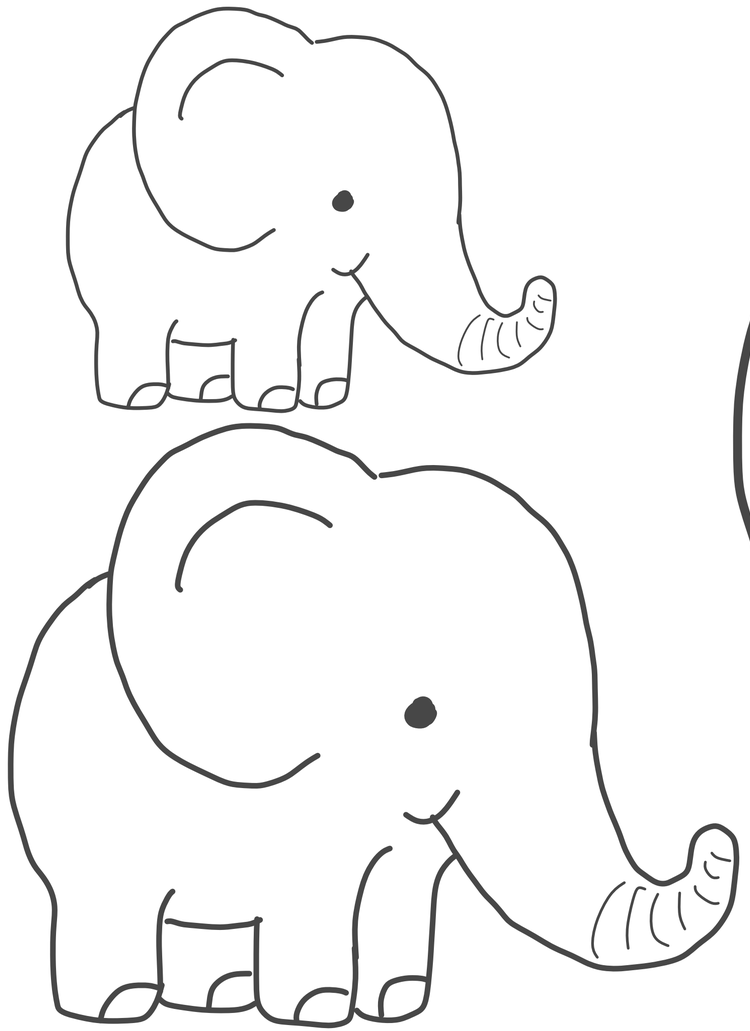
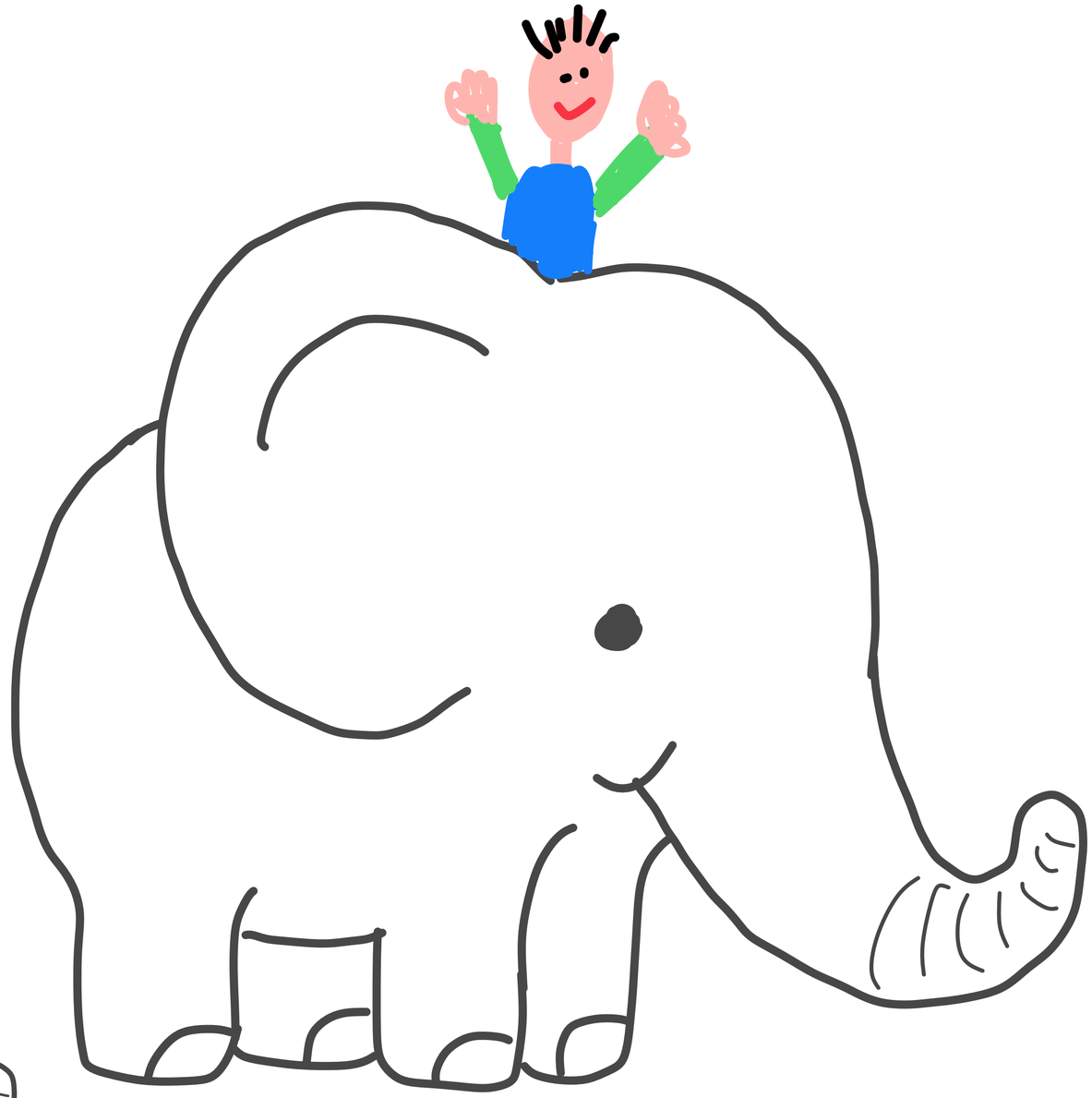
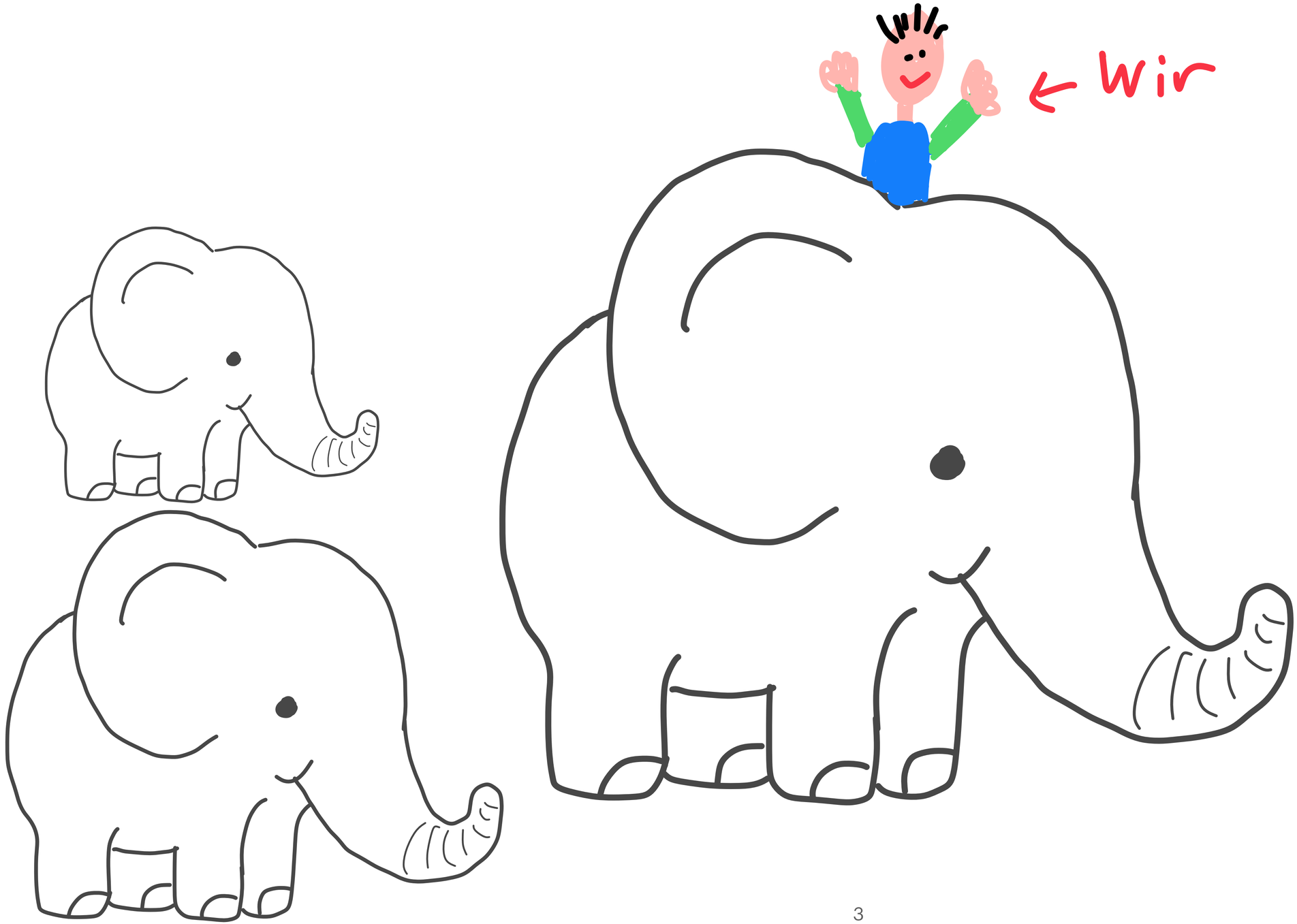
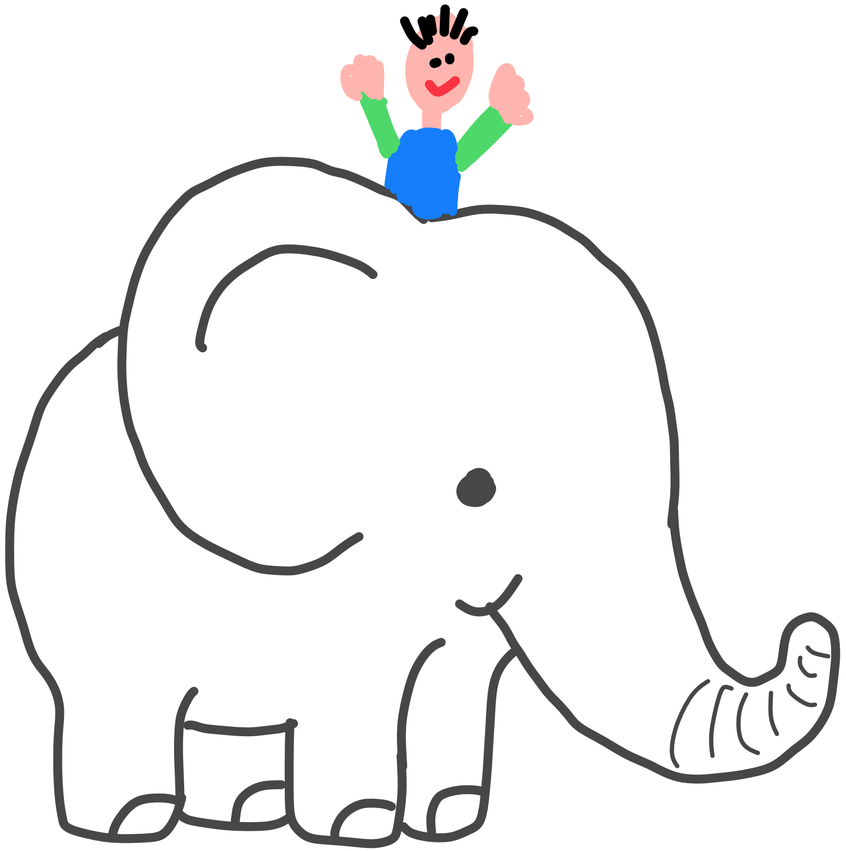
AOC 22/12/18 APL

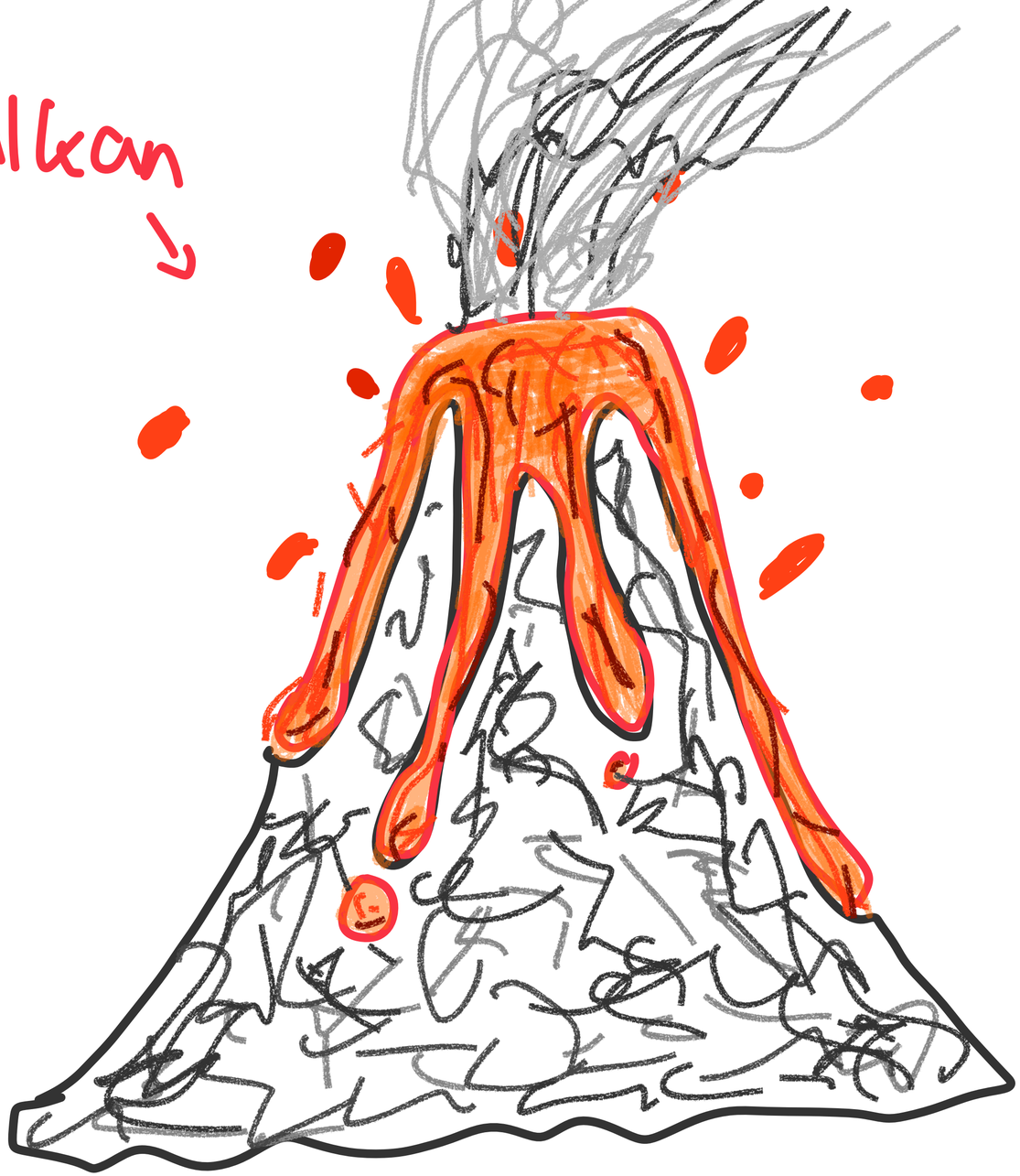
Marvin Borner

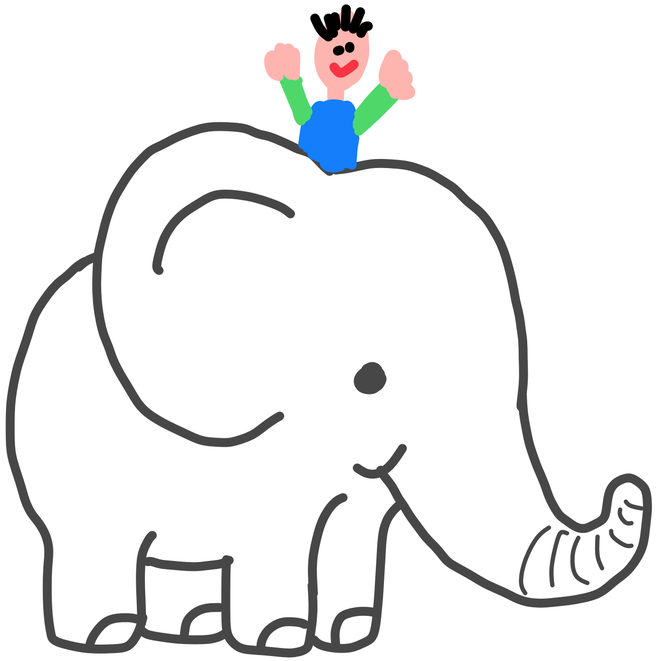




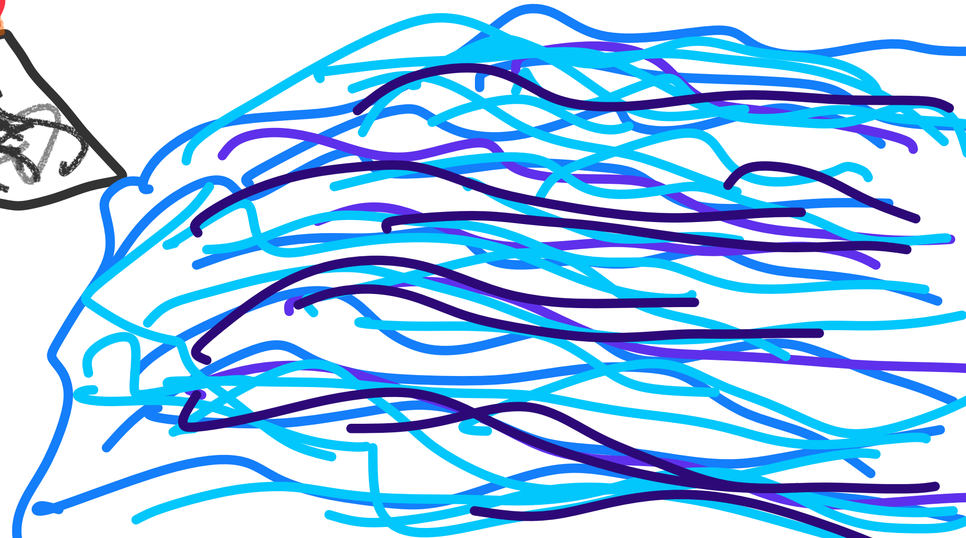

Wir



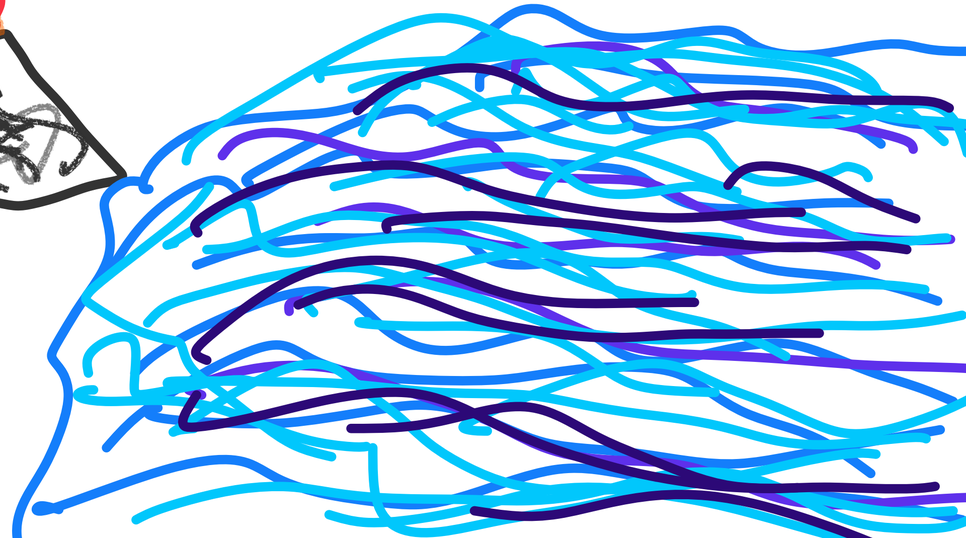

Vulkan



Wasser



Höhle

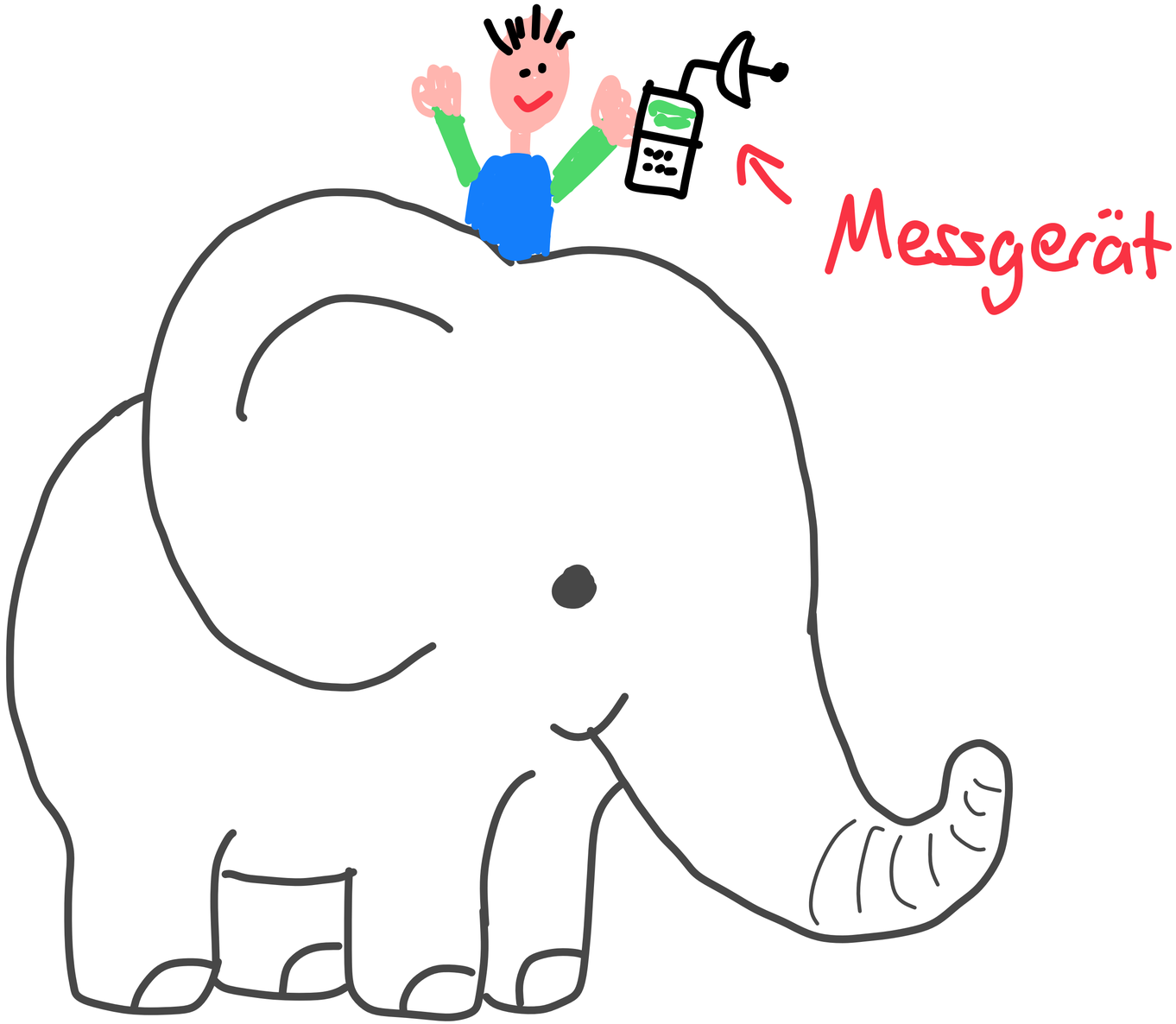


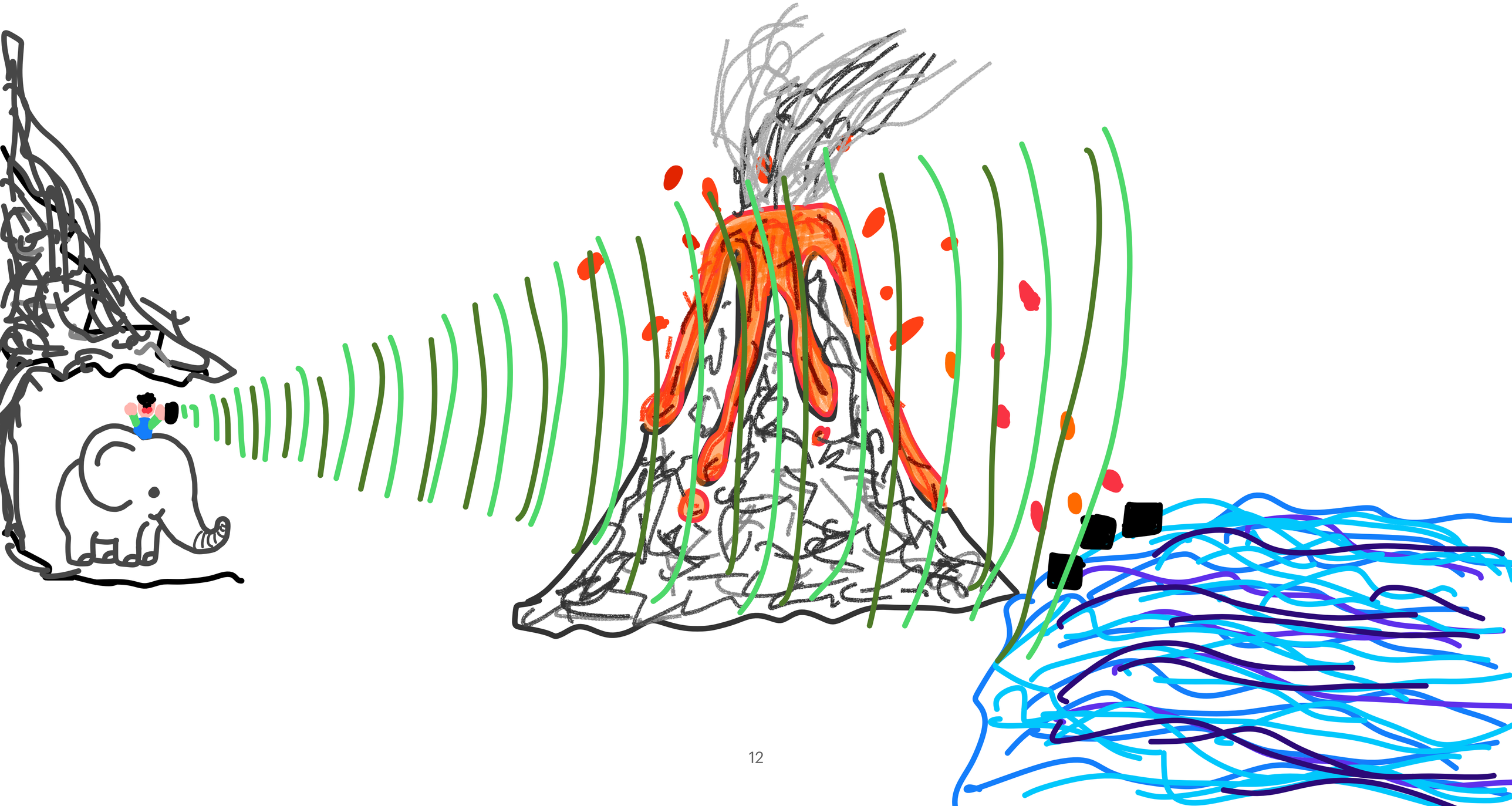

Obsidian?!

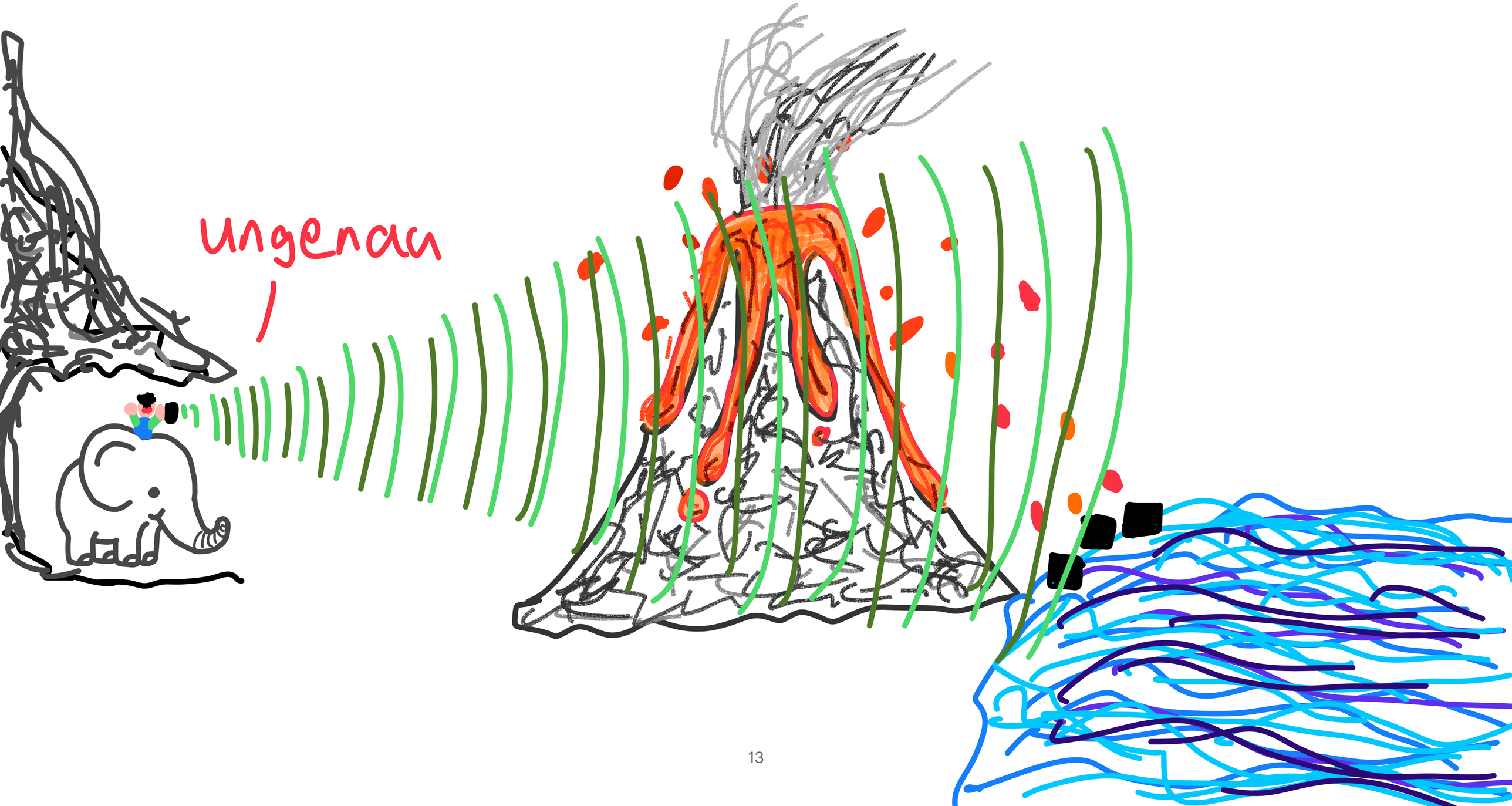

Abhängig
von Kühlrate!

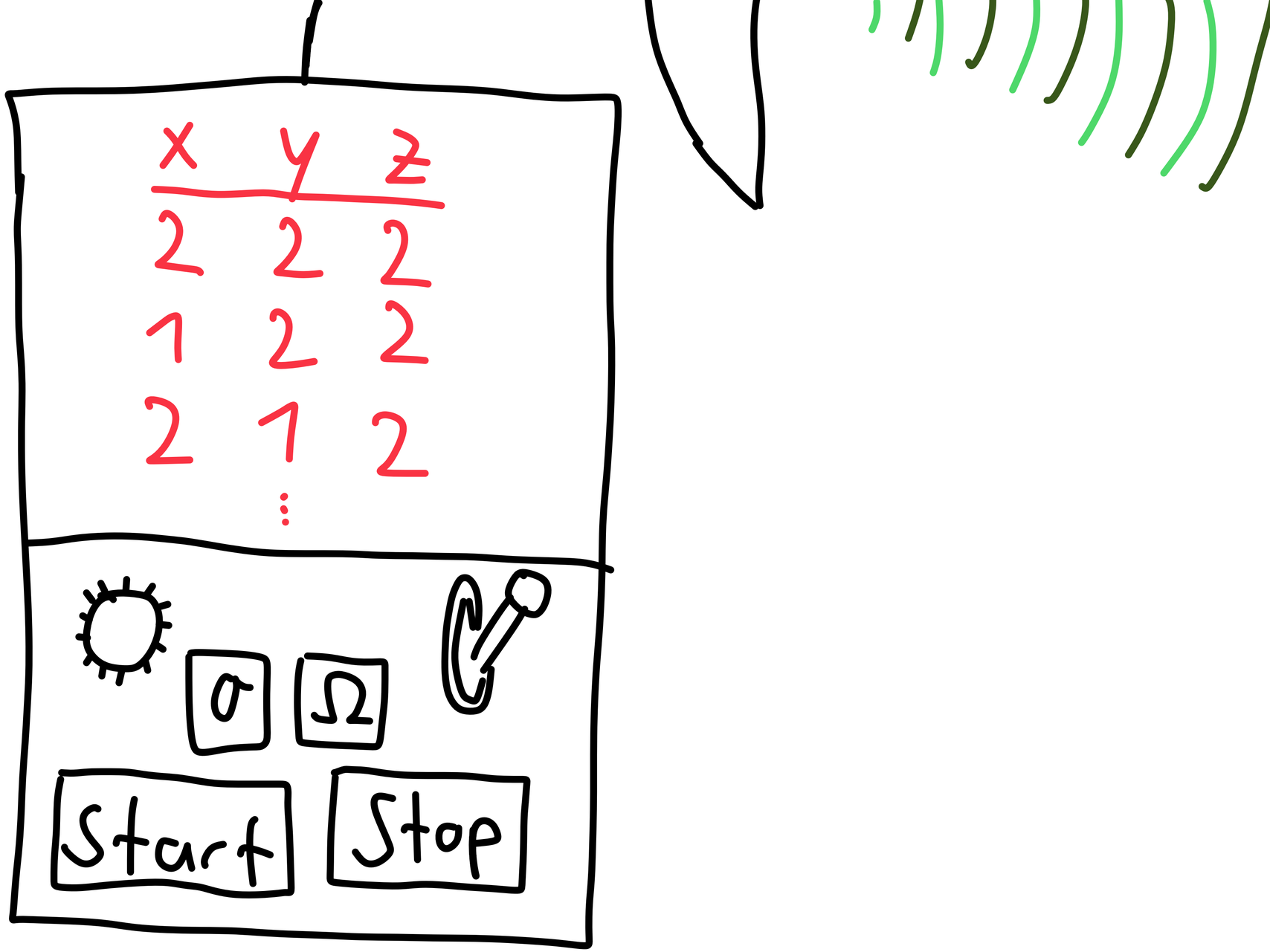
Oberfläche

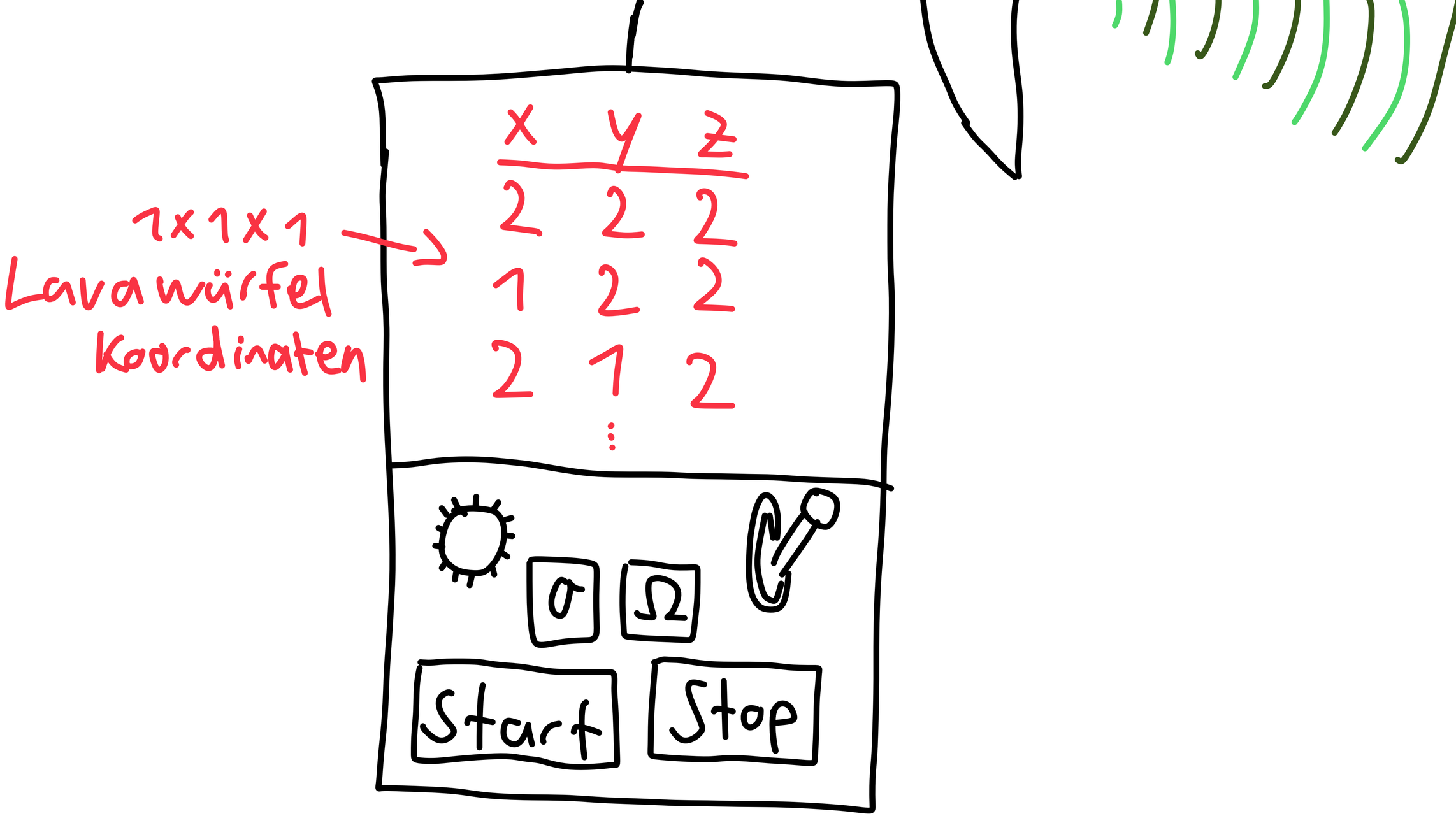
=> Kühlrate => Obsidian?

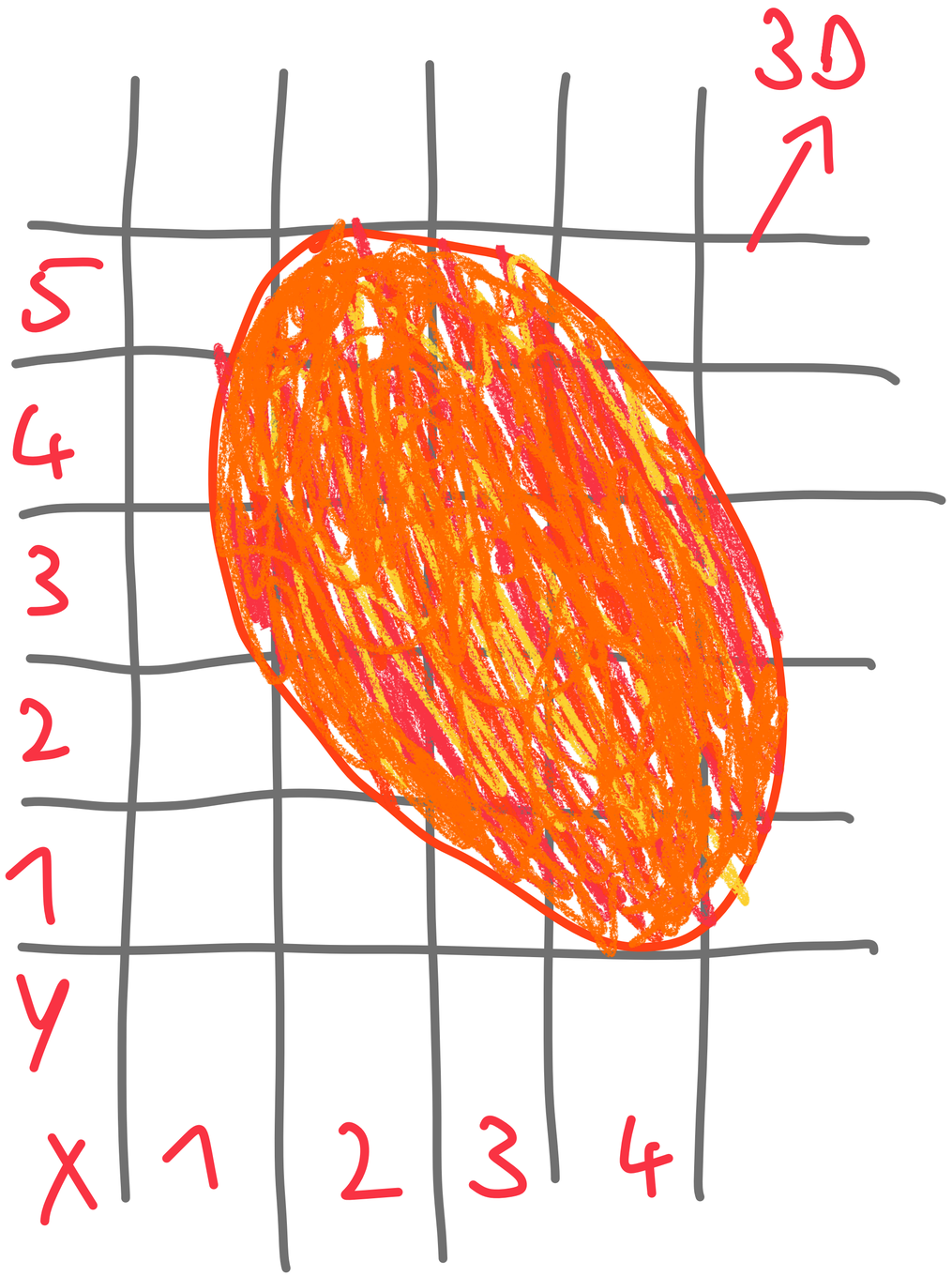

Messgerät



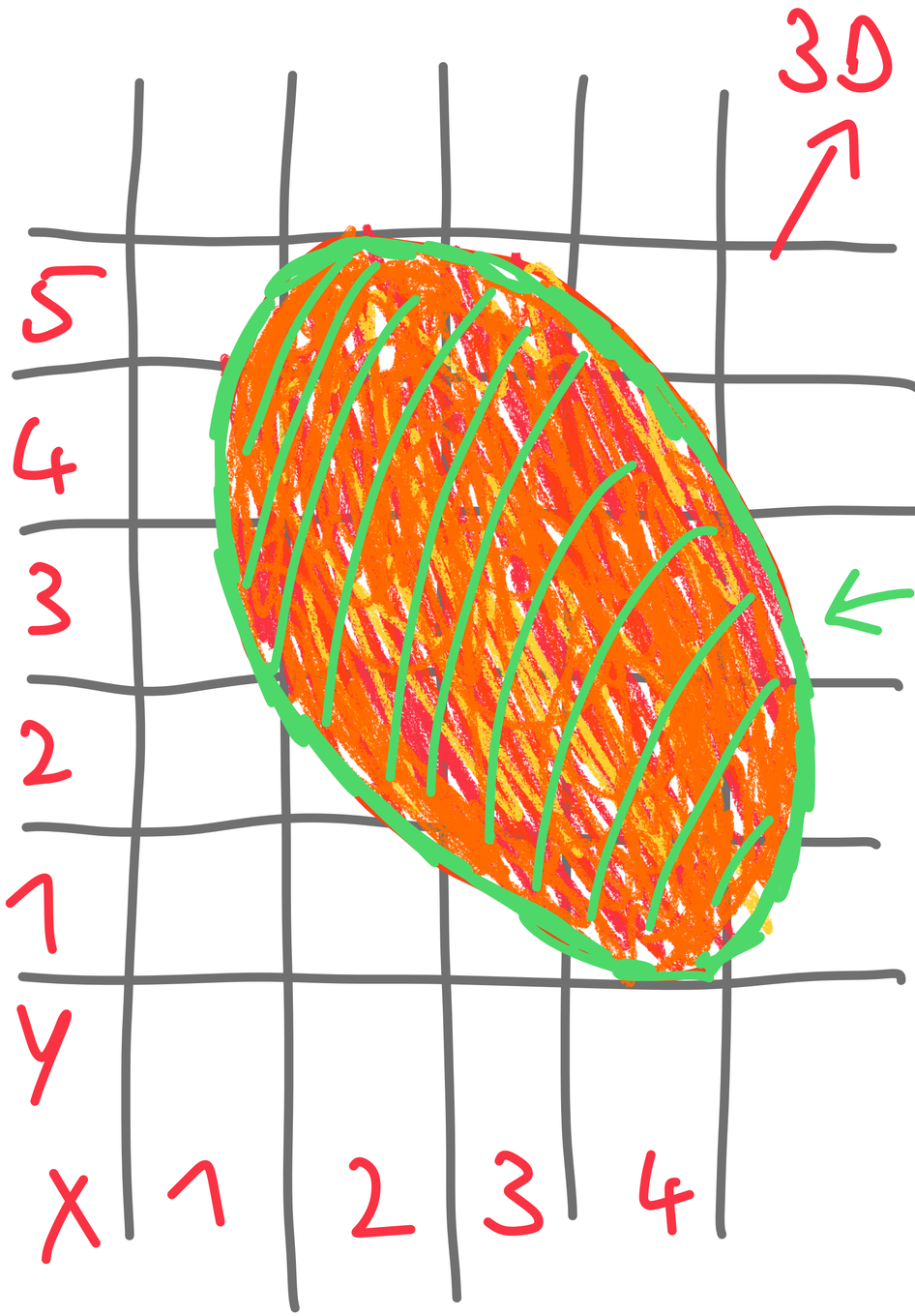

ungenau


x y z
2 2 2
1 2 2
2 1 2
σ
Ω
Start
Stop


1x1x1
Lavawürfel
Koordinaten
x y z
2 2 2
1 2 2
2 1 2
σ
Ω
Start
Stop

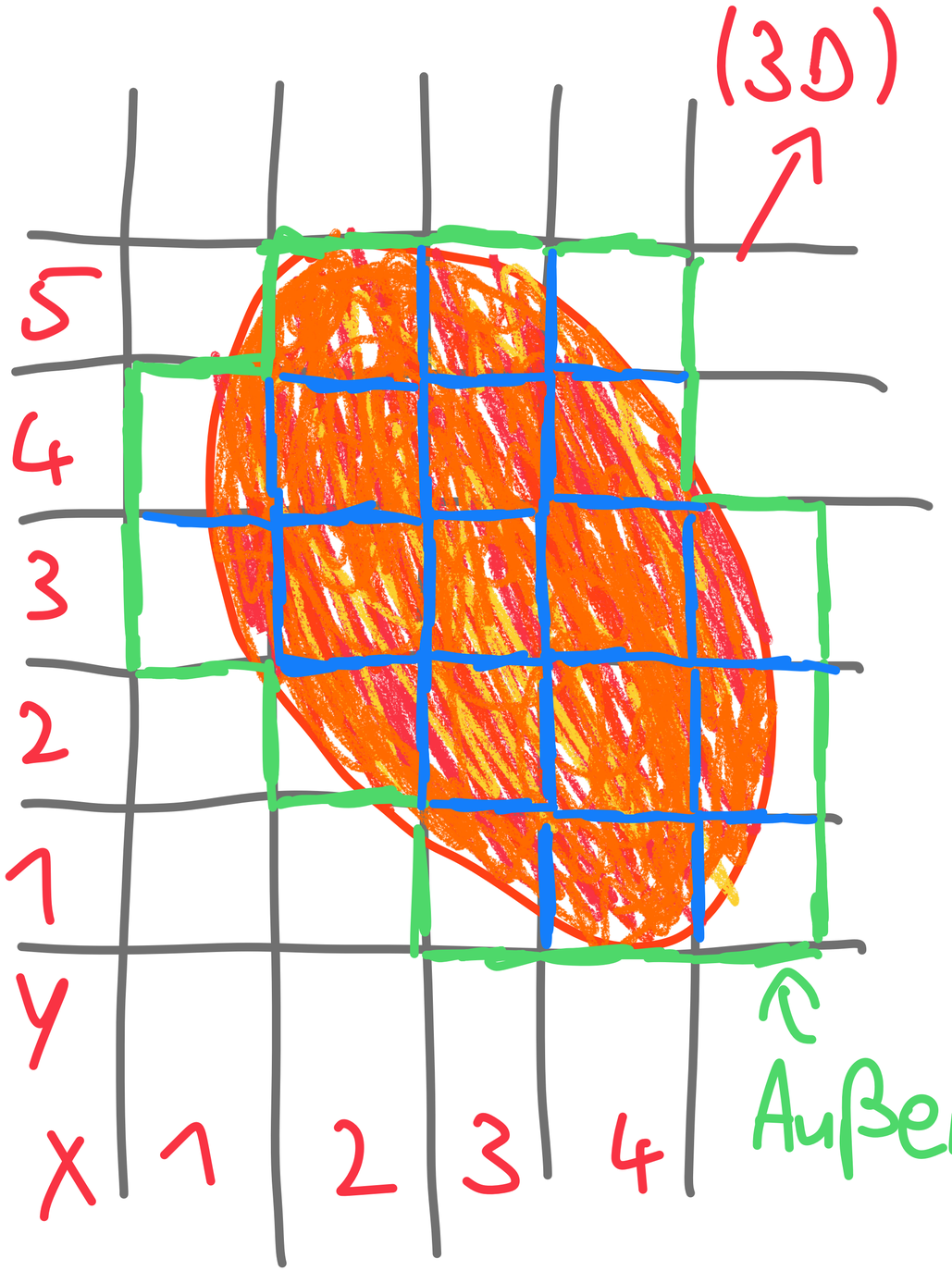


x	y	z
1	3	2
1	4	2
2	2	2
2	3	1
2	4	1
2	5	2
3	1	1
⋮		



← Fläche

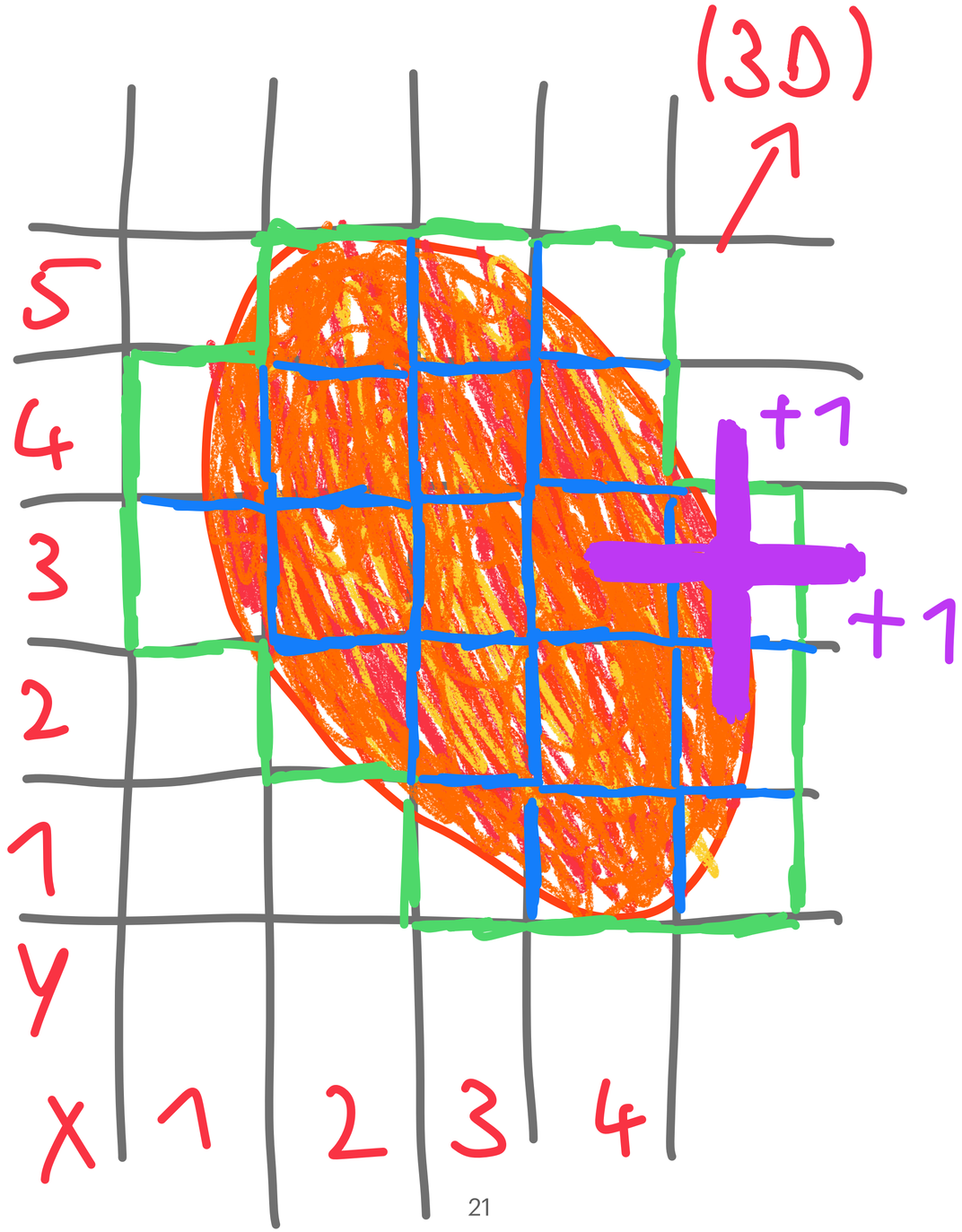
x	y	z
1	3	2
1	4	2
2	2	2
2	3	1
2	4	1
2	5	2
3	1	1
⋮		



x	y	(z)
1	3	
1	4	
2	2	
2	3	
2	4	
2	5	
3	1	
⋮		

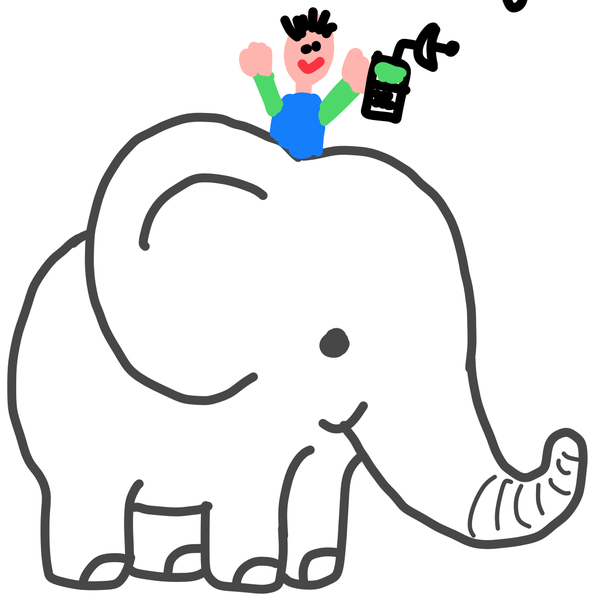
Part 1

Alle Seiten zählen, die nicht mit einem anderen Würfel verbunden sind

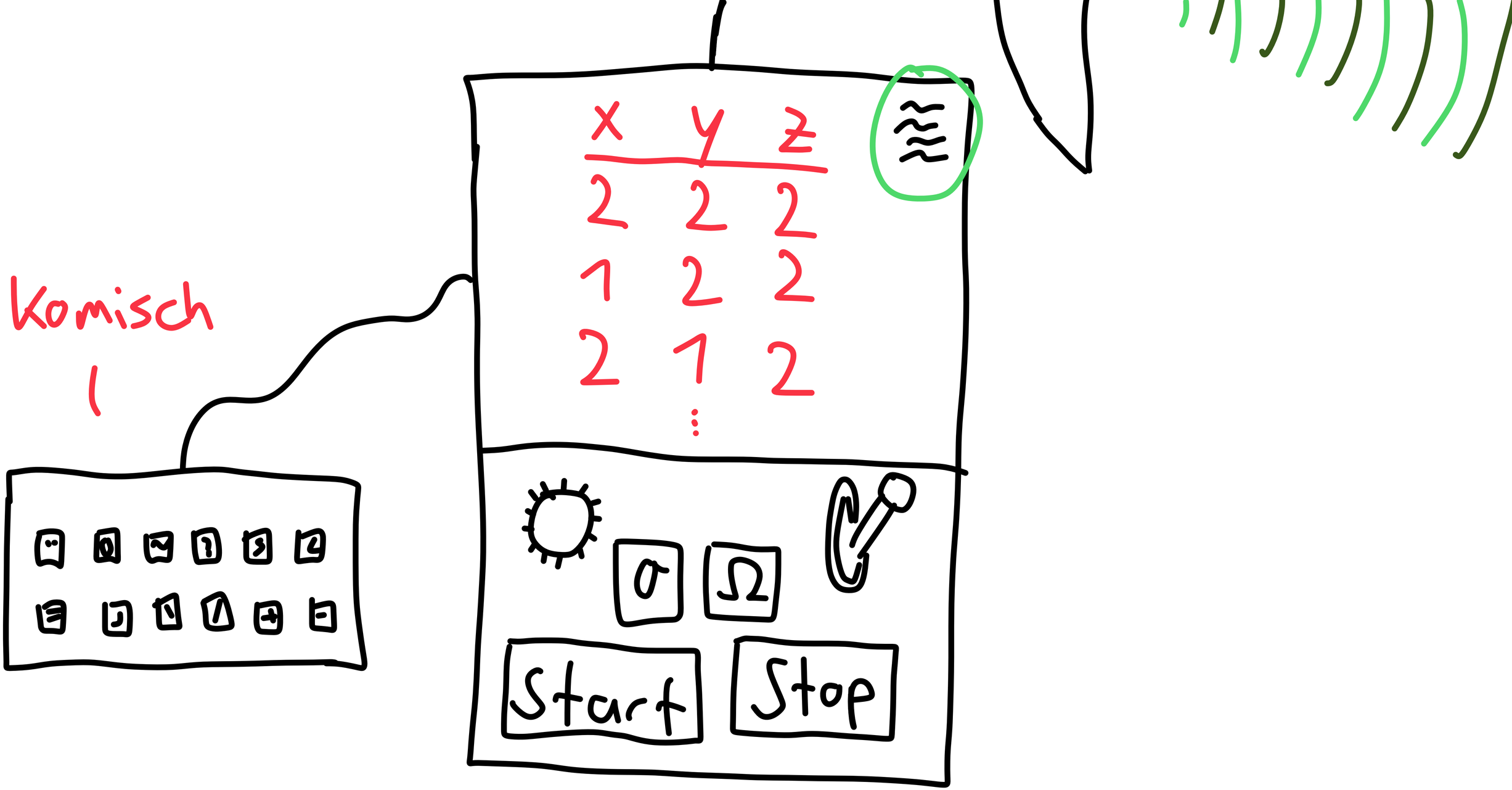

(3D)
5
4
3
2
1
y
x
1
2
3
4
+1
+1

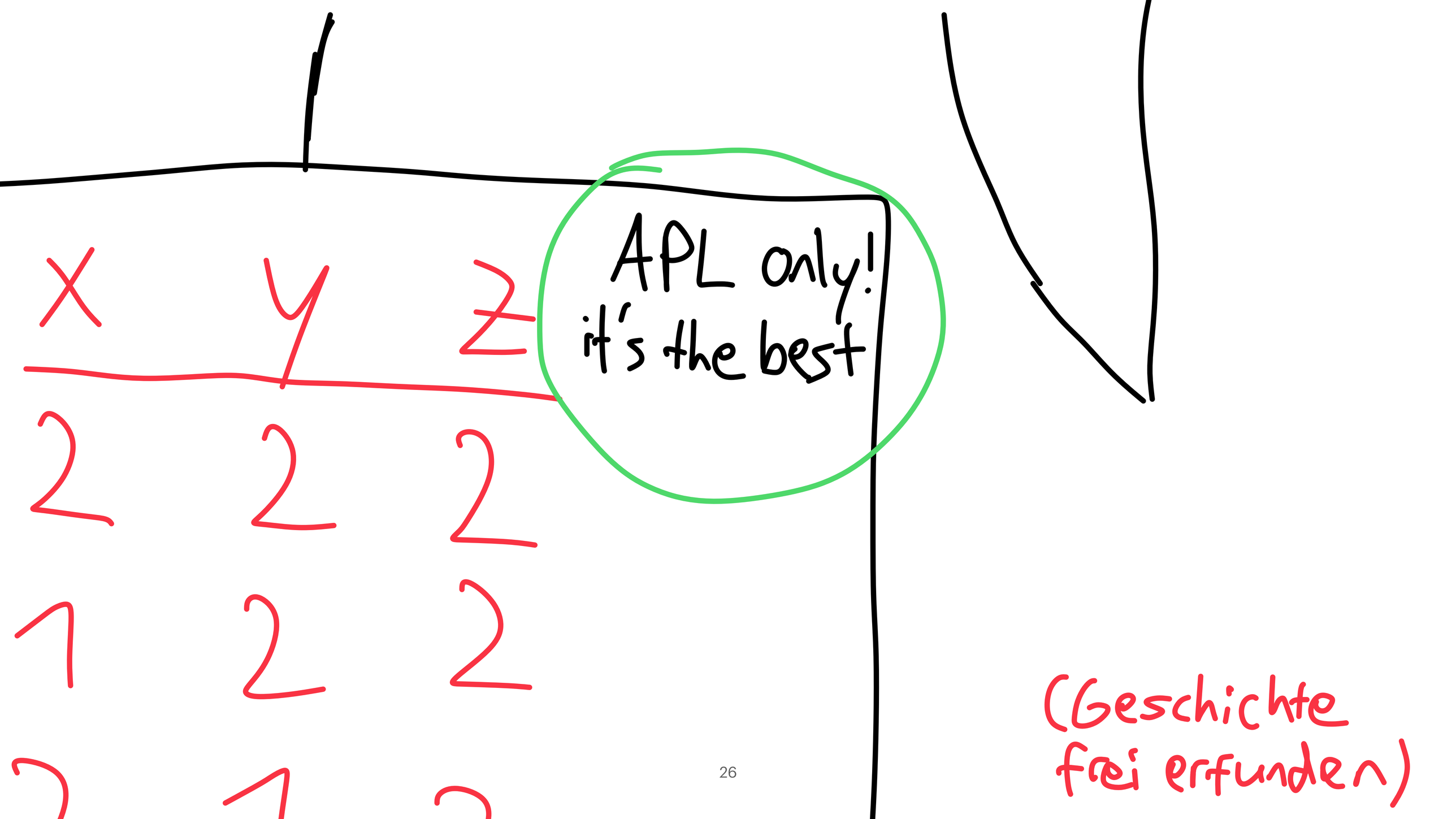
```
n = 0
for cube in cubes:
    for neighbor in neighbors(cube):
        if neighbor not in cubes:
            n += 1
```

```
n = 0
for cube in cubes:
    for neighbor in neighbors(cube):
        if neighbor not in cubes:
            n += 1
```



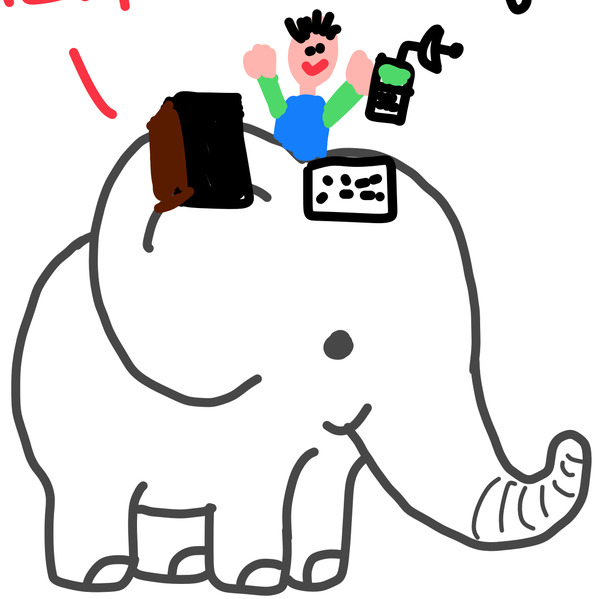
```
n = 0
for cube in cubes:
    for neighbor in neighbors(cube):
        if neighbor not in cubes:
            n += 1
```


Komisch
x y z
2 2 2
1 2 2
2 1 2
Start
Stop



(Geschichte frei erfunden)

Buch



,,oo APL2!

Buch: APL Hints

- Von rechts nach links lesen (Klammerung)
- Operatoren auf Arrays (monadisch/dyadisch)
- Komische Unicode Zeichen
- Bekannt für Unlesbarkeit (muss aber nicht!)
- Trains?!

APL IDE

```python
# Part 1
cubes = [eval(f"({line})") for line in open("input").readlines()]

movements = (lambda w: np.vstack([w, -w]))(np.identity(3, dtype=int))

sides = [tuple(c + m) for c in cubes for m in movements]

print(sum(1 for side in sides if side not in cubes))
```

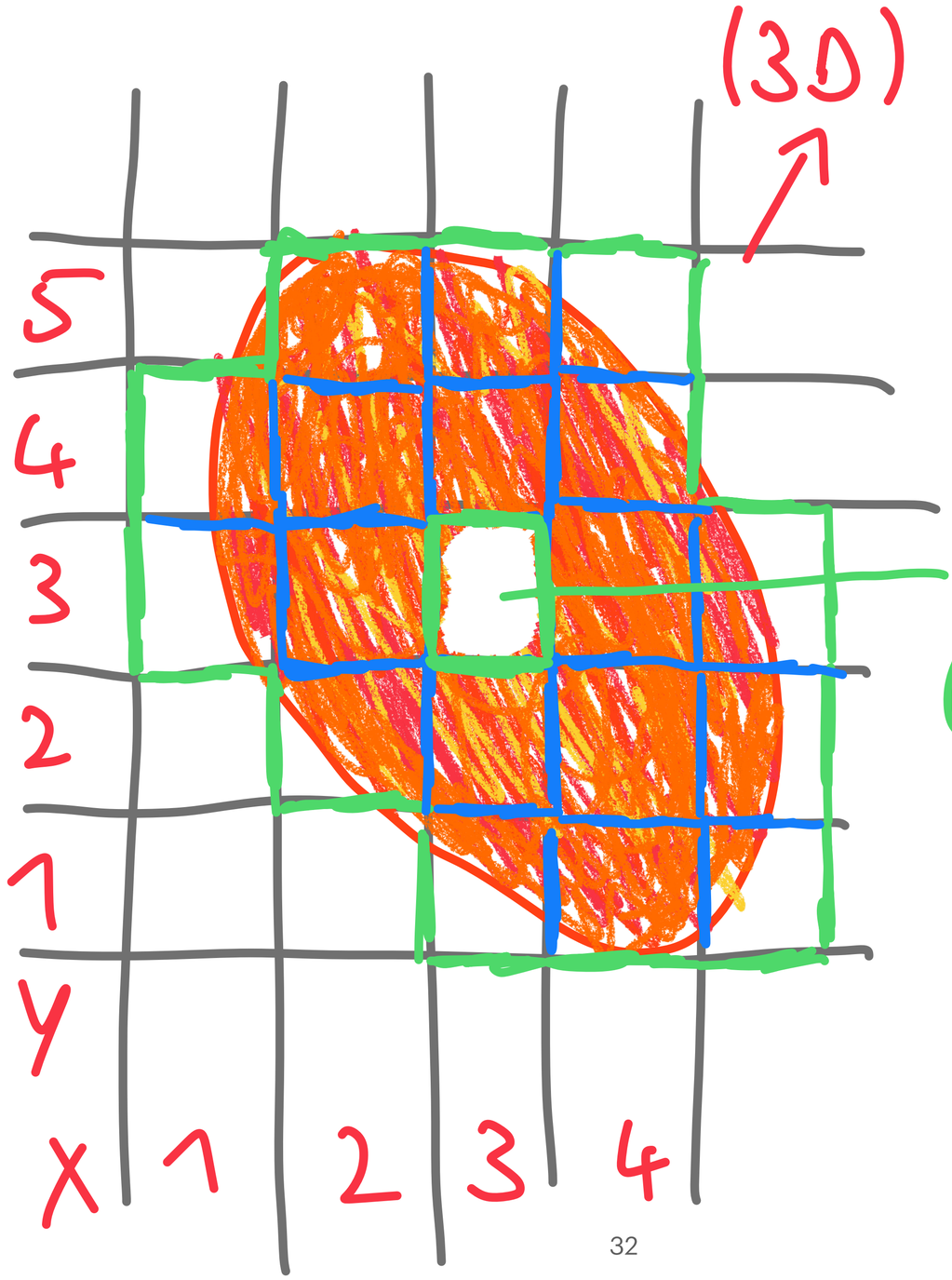
```
⍝ Part 1
cubes←⍎¨⊃⎕NGET'input'1

movements←{⍵,-⍵}↓(3 3⍴4↑1)

sides←,cubes∘.+movements

≠sides~cubes
```

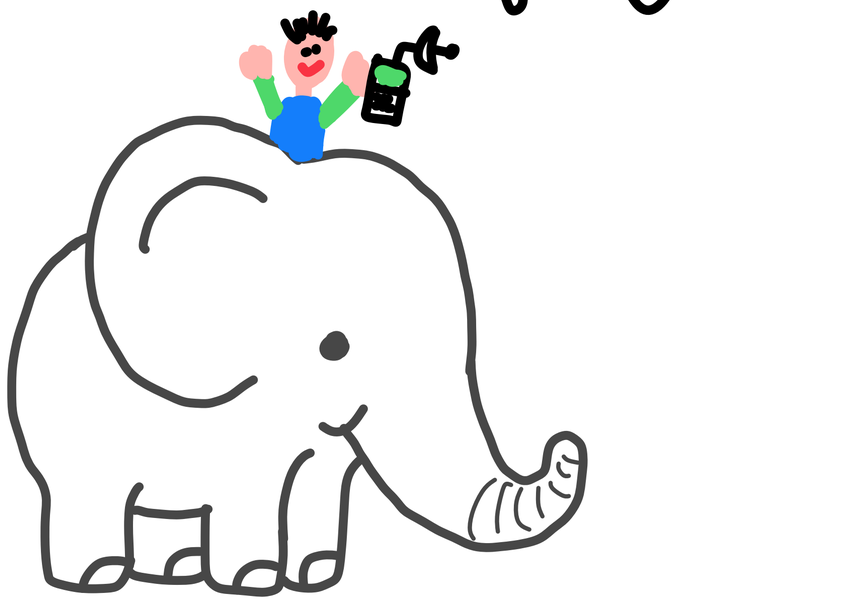
Part 2

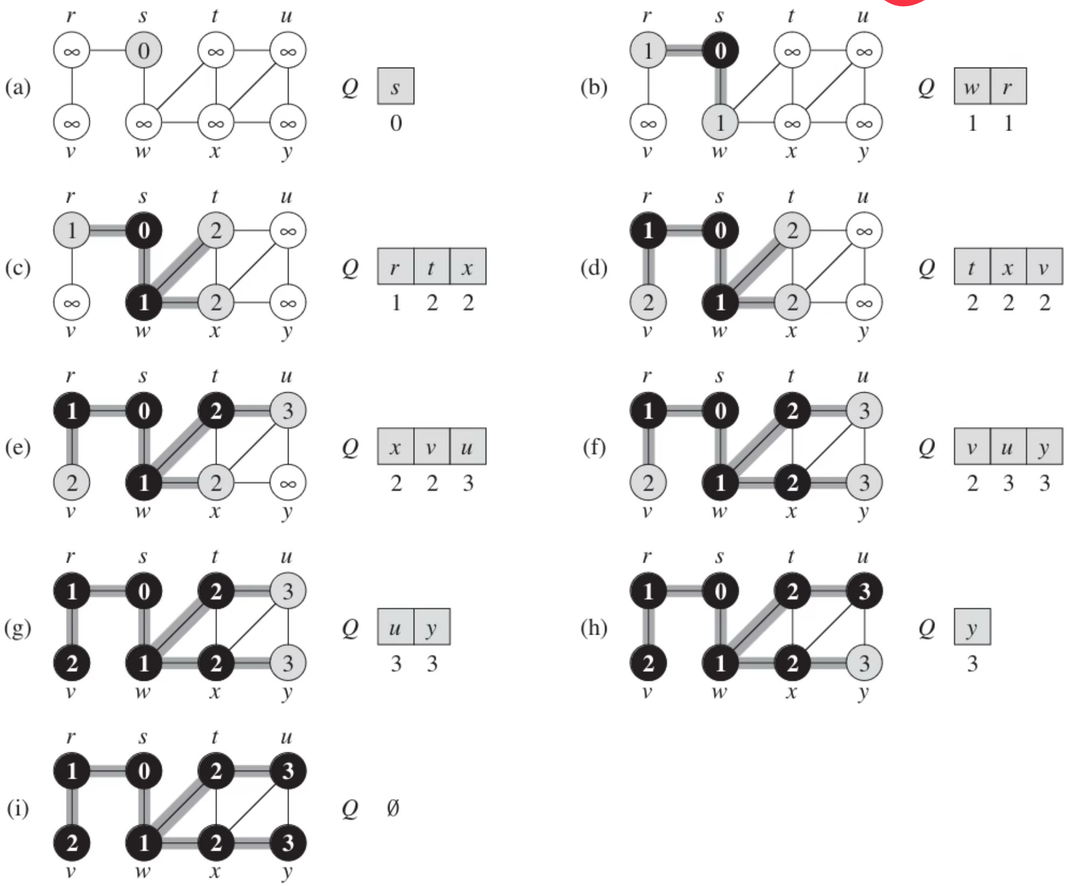


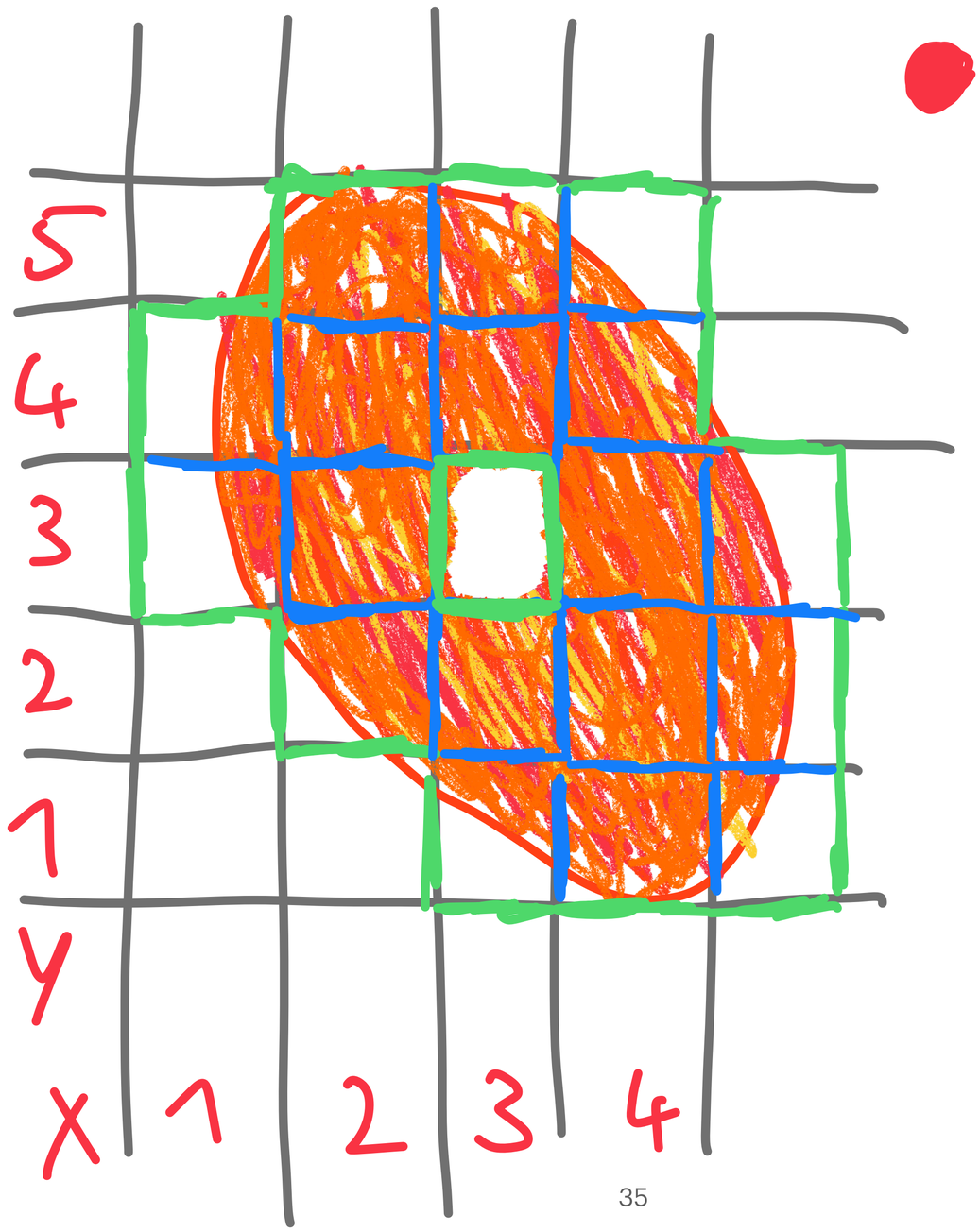
in Part 1
mitgezählt!

Wie groß ist die Fläche, die von außen allein durch ⬅️➡️⬆️⬇️ erreicht werden kann?

Algo Vorlesung !

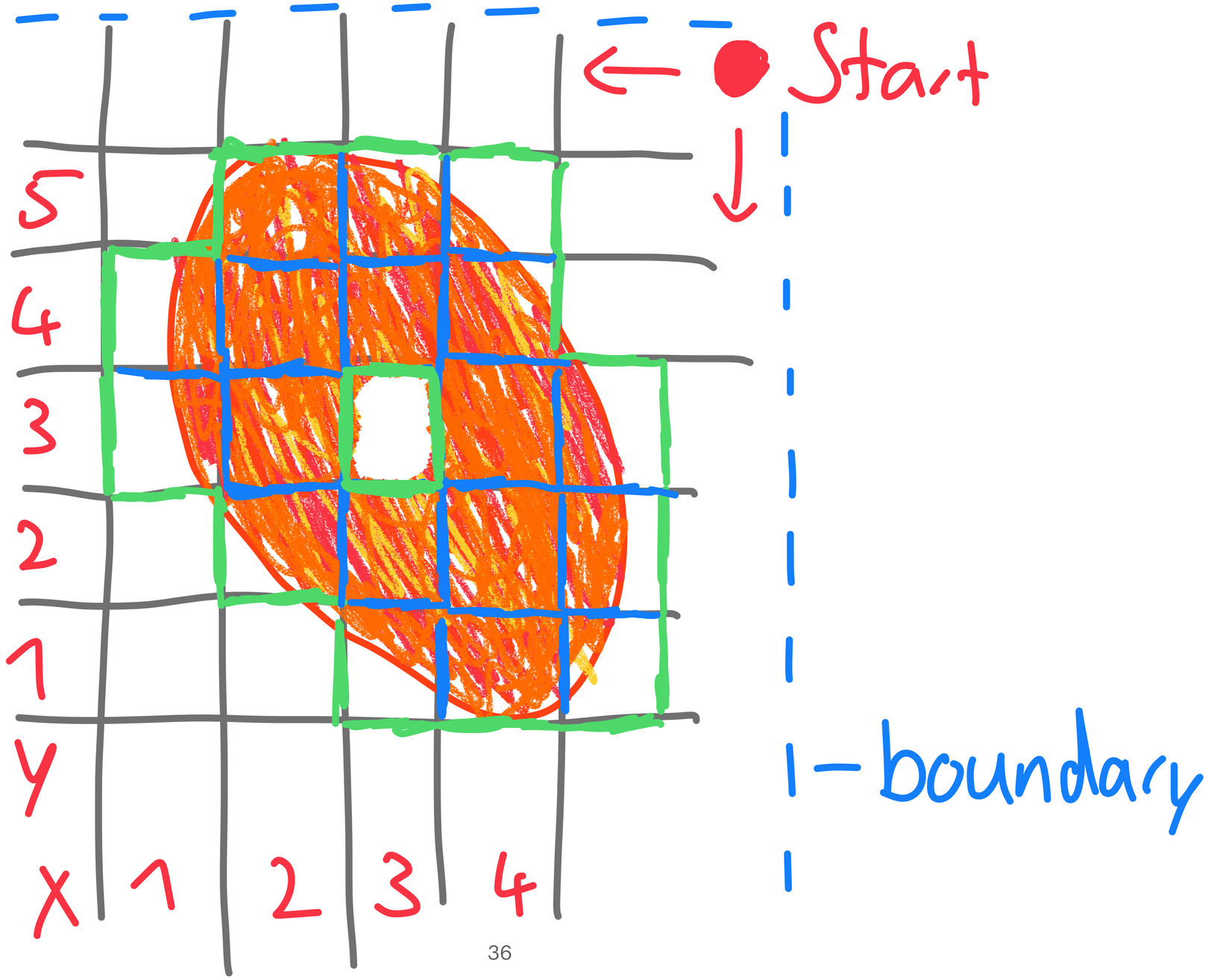


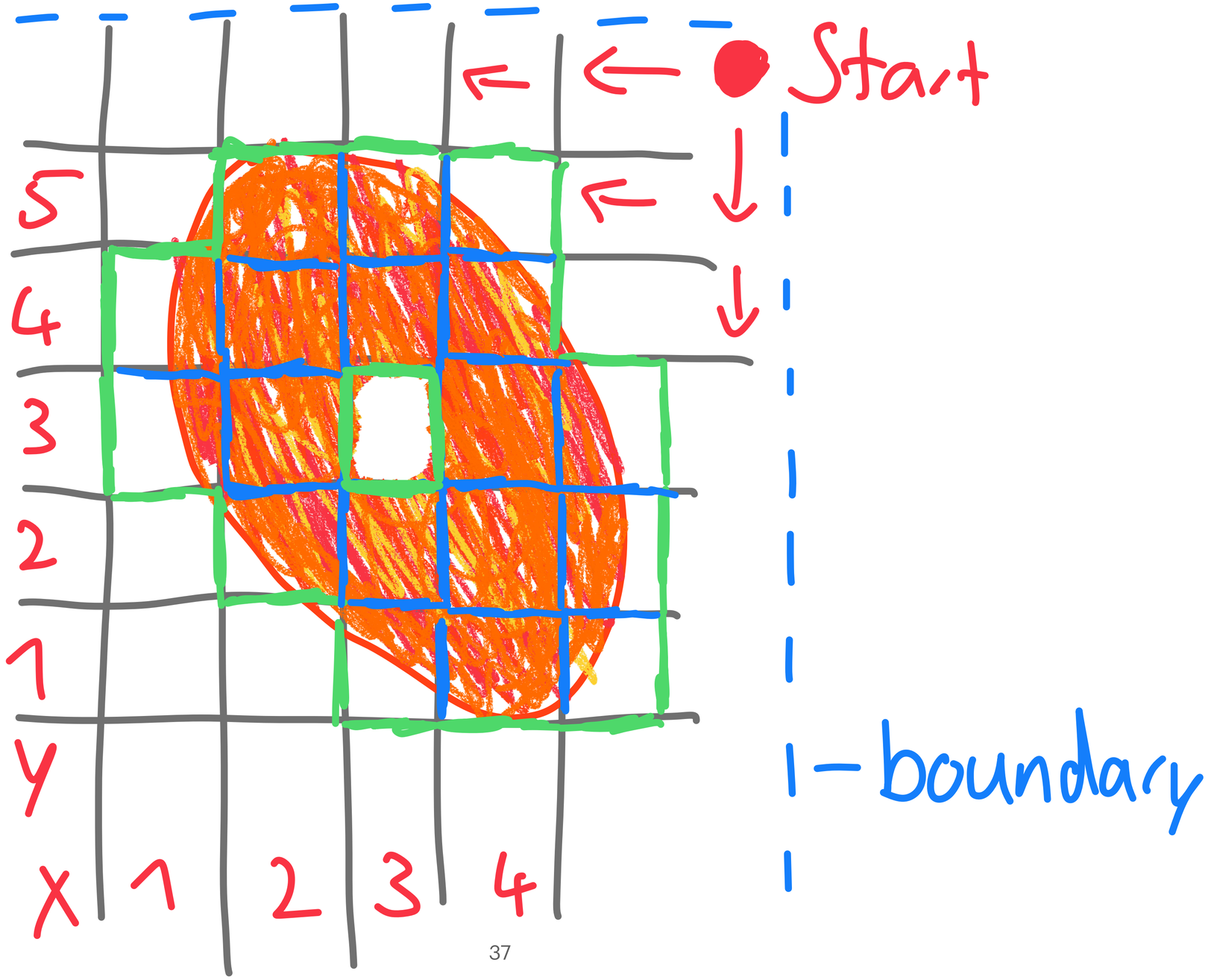


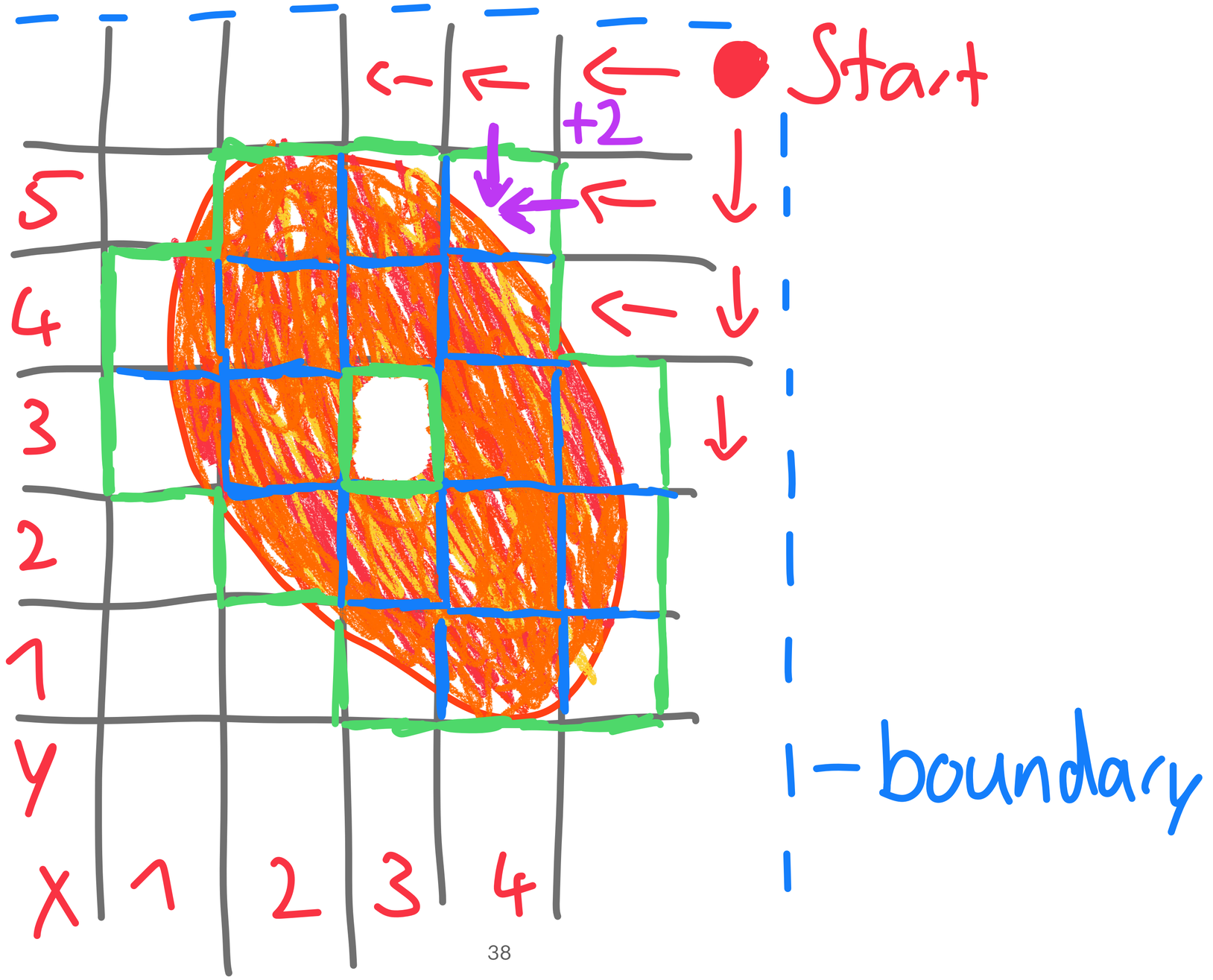


- Start: $\begin{pmatrix} \max x \\ \max y \\ \max z \end{pmatrix} + 1$

→ boundary analog


Start
5
4
3
2
1
y
x
1
2
3
4
boundary


Start
5
4
3
2
1
y
x
1
2
3
4
boundary


Start
+2
5
4
3
2
1
y
x
1
2
3
4
boundary

```
n = 0  # result
visited = set(cubes)
queue = [(xmax, ymax, zmax)]
while queue:
    cube = queue.pop(0)
    for neighbor in neighbors(cube):
        if not in_boundary(neighbor):
            continue
        if neighbor in cubes:
            n += 1
        if neighbor in visited:
            continue
        visited.add(neighbor)
        queue.append(neighbor)
```

APL IDE

```python
# Part 2
mins = tuple(min(c) - 1 for c in zip(*cubes))
maxs = tuple(max(c) + 1 for c in zip(*cubes))
in_boundary = lambda c: 0 == functools.reduce(
    lambda a, b: a + b,
    (int(c[i] < mins[i] or c[i] > maxs[i]) for i in range(3)),
    0,
)
count = 0
```

```apl
⍝ Part 2
mins←(↑⌊/cubes)-1 1 1
maxs←(↑⌈/cubes)+1 1 1
in_boundary←{0≡+/((⍵<mins),(⍵>maxs))}
count←0
```

```python
def rec(left, right):
    global count
    if not right:
        return count
    head, tail = right[0], right[1:]
    neighbors = [tuple(head + m) for m in movements]
    next = set(filter(in_boundary, neighbors))
    count += sum((n in cubes) for n in neighbors)
    return rec(left | set([head]), list((set(tail) | next) - left))
print(rec(set(cubes), [maxs]))
```

```
cubes{ ⍝ https://dfns.dyalog.com/n_bfs.htm
    0=≢⍵:count
    head←1↑⍵ ⋄ tail←1↓⍵
    neighbors←movements+¨head
    next←{(in_boundary¨⍵)/⍵}neighbors
    count⊢←count+(+/{(↓⍵)∊cubes}¨neighbors)
    (⍺,head)∇((tail∪next)~⍺)
}↓maxs
```

Visualisierung

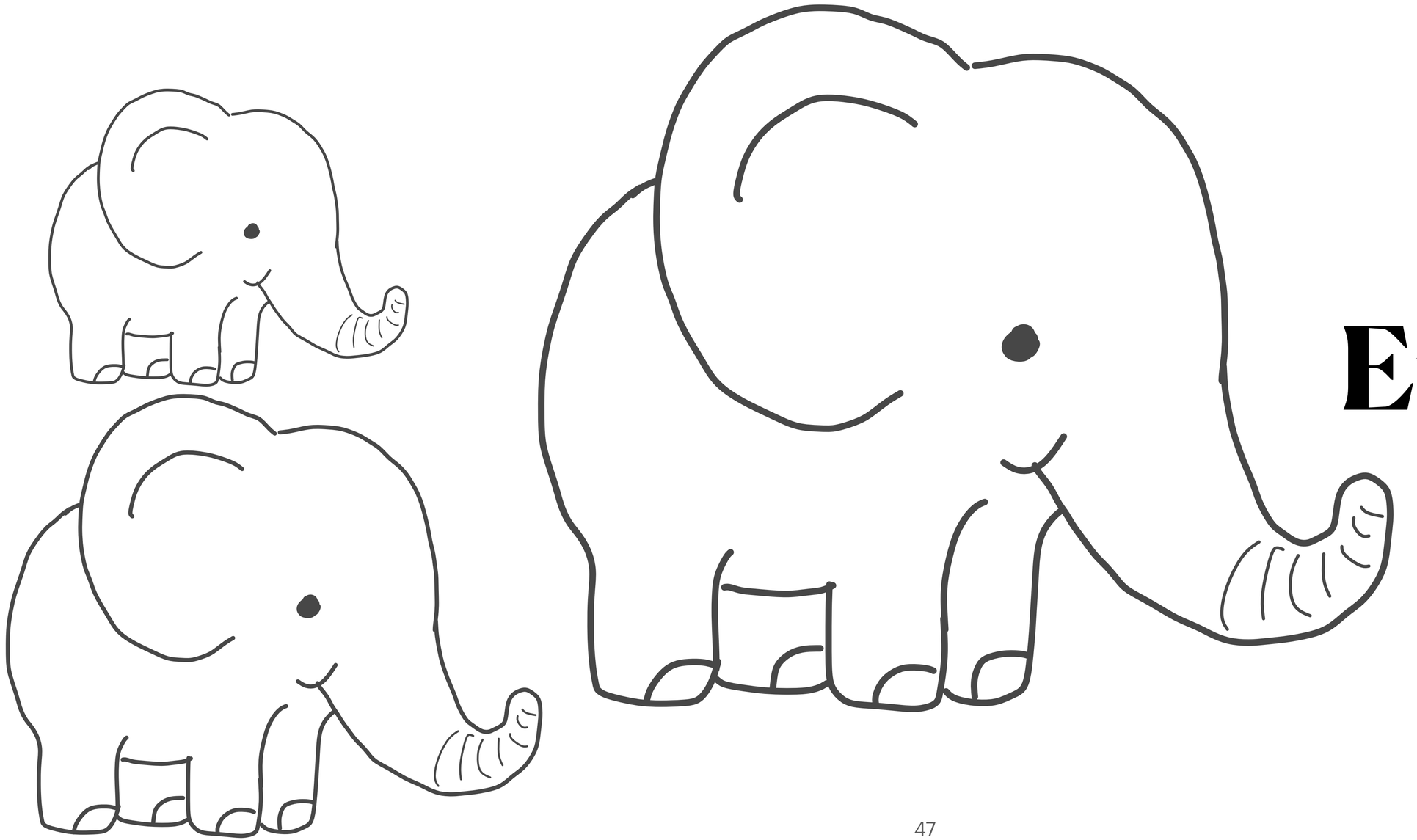
Diskussion

APL?!

- Warum?? Keine Erfahrung mit Array Programmiersprachen
- Macht Spaß, empfehlenswert
- Erst mit Übung leserlich
- Elegante Lösungen „for free“
- Gut für golfing

Andere Ansätze

- Graph: Strongly Connected Components
- D/BFS Varianten (parallelisiert, Flood Fill)



Ende

Worst-Case Part 1

- Großer Lavatropfen (Volumen)
- O(n), |Würfel in Tropfen|

Worst-Case Part 2

- Große boundary (n^3), kleines Volumen
- O(n), |Würfel in boundary|