SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Računalna animacija, 3. lab. vježba

RAČUNALNA VIZUALIZACIJA MANDELBROTOVOG SKUPA

Nikola Radelić

Zagreb, siječanj, 2023.

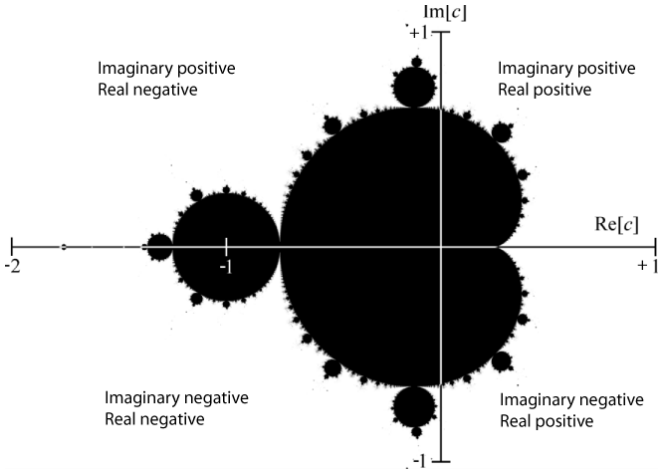
Sadržaj

1. Mandelbrotov skup

Mandelbrotov skup je definiran kao skup kompleksnih brojeva za koje vrijedi da nakon određenog broja iteracija u formuli

$$z_{n+1} = z_n^2 + c$$

ne divergiraju više od postavljenog uvjeta. Početni uvjeti su $z_n = 0$. c predstavlja broj za koji ispitujemo nalazi li se u skupu ili ne. Uvjet za ulazak u skup je da broj nakon određenog broja iteracija ne prelazi apsolutnu vrijednost veću od 2. Pokazuje se da svaki kompleksni broj s apsolutnom vrijednošću većom od 2 divergira i ne spada u Mandelbrotov skup.



Slika 1: Mandelbrotov skup u kompleksnom koordinatnom sustavu (http://www.scienceandcode.com/2020/09/04/the-mandelbrot-set/)

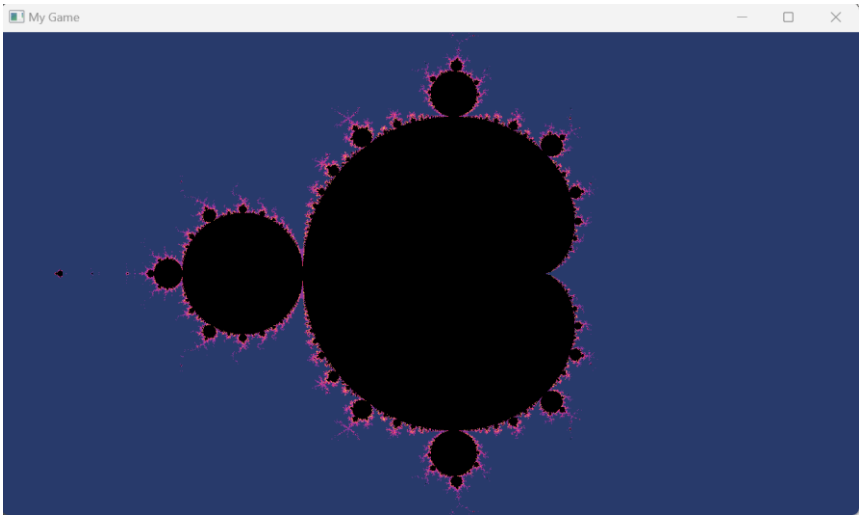
Slika 1 prikazuje vizualizaciju Mandelbrotovog skupa u kompleksnom koordinatom sustavu. Točke koje pripadaju skupu su prikazane crnom bojom dok su bijelom bojom prikazane točke koje divergiraju i ne pripadaju skupu.

Takva vizualizacija je matematički najtočnija ali ako želimo dodati dodatne estetske karakteristike možemo točke koje ne pripadaju skupu bojati ovisno o tome koliko im iteracija treba da pređu uvjet divergencije.

2. Vizualizacija Mandelbrotovog skupa

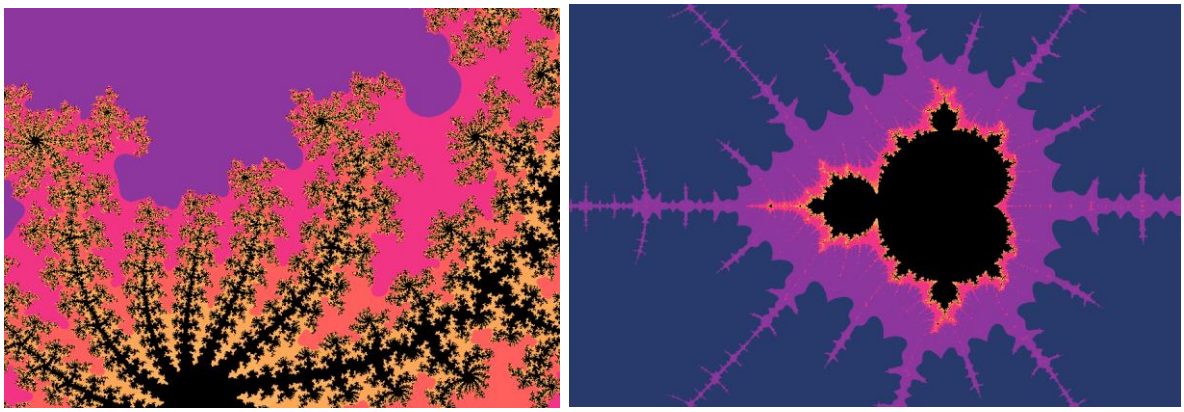
U ovoj laboratorijskoj vježbi vizualizacija Mandelbrotovog skupa ostvarena je korištenjem programskog jezika Python, knjižnice GLFW i jezika za sjenčare GLSL.

U programu je moguće pomoću miša kretati se po skupu i korištenjem kotačića na mišu zoomirati u sami skup. Također je moguće mijenjati korištenje *float* ili *double* vrijednosti za računanje skupa. Ako se koriste *float* vrijednost, onda su performanse puno bolje, ali je ograničeno koliko možemo zoomirati u skup, dok korištenje *double* vrijednosti smanjuje performanse dok povećava koliko možemo zoomirati u skup.



Slika 2: Prikaz vizualizacije

Slika 2 prikazuje prikaz koji dobijemo pokretanjem programa. Kretanje po skupu je ostvareno samo korištenjem miša. Iz programa se izlazi korištenjem tipke ESC.



Na ovim slikama možemo vidjeti neke zanimljive fraktale i uzorke koje dobijemo kako se krećemo kroz skup.

3. Programsko ostvarenje vizualizacije

Glavni element programa koji se želi naglasiti je sustav sjenčara koji omogućuje efikasno i brzo prikazivanje elemenata na ekranu. Koristi se GLSL jezik koji je sličan C-u. Oni se izvode na grafičkoj kartici koja ih puno brže i paralelno obrađuje.

3.1. Fragment.txt

```
#version 330 core
#define MAX_ITERATIONS 1000

uniform vec2 resolution;

uniform float CenterX;
uniform float CenterY;
uniform float ZoomScale;

in vec4 gl_FragCoord;
out vec4 frag_color;
out float frag_depth;

vec4 GetIterations()
{
    //if you change the screen , change the value of offset X and offset Y
till it the mandelbrot is centered on your screen
    float offsetX = 1.0f;
    float offsetY = 0.5f;
    float real = ((gl_FragCoord.x / resolution.y - offsetX) * ZoomScale +
CenterX )* 2.0;
    float imag = ((gl_FragCoord.y / resolution.y - offsetY) * ZoomScale +
CenterY )* 2.0;

    int iterations = 0;
    float real_number = real;
    float imaginary = imag;

    while (iterations < MAX_ITERATIONS)
    {
        float tmp_real = real;
        real = (pow(real, 2) - pow(imag, 2)) + real_number;
        imag = (2.0 * tmp_real * imag) + imaginary;

        float dist = pow(real, 2) + pow(imag, 2);

        if (dist > 2.0){
            break;
        }

        ++iterations;
    }
    float k = float(iterations) / MAX_ITERATIONS;

    vec4 color_f;
    if (k < 0.14f){
        color_f = vec4(0.157f, 0.227f, 0.421f, 1.0f);
```

```
    }
    else if (k < 0.28f){
        color_f = vec4(0.553f, 0.216f, 0.619f, 1.0f);
    }
    else if (k < 0.43f){
        color_f = vec4(0.945f, 0.204f, 0.518f, 1.0f);
    }
    else if (k < 0.57f){
        color_f = vec4(1.0f, 0.376f, 0.365f, 1.0f);
    }
    else if (k < 0.71f){
        color_f = vec4(0.996f, 0.663f, 0.349f, 1.0f);
    }
    else if (k < 0.86f){
        color_f = vec4(1.0f, 0.7f, 0.378f, 1.0f);
    }
    else{
        color_f = vec4(0.0f, 0.0f, 0.0f, 1.0f);
    }

    color_f = color_f.rgba;
    color_f.a = 1.0f;
    return color_f;
}

void main()
{
    vec4 color = GetIterations();
    frag_color = color;
}
```

Fragment.txt je fragment sjenčar koji se poziva za svaki fragment (piksel) koji se treba prikazati na ekranu. Kroz vektor gl_FragCoord dobiva koordinate elementa koji se trenutno obrađuje. Dalje za trenutni element njegovu koordinatu prebacujemo u koordinatni sustav prikladan za Mandelbrotov skup. Zbog nemogućnosti korištenja kompleksnih brojeva koristimo realne brojeve ali y koordinate tretiramo kao imaginarni dio, dok x koordinatu tretiramo kao realni dio. Sljedeće vrtimo trenutni broj kroz iteracije Mandelbrotove jednadžbe i provjeravamo koliko iteracija treba da broj pređe apsolutnu vrijednost od 2.

Zatim izabiremo kojom ćemo bojom obojati fragment. Ovisno o broju iteracija izabiremo boju. Tako će fragment biti obojan u jednu od 7 boja.

3.2. Mandelbrot_set.py

Ovdje ću objasniti neke zanimljive dijelove glavnog programskog koda u pythonu.

```
def createShader(self, vertexFilepath, fragmentFilepath):  # funkcija za
učitavanje shadera
    with open(vertexFilepath, 'r') as f:
        vertex_src = f.readlines()

    with open(fragmentFilepath, 'r') as f:
        fragment_src = f.readlines()

    shader = shaders.compileProgram(
        shaders.compileShader(vertex_src, GL_VERTEX_SHADER),
        shaders.compileShader(fragment_src, GL_FRAGMENT_SHADER)
    )
    return shader
```

Funkcija createShader prima dva filepatha do sjenčara fragmenata i vrhova. Od njih kompajlira sjenčar koji dalje zovemo za prikaz slike.

```
def handleScroll(self, window, xoffset, yoffset):  # funkcija za obradu
zooma
    global z
    if yoffset == 1:
        z = z * zoomIn
        if z > 1.0:
            z = 1.0
    elif yoffset == -1:
        z = z * zoomOut
        if z > 1.0:
            z = 1.0

def handlePosition(self):
    global x, y, z

    mouseX, mouseY = glfw.get_cursor_pos(self.window)  # dohvati položaj
miša
    if glfwGetMouseButton(self.window, GLFW_MOUSE_BUTTON_LEFT):  # mijenjaj
položaj samo ako je lijevi gumb stisnut
        x = x + (SCREEN_WIDTH / 2 - mouseX) * z * 0.001
        y = y - (SCREEN_HEIGHT / 2 - mouseY) * z * 0.001
    glfw.set_cursor_pos(self.window, SCREEN_WIDTH / 2, SCREEN_HEIGHT / 2)
# postavi miš na centar ekrana

def mainLoop(self):
    global x, y, z
    glfw.set_scroll_callback(self.window, self.handleScroll)

    while not glfw.window_should_close(self.window) and not
glfw.get_key(self.window, GLFW_CONSTANTS.GLFW_KEY_ESCAPE) ==
GLFW_CONSTANTS.GLFW_PRESS:
        glClearColor(0.0, 0.0, 0.0, 1.0)
```

```
        glClear(GL_COLOR_BUFFER_BIT)

        # dohvati sve događaje
        glfw.poll_events()

        # obradi položaj miša
        self.handlePosition()

        # pošalji varijable shaderu
        if high_precision == True:
            glUniform1d(self.CenterX, x)
            glUniform1d(self.CenterY, y)
            glUniform1d(self.ZoomScale, z)
        else:
            glUniform1f(self.CenterX, x)
            glUniform1f(self.CenterY, y)
            glUniform1f(self.ZoomScale, z)

        # crtaj
        glDrawArrays(GL_TRIANGLE_FAN, 0, 4)

        glFlush()

    glfw.terminate()
```

Funkcija handleScroll se zove za svaki pomak kotačića miša i ažurira varijable za zoomiranje koje se koriste u sjenčaru.

Funkcija handlePosition uzima pomicanje miša i ažurira vrijednosti x i y koje se šalju sjenčaru kako bi se mogli kretati po skupu.

U mainLoop-u je zanimljivo slanje podataka sjenčaru preko adresa koje smo na početku inicijalizrali.

4. Upute za pokretanje programa

Program se nalazi na github repozitoriju na adresi https://github.com/nikolaRadelicFER/racAnim u direktoriju rac_anim_lab3. Skinuti cijeli direktorij i pokrenuti u odgovarajućem Python okruženju. Za izradu je korišteno PyCharm okruženje pa se ono preporuča. Treba imati instalirane knjižnice OpenGL, numpy i glfw. Također instalirati PyOpenGL kako bi Python mogao komunicirati s OpenGL sustavom.