Colegiul Național „Ferdinand I” Bacău

Lucrare de atestare a competențelor la informatică

Candidat: Covaci Patrick David Prof. Coordonator: Trofin Gabriela

Mai 2024

Argument

Motivul pentru care am ales să fac acest proiect este de a experimenta cu a înțelege sistemele low-level de interacțiune între program și sistemul de operare, de a concepe un motor de joc video de la zero și de a ieși din aria de comfort din sfera informaticii, lucrând pe o bibliotecă ce nu este predată la școală, cu puțin sau aproape deloc ajutor din mediul extern.

Programul în sine este făcut în limbajul de programare C. Am ales acest limbaj datorită ecosistemului său matur, utilizare foarte mare în Embedded și Systems Programming și sintaxei sale corente și disciplinate, prezentate foarte bine în cartea „The C Programming Language". Abstractizarea este fundamentală pentru dezvoltarea aplicațiilor software, din acest motiv am ales biblioteca Simple DirectMedia Layer care este disponibilă în mai multe platforme (Windows, GNU/Linux, BSD, MacOS, etc.). Această bibliotecă oferă posibilitatea de a interacționa cu grafici și sisteme de input pentru a concepe un software complex, rapid, eficient și portabil. Foarte multe jocuri video create de la zero utilizează biblioteca SDL2.

Acest proiect este o demonstrație a afișării pe ecran unui pătrat pe o fereastră, acesta mișcându-se constant într-o direcție diagonală până când atinge o margine a ferestrei, sau când cursorul mouse-ului atinge forma geometrică. În acest sens, programul demonstrează logica randării pe ecran, cadru cu cadru, a canvasului și de procesare a evenimentelor de input. Fiecare atingere a pătratului cu mouse-ul va fi numărat într-un counter.

Logica programului

Din natura complexității acestei biblioteci, logica de inițializare este foarte verboasă. Întâi se declară variabilele globale ce vor fi folosite pe parcursul rulării (automat vor fi inițializate cu valori nule), apoi în interiorul funcției init_program() se inițializează întâi în memorie scheletul bibliotecii SDL2 respectiv SDL2_TTF, în același timp verificând dacă inițializarea a fost efectuată cu succes. Ulterior, restul variabilelor vor fi inițializate de regulă dinamic utilizând fie funcțiile prestabilite furnizate de SDL precum SDL_CreateWindow() sau manual utilizând calloc(). Există de asemenea și multe verificări de inițializare, pentru a asigura un comportament normal al programului.

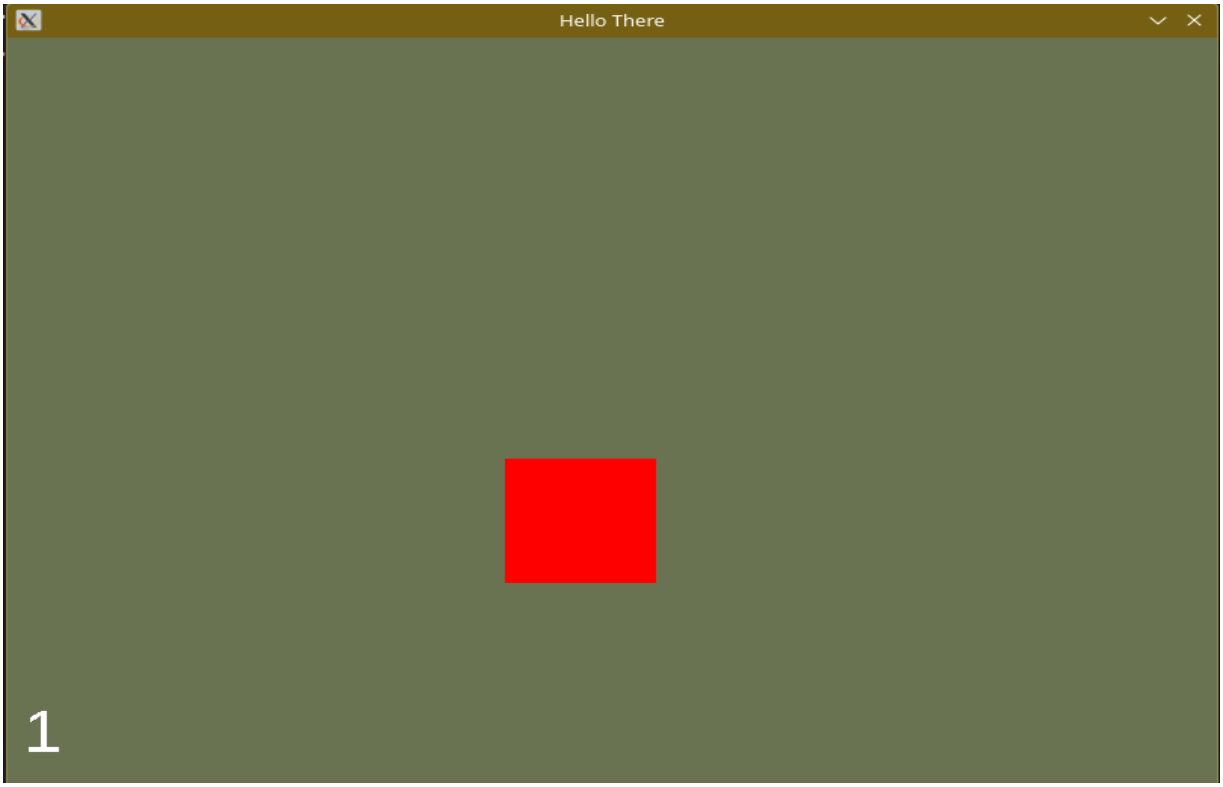
Imediat după, se inițializează incrementarea poziției pătratului prin intermediul structurii pos, definiția acesteia fiind regăsită în headerul window.h, împreună cu poziția inițială a pătratului, acesta fiind amplasat aproximativ pe la mijloc. Se mai inițializează și un vector de caractere care are ca scop afișarea pe ecran a counterului.

Ulterior, se intră într-o structură repetitivă nedeterminată pentru a menține fereastra programului deschis. Se crează de fiecare dată valori de tip boolean pentru determinarea marginilor ferestrei, apoi se crează counterul în sine pentru a fi pregătit de afișare, apoi se trece la logica randării în sine. Pe SDL, orice manipulare de canvas se face în culise, ca apoi să fie afișat rezultatul final prin intermediul funcției SDL_RenderPresent(). După ce se verifică limitele pentru a schimba direcția formei geometrice, se trece la logica procesării evenimentelor.

Pe SDL, mișcarea ferestrei, apăsarea tastaturii, mișcarea cursorului, sau apăsarea butonului de închidere sunt considerate evenimente. Aceste evenimente se adună într-o stivă ce va fi eliberată odată cu procesarea fiecărui eveniment în parte. Pentru a asigura un comportament previzibil, se crează o altă structură repetitivă pentru a elibera complet stiva evenimentelor pentru fiecare iterație a structurii repetitve principale. Pentru a filtra ce evenimente vor fi procesate, se utilizează un switch care se ocupă strict de evenimentul de închidere respectiv cel al cursorului. Pentru evenimentul de tip SDL_MOUSEMOTION, este impusă condiția check_interaction_in_rect(), unde se verifică dacă cursorul se află în interiorul pătratului. În caz afirmativ, se schimbă sensul, se amplasează într-un loc aleatoriu și se schimbă culoarea fundalului tot cu o valoare aleatorie.

Viteza structurilor repetitive depinde de viteza nucleului procesorului, așadar pentru a tranziționa la următorul cadru la fel pe toate platformele, s-a introdus SDL_Delay() pentru a opri de execuția timp de 3 milisecunde și pentru a reduce utilizarea procesorului.

Instalarea programului



Codul sursă este disponibil pe GitHub pe pagina https://github.com/Ty3r0X/funni-square. Deoarece am lucrat la acest proiect pe GNU/Linux, instrucțiunile din fișierul README.md se aplică doar acolo.

Instalarea pe GNU/Linux

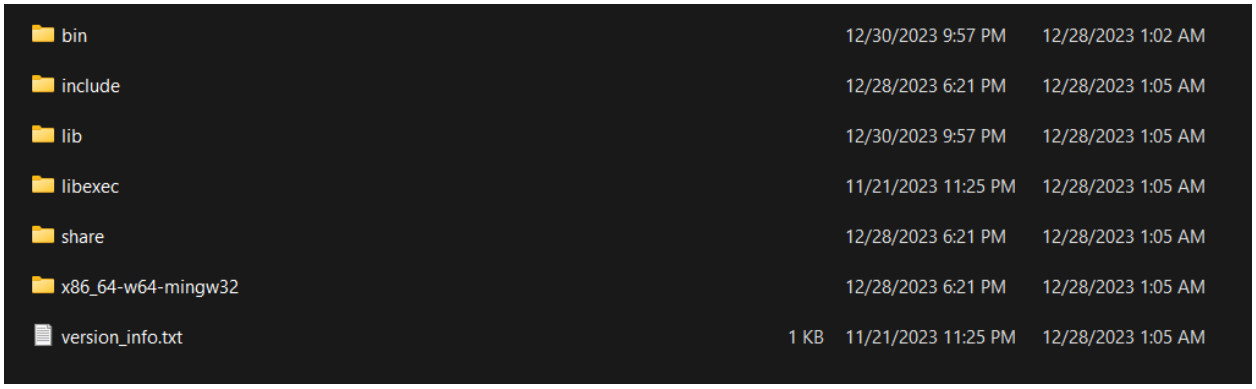
Se presupune utilizarea unei distribuții bazate pe Debian. Pe un terminal se va rula:

```
# instalare dependency-uri
sudo apt install git libsdl2-dev libsdl2-ttf-dev
# furnizare cod sursă direct de pe GitHub
git clone https://github.com/Ty3r0X/funni-square
```

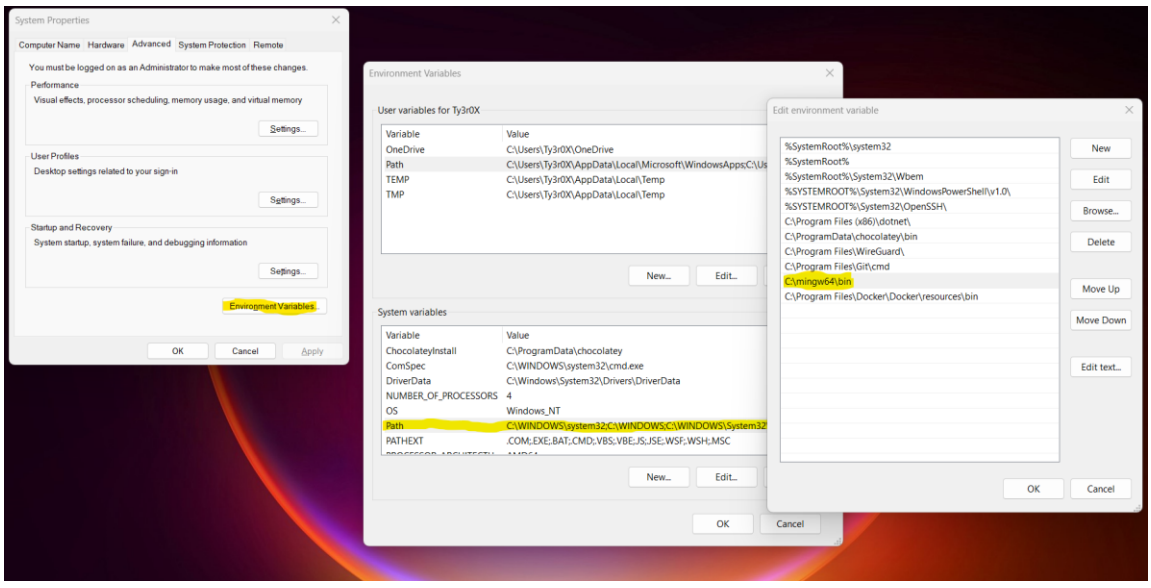
```
cd funni-square/
# compilarea programului
make
# rularea programului
./project
```

Instalarea pe Windows

Se presupune că este instalat Code::Blocks împreună cu MingW. Întâi se identifică folderul MingW, de regulă este situat în C:\Program Files\CodeBlocks\MingW\.



Întâi se vor descărca bibliotecile SDL2 și SDL2_ttf pentru MingW. Apoi, tot conținutul din folderul x86_64-w64-mingw32 din ambele arhive va fi mutat pe folderul MingW respectiv folderul x86_64-w64-mingw32 situat tot în interiorul MingW.



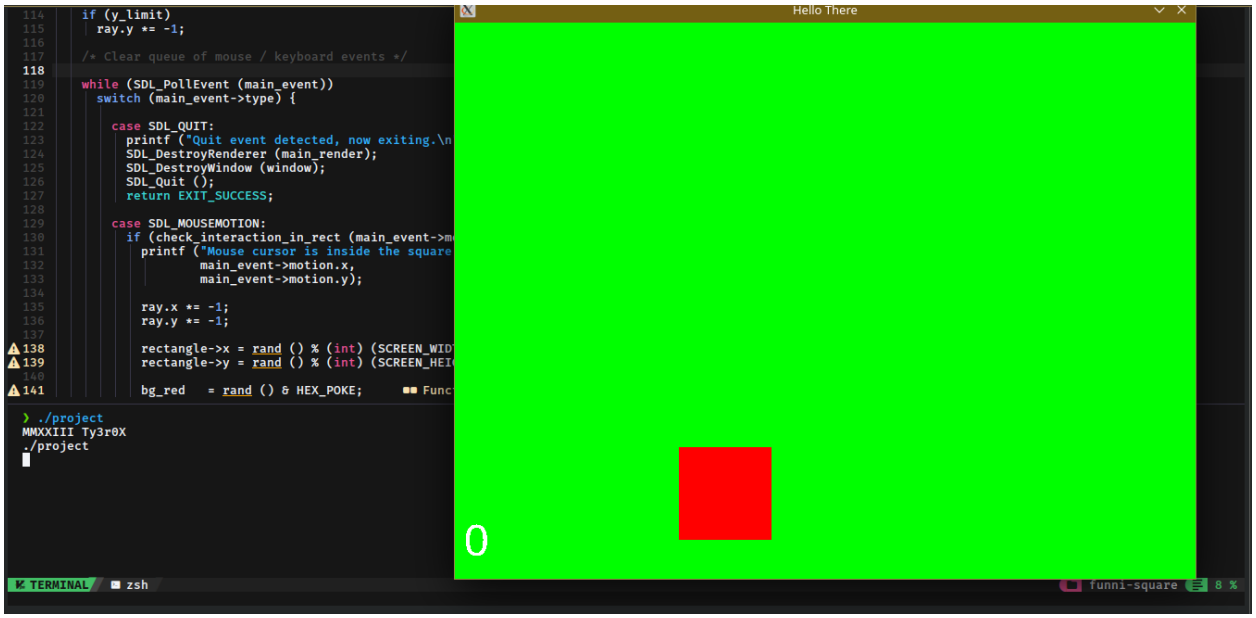
Apoi, se vor deschide setările pentru modificarea Enviroment Variables. Pe User Variables sau System Variables, se identifică intrarea Path, apoi se dă click pe butonul Edit, New, Browse, iar apoi idenificați folderul bin din interiorul MingW. După, se dă click pe OK, se va deschide un Command Prompt și se va scrie gcc. Dacă apare `gcc: fatal error: no input files compilation terminated`, uneltele pentru compilarea programului au fost instalate cu succes. Pe un browser, deschideți https://github.com/Ty3r0X/funni-square, apăsați butonul verde Code, apoi Download ZIP. După ce fișierele sunt dezarhivate, se va deschide un nou Command Prompt în acel folder și se va rula comanda ming32-make. Dacă nu au apărut erori, se va deschide executabilul project.exe.

```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ty3r0X\Desktop\funni-square>mingw32-make
gcc src/rectangle.c -c -std=c99 -Werror -Wno-padded -Wno-disabled-macro-expansion -pedantic -Isrc/ -g -o "src/rectangle.
o"
gcc src/init.c -c -std=c99 -Werror -Wno-padded -Wno-disabled-macro-expansion -pedantic -Isrc/ -g -o "src/init.o"
gcc src/main.c -c -std=c99 -Werror -Wno-padded -Wno-disabled-macro-expansion -pedantic -Isrc/ -g -o "src/main.o"
gcc src/rectangle.o src/init.o src/main.o  -lmingw32 -lSDL2main -lSDL2 -lSDL2_ttf -o "project"

C:\Users\Ty3r0X\Desktop\funni-square>project.exe
MMXXIII Ty3r0X - The Eye shall forsee...
project.exe
Quit event detected, now exiting.
```

Codul sursă




```
/* ------------------------------------------------------------

*

* init.c - executes initialization functions + allocations
```

```
*
* Copyright (C) MMXXIII Patrick D. Covaci <ty3r0x@chaox.ro>
*
* The source is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* FITNESS FOR A PARTICULAR PURPOSE.
*
*/

#include "window.h"

#include <SDL2/SDL_error.h>
#include <SDL2/SDL_events.h>
#include <SDL2/SDL_pixels.h>
#include <SDL2/SDL_rect.h>
#include <SDL2/SDL_render.h>
#include <SDL2/SDL_surface.h>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_video.h>
#include <stdio.h>
#include <stdlib.h>

/* Global variables */

SDL_Window *window;
SDL_Renderer *main_render;
SDL_Event *main_event;
SDL_Texture *background;
SDL_Rect *rectangle;
SDL_Surface *text_surface;
SDL_Texture *text_texture;
TTF_Font *text_font;
SDL_Color *text_color;
```

```
SDL_Rect counter_box;


int

init_program (void) {

/* Initialization Checks */


if (SDL_Init (SDL_INIT_VIDEO) < 0)

return fprintf (stderr, "SDL initialization error: %s\n", SDL_GetError ());


if (TTF_Init () < 0)

return fprintf (stderr, "Font initialization error: %s\n", SDL_GetError ());


/* window variable initialization */


window = SDL_CreateWindow ("Hello There",

POS_X,

POS_Y,

SCREEN_WIDTH,

SCREEN_HEIGHT,

SDL_WINDOW_SHOWN);


if (window == NULL) {

return fprintf (stderr, "SDL Window Creation Error: %s\n", SDL_GetError ());

}


/* Background initialization */


main_render = SDL_CreateRenderer (window, -1, 0);

background = SDL_CreateTexture (main_render,

SDL_PIXELFORMAT_RGBA8888,

SDL_TEXTUREACCESS_TARGET,

SCREEN_WIDTH,

SCREEN_HEIGHT);
```

```
/* Initialize rectangle*/

rectangle = calloc (1, sizeof (SDL_Rect));
rectangle->w = RECT_SIZE;
rectangle->h = RECT_SIZE;

/* Counter initialization */

counter_box.x = COUNTER_BOX_X;
counter_box.y = COUNTER_BOX_Y;

text_font = TTF_OpenFont ("arimo.ttf", COUNTER_TEXT_SIZE);

if (text_font == NULL)
fprintf (stderr, "Failed to load counter - arimo.ttf may be missing\n");

text_surface = NULL;
text_texture = NULL;

text_color = calloc (1, sizeof (SDL_Color));
text_color->r = 255;
text_color->g = 255;
text_color->b = 255;

/* A new variable is created called main_event, which is a SDL_Event type, which itself is a structure
* of multiple structures... of events... */

main_event = calloc (1, sizeof (SDL_Event));

if (main_event == NULL) {
return fprintf (stderr, "Failed to allocate memory for main event structure %s\n", SDL_GetError ());
}
```

```
return EXIT_SUCCESS;
}
/* ------------------------------------------------------------
*
* rectangle.c - defines special functions for rectangles
*
* Copyright (C) MMXXIII Patrick D. Covaci <ty3r0x@chaox.ro>
*
* The source is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* FITNESS FOR A PARTICULAR PURPOSE.
*
*/

#include "window.h"

#include <SDL2/SDL_stdinc.h>

/* Function to check if a coordinate (x, y) is inside a rectangle */
int
check_interaction_in_rect (int x_cursor, int y_cursor, struct SDL_Rect *rect) {
/* Check X coordinate is within rectangle range */
if (x_cursor >= rect->x && x_cursor < (rect->x + rect->w)) {
/* Check Y coordinate is within rectangle range */
if (y_cursor >= rect->y && y_cursor < (rect->y + rect->h)) {
/* X and Y is inside the rectangle */
return 1;
}
}

/* X or Y is outside the rectangle */
return 0;
```

```
}

/* ----------------------------------------------------------

*

* main.c

*

* Copyright (C) MMXXIII Patrick D. Covaci <ty3r0x@chaox.ro>

*

* The source is distributed in the hope that it will be useful,

* but WITHOUT ANY WARRANTY; without even the implied warranty of

* FITNESS FOR A PARTICULAR PURPOSE.

*

*/


#include "window.h"


#include <SDL2/SDL_events.h>

#include <SDL2/SDL_render.h>

#include <SDL2/SDL_stdinc.h>

#include <SDL2/SDL_video.h>

#include <stdbool.h>

#include <stdlib.h>


#define HEX_POKE 0xFF

#define COUNTER_SIZE 100


int

main (int argc, const char *argv[]) {


struct pos ray;

int counter = 0;


Uint8 bg_red = 0x00;

Uint8 bg_green = HEX_POKE;
```

```
Uint8 bg_blue = 0x00;


printf ("MMXXIII Ty3r0X\n");


if (init_program () != EXIT_SUCCESS)

return EXIT_FAILURE;


/* Print argv - useless but used to stop

* my compiler from yelling */


for (int i = 0; i < argc; i++)

printf ("%s\n", argv[i]);


ray.x = 1;

ray.y = 1;


rectangle->x = (int) SCREEN_WIDTH / 2;

rectangle->y = (int) SCREEN_HEIGHT / 2;


for (;;) {


/* Initialize counter array */


char counter_text[COUNTER_SIZE];


/* Constantly initialize boolean values for corners, each for loop

* iteration they change values */


bool x_limit = (rectangle->x <= 0 || rectangle->x >= SCREEN_WIDTH - RECT_SIZE);

bool y_limit = (rectangle->y <= 0 || rectangle->y >= SCREEN_HEIGHT - RECT_SIZE);


/* Insert counter values into the array */
```

```
snprintf (counter_text, sizeof (counter_text), "%d", counter);

text_surface = TTF_RenderText_Solid (text_font, counter_text, *text_color);
text_texture = SDL_CreateTextureFromSurface (main_render, text_surface);

/* Snatch counter_box's width and size by querying text_surface */

SDL_QueryTexture (text_texture, NULL, NULL, &counter_box.w, &counter_box.h);

/* Background render*/

SDL_SetRenderTarget (main_render, background);
SDL_SetRenderDrawColor (main_render, bg_red, bg_green, bg_blue, 0x00);
SDL_RenderClear (main_render);

/* Rectangle render */

SDL_RenderDrawRect (main_render, rectangle);
SDL_SetRenderDrawColor (main_render, HEX_POKE, 0x00, 0x00, 0x00);
SDL_RenderFillRect (main_render, rectangle);
SDL_SetRenderTarget (main_render, NULL);
SDL_RenderCopy (main_render, background, NULL, NULL);

/* Counter render */

SDL_RenderCopy (main_render, text_texture, NULL, &counter_box);

/* All finished, time to show on screen */

SDL_RenderPresent (main_render);

/* Normally the rect constantly goes up diagonally, if it reaches a
* corner the next position gets multiplied with -1 on its respective
```

```
* axis so it goes the opposite direction*/

if (x_limit)
ray.x *= -1;

if (y_limit)
ray.y *= -1;

/* Clear queue of mouse / keyboard events */

while (SDL_PollEvent (main_event))
switch (main_event->type) {

case SDL_QUIT:
printf ("Quit event detected, now exiting.\n");
SDL_DestroyRenderer (main_render);
SDL_DestroyWindow (window);
SDL_Quit ();
return EXIT_SUCCESS;

case SDL_MOUSEMOTION:
if (check_interaction_in_rect (main_event->motion.x, main_event->motion.y, rectangle)) {
printf ("Mouse cursor is inside the square at position (%d,%d)\n",
main_event->motion.x,
main_event->motion.y);

ray.x *= -1;
ray.y *= -1;

rectangle->x = rand () % (int) (SCREEN_WIDTH - 1 - RECT_SIZE);
rectangle->y = rand () % (int) (SCREEN_HEIGHT - 1 - RECT_SIZE);

bg_red = rand () & HEX_POKE;
```

```
bg_green = rand () & HEX_POKE;
bg_blue = rand () & HEX_POKE;

counter++;
}
break;

default:
break;
}

/* Constantly increment / decrement rectangle position */

rectangle->x = rectangle->x + (1 * ray.x);
rectangle->y = rectangle->y + (1 * ray.y);

/* Add a delay, let the CPU breathe */

SDL_Delay (3);
}
}
/* ------------------------------------------------------------
* window.h
*
* Copyright (C) MMXXIII Patrick D. Covaci <ty3r0x@chaox.ro>
*
* The source is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* FITNESS FOR A PARTICULAR PURPOSE.
*
*/
#ifndef WINDOW_H
#define WINDOW_H
```

```
#include <SDL2/SDL.h>
#include <SDL2/SDL_pixels.h>
#include <SDL2/SDL_rect.h>
#include <SDL2/SDL_render.h>
#include <SDL2/SDL_stdinc.h>
#include <SDL2/SDL_ttf.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>

#define SCREEN_WIDTH 800.0
#define SCREEN_HEIGHT 600.0
#define POS_X 100
#define POS_Y 100

#define COLOR_R 0xFF
#define COLOR_G 0xA5
#define COLOR_B 0x00

#define RECT_SIZE 100

#define COUNTER_BOX_X 10
#define COUNTER_BOX_Y 530
#define COUNTER_TEXT_SIZE 48

extern SDL_Window *window;
extern SDL_Renderer *main_render;
extern SDL_Texture *background;
extern SDL_Rect *rectangle;
extern SDL_Event *main_event;
extern SDL_Surface *text_surface;
extern SDL_Texture *text_texture;
```

```
extern TTF_Font *text_font;

extern SDL_Color *text_color;

extern SDL_Rect counter_box;


struct pos {

int x;

int y;

};


int init_program (void);

int check_interaction_in_rect (int x_cursor, int y_cursor, struct SDL_Rect *rect);


#endif
```

Bibliografie:

Kernighan, B. W., & Ritchie, D. M. (1978). *C Programming Language, 2nd Edition.* Prentice Hall.

Lazy Foo' Productions. (2024). Retrieved from https://lazyfoo.net/tutorials/SDL/

SDL2 Wiki. (2024). Retrieved from https://wiki.libsdl.org

StackOverflow. (2024). Retrieved from https://stackoverflow.com/

Cuprins