene 2 arrays: ítems y shop, en ellas se almacenan las tools disponibles.

El control de esta clase se lleva desde la función Draw() que es llamada desde el player.

Tools.js:

En este escript se encuentran las herramientas, hay 2: hand y Pick.

La clase hand la tiene el jugador por defecto y la clase pick se tiene que comprar en la tienda y se almacena en el inventario.

Estas 2 clases son muy similares, para usarlas, las 2 disponen de una función llamada working:

```
Working = function(deltaTime){
```

Esta función se encarga de realizar un trabajo, cuando lo acaba, devuelve un true.

Las demás diferencias entre las 2 clases tienen que ver con la durabilidad de las herramientas y con el aspecto gráfico, puesto que hand no dispone de ninguna de estas cosas.

World.js:

Este script contiene el mundo y los tiles que lo constituyen:

En la clase world: se genera el mundo mediante la función build_world

```
build_world: function(){
```

En el Start() puedes mediante unos parámetros ajustar el porcentaje de tierra, agua y clicks que quieres que haya en el mundo, además del tamaño del mismo:

```
Start: function(size, amount_of_clicks,amount_of_water,amount_of_green){
```

Como decía el mundo está constituido por tiles, estos tiles se almacenan en la variable tile_world y las funciones de Update() y Draw() llama a cada una de esta baldosas para actualizarlas.

En la clase Tile se almacena en el caso de que haya clics, la cantidad de clicks, su gráfica, su posición real y también la posición con respecto a la posición del jugador(la camara) y también, esta clase contiene una serie de funciones que se llaman cuando se genera el mundo que expande su tipo a las baldosas vecinas, la función de esto es que no se generaran cortes de biomas bruscos.

```
transfer_clicks = function() {
    for(var i = 0;i < this.near_Tiles.length;i++){
        if(this.near_Tiles[i] != null){
            if(this.clicks > 500 && this.clicks/2 > 100){
                this.near_Tiles[i].clicks = this.clicks/2;
                this.near_Tiles[i].walkable = true;
                this.near_Tiles[i].kind_of_tile=4;
                this.near_Tiles[i].select_kind_of_Tile();

            }
        }
    }
};
transfer_green = function(){
    for(var i = 0;i < this.near_Tiles.length;i++){
        if(this.near_Tiles[i] != null){
            if(this.kind_of_tile == 2){
                if(this.near_Tiles[i] != 0){
                    if(Math.floor(Math.random() * 100) < 40){
                        this.near_Tiles[i].kind_of_tile=2;
                        this.near_Tiles[i].walkable = true;
                        this.near_Tiles[i].select_kind_of_Tile();
                    }

                }


            }
        }
    }

};
transfer_water = function(){
    for(var i = 0;i < this.near_Tiles.length;i++){
        if(this.near_Tiles[i] != null){
            if(this.kind_of_tile == 0){
                if(Math.floor(Math.random() * 100) < 20){
                    this.near_Tiles[i].kind_of_tile=0;
                    this.near_Tiles[i].walkable = false;
                    this.near_Tiles[i].select_kind_of_Tile();
                }

            }
        }
    }

};
```

Camera.js:

Contiene la clase camera, esta clase solo contiene el constructor y el Update(). En esta clase se genera una posición relativa al jugador para enviársela a todos los objetos estáticos del juego, que son los tiles del mapa.

Player_inputs.js:

Contiene la clase Controls, y sirve para poder administra los controles, básicamente asocia los controles con las acciones que se van a realizar si se accionan, de esta forma si se hiciera un

menú para cambiar los controles sería mucho más fácil, también me sirve para poder asociar mas de un control para una acción:

```
class Controls {
    up ={ b1: KEY_UP, b2: KEY_W};
    down ={ b1: KEY_DOWN, b2: KEY_S};
    left ={ b1: KEY_LEFT, b2: KEY_A};
    right ={ b1: KEY_RIGHT, b2: KEY_D};
    settings = KEY_SCAPE;
    action = {b1:KEY_E , b2: KEY_E};
    right_bar={b1:null , b2:null};
    left_bar={b1:null, b2:null};
    item1 = KEY_1;
    item2 = KEY_2;
    item3 = KEY_3;
    item4 = KEY_4;
    item5 = KEY_5;
    item6 = KEY_6;
    build = null;
    shoot = null;
}
```

Conclusiones y problemas:

El principal problema han sido algunos controles como el de abrir la tienda o el de mutear la música, porque son botones que funcionan como swiches e intenté hacer que se accionen cuando levantas el botón pero, cuando haces eso, se queda en un bucle infinito y se está accionando y des accionando constantemente, por lo que lo he dejado para que se accionen cuando presionas el botón.

Mis conclusiones finales sobre el proyecto y la asignatura es que he aprendido un montón de web, especialmente de JavaScript , que nunca lo había tocado mucho. Ha sido una de las mejores asignatura de este año, y no solo por la materia sino por lo mucho que he podido experimentar con el código, al final se me ha echado el tiempo encima, porque pensaba conectarlo con firebase pero bueno, de eso he sacado que la próxima vez hago un juego mas simple o si lo hago así de complejo buscaré tecnologías que me ayuden ha hacerlo menos pesado.