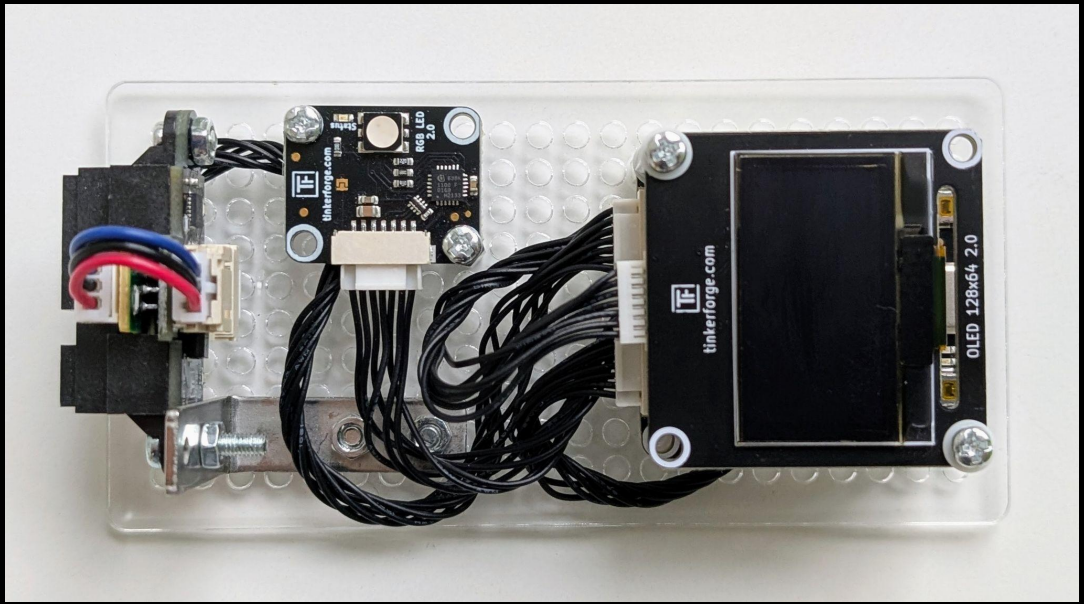
These slides serve as a visual aid for the lecture, not as a comprehensive document or script.

Please refrain from printing these slides to help protect the environment.

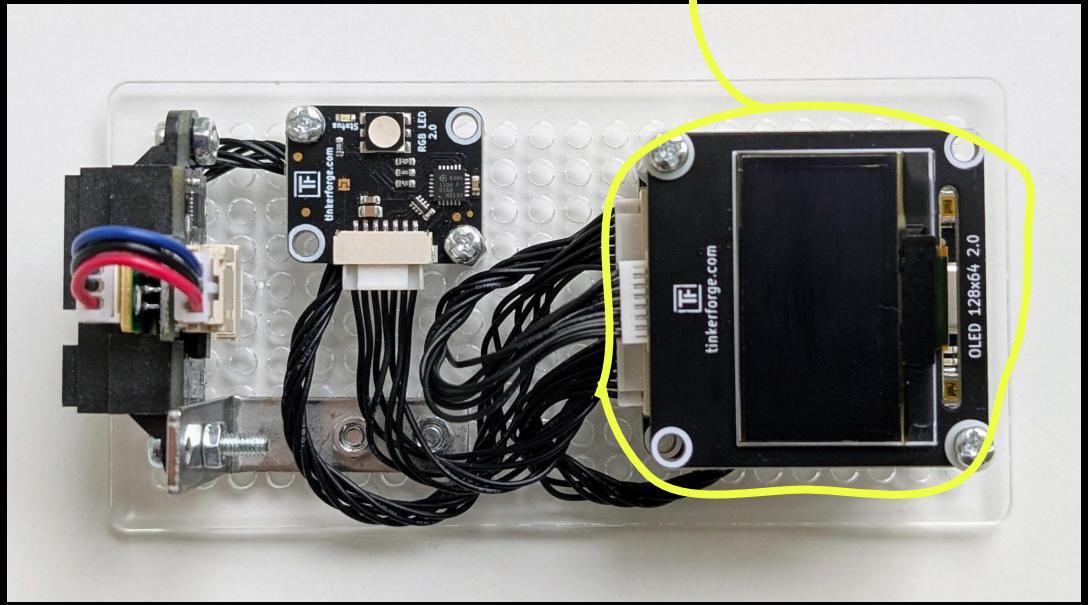
For any comments or feedback, please contact n.meseth@hs-osnabrueck.de.

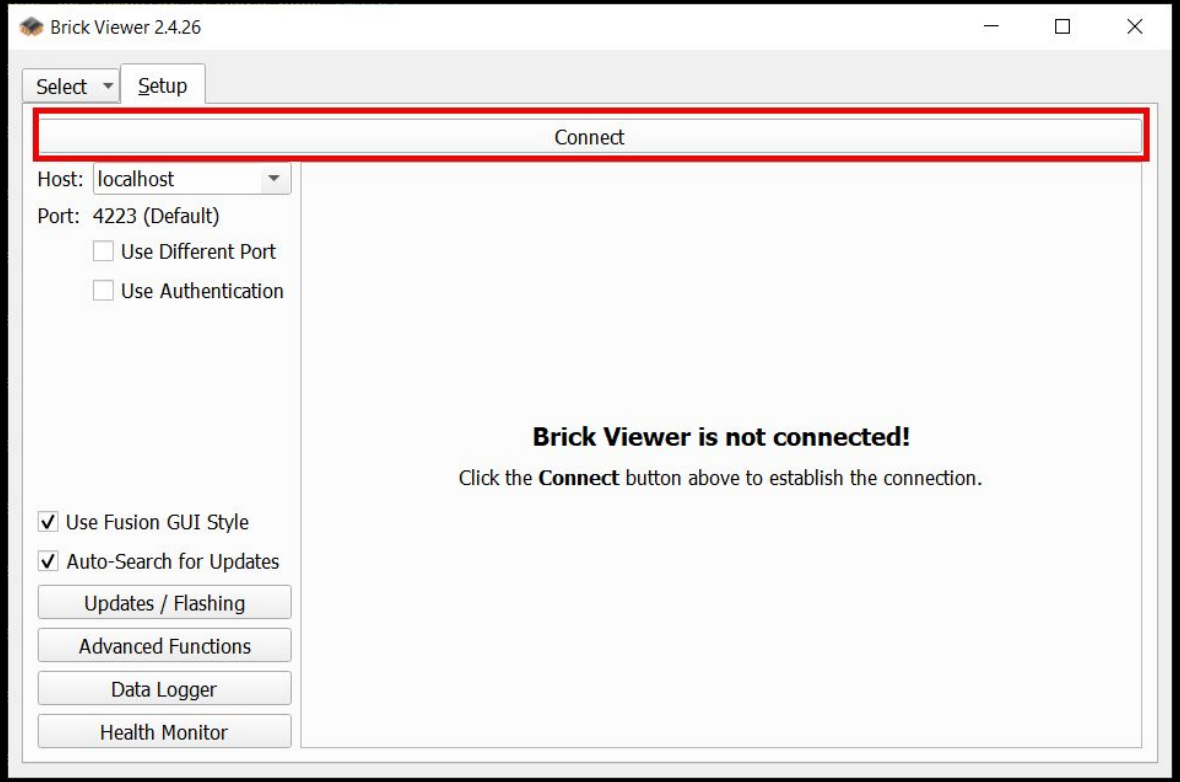
IMAGES

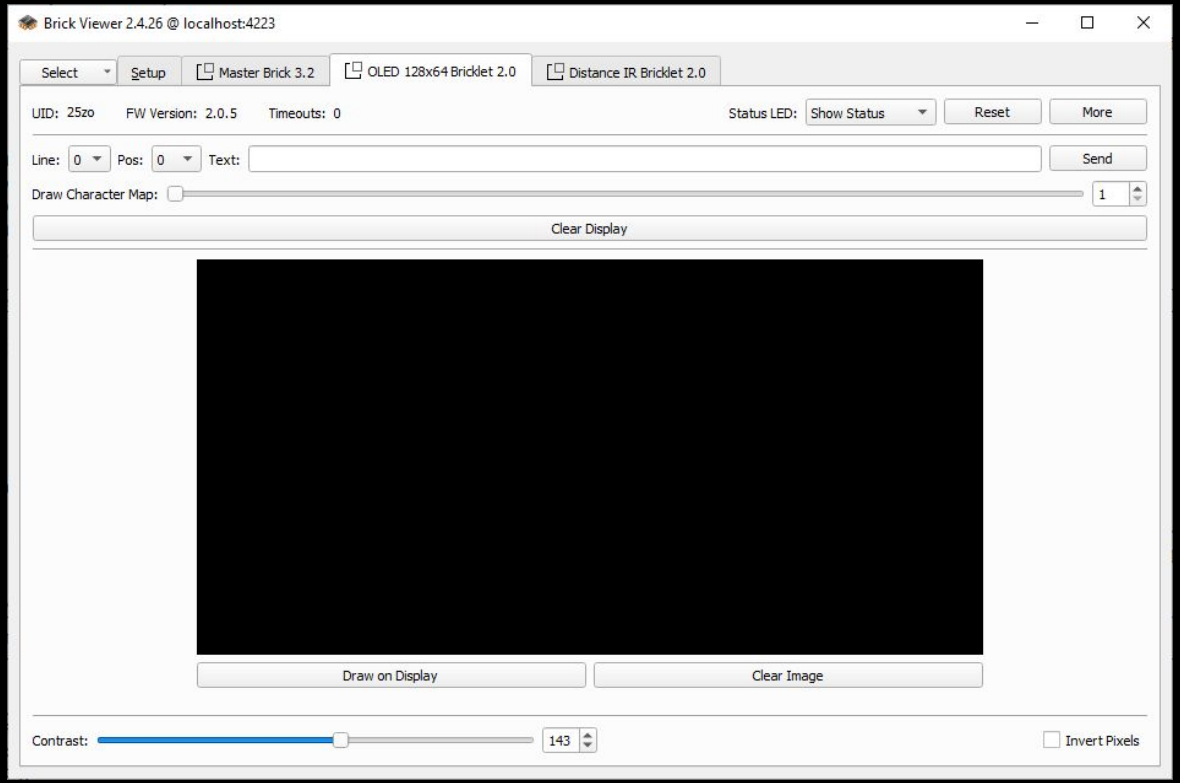
Supporting slides for chapter 4 of the book
Hands-On Computer Science

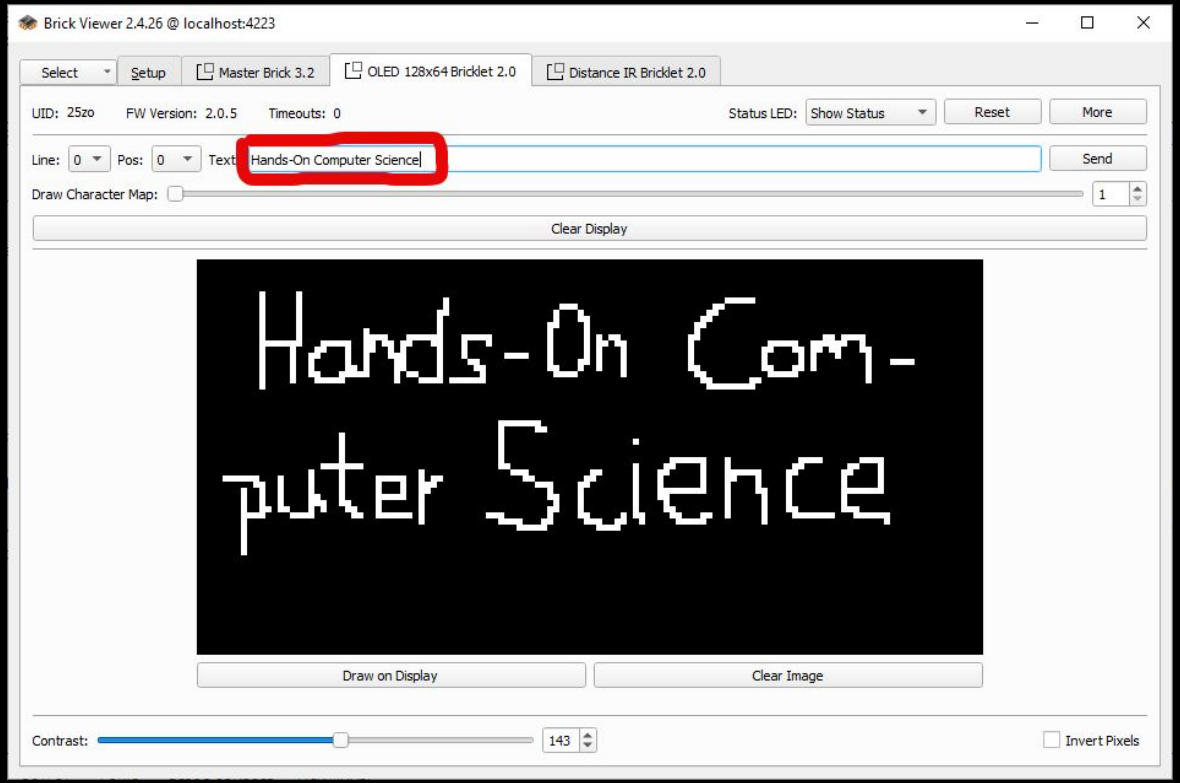

RGB LED 2.0
tinkerforge.com
tinkerforge.com
OLED 128x64 2.0

OLED Display




Brick Viewer 2.4.26
Select
Setup
Connect
Host: localhost
Port: 4223 (Default)
Use Different Port
Use Authentication
Brick Viewer is not connected!
Click the Connect button above to establish the connection.
Use Fusion GUI Style
Auto-Search for Updates
Updates / Flashing
Advanced Functions
Data Logger
Health Monitor


Brick Viewer 2.4.26 @ localhost:4223
Select
Setup
Master Brick 3.2
OLED 128x64 Bricklet 2.0
Distance IR Bricklet 2.0
UID: 25zo
FW Version: 2.0.5
Timeouts: 0
Status LED:
Show Status
Reset
More
Line:
0
Pos:
0
Text:
Send
Draw Character Map:
1
Clear Display
Draw on Display
Clear Image
Contrast:
143
Invert Pixels


Brick Viewer 2.4.26 @ localhost:4223
Select
Setup
Master Brick 3.2
OLED 128x64 Bricklet 2.0
Distance IR Bricklet 2.0
UID: 25zo
FW Version: 2.0.5
Timeouts: 0
Status LED:
Show Status
Reset
More
Line:
0
Pos:
0
Text:
Hands-On Computer Science
Send
Draw Character Map:
1
Clear Display
Hands-On Com-
puter Science
Draw on Display
Clear Image
Contrast:
143
Invert Pixels

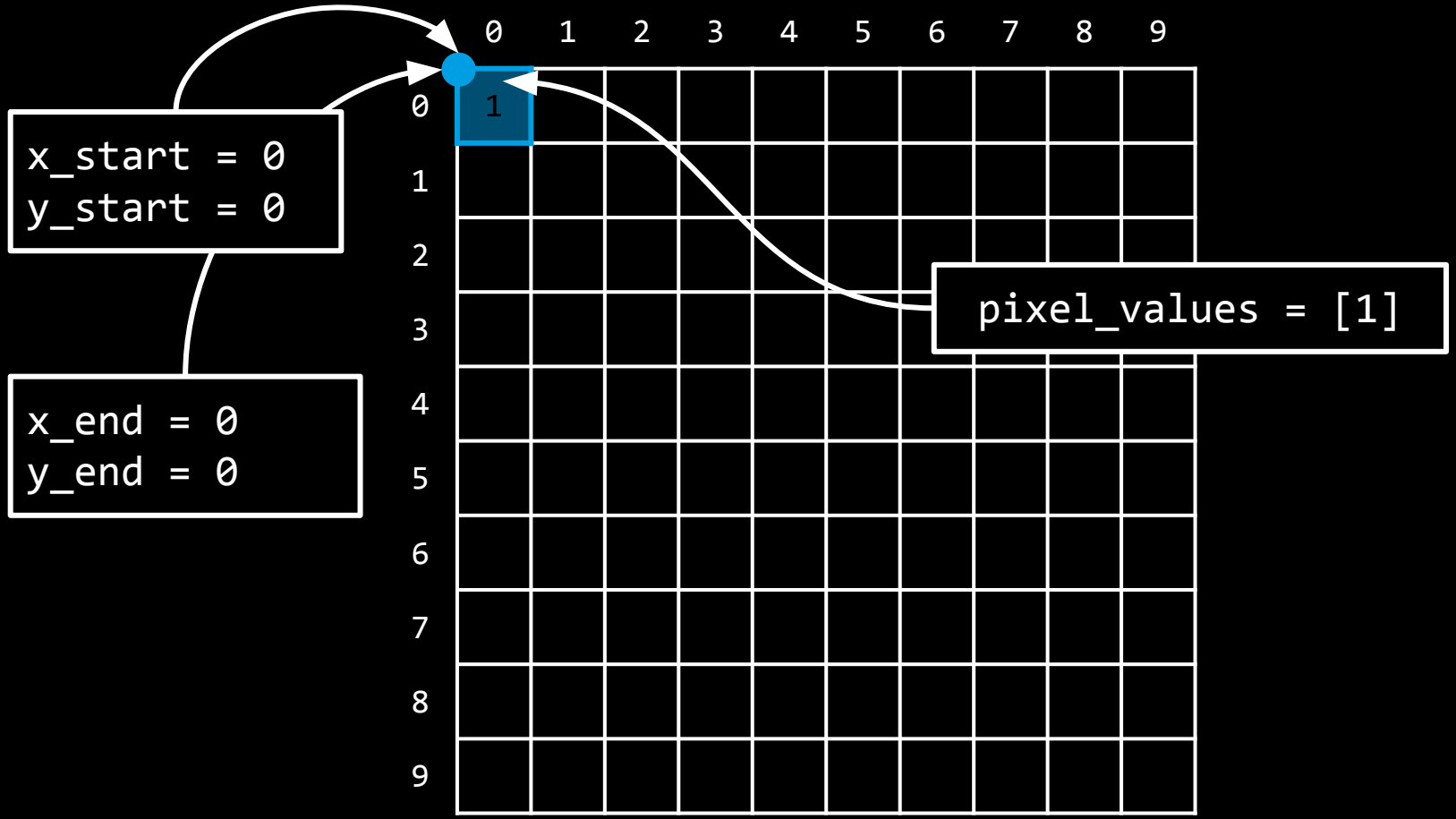
boilerplate code

```
from tinkerforge.ip_connection import IPConnection
from tinkerforge.bricklet_oled_128x64_v2 import BrickletOLED128x64V2

ipcon = IPConnection()
ipcon.connect("localhost", 4223)
oled = BrickletOLED128x64V2("25zo", ipcon)
```

clearing the display

```
from tinkerforge.ip_connection import IPConnection
from tinkerforge.bricklet_oled_128x64_v2 import BrickletOLED128x64V2

ipcon = IPConnection()
ipcon.connect("localhost", 4223)
oled = BrickletOLED128x64V2("25zo", ipcon)

oled.clear_display()
```

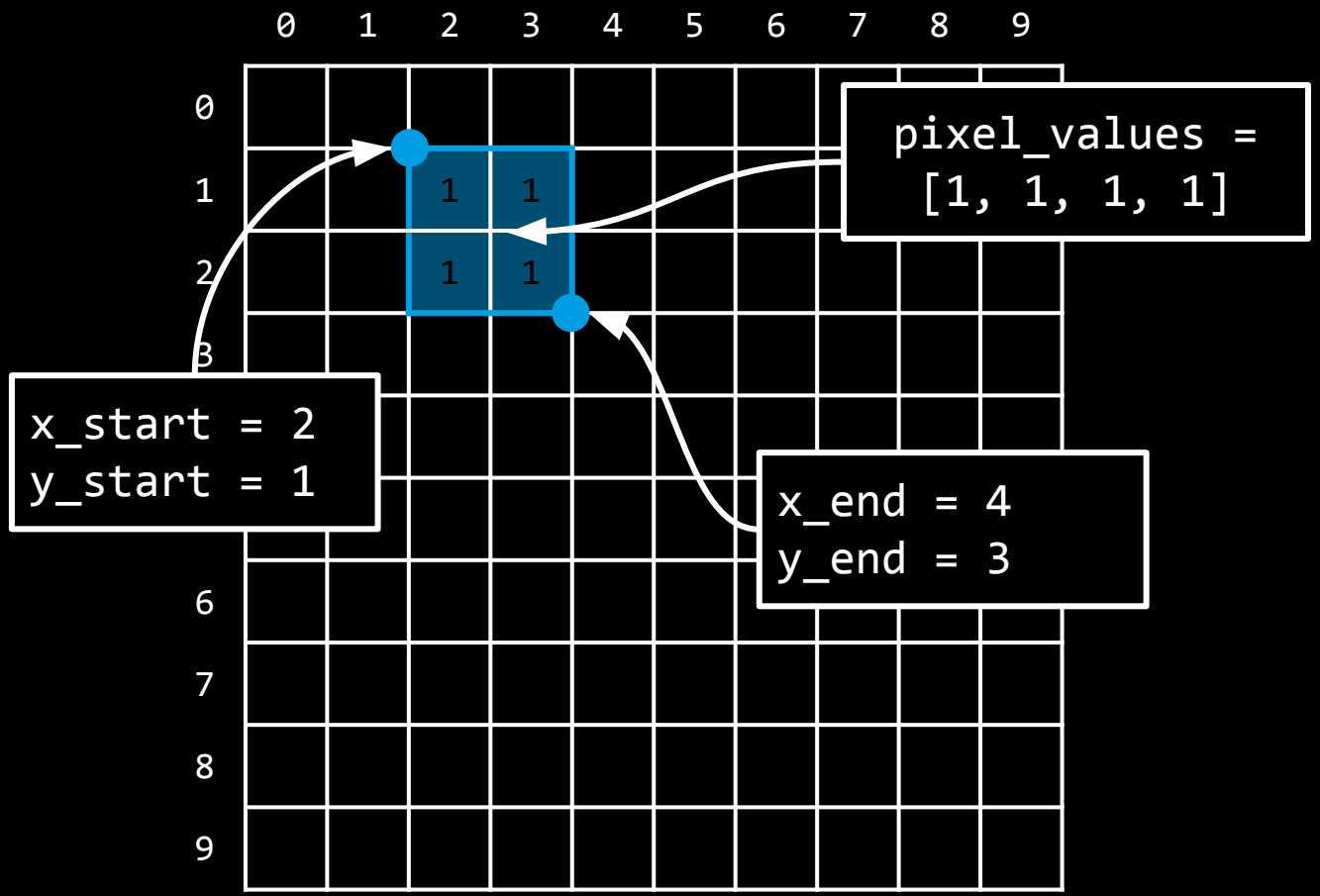
PIXELS


0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
1
x_start = 0
y_start = 0
x_end = 0
y_end = 0
pixel_values = [1]

writing a pixel

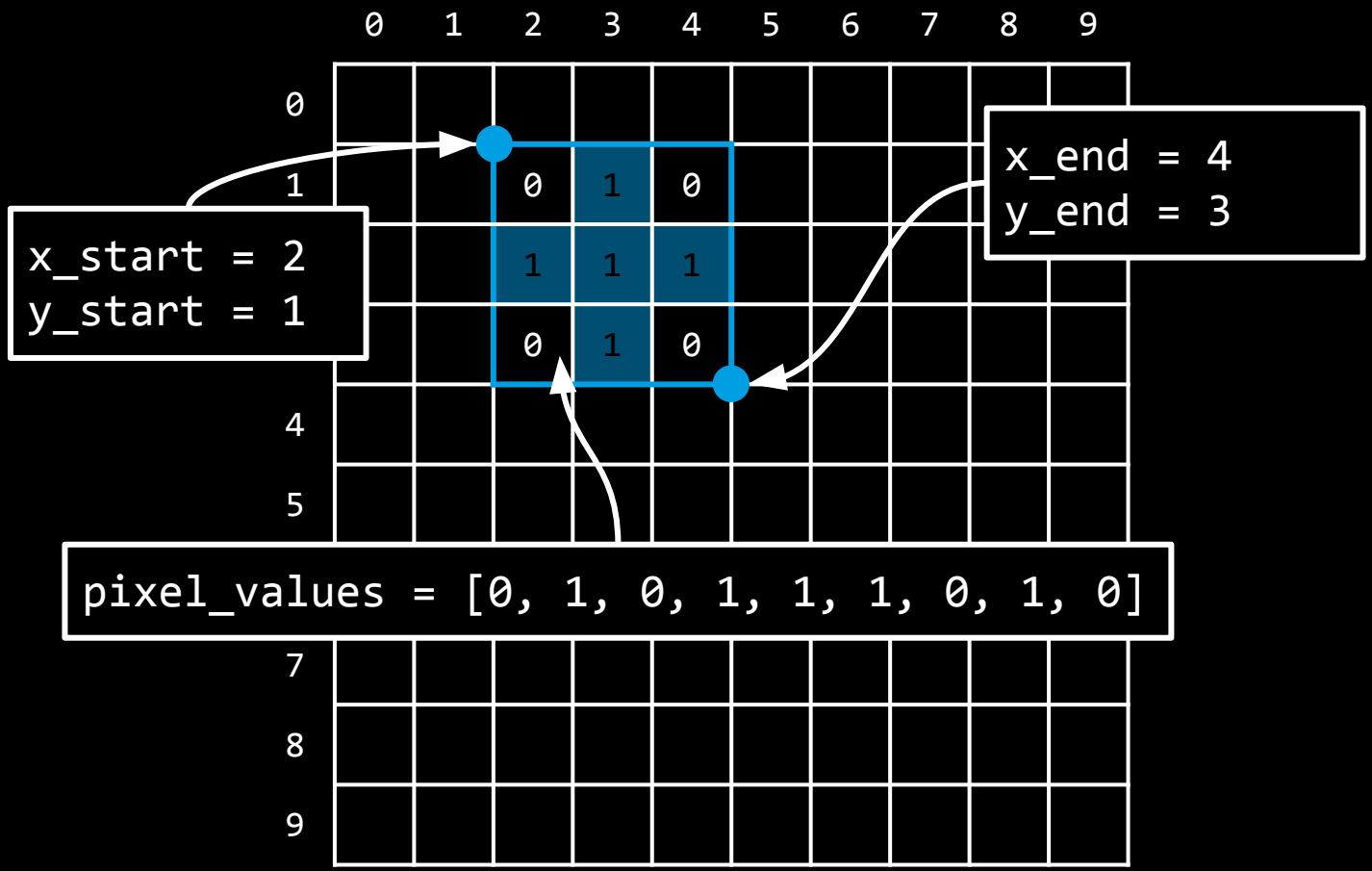
```
oled.write_pixels(0, 0, 0, 0, [1])
```

BITMAPS

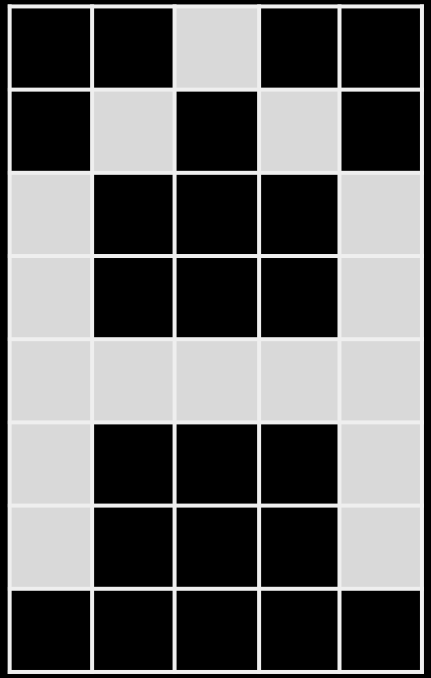

0 1 2 3 4 5 6 7 8 9
0
1
2
3
6
7
8
9
1 1
1 1
pixel_values =
[1, 1, 1, 1]
x_start = 2
y_start = 1
x_end = 4
y_end = 3

writing multiple pixels

```
oled.write_pixels(62, 30, 63, 31, [1, 1, 1, 1])
```


0 1 2 3 4 5 6 7 8 9
0
1
4
5
7
8
9
0 1 0
1 1 1
0 1 0
x_start = 2
y_start = 1
x_end = 4
y_end = 3
pixel_values = [0, 1, 0, 1, 1, 1, 0, 1, 0]

LETTERS



	0	1	2	3	4
0	0	0	1	0	0
1	0	1	0	1	0
2	1	0	0	0	1
3	1	0	0	0	1
4	1	1	1	1	1
5	1	0	0	0	1
6	1	0	0	0	1
7	0	0	0	0	0

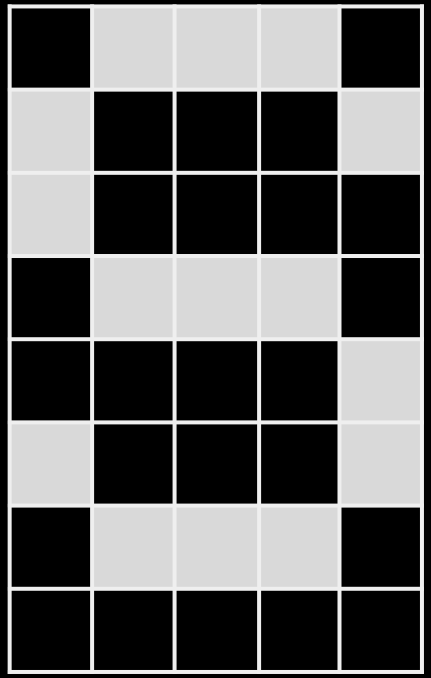
letter 'A' as a bitmap

```
letter_a_bitmap = [
    0, 0, 1, 0, 0,
    0, 1, 0, 1, 0,
    1, 0, 0, 0, 1,
    1, 0, 0, 0, 1,
    1, 1, 1, 1, 1,
    1, 0, 0, 0, 1,
    1, 0, 0, 0, 1,
    0, 0, 0, 0, 0
]
```

writing an 'A' as a bitmap

```
oled.write_pixels(1, 10, 5, 17, letter_a_bitmap)
```

```
letter_a_bitmap = [
    0, 0, 1, 0, 0,
    0, 1, 0, 1, 0,
    1, 0, 0, 0, 1,
    1, 0, 0, 0, 1,
    1, 1, 1, 1, 1,
    1, 0, 0, 0, 1,
    1, 0, 0, 0, 1,
    0, 0, 0, 0, 0
]
```



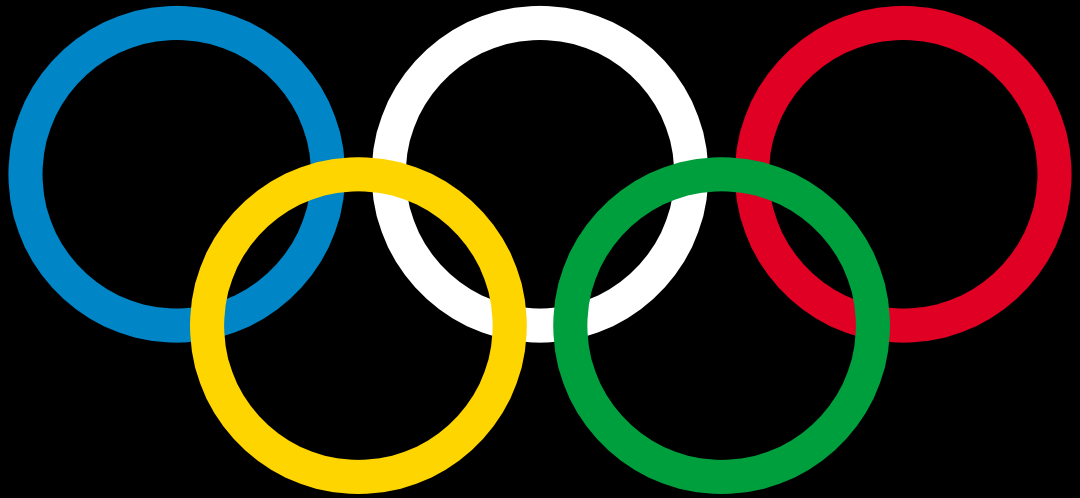
	0	1	2	3	4
0	0	1	1	1	0
1	1	0	0	0	1
2	1	0	0	0	0
3	0	1	1	1	0
4	0	0	0	0	1
5	1	0	0	0	1
6	0	1	1	1	0
7	0	0	0	0	0

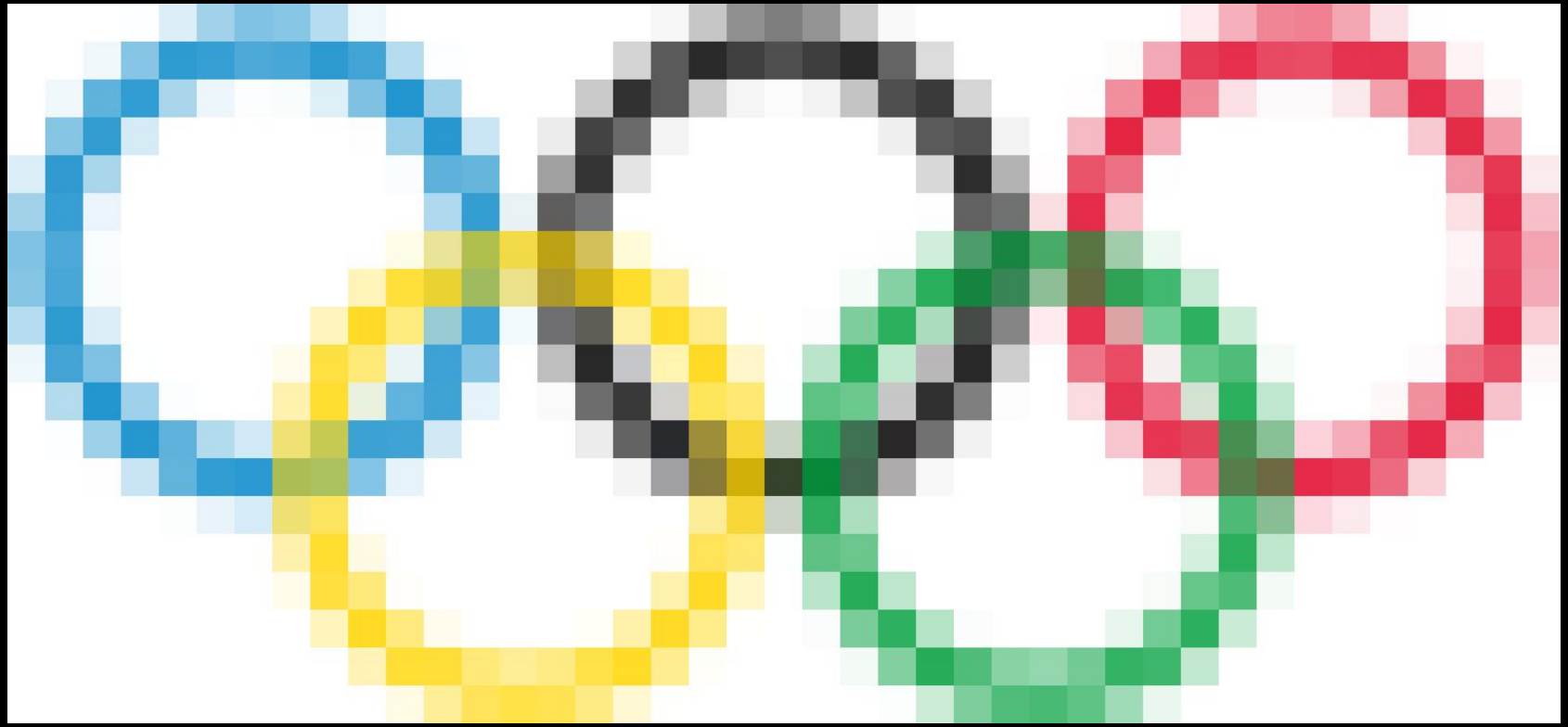
VECTOR GRAPHICS

Source: Wikipedia

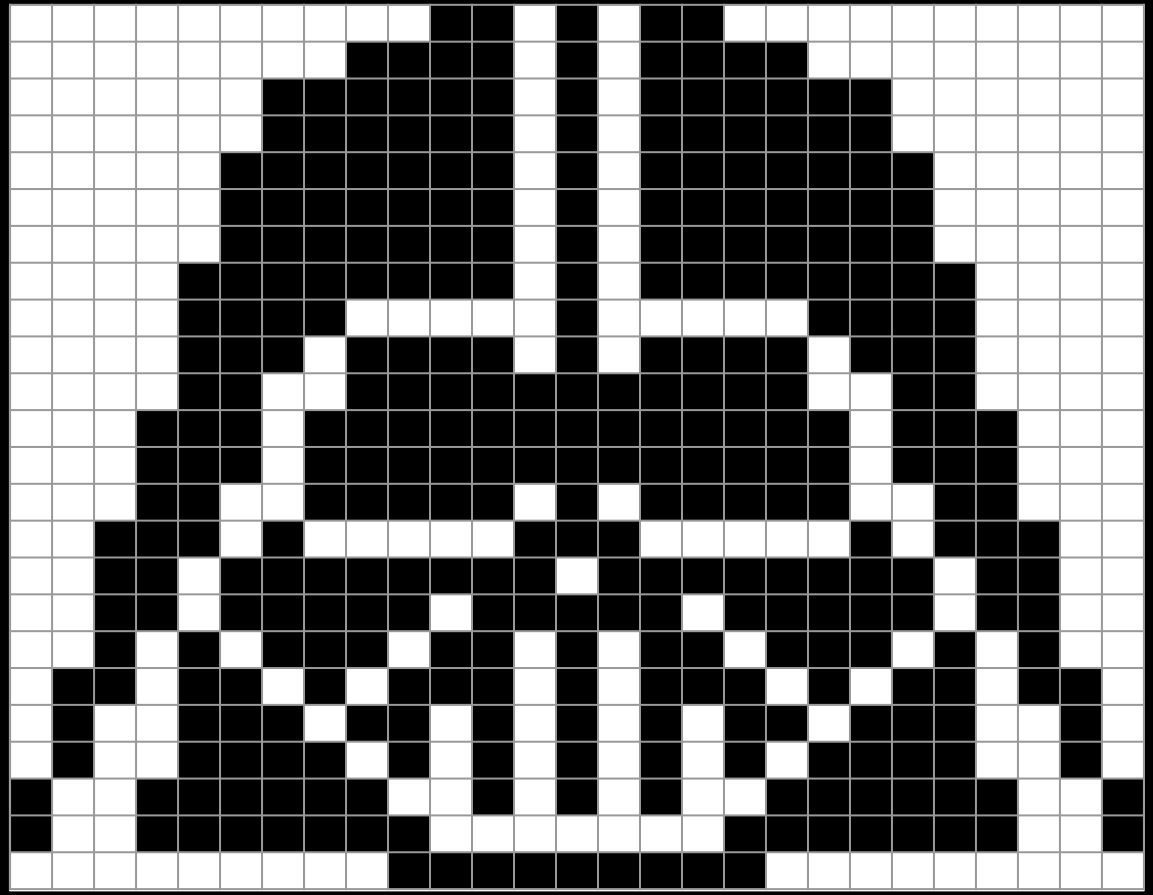
```
<svg width="440" height="220" xmlns="http://www.w3.org/2000/svg">
  <circle cx="60" cy="60" r="50" stroke="#0085C7" stroke-width="10" fill="none" />
  <circle cx="180" cy="60" r="50" stroke="#FFFFFF" stroke-width="10" fill="none" />
  <circle cx="300" cy="60" r="50" stroke="#DF0024" stroke-width="10" fill="none" />
  <circle cx="120" cy="110" r="50" stroke="#FFD500" stroke-width="10" fill="none" />
  <circle cx="240" cy="110" r="50" stroke="#009F3D" stroke-width="10" fill="none" />
</svg>
```

```
<svg width="440" height="220" xmlns="http://www.w3.org/2000/svg">
  <circle cx="60" cy="60" r="50" stroke="#0085C7" stroke-width="10" fill="none" />
  <circle cx="180" cy="60" r="50" stroke="#FFFFFF" stroke-width="10" fill="none" />
  <circle cx="300" cy="60" r="50" stroke="#DF0024" stroke-width="10" fill="none" />
  <circle cx="120" cy="110" r="50" stroke="#FFD500" stroke-width="10" fill="none" />
  <circle cx="240" cy="110" r="50" stroke="#009F3D" stroke-width="10" fill="none" />
</svg>
```

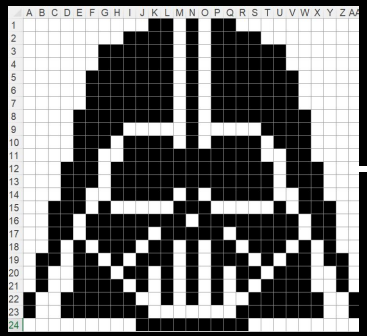




FROM BITS TO IMAGE



xlsx



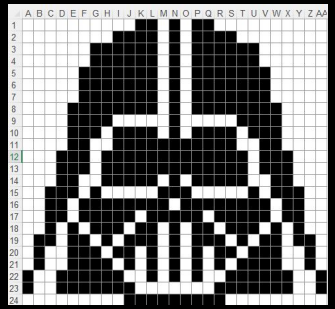
program

binary list

[0, 0, 0, …, 1, 1]

pixelart with excel

```
from openpyxl import load_workbook
workbook = load_workbook("Darth Vader Pixel Art.xlsx")
sheet = workbook["Darth Vader"]
```



pixelart with excel

```
for row in sheet.iter_rows():
    for cell in row:
        color = getattr(cell.fill.fgColor, "rgb", None)
        print(color)
```

pixelart with excel

```
for row in sheet.iter_rows():
    for cell in row:
        color = getattr(cell.fill.fgColor, "rgb", None)
        print(color)
```

RGB as hex values

pixelart with excel

```
for row in sheet.iter_rows():
    for cell in row:
        color = getattr(cell.fill.fgColor, "rgb", None)
        print(color)
```

RGB as hex values

```
FFFF0000
00000000
...
```

pixelart with excel

```
bits = []
for row in sheet.iter_rows():
    for cell in row:
        color = getattr(cell.fill.fgColor, "rgb", None)
        if color == "FF000000":
            bits.append(1)
        else:
            bits.append(0)

print(f"Bitmap with {len(bits)} bits: {bits}")
```

pixelart with excel

```
bits = []
for row in sheet.iter_rows():
    for cell in row:
        color = getattr(cell.fill.fgColor, "rgb", None)
        if color == "FF000000":
            bits.append(1)
        else:
            bits.append(0)

print(f"Bitmap with {len(bits)} bits: {bits}")
```

Black or whoite?

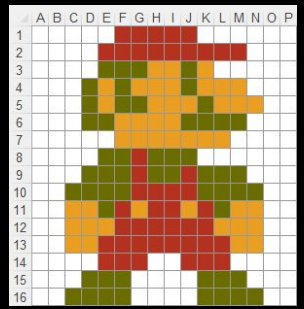
pixelart with excel

```
bits = []
for row in sheet.iter_rows():
    for cell in row:
        color = getattr(cell.fill.fgColor, "rgb", None)
        if color == "FF000000":
            bits.append(1)
        else:
            bits.append(0)

oled.write_pixels(50, 20, 76, 43, bits)
```

COLOR

pixelart with excel

```
workbook = load_workbook("Super Mario Pixel Art.xlsx")
sheet = workbook["Super Mario"]
```



pixelart with excel

```
for row in sheet.iter_rows():
    for cell in row:
        color = getattr(cell.fill.fgColor, "rgb", None)
        color = color[2:]
        print(color)
```

pixelart with excel

```
for row in sheet.iter_rows():
    for cell in row:
        color = getattr(cell.fill.fgColor, "rgb", None)
        color = color[2:]
        print(color)
```

Remove the alpha channel

pixelart with excel

```
for row in sheet.iter_rows():
    for cell in row:
        color = getattr(cell.fill.fgColor, "rgb", None)
        color = color[2:]
        print(color)
```

Remove the alpha channel

```
FFFFFF
FFFFFF
...
B53120
B53120
...
```

pixelart with excel

what we have

```
FFFFFF
FFFFFF
FFFFFF
…
```

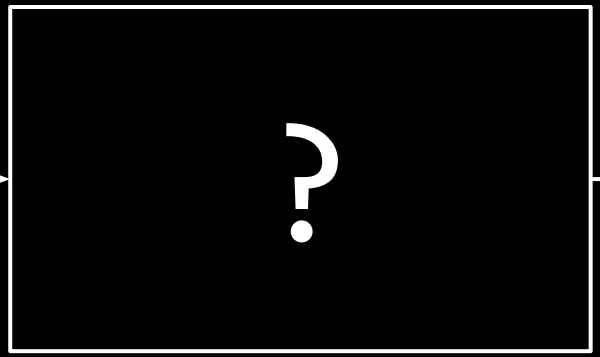
what we need

```
bitmap = [
    (255, 255, 255),
    (255, 255, 255),
    (255, 255, 255),
    …
]
```

pixelart with excel

what we have

```
FFFFFF
FFFFFF
FFFFFF
…
```



what we need

```
bitmap = [
     (255, 255, 255),
     (255, 255, 255),
     (255, 255, 255),
     …
]
```

pixelart with excel

what we have

```
FFFFFF
FFFFFF
FFFFFF
…
```

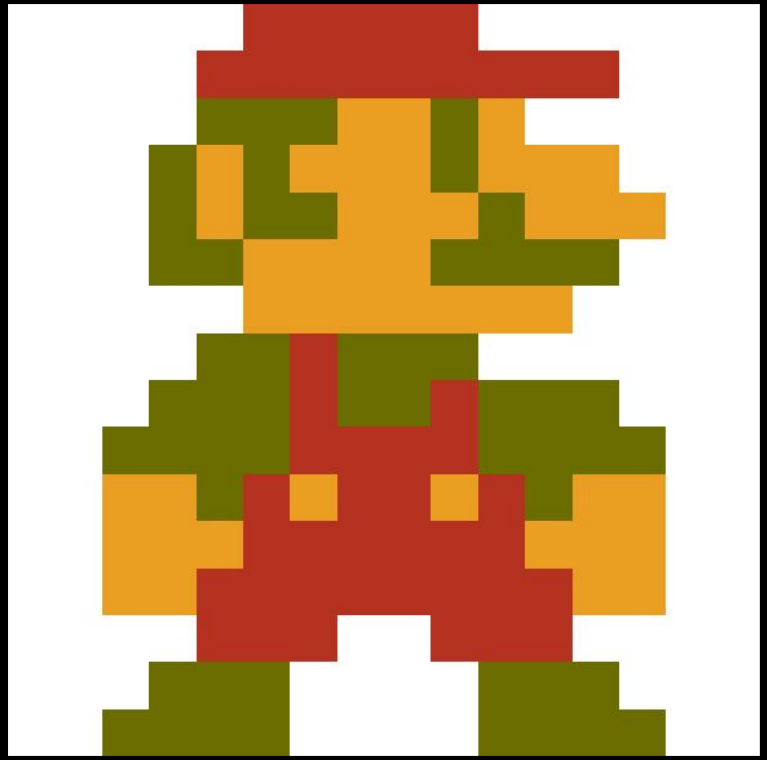
```
r = int(color[0:2], 16)
g = int(color[2:4], 16)
b = int(color[4:6], 16)
rgb_tuple = (r, g, b)
bitmap.append(rgb_tuple)
```

what we need

```
bitmap = [
     (255, 255, 255),
     (255, 255, 255),
     (255, 255, 255),
     …
]
```

saving a rgb bitmap

```
image = Image.new('RGB', (16, 16))
image.putdata(bitmap)
image.save("xlsx/super_mario_color.bmp")
```

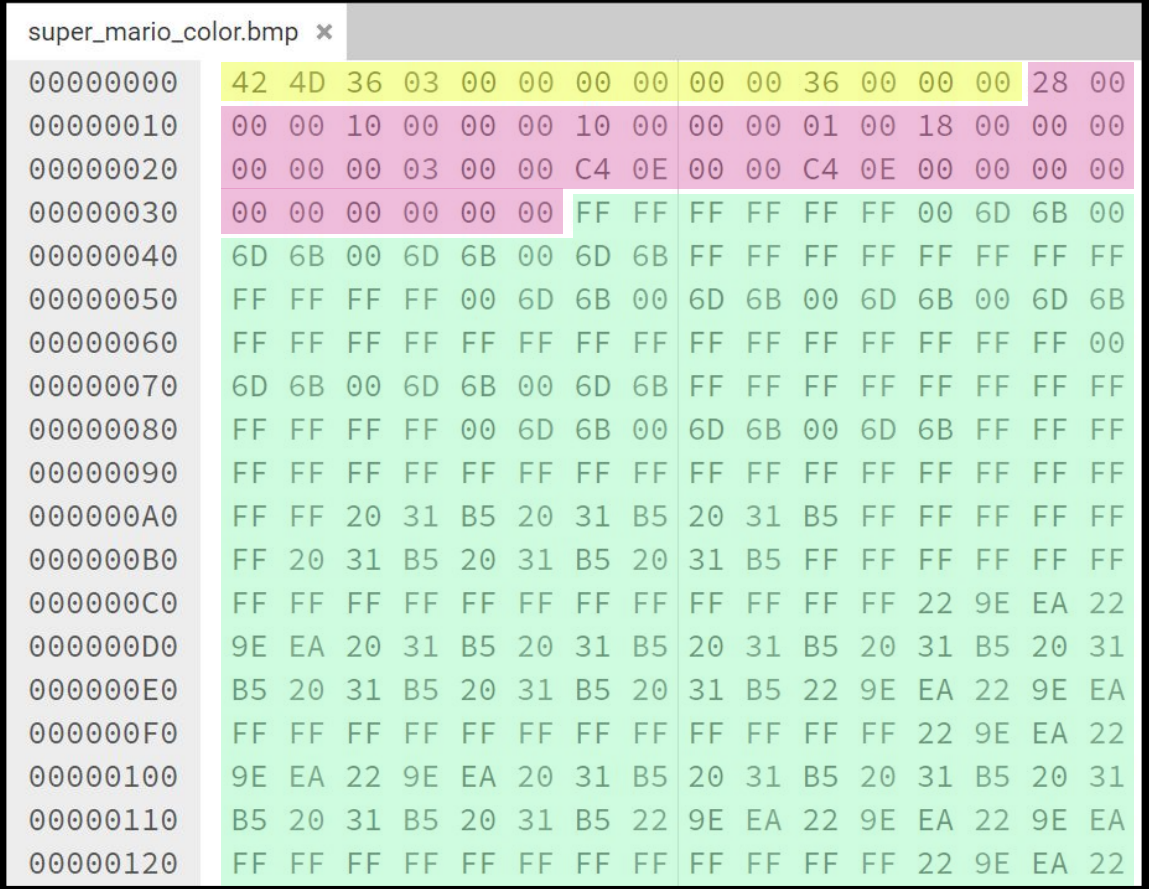


inside a bitmap file

bitmap file header

bitmap info header

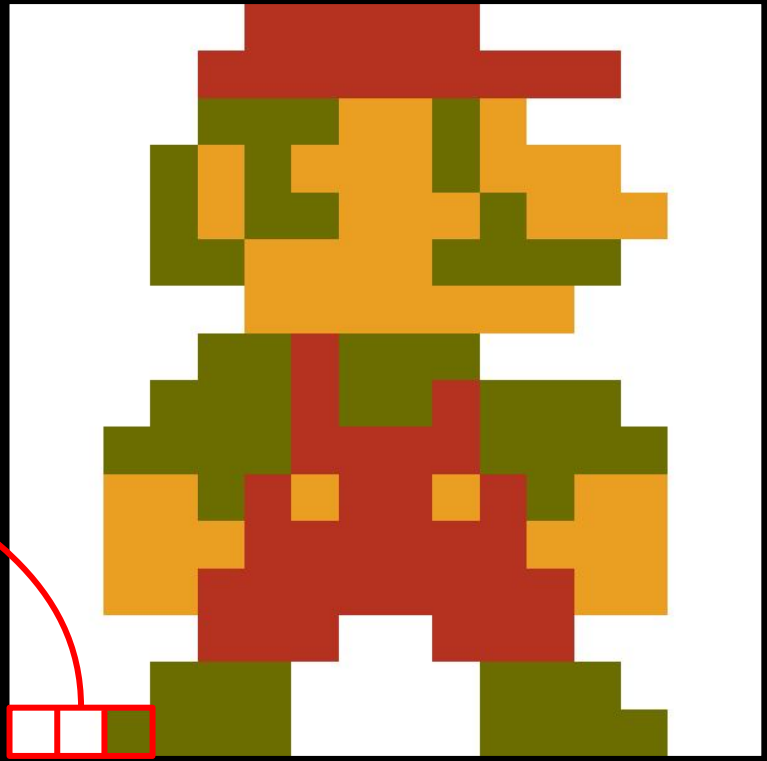
pixel data



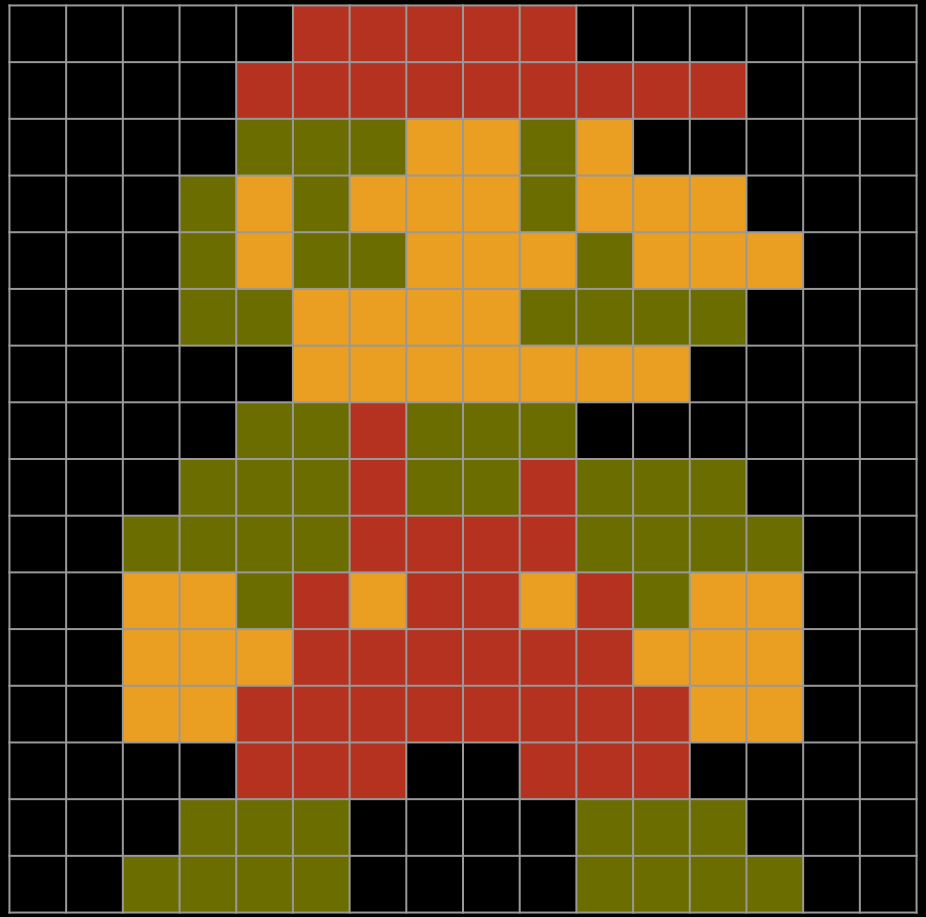
Try it: https://hexed.it/

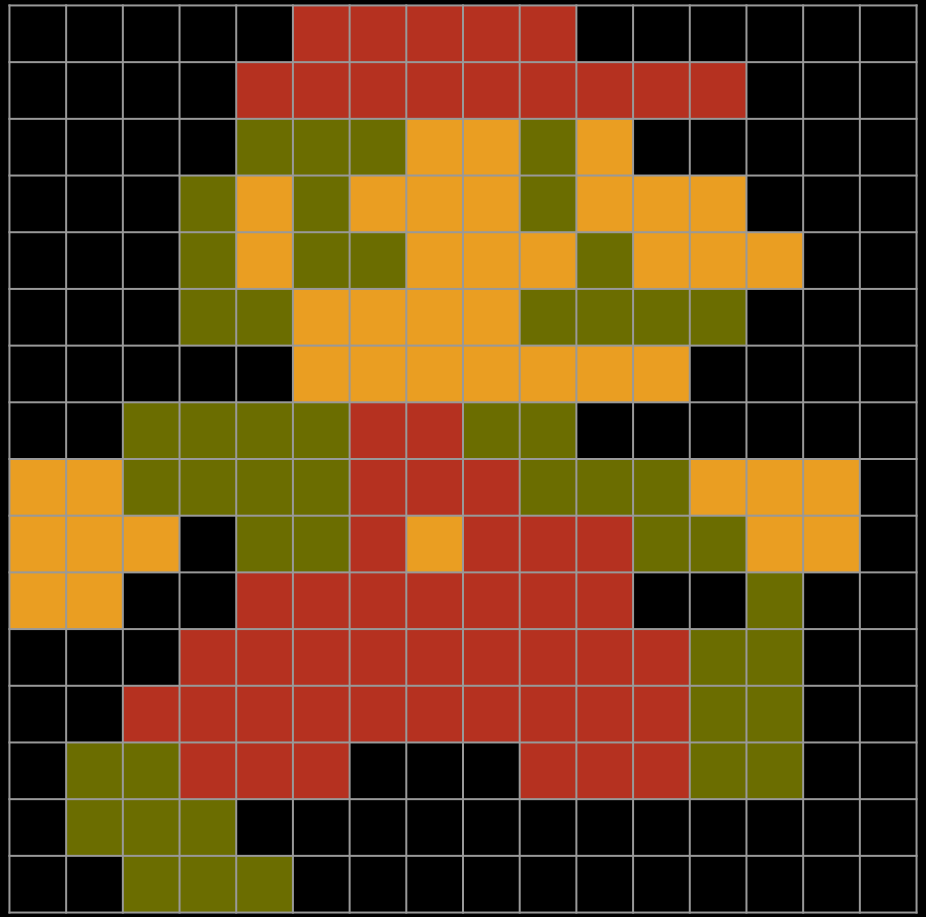
super_mario_color.bmp

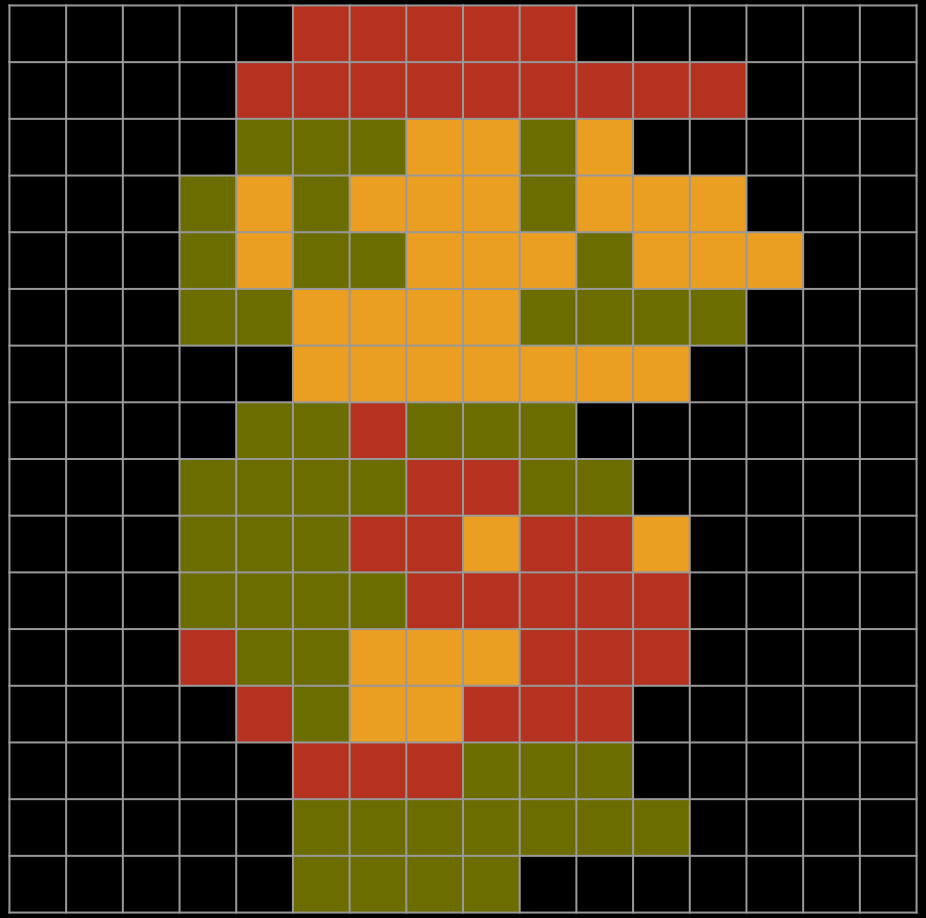
```
00000000  42 4D 36 03 00 00 00 00 00 00 36 00 00 00 28 00
00000010  00 00 10 00 00 00 10 00 00 00 01 00 18 00 00 00
00000020  00 00 00 03 00 00 C4 0E 00 00 C4 0E 00 00 00 00
00000030  00 00 00 00 00 00 FF FF FF FF FF FF 00 6D 6B 00
00000040  6D 6B 00 6D 6B 00 6D 6B FF FF FF FF FF FF FF FF
00000050  FF FF FF FF 00 6D 6B 00 6D 6B 00 6D 6B 00 6D 6B
00000060  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 00
00000070  6D 6B 00 6D 6B 00 6D 6B FF FF FF FF FF FF FF FF
00000080  FF FF FF FF 00 6D 6B 00 6D 6B 00 6D 6B FF FF FF
00000090  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000A0  FF FF 20 31 B5 20 31 B5 20 31 B5 FF FF FF FF FF
000000B0  FF 20 31 B5 20 31 B5 20 31 B5 FF FF FF FF FF FF
000000C0  FF FF FF FF FF FF FF FF FF FF FF FF 22 9E EA 22
000000D0  9E EA 20 31 B5 20 31 B5 20 31 B5 20 31 B5 20 31
000000E0  B5 20 31 B5 20 31 B5 20 31 B5 22 9E EA 22 9E EA
000000F0  FF FF FF FF FF FF FF FF FF FF FF FF 22 9E EA 22
00000100  9E EA 22 9E EA 20 31 B5 20 31 B5 20 31 B5 20 31
00000110  B5 20 31 B5 20 31 B5 22 9E EA 22 9E EA 22 9E EA
00000120  FF FF FF FF FF FF FF FF FF FF FF FF 22 9E EA 22
```

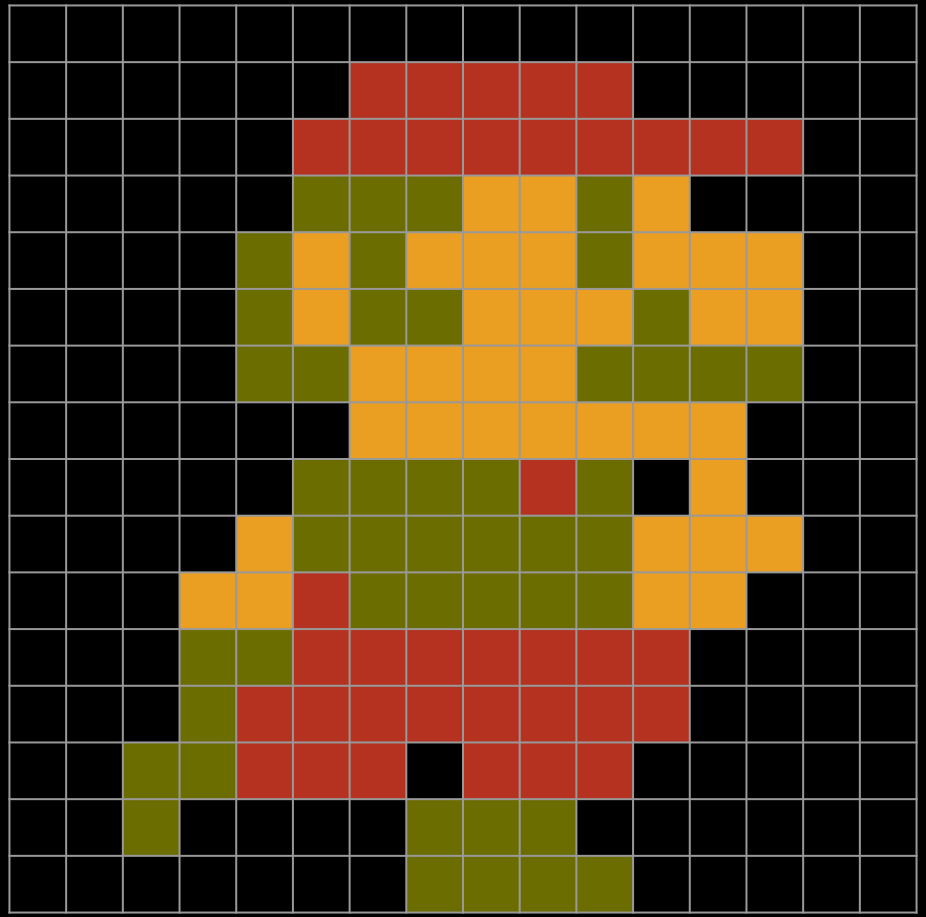


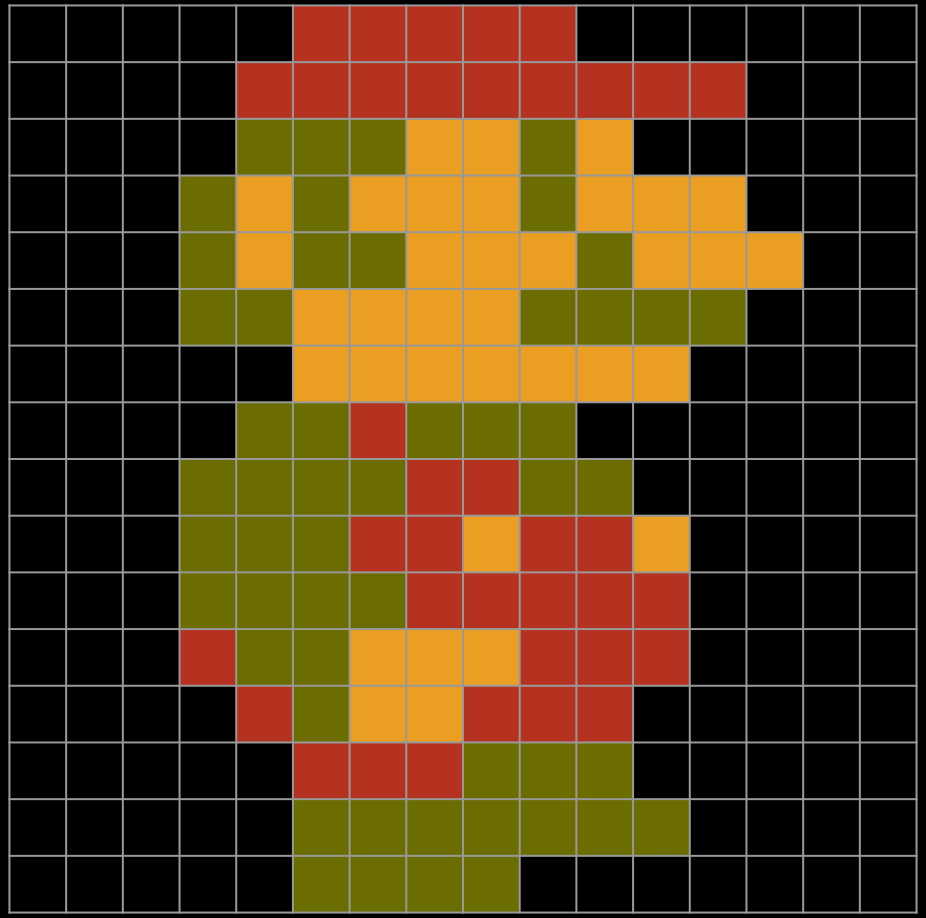
ANIMATION

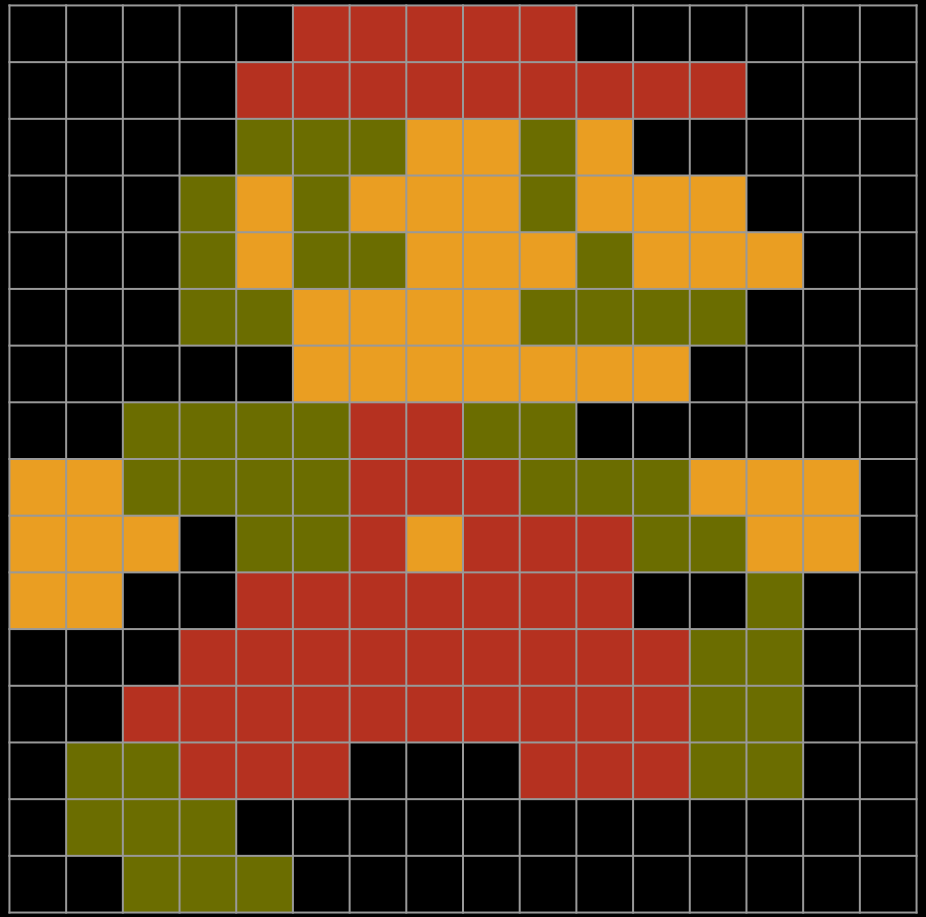












TRANSFORMATION

grayscale


to_grayscale()

rgb to luminance

```
luminance = 0.299 * r + 0.587 * g + 0.114 * b
luminance = round(luminance)
```

rgb to luminance as a function

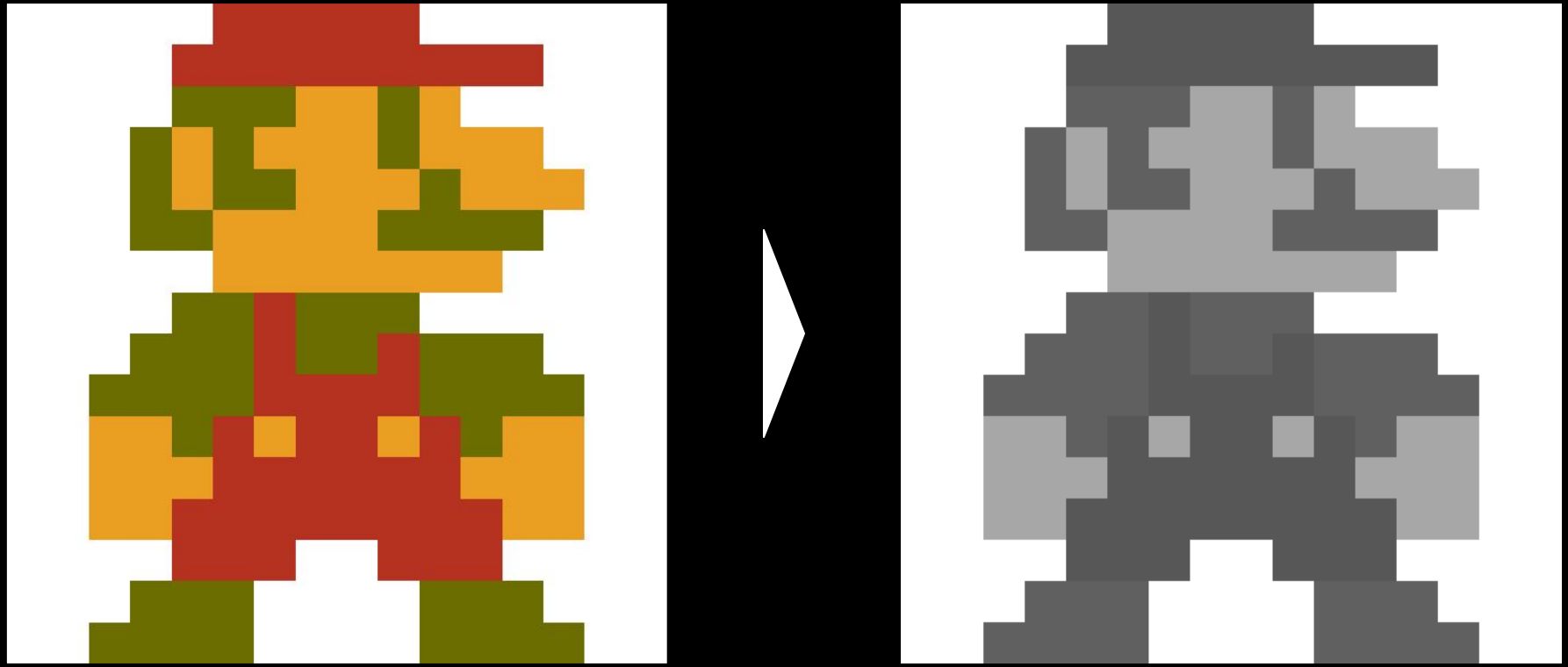
```
def rgb_to_luminance(rgb_tuple):
    r = rgb_tuple[0]
    g = rgb_tuple[1]
    b = rgb_tuple[2]

    luminance = 0.299 * r + 0.587 * g + 0.114 * b
    luminance = round(luminance)
    return luminance
```

image to grayscale

```
w, h = image.size
grayscale_values = []
for y in range(h):
    for x in range(w):
        r, g, b = image.getpixel((x, y))
        luminance = rgb_to_luminance((r, g, b))
        grayscale_values.append(luminance)
```

saving a grayscale image

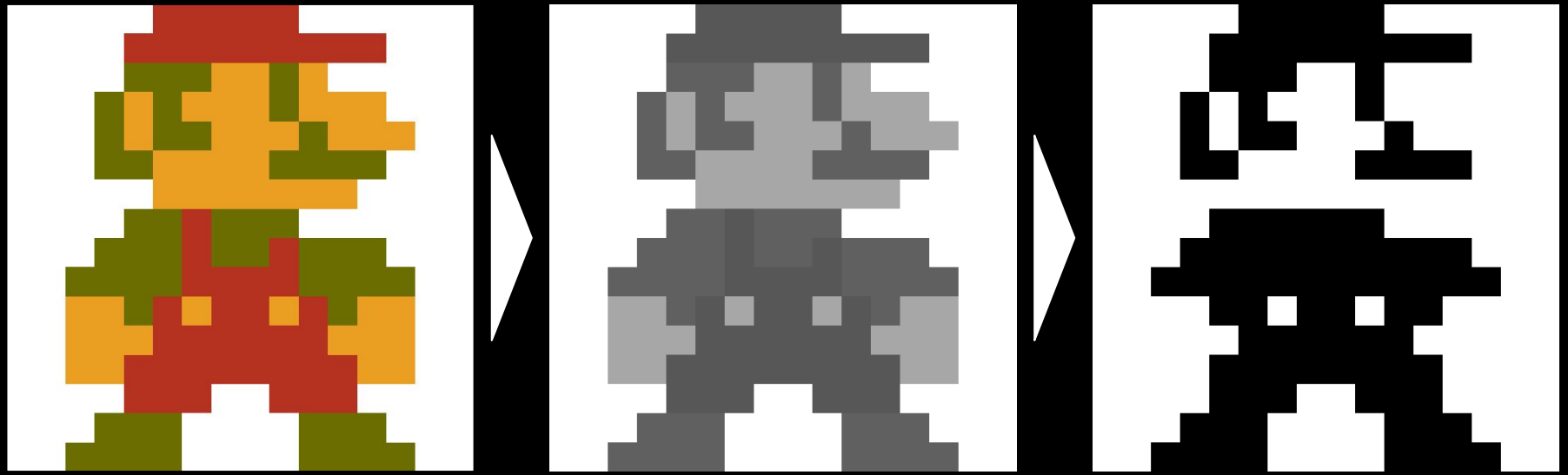
```
grayscale_image = Image.new("L", (w, h))
grayscale_image.putdata(grayscale_values)
grayscale_image.save("super_mario_grayscale.bmp")
```



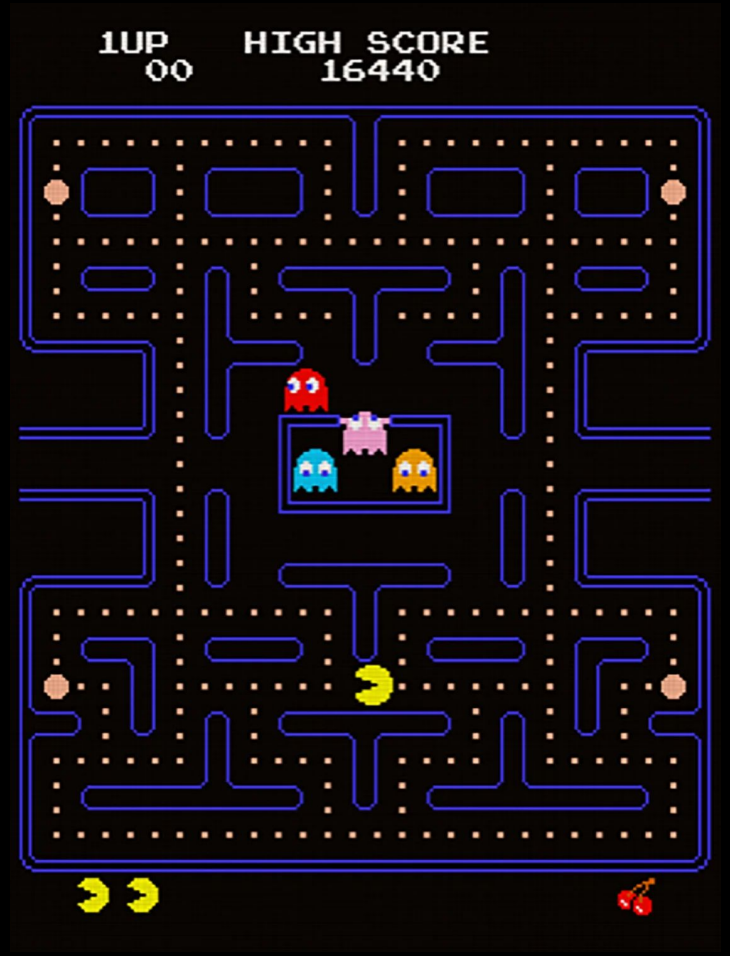
black/white

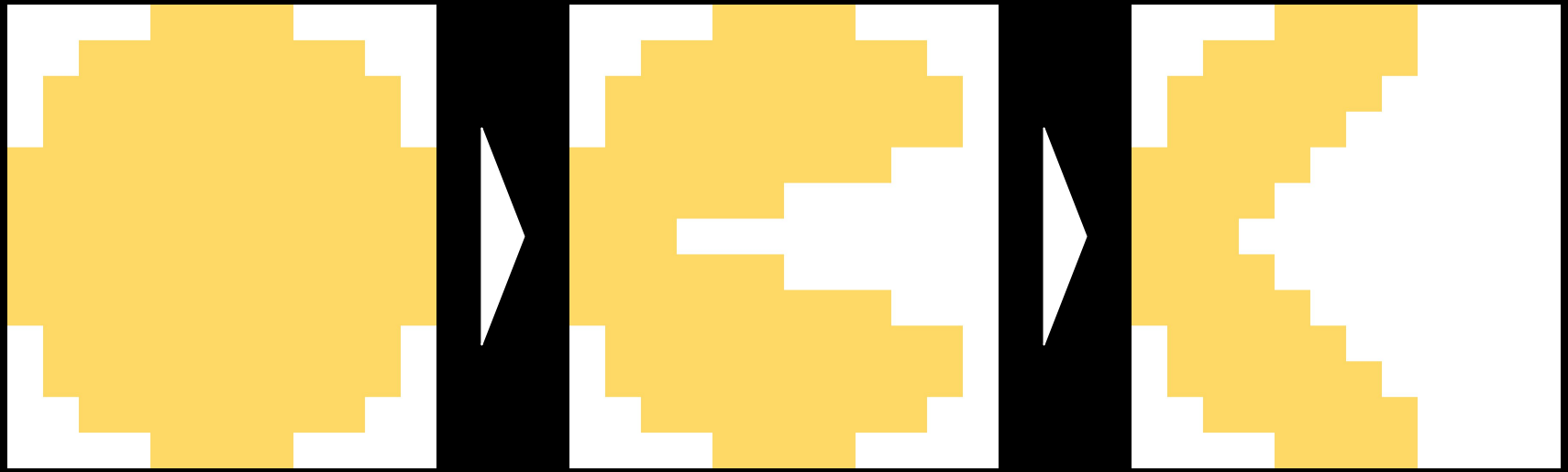
gray to b/w

```
def luminance_to_bw(luminance, threshold=128):
    if luminance < threshold:
        return 0
    else:
        return 1
```



PACMAN


1UP
00
HIGH SCORE
16440


closed
half
open

pacman animation

```
import time

wait_time = 0.1
while True:
    oled.write_pixels(10, 10, 21, 22, pacman_closed)
    time.sleep(wait_time)
    oled.write_pixels(10, 10, 21, 22, pacman_half)
    time.sleep(wait_time)
    oled.write_pixels(10, 10, 21, 22, pacman_open)
    time.sleep(wait_time * 2)
    oled.write_pixels(10, 10, 21, 22, pacman_half)
    time.sleep(wait_time)
```


tinkerforge.com
OLED 128x64 2.0