TEC – Sede central de Cartago – Escuela de computación – Estructuras de datos (IC-2001) – G2

Anteproyecto Matrix / Proyecto 1

Jose Pablo Agüero Mora / 2021126372 - Katerine Guzmán Flores / 2019390523

I Periodo – 2022 – Profesor: Victor Manuel Garro Abarca

Estrategia para la resolución del proyecto:

Teniendo en cuenta el material de referencia que se nos ha entregado con respecto a Matrix, hemos analizado dichos códigos para entender una posible implementación de solución para el presente proyecto 1.

- Con respecto al código implementado con allegro:

```
while (hecho) {

    ALLEGRO_EVENT eventos;

    al_wait_for_event(colaEventos, &eventos);
    //al_clear_to_color(transparente);

    if (eventos.type == ALLEGRO_EVENT_TIMER) {
        if (eventos.timer.source == primerTimer || eventos.timer.source == segundoTimer) { // Esta parte cambiada
            al_clear_to_color(al_map_rgb(0, 0, 0));
            caracter[0] = GenerarRandom();
            //AQUI VA LO QUE PASARIA EN EL EVENTO DADO POR EL PRIMER TIMER
            dibujar(caracter, stack, x, y);
            y = y + 20;

            caracter3[0] = GenerarRandom();
            //AQUI VA LO QUE PASARIA EN EL EVENTO DADO POR EL PRIMER TIMER
            dibujar(caracter3, stack3, x2, y2);
            y2 = y2 + 20;
        }


        if (y >= 700) {
            y = 10;

            y2 = crearRandom();
        }
    }
```

En este ciclo principal se repite el recorrido de las hileras, donde se crea un nuevo carácter random en cada iteración, las posiciones de “y” varían, además de la principal función de esta sección la cual es dibujar. Se usa el primer timer para poder ralentizar el proceso global de la impresión.

Como se puede ver en la imagen, hay una sección repetida de dibujar, esto es debido a que de forma experimental intentamos añadir una columna de caracteres que apareciera de forma aleatoria en “y” cada vez que llegara al final.

La sección más importante (dibujar) va a funcionar de igual forma para todas las hileras, por lo que entendimos que para lograr el efecto matrix en toda la pantalla es necesario crear un algoritmo que administre de forma aleatoria pero uniforme las impresiones logradas por la función dibujar.

```
char actual[1];
void dibujar(char caracter[1], char stack[6], int x, int y) {
    actual[0] = stack[0];
    al_draw_text(fuente, al_map_rgb(0, 200, 0), x, y - 20, ALLEGRO_ALIGN_CENTRE, actual);
    actual[0] = stack[1];
    al_draw_text(fuente, al_map_rgb(0, 150, 0), x, y - 40, ALLEGRO_ALIGN_CENTRE, actual);
    actual[0] = stack[2];
    al_draw_text(fuente, al_map_rgb(0, 100, 0), x, y - 60, ALLEGRO_ALIGN_CENTRE, actual);
    actual[0] = stack[3];
    al_draw_text(fuente, al_map_rgb(0, 50, 0), x, y - 80, ALLEGRO_ALIGN_CENTRE, actual);
    actual[0] = stack[4];
    al_draw_text(fuente, al_map_rgb(0, 25, 0), x, y - 100, ALLEGRO_ALIGN_CENTRE, actual);
    actual[0] = stack[5];
    al_draw_text(fuente, al_map_rgb(0, 10, 0), x, y - 120, ALLEGRO_ALIGN_CENTRE, actual);

    al_draw_text(fuente, al_map_rgb(255, 255, 255), x, y, ALLEGRO_ALIGN_CENTRE, caracter);

    al_flip_display();

    desplazar(stack);
    stack[0] = caracter[0];
}
```

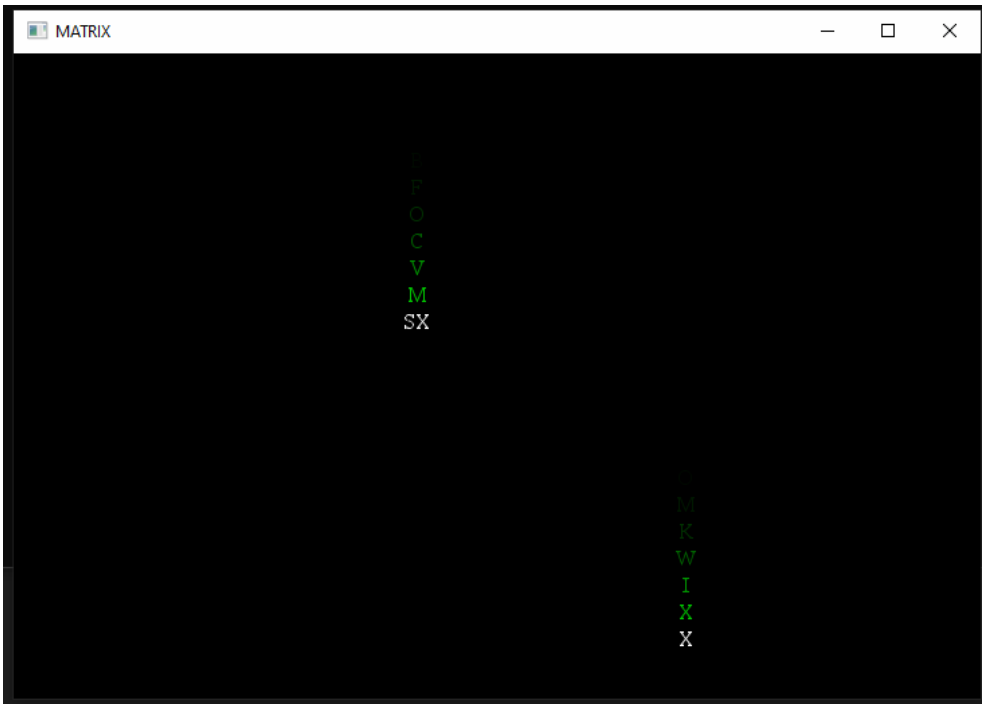
De esta forma, la función dibujar plantea una solución eficiente en la cual podemos inspirarnos para desarrollar nuestro programa. En esta se imprime en cada llamada el carácter aleatorio que se generó en el main en una letra blanca, al finalizar las impresiones este carácter ingresa a un stack de caracteres. En la siguiente llamada el carácter que antes se imprimió como blanco ahora presenta un color verde claro, ya que la función imprime el contenido de cada elemento del stack con ciertas características diferenciadas, pero manteniendo la posición original del carácter respectivo.

En cada llamada se pasan los elementos del stack un espacio hacia la derecha, al imprimirlos resulta en un efecto que simula un degradado de los caracteres en cada una de las iteraciones.

- Con respecto al código en ASCII:

```
void dibujar(hilera hilera[total]) {
    int coordenadaY;  /*lmacena temporalmente el valor de coordenada, es a esta variable que se le
    resta 1 en cada iteracion del ciclo para dibujar los caracteres que van arriba del mas reciente*/
    for (int i = 0; i < total; i++) { //Ciclo que alterna entre el total de hileras
        coordenadaY = hilera[i].Y;
        gotoxy(hilera[i].X, coordenadaY); //se posiciona en el caracter de mas abajo
        SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE), 10); //selecciona el color verde
        cout << hilera[i].caracter << endl;
        gotoxy(hilera[i].X, (coordenadaY-longitud));
        cout << ' ' << endl;
        //La hilera llegara al limite de coordenada Y=28
        if (hilera[i].Y < 28) {
            hilera[i].Y = hilera[i].Y + 1;
        }
        //Cuando Y esta en el limite, es necesario ir borrando uno a uno los caracteres de mas arriba a como la
        //hilera va bajando
        if (hilera[i].Y == 28) {
            gotoxy(hilera[i].X, (coordenadaY - (longitud- hilera[i].contador)));
            cout << ' ' << endl;
            hilera[i].contador = hilera[i].contador + 1;
                if (hilera[i].contador >longitud) {
                    hilera[i].Y = rand() % 20;
                    hilera[i].X = rand() % 120;
                    hilera[i].contador = 0;
                }
        }
    }
}
```

Al observar la implementación de matrix sin allegro se puede observar cómo al llegar al límite establecido de la pantalla, se genera una nueva posición aleatorio de X y Y. Esta es la estrategia que se probó de forma parcial en la sección anterior con allegro, lo que resultó correctamente:



Esto demuestra a grandes rasgos que la implementación de la lógica dentro de la función dibujar (presente en el código de allegro) junto con la administración aleatorizada de la versión en ASCII puede significar la solución definitiva del proyecto.

Contemplando todos estos puntos estamos diseñando la estrategia del proyecto y de esta forma podemos dividir cada una de las secciones a programar en un cronograma que distribuya equitativamente el proceso de desarrollo.

Secciones a desarrollar:

- Programar la base para mostrar una ventana. Definir los límites que tendrá en X y Y.
- Desarrollar y probar una cola de eventos que esté bien definida con los respectivos timers. Esto con el objetivo de organizar la ejecución de las instrucciones según el orden requerido. Probar las posibles velocidades de ejecución según los timers implementados.
- Plantear un posible diseño del main con cajas negras (asumiendo el funcionamiento de funciones las cuales no se han desarrollado).
- Crear una función similar a dibujar para los caracteres en el stack temporal de elementos.

- Escribir una función que permita rotar los caracteres en el stack para que la función dibujar pueda imprimirlos con la misma posición en que fueron impresos por primera vez, pero que en cada iteración cambie su color hasta que el carácter ya no sea visible.
- Programar una solución similar a la usada en el matrix ASCII para aleatorizar las posiciones de las hileras una vez que lleguen al límite de la pantalla.
- Usar la solución anterior para lograr que se visualicen hileras en la mayoría de columnas de la pantalla.
- El siguiente paso es buscar que se reproduzca más de una hilera por columna en cada ciclo. Ya analizamos que esto se puede lograr alargando la hilera y pintando una sección negra de caracteres en medio para dar la ilusión de que ahora son dos hileras de caracteres.
- Terminar de probar la estructura del main que mantiene el ciclo infinito sin problemas relacionados con salir de los rangos establecidos o mostrar de forma incorrecta las hileras en ciclos avanzados.
- Una vez que se ha probado lo suficiente la versión final se procede a realizar la documentación externa correspondiente.

División del trabajo:

Con base en las actividades redactadas anteriormente se dividen las responsabilidades de forma equitativa:

Nombre de la actividad	Semana de desarrollo	Encargado/s
Reunión para analizar los ejemplos de código y discutir una estrategia potencial de desarrollo.	Primera semana	Katerine Guzmán / Jose Pablo Agüero
Probar un prototipo que demuestre la validez de la estrategia planteada.	Primera semana	Katerine Guzmán / Jose Pablo Agüero
Desarrollo del anteproyecto que resuma los puntos logrados en las actividades anteriores.	Primera semana	Katerine Guzmán / Jose Pablo Agüero
Programar la base para mostrar una ventana.	Segunda semana	Katerine Guzmán
Desarrollar y probar una cola de eventos relacionada con los timers.	Segunda semana	Jose Pablo Agüero

Plantear un posible diseño del main con cajas negras.	Segunda semana	Jose Pablo Agüero
Crear una función similar a dibujar.	Segunda semana	Katerine Guzmán
Escribir una función que permita rotar los caracteres en el stack.	Segunda semana	Katerine Guzmán
Programar aleatorización de posición en hileras.	Segunda semana	Jose Pablo Agüero
Usar la solución anterior para lograr que se visualicen hileras en la mayoría de columnas de la pantalla.	Segunda semana	Jose Pablo Agüero
Buscar que se reproduzca más de una hilera por columna en cada ciclo.	Principio de tercera semana	Katerine Guzmán
Terminar de probar la estructura del main (pruebas finales).	Principio de tercera semana	Katerine Guzmán / Jose Pablo Agüero
Redacción de documentación externa.	Principio de tercera semana	Katerine Guzmán / Jose Pablo Agüero