Instituto Tecnológico de Costa Rica

Escuela de Computación

Análisis y Diseño de Algoritmos

Profesor: Victor Garro Abarca

Proyecto 1.0 Efecto Matrix

Estudiantes:

Melissa Navarro Villalobos

2018086497

Samuel Piedra Araica

2019007537

Septiembre, 2019

Índice

Manual de Usuario

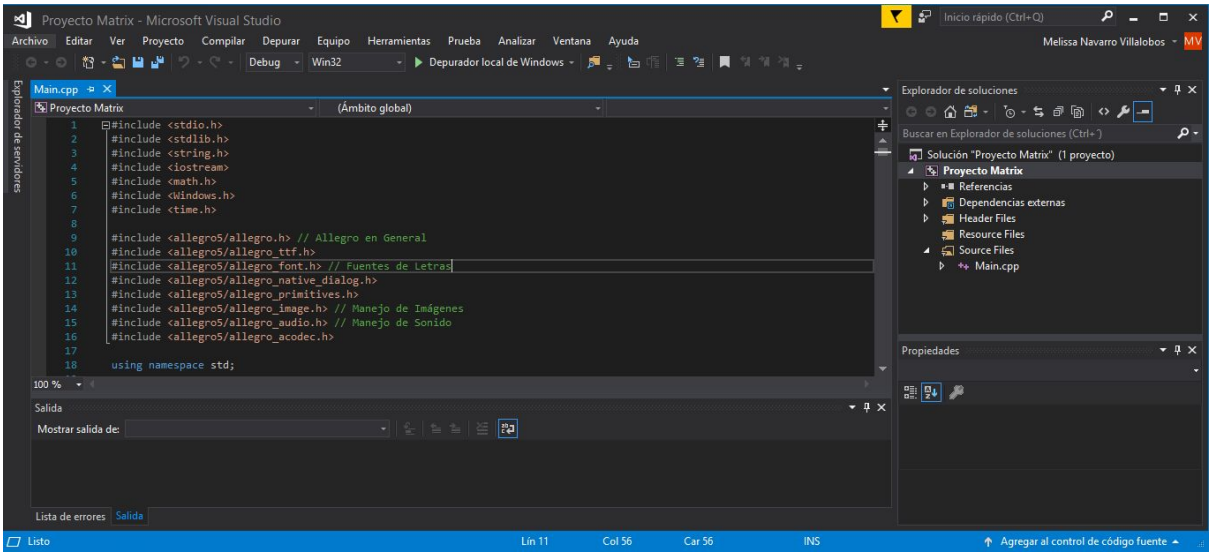
Descarga de proyecto

Mediante el link adjunto se podrá descargar la carpeta que contiene el Proyecto 1.0 Efecto Matrix.

https://drive.google.com/drive/folders/1ZtNjddNixsuIBaJGAK9Y57gJL3xubvnx?usp=sharing

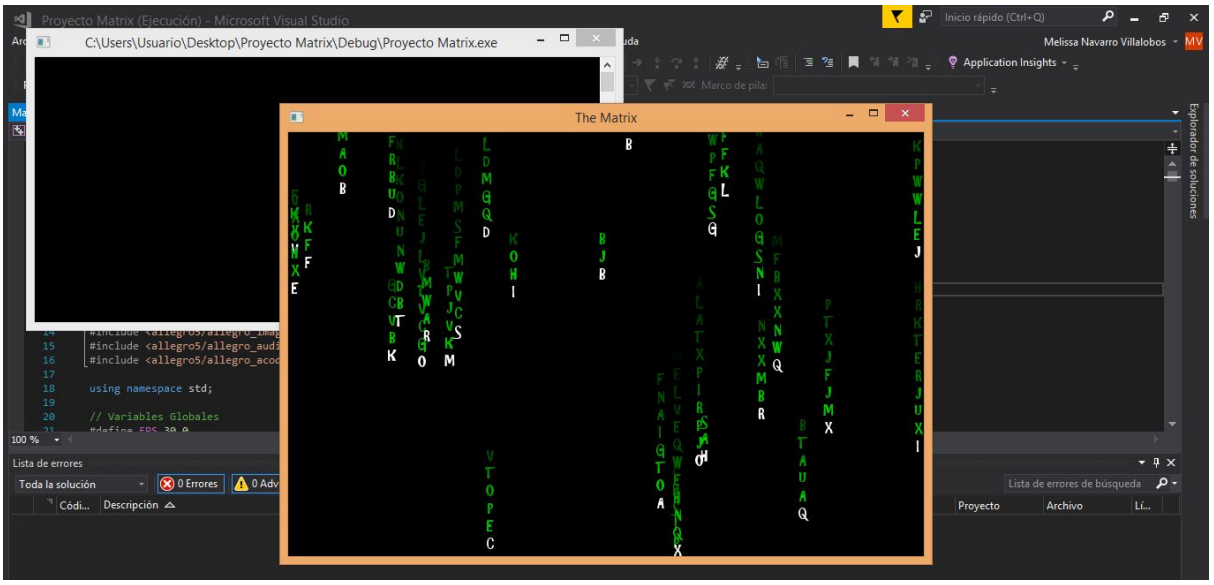
¿Cómo abrir el proyecto?

Dentro de la carpeta se encontrará un archivo llamado “Proyecto Matrix.sln” el cual deberá ser abierto en Visual Studio 2017



Una vez el archivo estando abierto, se procederá a darle click al Depurador Local de Windows (ícono verde)

El programa se deberá ver como lo indica la siguiente imagen



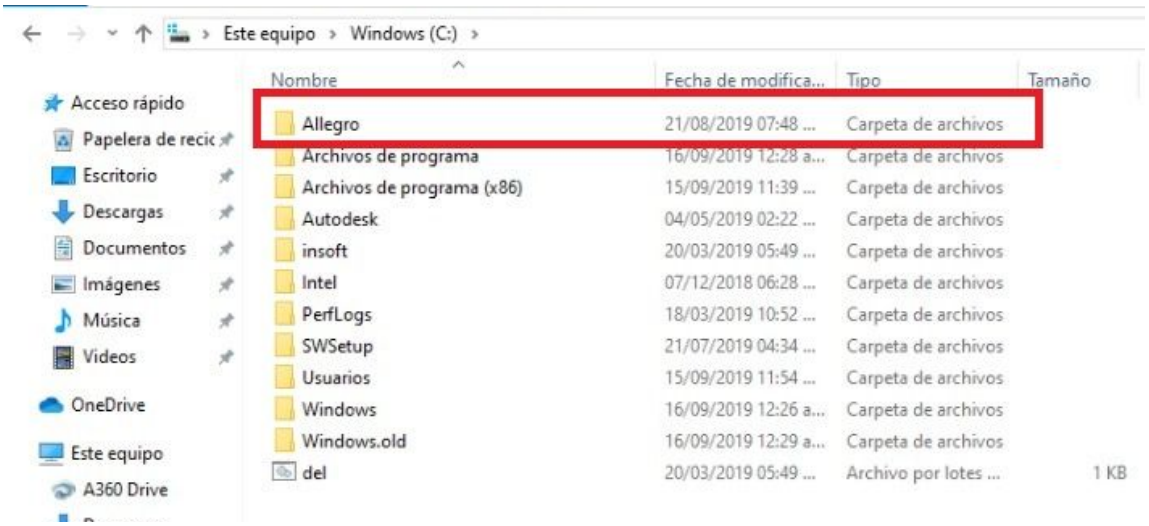
¿Cómo cerrar el programa?

Para finalizar así como cerrar el programa se utilizará la tecla ESC

Manual Técnico

Instalación de Allegro

La carpeta Allegro debe ser descargada y guardada en Disco Local (C:) como se muestra en la siguiente imagen



Importación de bibliotecas

Para poder utilizar Allegro se importaron las siguientes librerías

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <math.h>
#include <Windows.h>
#include <time.h>

#include <allegro5/allegro.h> // Allegro en General
#include <allegro5/allegro_ttf.h>
#include <allegro5/allegro_font.h> // Fuentes de Letras
#include <allegro5/allegro_native_dialog.h>
#include <allegro5/allegro_primitives.h>
#include <allegro5/allegro_image.h> // Manejo de Imágenes
#include <allegro5/allegro_audio.h> // Manejo de Sonido
#include <allegro5/allegro_acodec.h>

using namespace std;
```

Funciones generadas

Función para crear caracteres

Se encargan de generar un nuevo carácter aleatorio para posicionar en el extremo inferior de cada hilera.

```
// Declaración de funciones convenientes
long g_seed = 1;
inline int fastrand() {
    g_seed = (214013 * g_seed + 2531011);
    return (g_seed >> 16) & 0x7FFF;
}
char GenerarCaracter()
{
    char c = 'A' + fastrand() % 24;
    return c;
}
```

Descripción de cada función

- **inline int fastrand ( ):** Genera números aleatorios con gran rapidez.
- **char GenerarCaracter ( ):** Empleando la numeración de cada carácter en el formato ASCII regresa una letra aleatoria que se posicionará al final de cada hilera.

Funciones de control de hileras

Se encargan de generar, posicionar y dibujar las hileras en el display de Allegro.

```
void InicializarHileras(Hilera ConjuntoHileras[MaxHileras])
{
    for (int i = 0; i < MaxHileras; i++)
    {
        ConjuntoHileras[i].x = rand() % 720;
        ConjuntoHileras[i].y = rand() % 250;
    }
}
void DesplazarCaracteres(Hilera &Cadena)
{
    Cadena.Caracter8 = Cadena.Caracter7; // Desplaza los Caracteres dentro de una cadena
    Cadena.Caracter7 = Cadena.Caracter6;
    Cadena.Caracter6 = Cadena.Caracter5;
    Cadena.Caracter5 = Cadena.Caracter4;
    Cadena.Caracter4 = Cadena.Caracter3;
    Cadena.Caracter3 = Cadena.Caracter2;
    Cadena.Caracter2 = Cadena.Caracter1;
    Cadena.Caracter1 = Cadena.Caracter0;
}
```

```
103 void MostrarHileras(Hilera ConjuntoHileras[MaxHileras]) // Dibuja las hileras en pantalla
104 {
105     char CaracterActual[1]; // Buffer que almacena un caracter individual
106     for (int i = 0; i < MaxHileras; i++)
107     {
108         int X = ConjuntoHileras[i].x; // Variable que almacena la Coordenada X
109         char Caracter = GenerarCaracter(); // Se pide un nuevo caracter para imprimir al final
110         ContadorCaracteres++; // Incrementa el contador por cada vez que se genera una nueva letra
111 
112         CaracterActual[0] = ConjuntoHileras[i].Caracter0; // Almacena el caracter de la posición solicitada para luego imprimirlo
113         al_draw_text(Fuente, al_map_rgb(0, 200, 0), X, ConjuntoHileras[i].y - 20, ALLEGRO_ALIGN_CENTRE, CaracterActual);
114         CaracterActual[0] = ConjuntoHileras[i].Caracter1;
115         al_draw_text(Fuente, al_map_rgb(0, 180, 0), X, ConjuntoHileras[i].y - 40, ALLEGRO_ALIGN_CENTRE, CaracterActual);
116 
117         if (ConjuntoHileras[i].mode == 4)
118         {
119             CaracterActual[0] = ConjuntoHileras[i].Caracter2;
120             al_draw_text(Fuente, al_map_rgb(0, 80, 0), X, ConjuntoHileras[i].y - 60, ALLEGRO_ALIGN_CENTRE, CaracterActual);
121         }
122         else if (ConjuntoHileras[i].mode == 5)
123         {
124             CaracterActual[0] = ConjuntoHileras[i].Caracter2;
125             al_draw_text(Fuente, al_map_rgb(0, 160, 0), X, ConjuntoHileras[i].y - 60, ALLEGRO_ALIGN_CENTRE, CaracterActual);
```

```
258         CaracterActual[0] = Caracter; // Almacena el nuevo caracter generado para luego imprimirlo
259         al_draw_text(Fuente, al_map_rgb(255, 255, 255), X, ConjuntoHileras[i].y, ALLEGRO_ALIGN_CENTRE, CaracterActual);
260 
261         ConjuntoHileras[i].y += 15; // Desplaza la Coordenada Y de la hilera
262 
263         DesplazarCaracteres(ConjuntoHileras[i]); //Desplaza los caracteres de la hilera para hacer espacio al nuevo generado
264         ConjuntoHileras[i].Caracter0 = CaracterActual[0]; // Ingresa en la hilera el nuevo caracter
265 
```

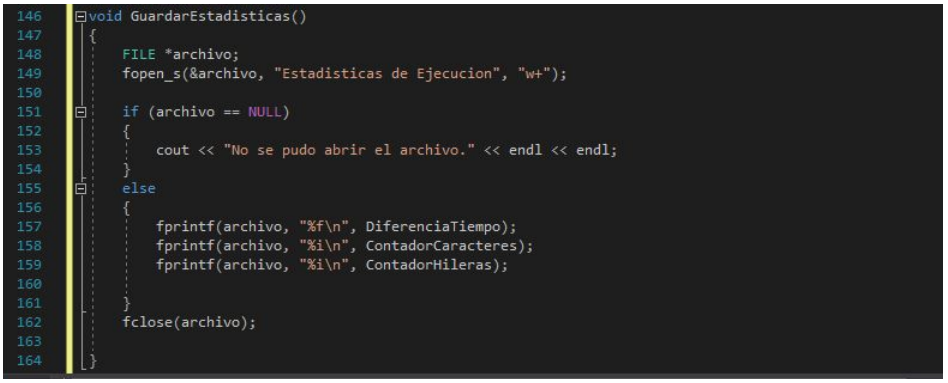
```
262 
263         DesplazarCaracteres(ConjuntoHileras[i]); //Desplaza los caracteres de la hilera para hacer espacio al nuevo generado
264         ConjuntoHileras[i].Caracter0 = CaracterActual[0]; // Ingresa en la hilera el nuevo caracter
265 
266         if (ConjuntoHileras[i].y >= 480) // Cuando las hileras han alcanzado el borde de la pantalla...
267         {
268             ConjuntoHileras[i].Caracter0 = ' '; // Se asignan nuevos valores arbitrarios para generar hileras nuevas
269             ConjuntoHileras[i].Caracter1 = ' ';
270             ConjuntoHileras[i].Caracter2 = ' ';
271             ConjuntoHileras[i].Caracter3 = ' ';
272             ConjuntoHileras[i].Caracter4 = ' ';
273             ConjuntoHileras[i].Caracter5 = ' ';
274             ConjuntoHileras[i].Caracter6 = ' ';
275             ConjuntoHileras[i].Caracter7 = ' ';
276             ConjuntoHileras[i].Caracter8 = ' ';
277             ConjuntoHileras[i].Caracter9 = ' ';
278             ConjuntoHileras[i].Caracter10 = ' ';
279             ConjuntoHileras[i].Caracter11 = ' ';
280             ConjuntoHileras[i].Caracter12 = ' ';
281             ConjuntoHileras[i].Caracter13 = ' ';
282             ConjuntoHileras[i].x = rand() % 720;
283             ConjuntoHileras[i].y = rand() % 40;
284             ConjuntoHileras[i].mode = 3 + rand() % 11;
285 
286             ContadorHileras++; // Se incrementa el contador de hileras
```

Descripción de cada función

- **void InicializarHileras ( ) :** Después de haberse creado un arreglo para las hileras que van a dibujarse en pantalla, esta función inicializa cada una con los valores mínimos requeridos para que puedan manipularse.
- **void DesplazarCaracteres ( ) :** Cada hilera almacena los caracteres que muestra en pantalla en su interior, pero solamente almacena una cantidad determinada. Esta función trabaja los caracteres internos con una lógica similar a la que se emplea en las colas.
- **void MostrarHileras ( ) :** Cada hilera almacena su posición en coordenadas en el display. Esta función dibuja en pantalla cada hilera en la posición que indican sus valores. Además, realiza el llamado de la función que genera nuevos caracteres.Se crearon diferentes modos que generarían el largo de hileras de forma aleatoria según un atributo adicional de cada hilera llamado “modo”, siendo el largo mayor un valor de 14.

Funciones de archivos

Se encargan de almacenar en un documento las estadísticas de la última ejecución del código: número de caracteres generados, número de hileras generadas y tiempo de ejecución.



```
void GuardarEstadisticas()
{
    FILE *archivo;
    fopen_s(&archivo, "Estadisticas de Ejecucion", "w+");

    if (archivo == NULL)
    {
        cout << "No se pudo abrir el archivo." << endl << endl;
    }
    else
    {
        fprintf(archivo, "%f\n", DiferenciaTiempo);
        fprintf(archivo, "%i\n", ContadorCaracteres);
        fprintf(archivo, "%i\n", ContadorHileras);

    }
    fclose(archivo);

}
```

```
ConjuntoEstadisticas *CargarEstadisticas()
{
    FILE *archivo;
    ConjuntoEstadisticas *Datos = NULL;
    fopen_s(&archivo, "Estadisticas de Ejecucion", "r");

    if (archivo == NULL)
    {
        cout << "No se pudo abrir el archivo" << endl << endl;
    }
    else
    {
        Datos = new(ConjuntoEstadisticas);
        while (!feof(archivo))
        {
            fscanf_s(archivo, "%f\n", &Datos->DiferenciaTiempo);
            fscanf_s(archivo, "%i\n", &Datos->ContadorCaracteres);
            fscanf_s(archivo, "%i\n", &Datos->ContadorHileras);
        }
    }
    fclose(archivo);
    return Datos;
}
```

Descripción de cada función

- **void GuardarEstadísticas ( ) :** Abre/Crea un nuevo archivo de texto. Luego almacena en su interior las estadísticas de la última ejecución del código.
- **ConjuntoEstadisticas *CargarEstadisticas ( ) :** Abre el archivo de texto generado. Toma sus valores y los asigna a una nueva estructura que los almacenará en su interior.

Función principal

Se refiere al void main. Ejecuta en orden las siguientes tareas:

- Inicialización de los elementos requeridos para trabajar la librería Allegro
- Creación de la cola de eventos y los timers
- Inicialización de las hileras iniciales
- Ejecución en ciclo de la espera de eventos
- Guardar en archivo las estadísticas de ejecución
- Carga del archivo las estadísticas

- Impresión en pantalla de las estadísticas
- Destrucción de los elementos Allegro creados

```
190 int main(int argc, char **argv)
191 {
192     if (!al_init()) {
193         fprintf(stderr, "failed to initialize allegro!\n");
194         return -1;
195     }
196
197     al_set_new_display_flags(ALLEGRO_WINDOWED | ALLEGRO_RESIZABLE);
198
199     Pantalla = al_create_display(720, 480);
200     al_set_window_title(Pantalla, "The Matrix");
201
202     if (!Pantalla) {
203         fprintf(stderr, "failed to create display!\n");
204         return -1;
205     }
206
207     // Inicializar constructores de Allegro
208     al_init_font_addon();
209     al_init_ttf_addon();
210     al_init_image_addon();
211     al_install_audio();
212     al_init_acodec_addon();
213     al_reserve_samples(1000);
214     al_init_primitives_addon();
```

```
214     al_init_primitives_addon();
215     al_install_keyboard();
216
217     // Inicializar Font
218     Fuente = al_load_font("ARCHRISTY.ttf", 16, NULL);
219
220     // Inicializar Fuente de Audio
221     Sample = al_load_sample("Soundd.ogg"); //sample always NULL
222     al_reserve_samples(1);
223     al_play_sample(Sample, 1.0, ALLEGRO_AUDIO_PAN_NONE, 1.0, ALLEGRO_PLAYMODE_LOOP, 0);
224
225     // Creación de Timers
226     ALLEGRO_TIMER *PrimerTimer = al_create_timer(5 / FPS);
227     ALLEGRO_TIMER *SegundoTimer = al_create_timer(5.0 / FPS);
228     ALLEGRO_TIMER *TercerTimer = al_create_timer(1.0 / FPS); // Flip Display
229
230     // Creación de Cola de Eventos
231     ALLEGRO_EVENT_QUEUE *ColaEventos = al_create_event_queue();
232
233     // Registro de Eventos
234     al_register_event_source(ColaEventos, al_get_timer_event_source(PrimerTimer));
235     //al_register_event_source(ColaEventos, al_get_timer_event_source(SegundoTimer));
236     al_register_event_source(ColaEventos, al_get_timer_event_source(TercerTimer));
```

```
237     al_register_event_source(ColaEventos, al_get_keyboard_event_source());
238
239     // Inicialización de Timers
240     al_start_timer(PrimerTimer);
241     //al_start_timer(SegundoTimer);
242     al_start_timer(TercerTimer);
243
244     ALLEGRO_KEYBOARD_STATE estadoTeclado;
245     al_get_keyboard_state(&estadoTeclado);
246
247     Hilera ConjuntoHileras[MaxHileras];
248     InicializarHileras(ConjuntoHileras);
249
250     // Estadísticas
251     ConjuntoEstadisticas *DatosEjecución = new(ConjuntoEstadisticas);
252
253     // Cronómetros
254     time_t inicio, fin;
255     inicio = time(NULL);
256
```

Incluir Audio y Fuentes de Letra

La siguiente imagen muestra las funciones utilizadas para insertar audio al programa, así como las fuentes de letra utilizadas, así como los timers utilizados

```
216     // Inicializar Font
217     Fuente = al_load_font("ARCHRISTY.ttf", 16, NULL);
218
219     // Inicializar Fuente de Audio
220     Sample = al_load_sample("Soundd.ogg"); //sample always NULL
221     al_reserve_samples(1);
222     al_play_sample(Sample, 1.0, ALLEGRO_AUDIO_PAN_NONE, 1.0, ALLEGRO_PLAYMODE_LOOP, 0);
223
224     // Creación de Timers
225     ALLEGRO_TIMER *PrimerTimer = al_create_timer(5 / FPS);
226     ALLEGRO_TIMER *SegundoTimer = al_create_timer(5.0 / FPS);
227     ALLEGRO_TIMER *TercerTimer = al_create_timer(1.0 / FPS); // Flip Display
228
229     // Creación de Cola de Eventos
230     ALLEGRO_EVENT_QUEUE *ColaEventos = al_create_event_queue();
231
232     // Registro de Eventos
233     al_register_event_source(ColaEventos, al_get_timer_event_source(PrimerTimer));
234     //al_register_event_source(ColaEventos, al_get_timer_event_source(SegundoTimer));
235     al_register_event_source(ColaEventos, al_get_timer_event_source(TercerTimer));
236     al_register_event_source(ColaEventos, al_get_keyboard_event_source());
237
```

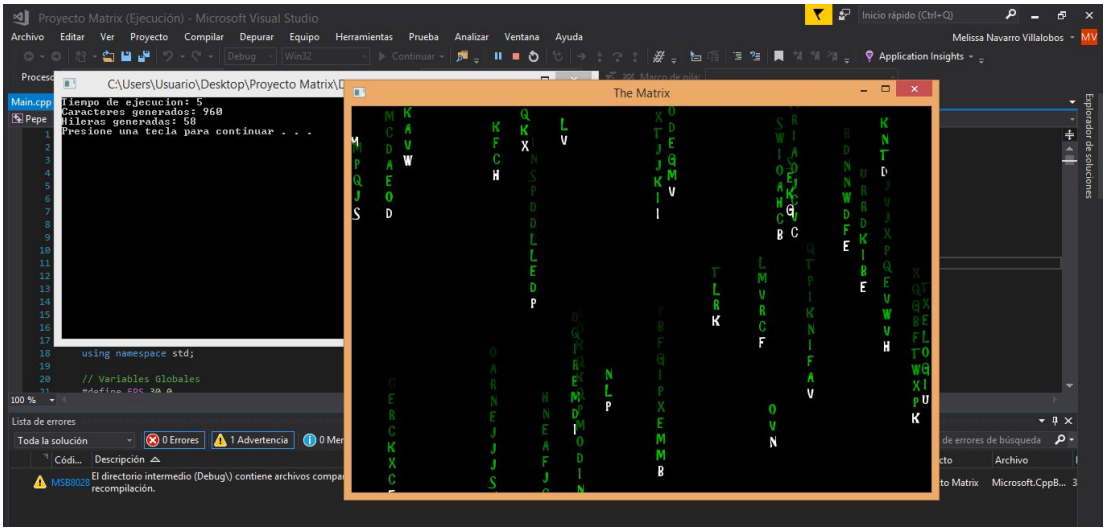
Salir del programa

Con la tecla ESC se cierra el programa

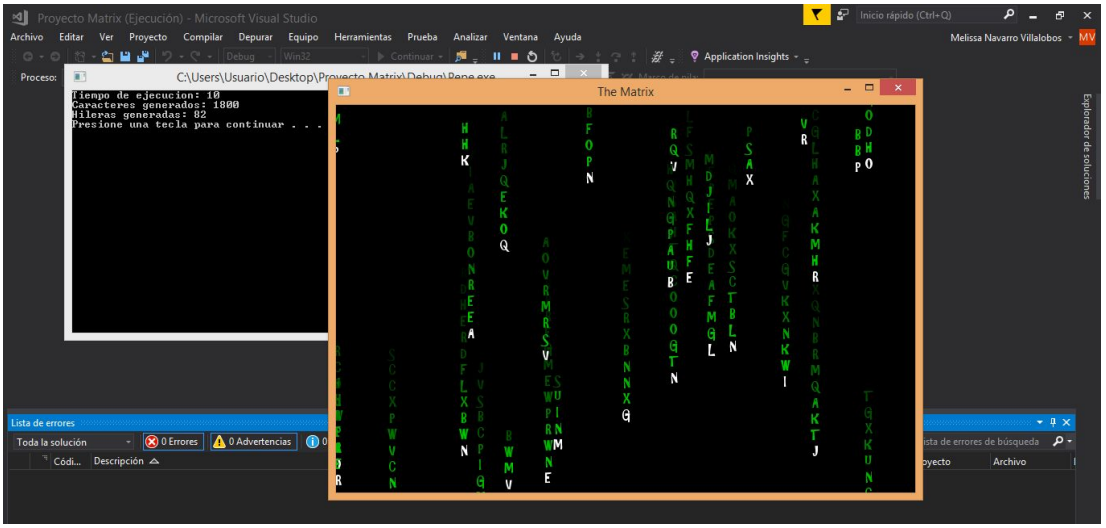
```
256     while (Condicion == true)
257     {
258         ALLEGRO_EVENT eventos;
259         al_wait_for_event(ColaEventos, &eventos);
260
261         if (eventos.type == ALLEGRO_EVENT_KEY_DOWN)
262         {
263             switch (eventos.keyboard.keycode)
264             {
265             case ALLEGRO_KEY_ESCAPE:
266             {
267                 Condicion = false;
268                 break;
269             }
270             }
271         }
```

Prueba de comprobación

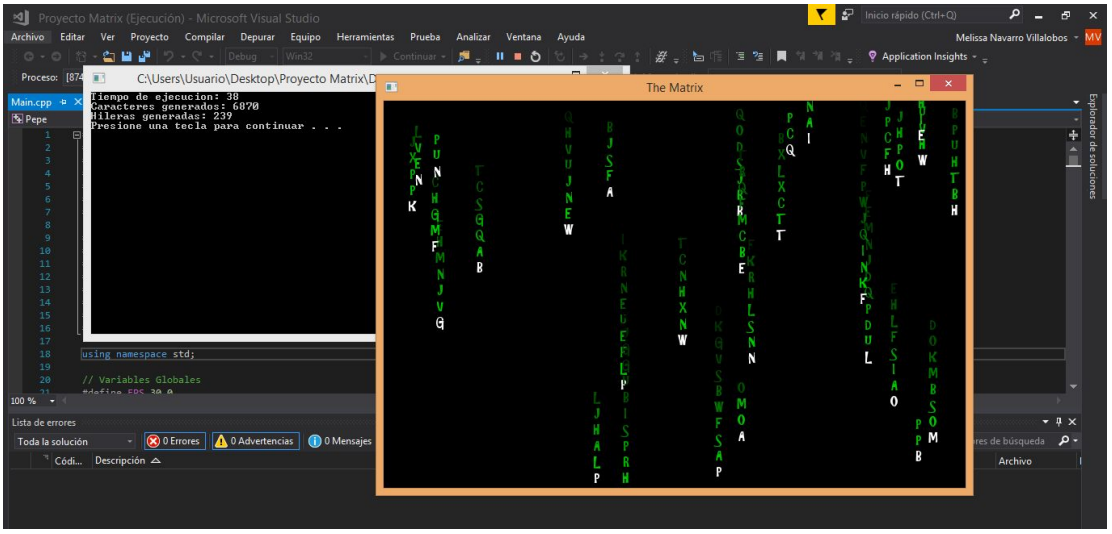
5 segundos transcurridos desde la ejecución:



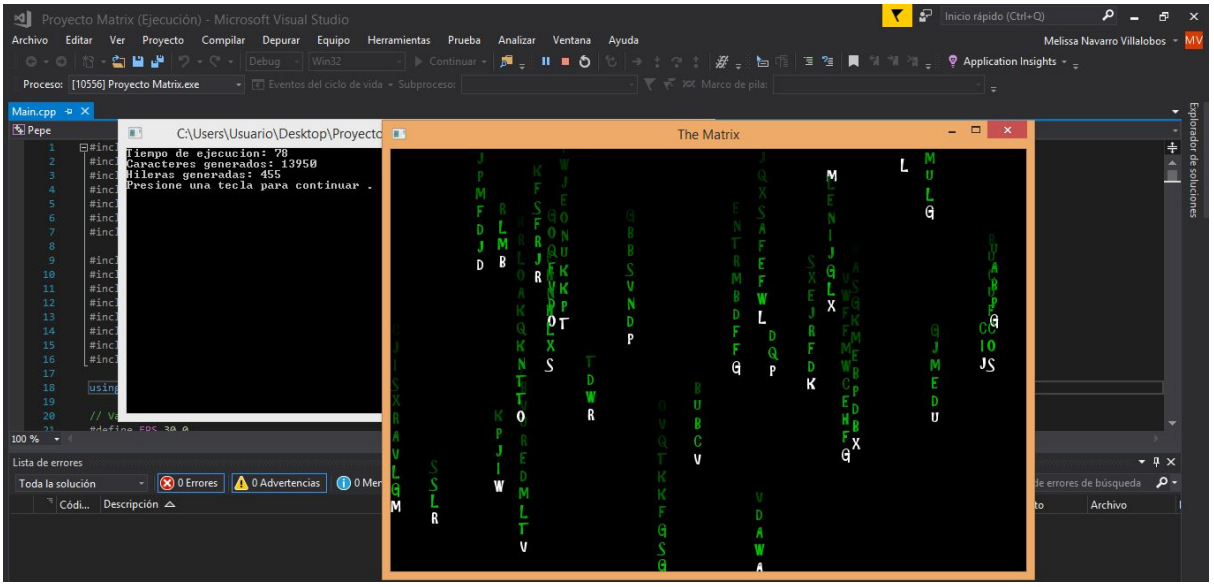
10 segundos transcurridos desde la ejecución:



38 segundos transcurridos desde la ejecución:

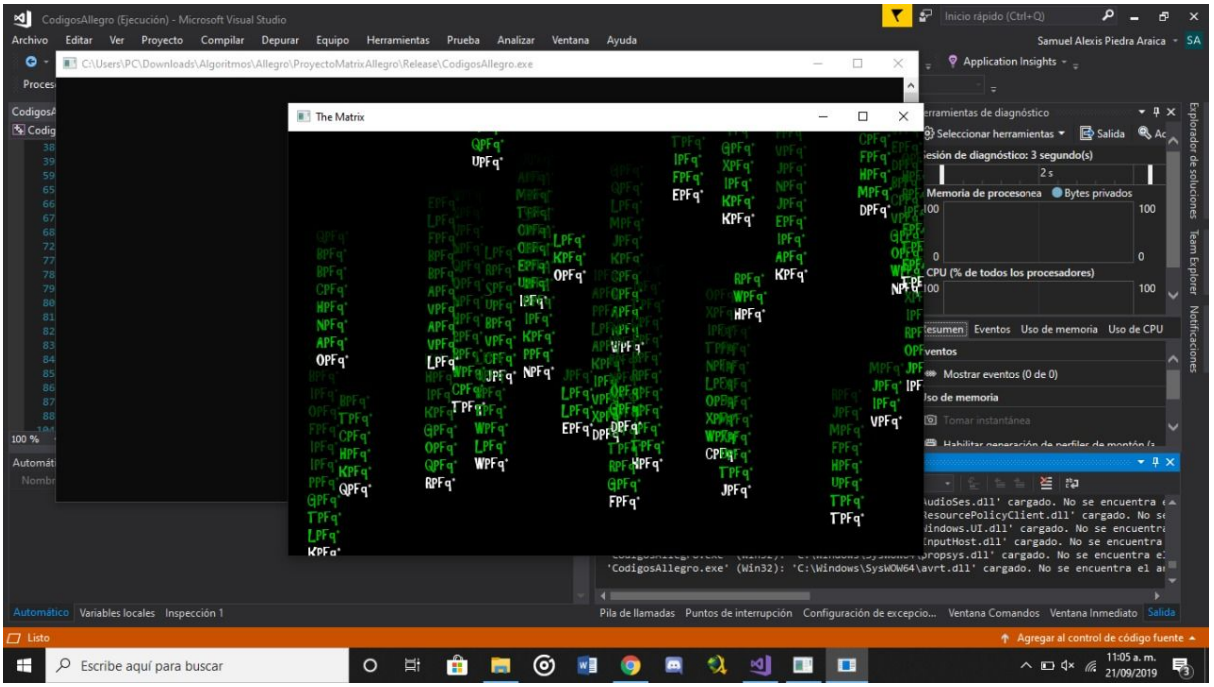


78 segundos después de la ejecución:

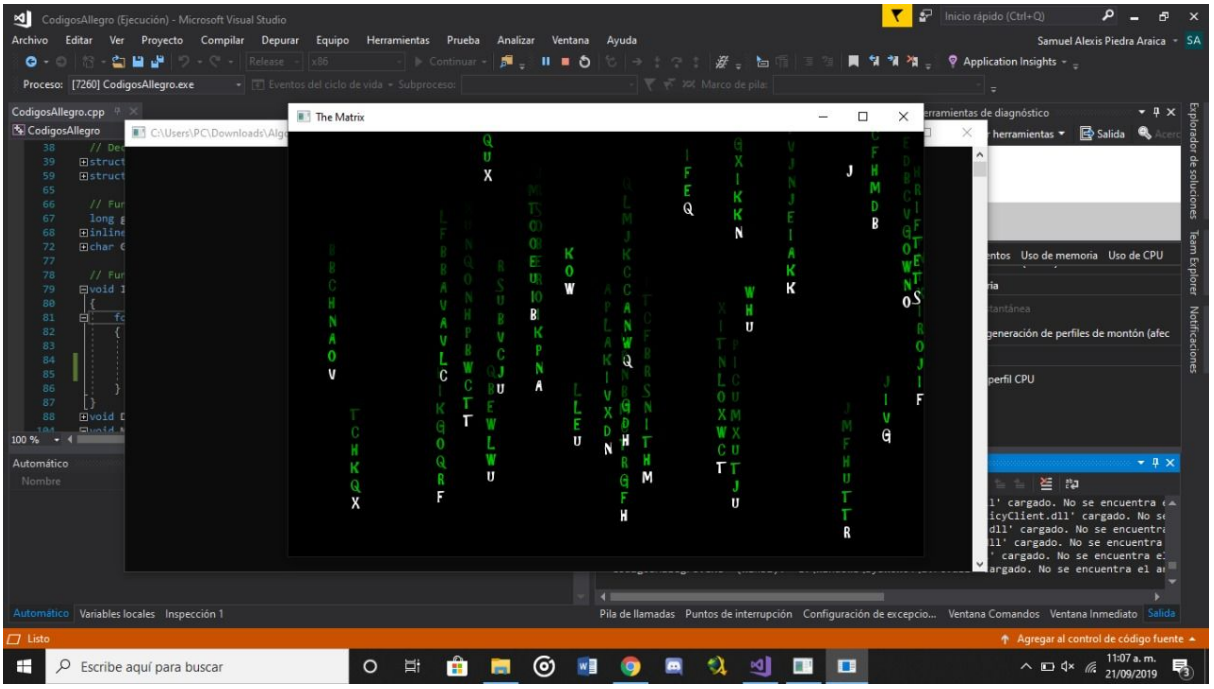


Revisión, Confesión, Pulgas o Errores

En alguna computadoras, el archivo puede correr mal la primera vez que se ejecuta debido a que aún no existe memoria caché que alivie el proceso, esto se soluciona al correrlo una segunda vez



La siguiente imagen muestra el código funcionando de forma correcta después de correrlo una segunda vez



Otro problema presentado es que las hileras desaparecen de forma inmediata al llegar al final de la pantalla, es decir, no se borran caracter por caracter como en las simulaciones ejemplo.