– Praktikumsaufgabe 2+ –

Thema: *Einfache Grafik in Rust*

Zielstellung: Erlernen des Crates `minifb`, Erzeugung eines Fensters, Setzen von Pixeln, Zeichnen der Mandelbrotmenge

1. Das Crate `minifb` gestattet einfache grafische Ausgaben. Legen Sie ein Fenster mit geeigneter Auflösung an und Versuchen Sie, in ihm ein Apfelmännchen (die berühmte *Mandelbrot*-Menge) darzustellen.

   Hinweise:

   - Ein Fenster der Größe `WIDTHxHEIGHT` Pixel kann beispielsweise durch folgenden Code angelegt werden:

     ```
     let mut window = Window::new(
         "Window Title",
         WIDTH,
         HEIGHT,
         WindowOptions::default(),
     )
     .unwrap_or_else(|e| {
         panic!("{}", e);
     });
     ```

   - Für jedes Pixel des Fensterinhalts steht ein vorzeichenloser 32-Bit-Integer zur Verfügung, der die Farbinformation als RGB-Tupel kodiert, wie in Abbildung 1 gezeigt.

     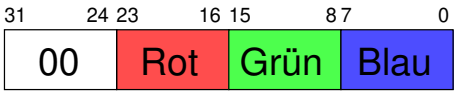

     Abbildung 1: Kodierung der Farbinformation im 32-Bit-Integer

   - Zur Repräsentation des gesamten Bildes müssen Sie sich einen Vektor des Typs `u32` definieren, der für jedes Pixel ein Element besitzt.
   - Der Fensterinhalt wird aktualisiert mittels des Aufrufs

     ```
     window.update_with_buffer(&buffer, WIDTH, HEIGHT).unwrap();
     ```

     Der Vektor `buffer` wird als so genannte *Referenz* übergeben (→ nächste Vorlesung)
   - Das Fenster repräsentiert die x-y-Ebene, wobei x (anfangs) im Intervall [-2;1] und y im Intervall [-1;1] liegt.
   - Für jeden Punkt (x,y) in dieser Ebene werden die folgenden Iterationsgleichungen[1]

[1] In Wirklichkeit wird nur $z_{n+1} := z_n^2 + c$ iteriert, wobei z und c komplexe Zahlen sind.

ausgeführt:

$$a_{n+1} = a_n^2 - b_n^2 + x$$
$$b_{n+1} = 2a_n b_n + y$$

Startwerte für a und b sind $a_0 = b_0 = 0$.

- Die beiden Gleichungen werden iteriert, bis
  - entweder die Maximalanzahl Iterationen (1000) erreicht ist. In diesem Falle erhält der Punkt die Farbe Schwarz.
  - oder ein bestimmter Schwellwert überschritten wird:

    $$a_{n+1}^2 + b_{n+1}^2 > 4.0$$

    Dieser Punkt wird weiß (oder später farbig) gezeichnet.
- Es empfiehlt sich aus Performancegründen, nicht nach jedem einzelnen Pixel `window.update_with_buffer()` aufzurufen (sondern nach jeder Zeile oder nach Komplettierung des Bildes).
- Wie Sie zur Programmbeendigung auf die Betätigung einer bestimmten Taste warten (und dabei nicht zuviel CPU-Zeit verschwenden), finden Sie in der Dokumentation zu `minifb`.
- Wenn Sie alles richtig implementiert haben, sollte ungefähr eine Figur wie in Abbildung 2 gezeigt resultieren.

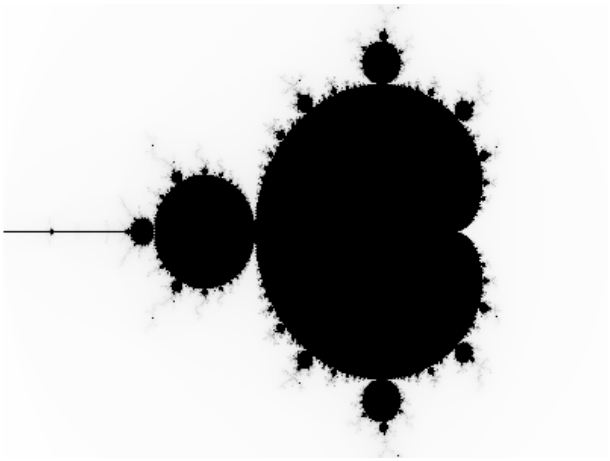

Abbildung 2: Das Apfelmännchen in Schwarz-Weiß

2. Überlegen Sie, warum in Rust weder Prä- noch Post-Inkrement- bzw. Dekrementoperator, also das beliebte `++` und `--` in C, verwirklicht ist. Versuchen Sie dazu das folgende C-Programm nach Rust zu portieren.

```c
#include <stdio.h>

int main(void)
```

```
{
  int x = 23, y;

  y = x++ + ++x;
  printf("x=%d, y=%d\n", x, y);
  return 0;
}
```