Aufgabe 2: Verzinkt

Teilnahme-ID: 66432

Team-ID: 00662

Bearbeiter/-in dieser Aufgabe:
Linus Schumann

20. November 2022

Inhaltsverzeichnis

1 Lösungsidee

1.1 Grundsätzliche Idee

Wie in der Aufgabenstellung angegeben handelt es sich bei dieser Aufgabe um eine Simulation, daher wird die Lösungsidee auch so entwickelt.
Generell ist die Idee sehr einfach. Zuerst einmal müssen ein paar Keime definiert werden, von denen die Kristalle wachsen. Dann werden alle Keime einmal "wachsen"gelasssen. Dabei wird jeder Keim waagerecht und senkrecht, mit einer für diese Richtung vorher definierten Geschwindigkeit erweitert.
Bei jedem Schritt der auf diesem Weg dann zurückgelegt wird, wird dann einer neuer Keim mit den selben Einstellungen erstellt, der dann im nächsten Schritt auch "wachsen"gelasssen wird. Die Simulation endet dann, wenn das ganze Bild ausgefüllt ist und somit kein Keim mehr "wachsen"kann, da ein Keim nur auf freie Felder wachsen kann.

1.2 Input Parameter

Als Input-Parameter muss die Anzahl an Keimen, die Breite und Höhe des Bildes und die maximale Geschwindigkeit definiert werden. Genaueres zur Umsetzung dieser Input-Datei lässt sich in Abschnitt 2.2 finden.

1.3 Datenstruktur eines Keims

Wie in Abschnitt 1.1 beschrieben, müssen verschiedene Werte für jeden Keim gespeichert werden. Dazu gehören die x bzw. y Koordinaten des Keims, sowie die Geschwindigkeit für jede Richtung. Diese wird beim Erstellen der ersten Keime zufällig für jede Richtung definiert. Dabei gilt für eine Geschwindigkeit $1 \leq speed \leq maxSpeed$. Als letztes muss auch noch gespeichert werden, ob der Keim schon aktiviert ist. Dies liegt daran, dass um die Simulation realer zu machen, nicht alle Keime gleichzeitig das erste Mal anfangen zu wachsen, sondern zufällig nacheinander starten.

1.4 Unterschiedliche Farben

Um die unterschiedlichen Richtungen darzustellen werden diese 4 verschiedenen Grautöne genutzt:

1. rgb(75,75,75)
2. rgb(90,90,90)
3. rgb(105,105,105)
4. rgb(120,120,120)

Außerdem wird jeder unsprüngliche Startpunkt eines Keims mit der Farbe schwarz (rgb(0,0,0)) dargestellt.

2 Umsetzung

2.1 Allgemeines

Im Folgenden wird die Umsetzung, der in Abschnitt 1 beschriebene Lösungsidee, näher erläutert. Grundsätzlich wurde diese Idee dabei in Python, genauer gesagt in der Datei ”Aufgabe_2.py” implementiert. Diese Datei befindet sich im Verzeichnis ”./source/”.

Um das implementierte Programm zu starten, kann das Batch-Script genutzt werden. Dieses befindet sich in ”./executables/”. Eine direkte Ausführung der Python-Datei ist auch möglich, allerdings nur wenn sie man sich im ”./source/” Verzeichnis befindet.

Unter dem Verzeichnis ”./beispieleingaben/” befinden sich alle in dieser Dokumentation aufgeführten Beispiele und unter dem Verzeichnis ”./beispielausgaben/” befinden sich dementsprechend die gesicherten Ausgaben, die auch bei Ausführung des Scripts ausgegeben werden. Letztere werden auf Grund der Bild-Ausgabe als ”.png” gepspeichert.

2.2 Input Datei

Die Input-Datei, deren Parameter in Abschnitt 1.2 beschrieben wurden, muss wie folgt aufgebaut sein:

```
nOfSeeds -> 50
width -> 5000
height -> 5000
maxSpeed -> 5
```

Listing 1: Eingabe Beispiel 1

Dies ist natürlich ein Beispiel, es können also natürlich auch andere Zahlen eingegeben werden. Dabei muss nur für alle Zahlen $1 \leq Zahl$ gelten.

2.3 Ablauf der Simulation

Zuerst werden die Keime initalisiert und als Dictonary in einer Liste gespeichert. Außerdem werden die erste Postionen der Keime schon mal schwarz gefärbt. Dann kann die Varibale "pixelCount", die immer die aktuelle Anzahl bereits gefärbter Pixel beinhaltet, auf die anfängliche Anzahl an Keimen gesetzt werden. Dann startet die eigentliche Simulation. Zuerst wird dabei eine While-Schleife definiert, die nur endet wenn alle Felder ausgefüllt sind, also $pixelCount \geq width * height$ gilt. Innerhalb dieser Schleife wird zuerst eine Liste definiert in der alle neuen Keime, die in dieser Iteration entstehen gespeichert werden. Dann wird die aktuelle Liste an Keimen durchgegangen und jeder Keim bearbeitet, wenn er schon aktiviert ist (Abschnitt 2.4). Wenn der Keim noch nicht aktiviert wurde, wird er mit einer Chance von $1 : ((width * height)/20000)$ aktiviert.
Nachdem Durchgehen der Keime wird die alte Liste der Keime durch die neue Liste der Keime ersetzt. Dann startet die nächste Iteration.

Am Ende der While-Schleife wird das Bild gespeichert und die Simulation endet.

2.4 Keim bearbeiten

Beim Bearbeiten eines Keims, wird zuerst die Reihenfolge der Richtungen gemischt. Dann wird für jede Richtung jedes Feld bis zur maximalen Geschwindigkeit durchgegangen. Für jedes berechnete Feld muss dann geprüft werden, ob es innerhalb des Bildes liegt und ob es noch nicht belegt ist.
Wenn es frei ist wird es gefärbt und dann an dieser Stelle ein neuer Keim erstellt, der dann in die Liste für neue Keime hinzugefügt wird.

3 Beispiele

3.1 Beispiel 1

```
nOfSeeds -> 50
width -> 5000
height -> 5000
maxSpeed -> 5
```

Listing 2: Eingabe Beispiel 1

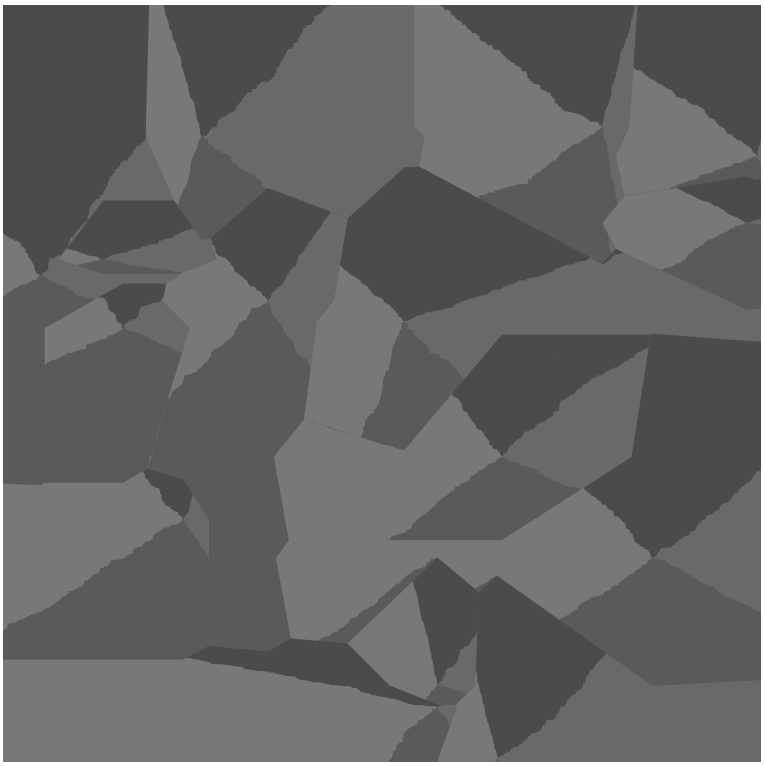

Abbildung 1: Ausgabe Beispiel 1

3.2 Beispiel 2

```
nOfSeeds -> 40
width -> 1920
height -> 1080
maxSpeed -> 3
```

Listing 3: Eingabe Beispiel 2

Abbildung 2: Ausgabe Beispiel 2

3.3 Beispiel 3

```
nOfSeeds -> 100
```

```
width -> 7680
height -> 4320
maxSpeed -> 8
```

Listing 4: Eingabe Beispiel 3

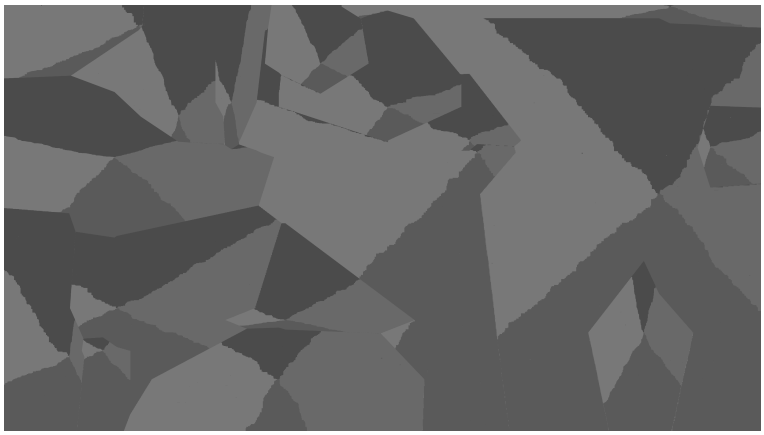

Abbildung 3: Ausgabe Beispiel 3

4 Quellcode

```python
from random import randint, shuffle
from PIL import Image
# inputData
# "nOfSeeds" : total number of seeds
# "width" : width of image
# "height" : height of image
# "maxSpeed" : max grow speed of seed

directions = [(0,-1,"up"),(0,1,"down"),(1,0,"right"),(-1,0,"left")] # direction list
dict = {"up":0,"down":1,"right":2,"left":3} # direction dictory
colors = [75, 90, 105, 120] # list of colors

def calculateImage(inputData, filename: str): # function to calculate new image
    img = Image.new('RGB',(inputData["width"],inputData["height"]), "black") # create new PIL image
    pixels = img.load() # load data into pixels matrix
    seeds = [] # list to save all seeds
    for _ in range(inputData["nOfSeeds"]): # create n seeds
        seedData = {} # create dictionary
        seedData["x"] = randint(0,inputData["width"]-1) # random x pos
        seedData["y"] = randint(0,inputData["height"]-1) # random y pos
        seedData["speeds"] = [randint(1,inputData["maxSpeed"]) for _ in range(4)] # random speed
        seedData["enabled"] = False
        seeds.append(seedData) # append seed to list
        pixels[seedData["x"],seedData["y"]] = (1,1,1) # color start of seed black
    pixelCount = inputData["nOfSeeds"]
    c = 5
    while pixelCount < inputData["width"]*inputData["height"]:
        newseeds = [] # create new list of seeds
        for seed in seeds:
            if seed["enabled"]: # seed is enabled
                shuffle(directions)
                for (x,y,direction) in directions:
                    speed = seed["speeds"][dict[direction]] # get speed of direction
                    pixelToAdd = 0
                    breaked = False
                    for j in range(1, speed+1):
                        # Check if field is free and in picture
```

```
                        if seed["x"]+(x*j) >= 0 and seed["x"]+(x*j) < inputData["width"] and seed["y"]+(y*j) >= 0 and seed["y"]+(y*j) < inputData["height"] and pixels[seed["x"]+(x*j),seed["y"]+(y*j)] == (0,0,0):
                            pixels[seed["x"]+(x*j),seed["y"]+(y*j)] = tuple([colors[dict[direction]] for _ in range(3)]) # color new pixel
                            newSeed = seed.copy() # copy seed
                            newSeed["x"] += j*x # change x
                            newSeed["y"] += j*y # change y
                            newseeds.append(newSeed.copy()) # append new seed to list
                        else:
                            pixelToAdd += j-1
                            breaked = True
                            break
                    if not(breaked):
                        pixelCount += speed # add full speed to pixelCount
                    else:
                        pixelCount += pixelToAdd # add possible speed to pixelCount
                    # prinout status
                    if ((pixelCount / (inputData["width"]*inputData["height"]))*100) > c:
                        print(int((pixelCount / (inputData["width"]*inputData["height"]))*100), "%", "abgeschlossen")
                        c+=5
            else:
                newSeed = seed.copy()
                if randint(1,(inputData["width"]*inputData["height"])//20000) == 1:
                    newSeed["enabled"] = True
                newseeds.append(newSeed.copy())
        seeds = newseeds.copy()

    img.save("../beispielausgaben/beispiel"+filename+".png")

def readFile():
    print("Enter number of example: ",end="")
    filename = input() # get file number
    data = {} # init data dictionary
    try:
        with open("../beispieleingaben/beispiel"+filename+".txt") as f: # open input file
            for line in f.readlines():
                line = line.strip()
                data[line.split(" -> ")[0]] = int(line.split(" -> ")[1]) # save line into data dictionary
    except:
        print("Invalid file")
        exit(1)
    return data, filename

def solve():
    inputData, filename = readFile()
    calculateImage(inputData, filename)

if __name__ == '__main__':
    solve()
```