Irgendetwas mit ASCII-Art

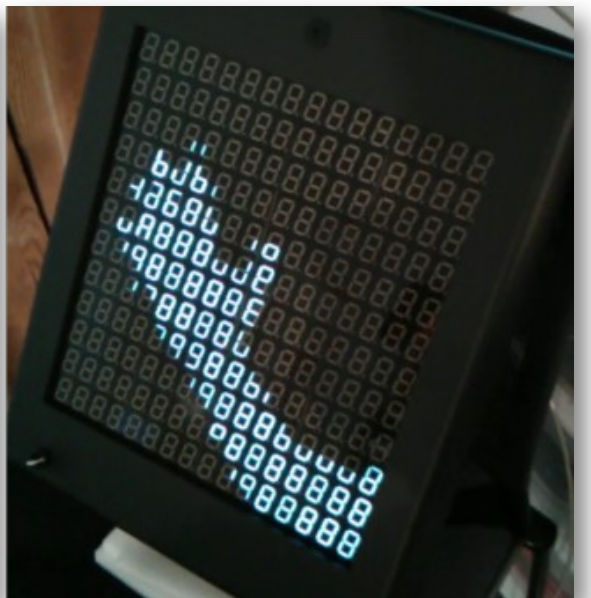
Uwe Berger
bergeruw@gmx.net

Uwe Berger

Motivation

Hackaday 2013

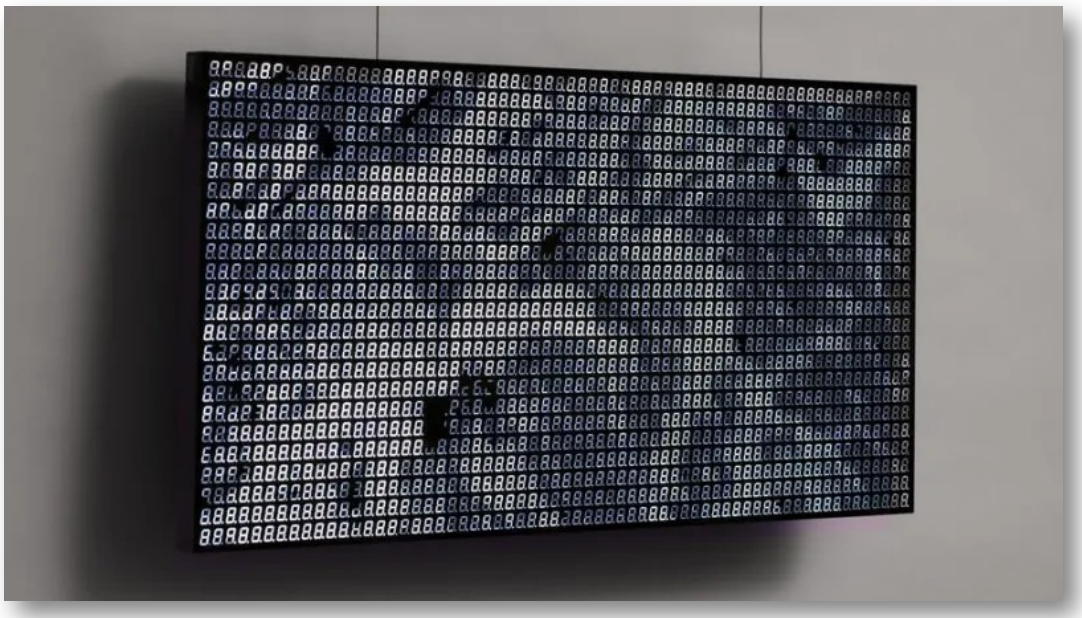
→ https://hackaday.com/2013/11/21/7-segment-display-matrix-visualizes-more-than-numbers/



Motivation

Hackaday 2023

→ https://hackaday.com/2023/02/23/sailing-on-a-sea-of-seven-segment-displays/



Was erzähle ich heute?

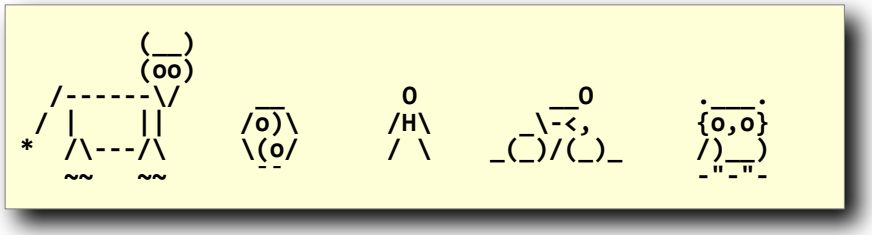
- ASCII-Art
  - Was ist das?
  - Programme, Tools etc.

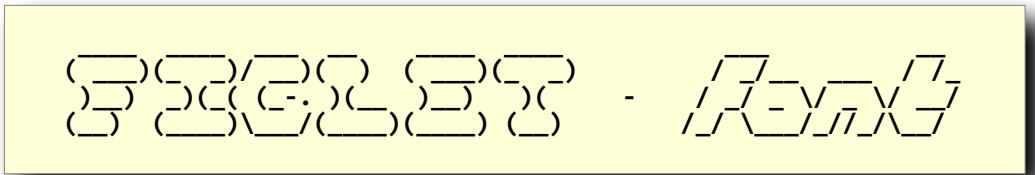
- Eine 7-Segment Display Matrix
  - Wie funktioniert das?
  - 7s-Matrix-Simulator
  - Was kommt jetzt (vielleicht)?

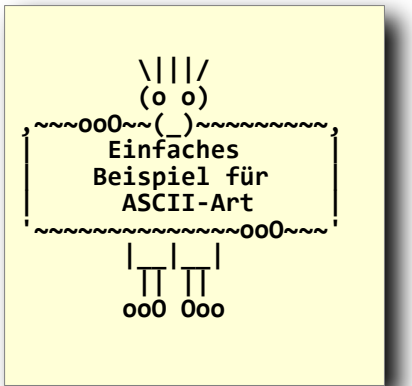
ASCII-Art

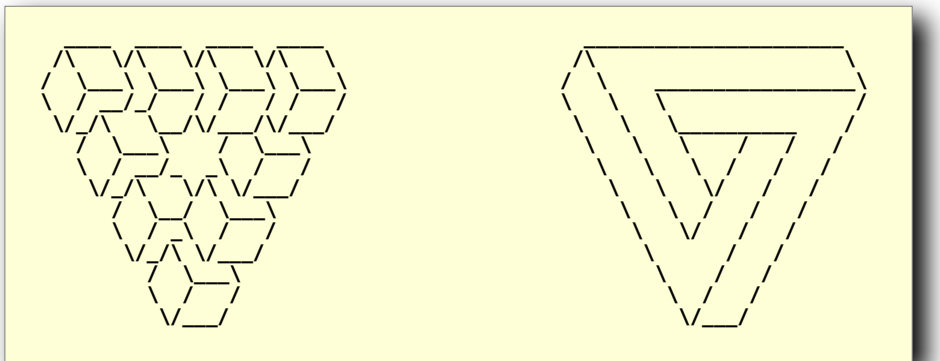
- Kunstrichtung, bei der man mit Buchstaben, Ziffern und Sonderzeichen Piktogramme, Bilder, Videos u.ä. darstellt
- Früher (vor Erfindung des Terminals) gab es auch den Begriff „Typewriter Art“ → Paul Smith

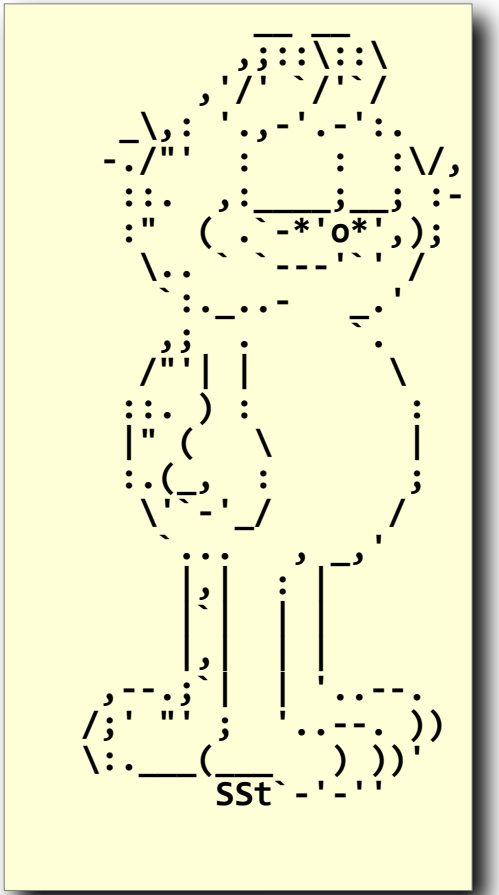
Beispiele für „einfache“ ASCII-Art



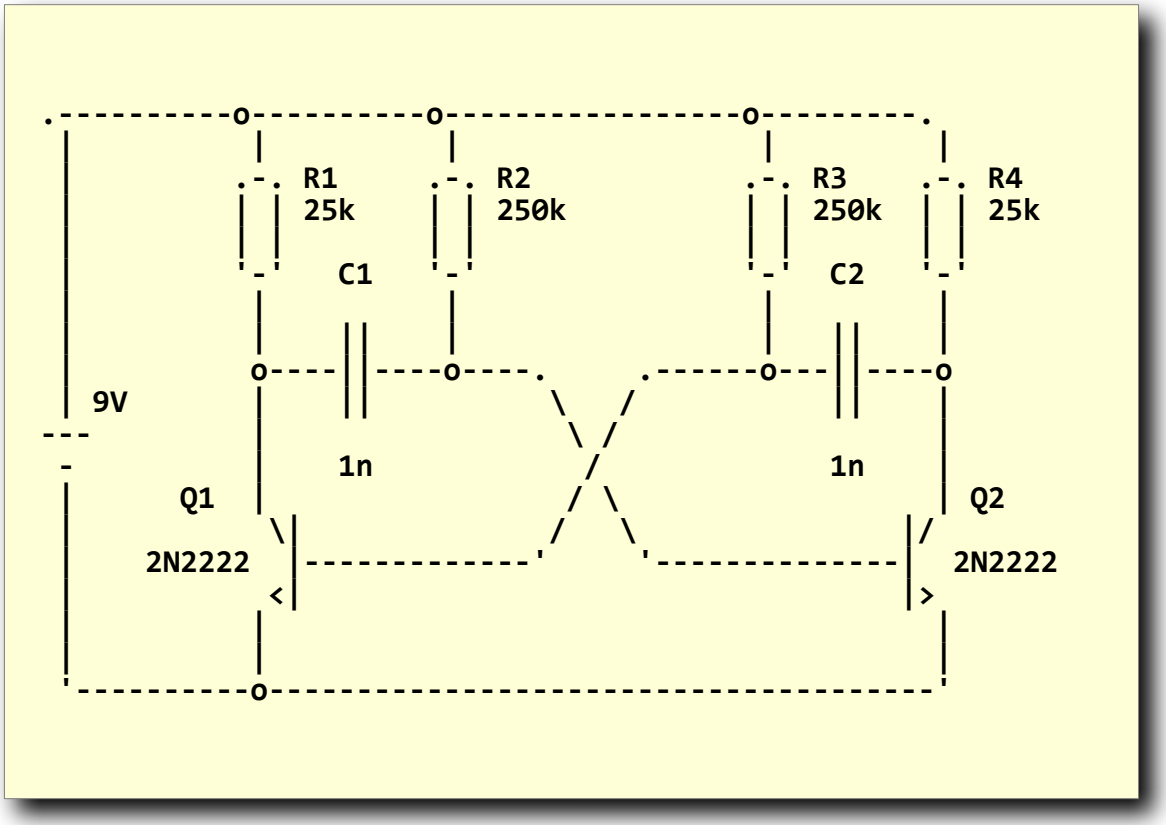




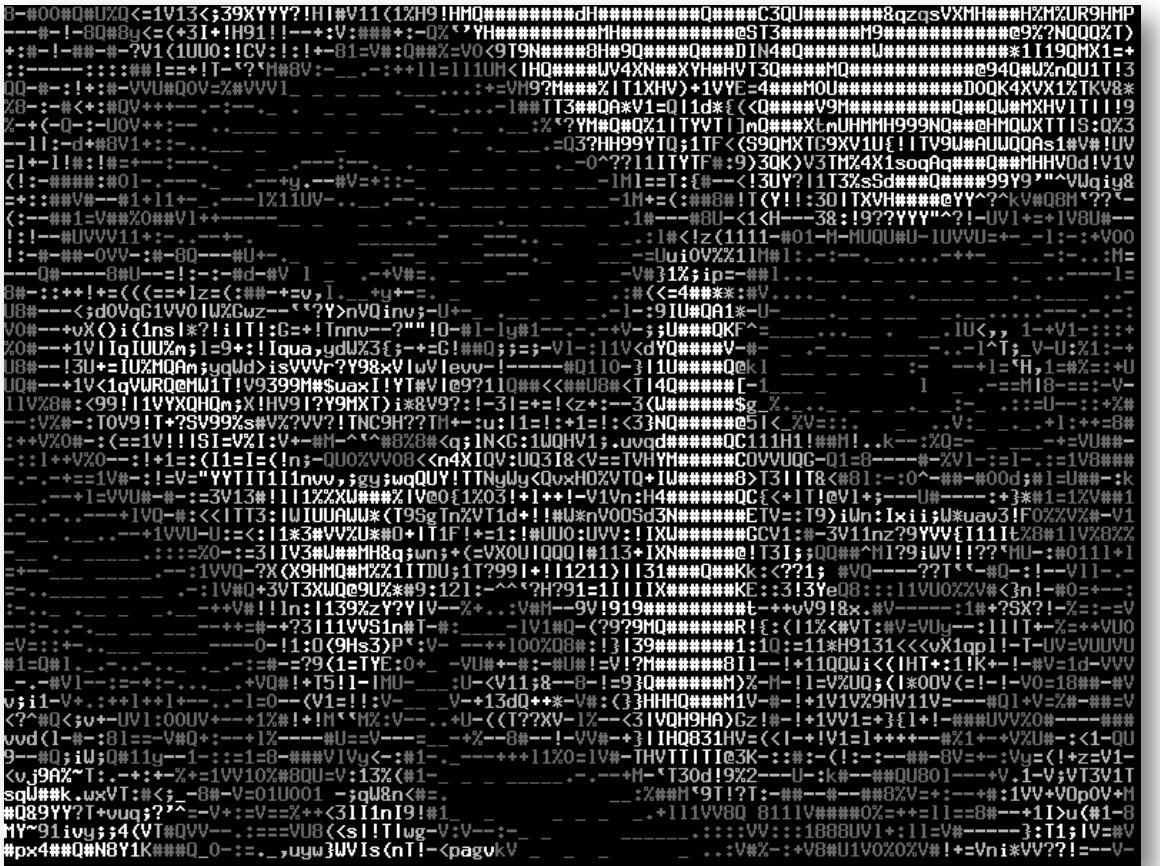




Schaltpläne, Organigramme u.ä.



„Aufwendige“ ASCII-Art



Programme, Tools ...

…zur Erstellung/Konvertieren von Bildern oder Video-Sequenzen in ASCII-Art basieren meist auf:

- aalib → https://aa-project.sourceforge.net/aalib/
- libcaca → http://caca.zoy.org/wiki/libcaca

Die zugrundeliegenden Algorithmen versuchen meist Bildbereiche des Originals durch äquivalente ASCII-Zeichen zu ersetzen (Kontur, visuelle Helligkeit/Kontrast, Farben etc.).

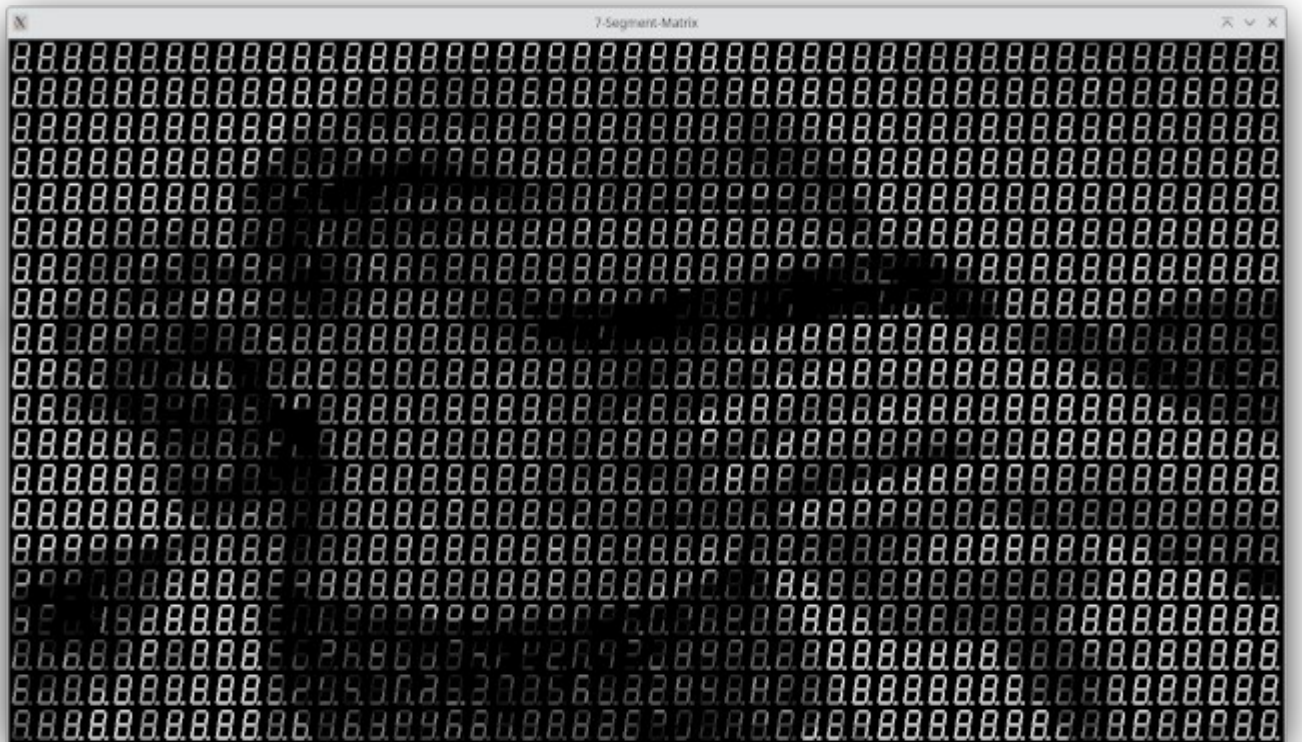
Programme, Tools (Beispiele)

- Bilder in ASCII konvertieren:
  - aview (nur .pnm-Bilder als Input)
  - jp2a (für .jpg, .png)

- Videos als ASCII anzeigen:
  - mplayer -vo aa video.avi

- Webcam-Output (o.ä.) als ASCII-Art-Stream:
  - Hasciicam

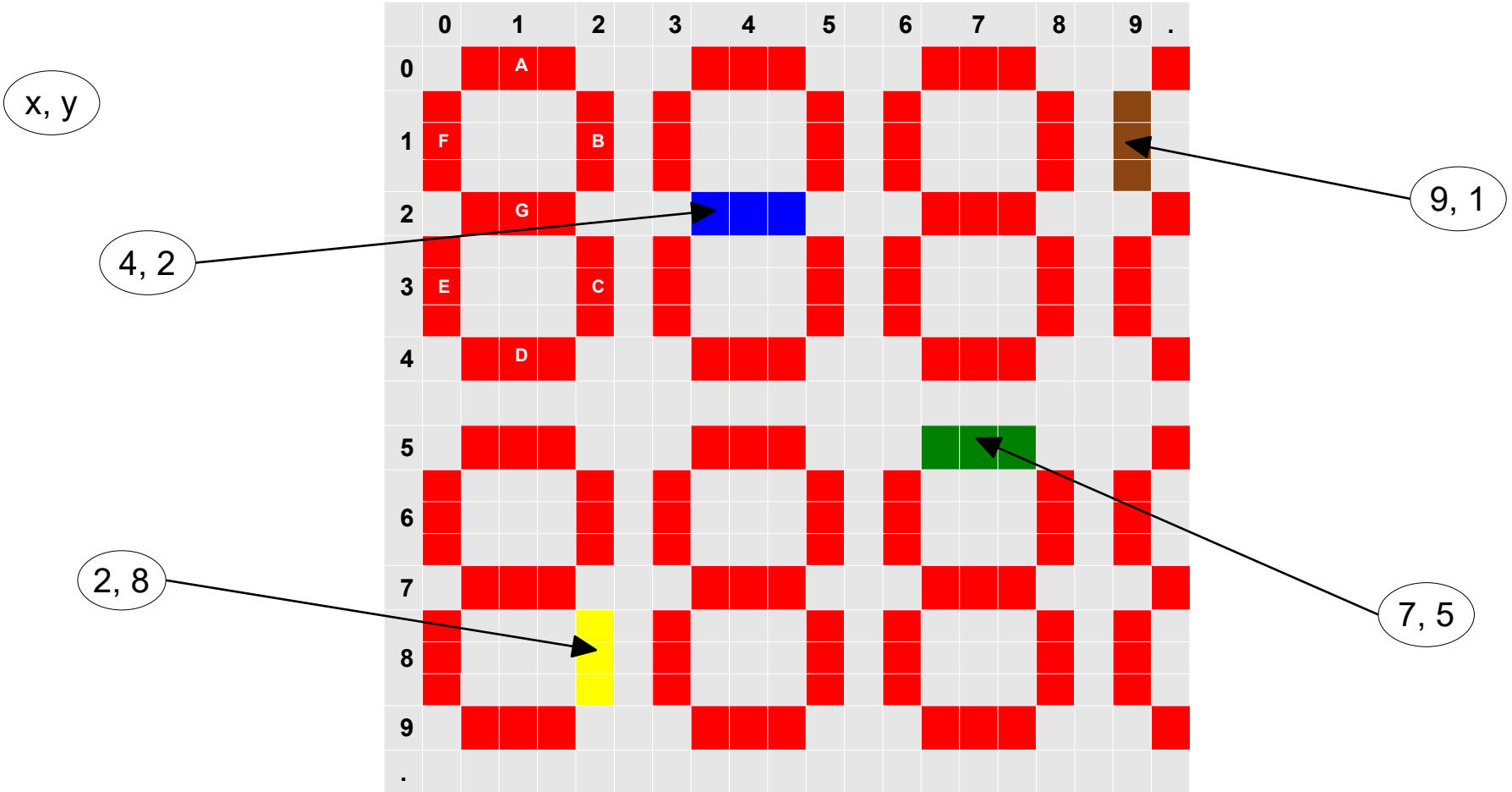
7-Segment Display Matrix



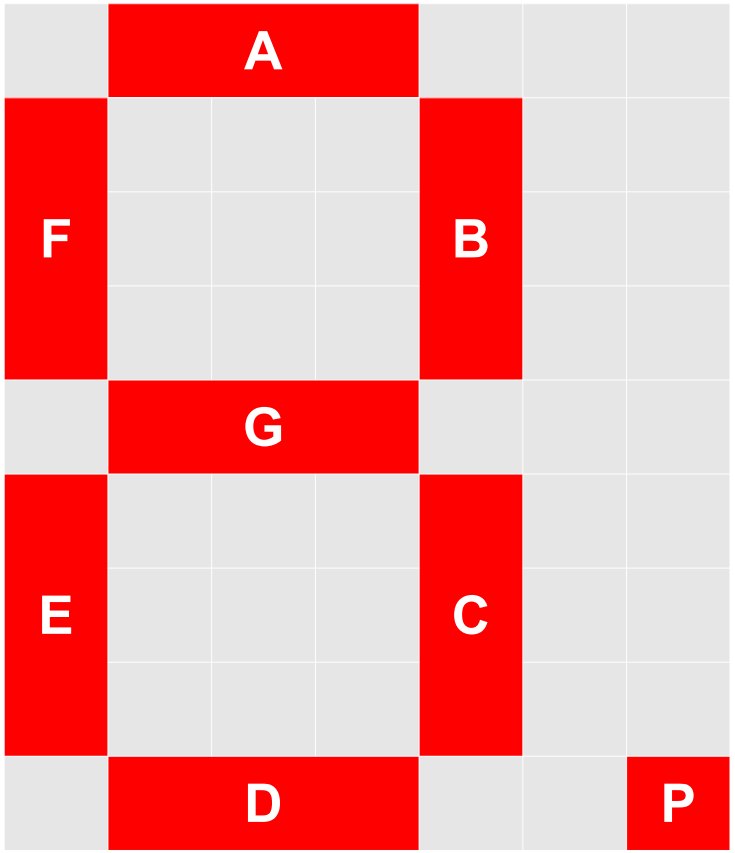
Wie funktioniert es?

- Ich verwende keine aalib, libcaca o.ä.!
- Das Bild wird „Pixel für Pixel“ in der 7s-Matrix gezeichnet
- Pixel-Koordinaten in der 7s-Matrix sind „korrespondieren-de“ Segmente eines Digits:
  - Variante 1: symmetrische Verteilung der Segmente
  - Variante 2: gepackte/komprimierte Verteilung der Segmente

V1: Sym. Verteilung der Segmente



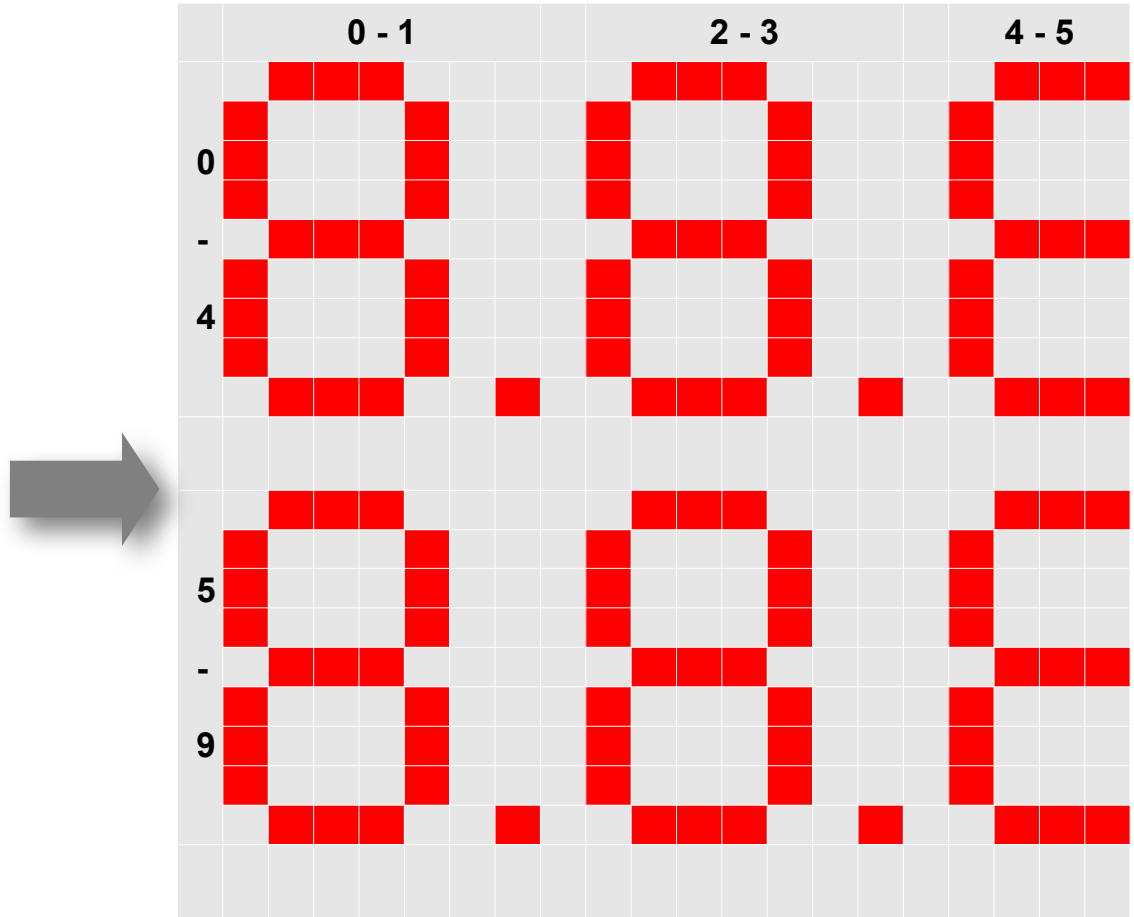
V2: gepackte Verteilung der Segm.



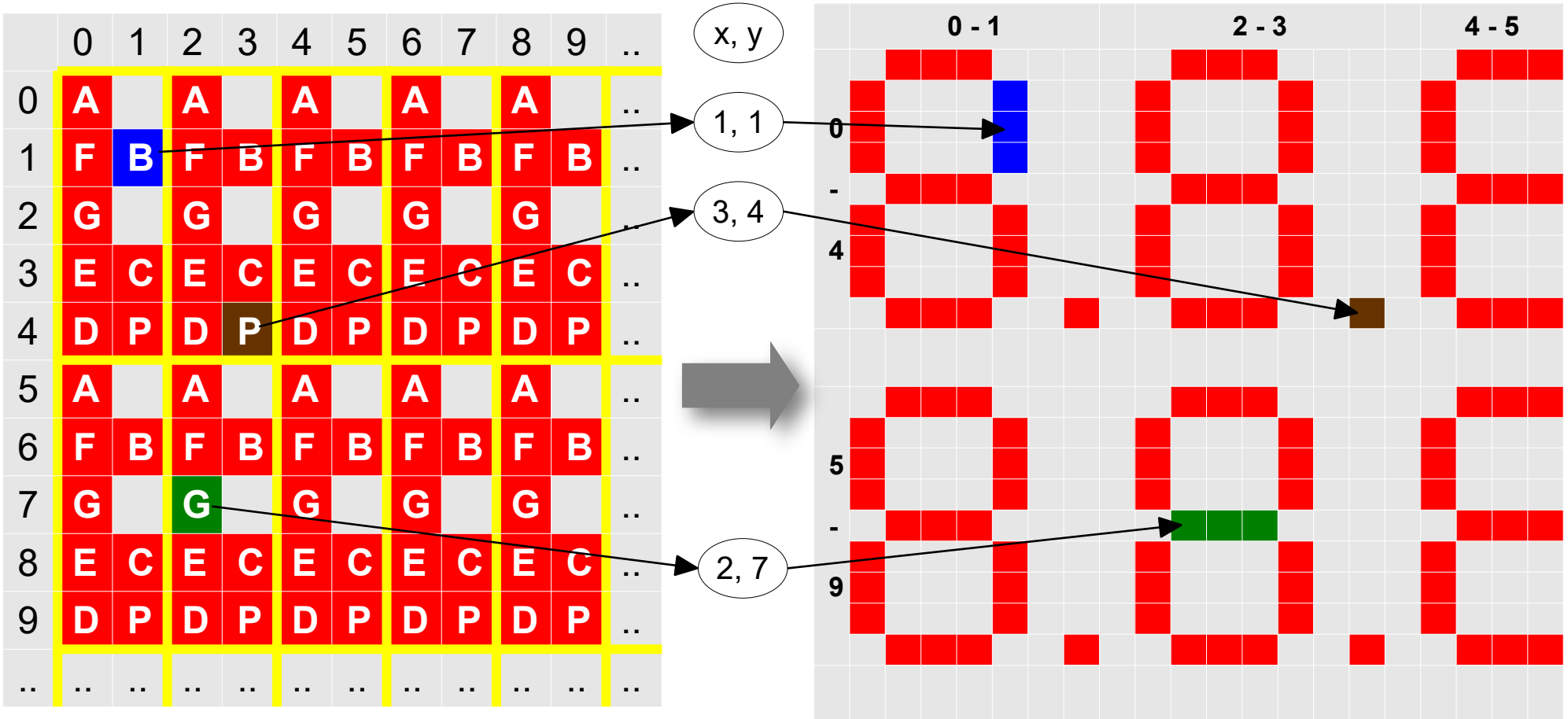
		X	
		0	1
Y	0	A	
	1	F	B
	2	G	
	3	E	C
	4	D	P

V2: gepackte Verteilung der Segm.

	0	1	2	3	4	5	6	7	8	9	..
0	A		A		A		A		A		..
1	F	B	F	B	F	B	F	B	F	B	..
2	G		G		G		G		G		..
3	E	C	E	C	E	C	E	C	E	C	..
4	D	P	D	P	D	P	D	P	D	P	..
5	A		A		A		A		A		..
6	F	B	F	B	F	B	F	B	F	B	..
7	G		G		G		G		G		..
8	E	C	E	C	E	C	E	C	E	C	..
9	D	P	D	P	D	P	D	P	D	P	..
..	..	..	..	..	..	..	..	..	..	..	..



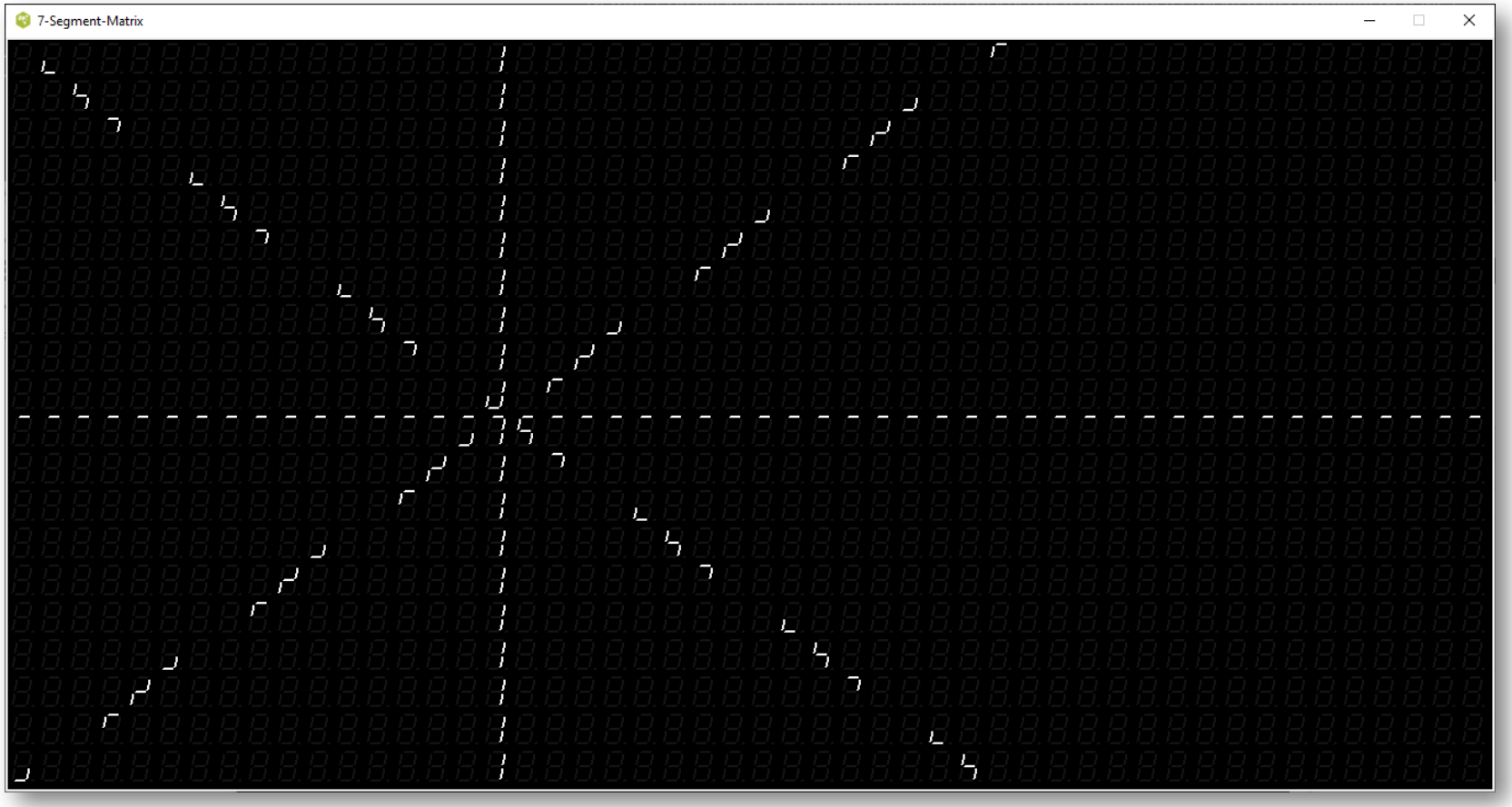
V2: gepackte Verteilung der Segm.



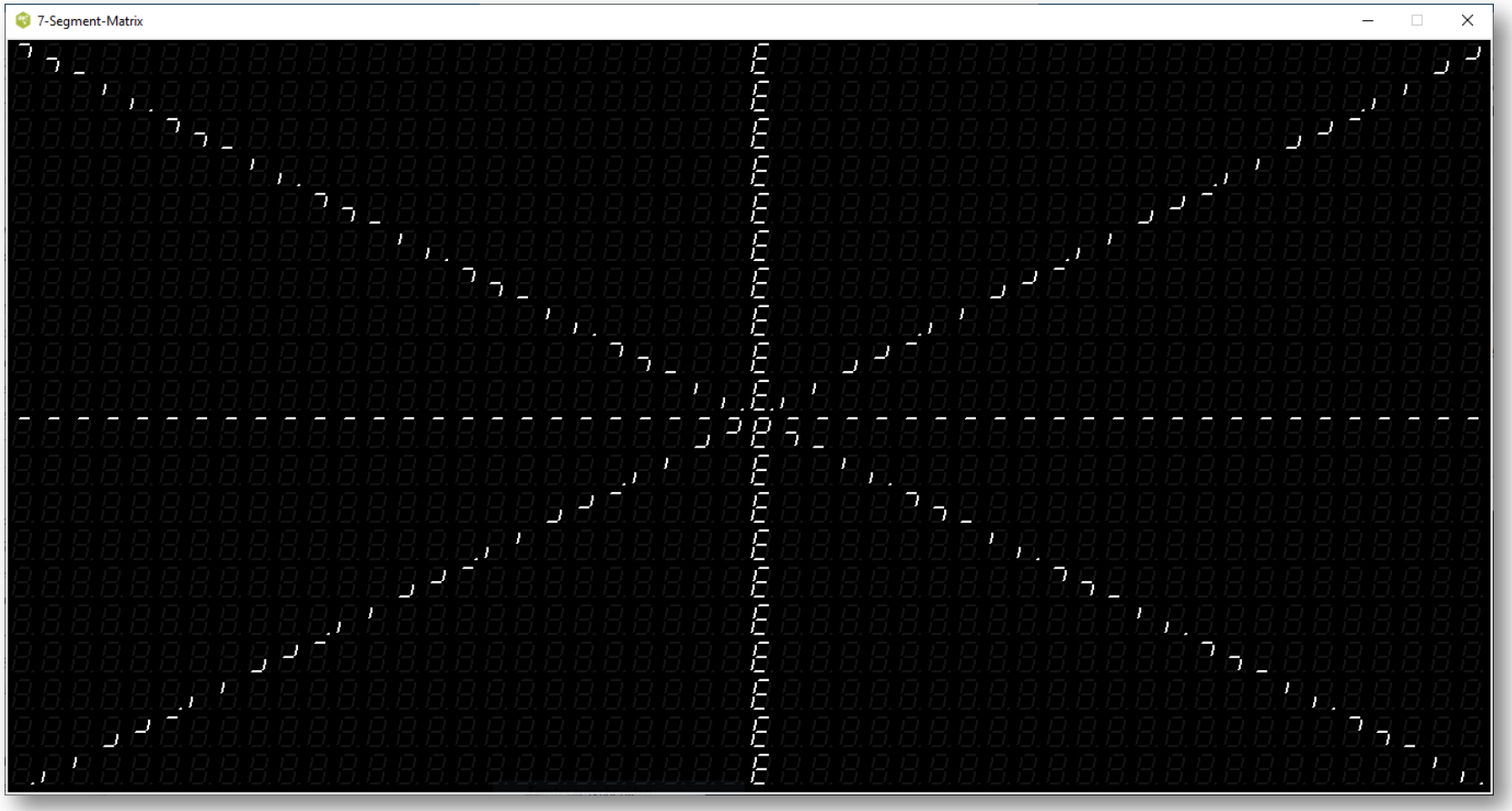
7s-Matrix-Simulator

- Client/Server-Architektur
  - Der Client generiert/konvertiert Bilder und sendet das Ergebnis zum Server
  - Der Server generiert initial die 7s-Matrix und stellt empfangene Bilddaten auf dieser dar
- (derzeit) in Tcl/Tk geschrieben (wegen einiger cooler Eigenschaften von Tk)

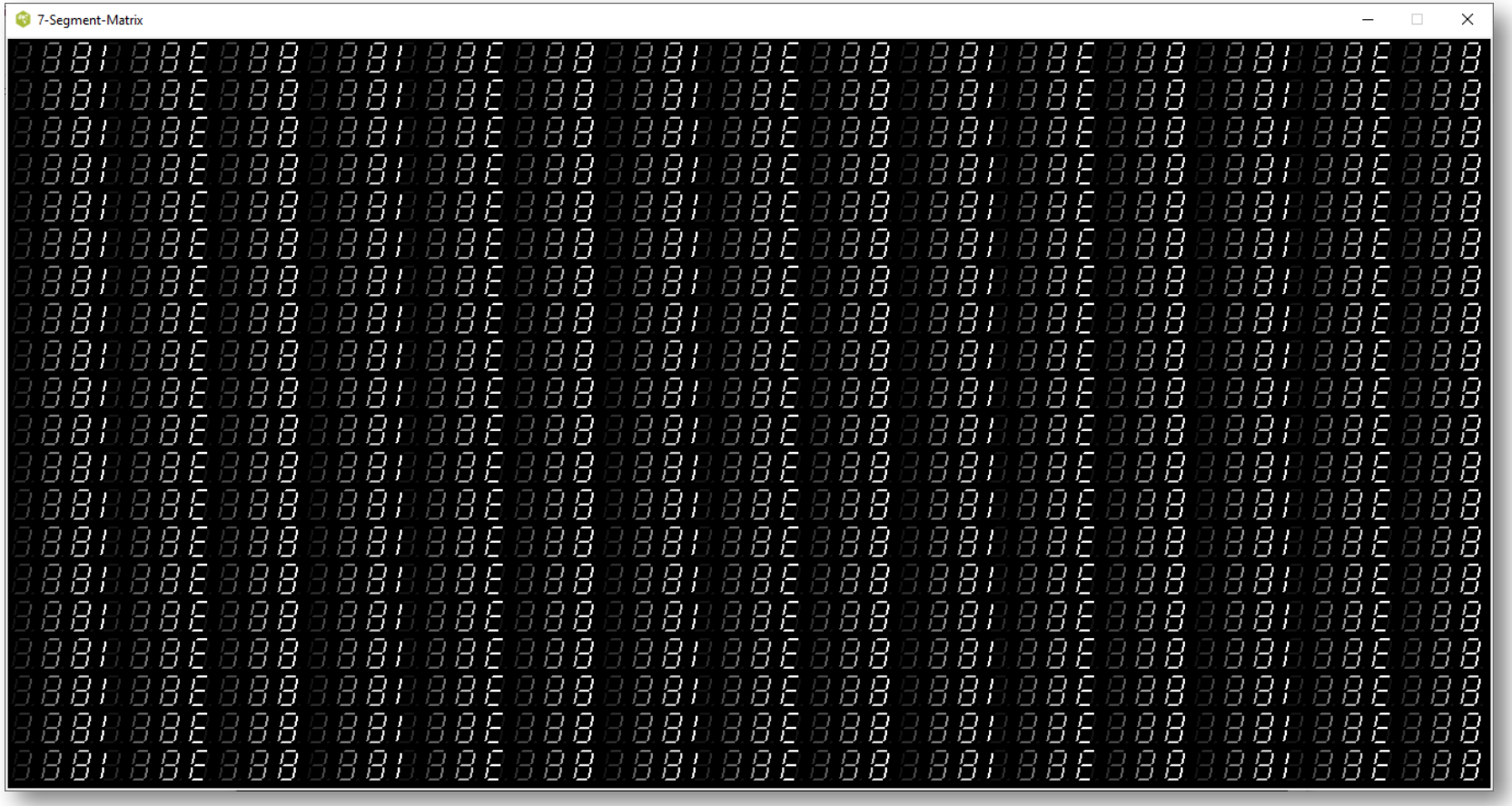
7s-Matrix-Simulator (mit V1)



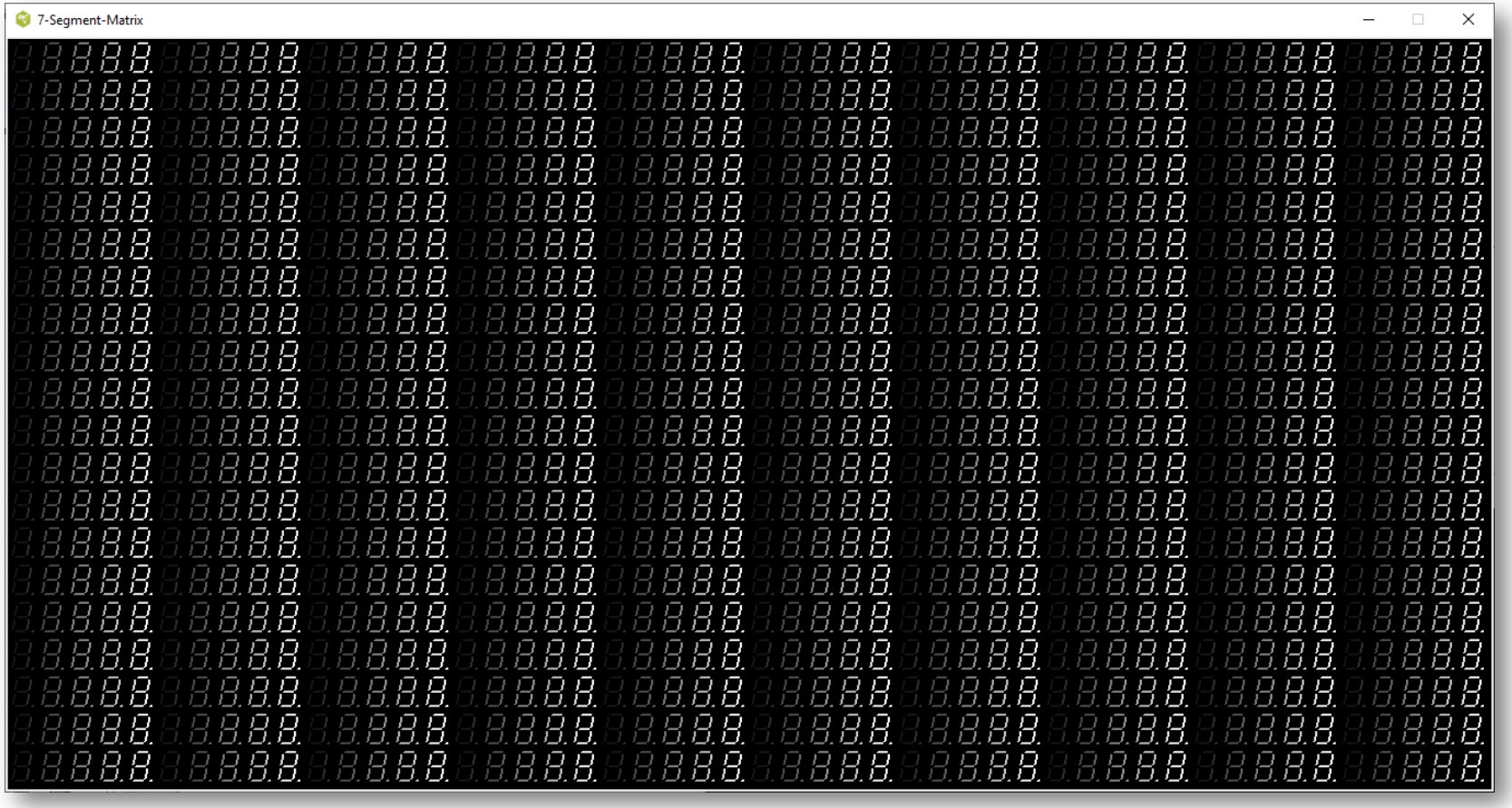
7s-Matrix-Simulator (mit V2)



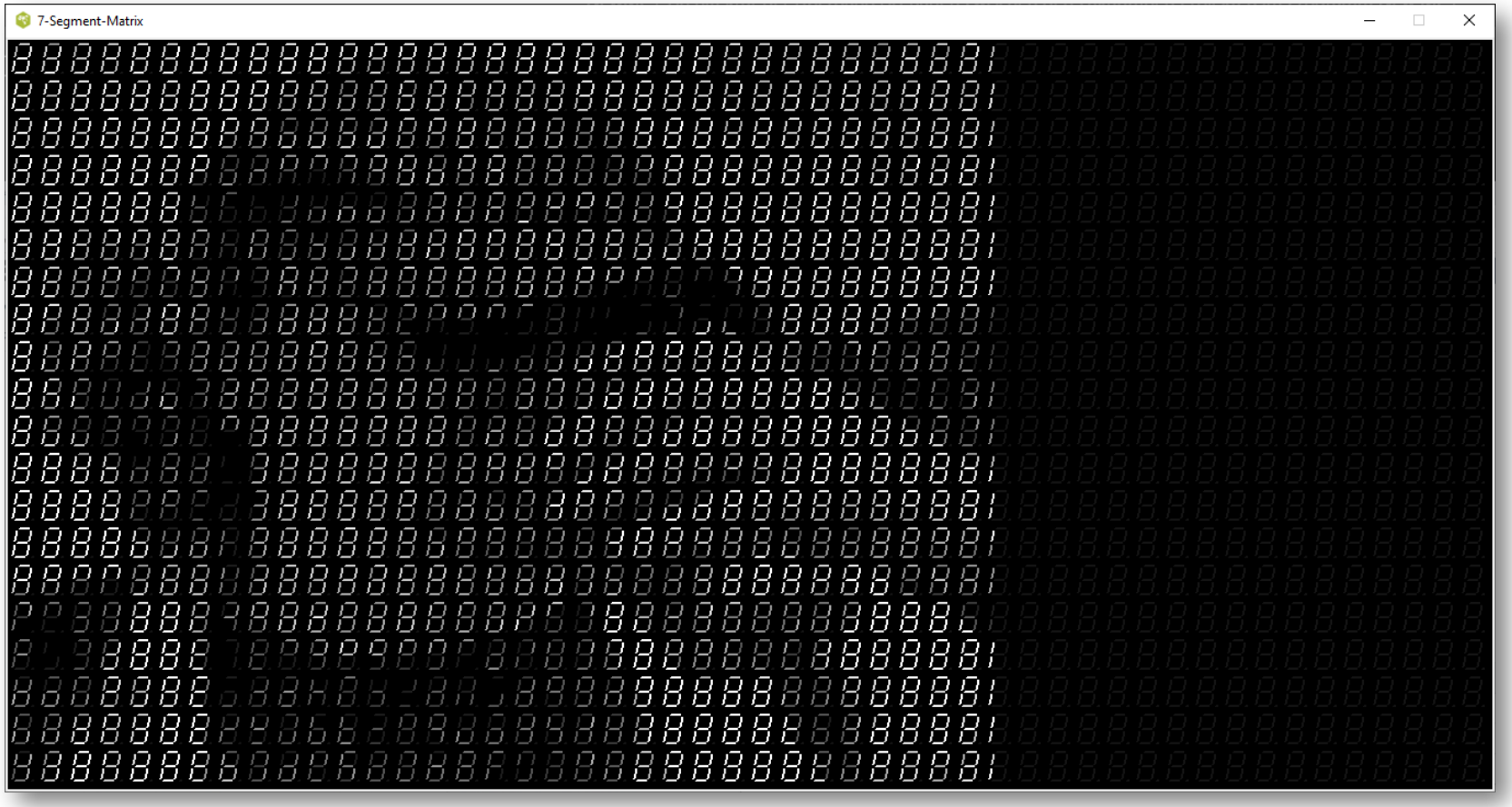
7s-Matrix-Simulator (mit V1)



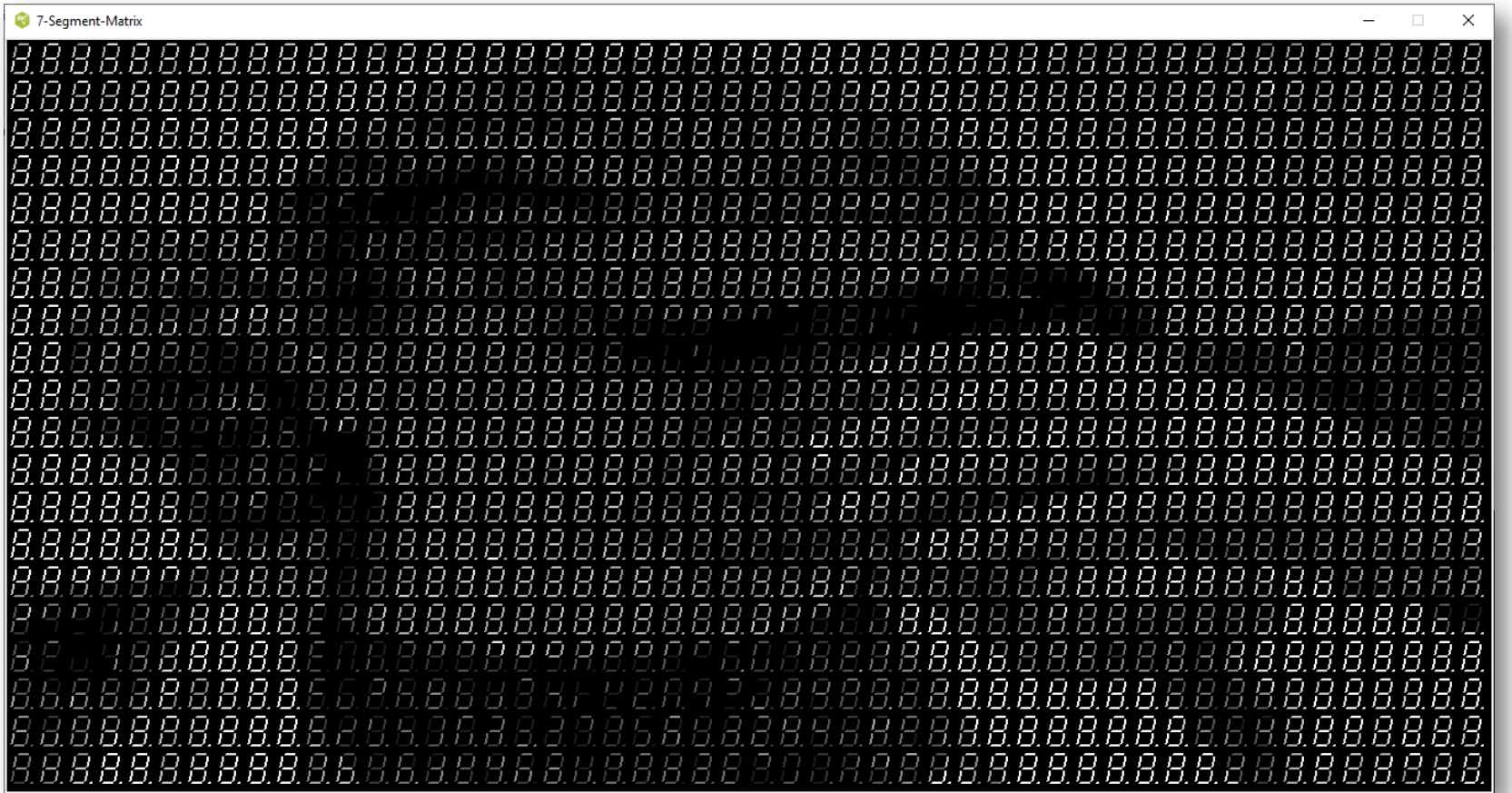
7s-Matrix-Simulator (mit V2)



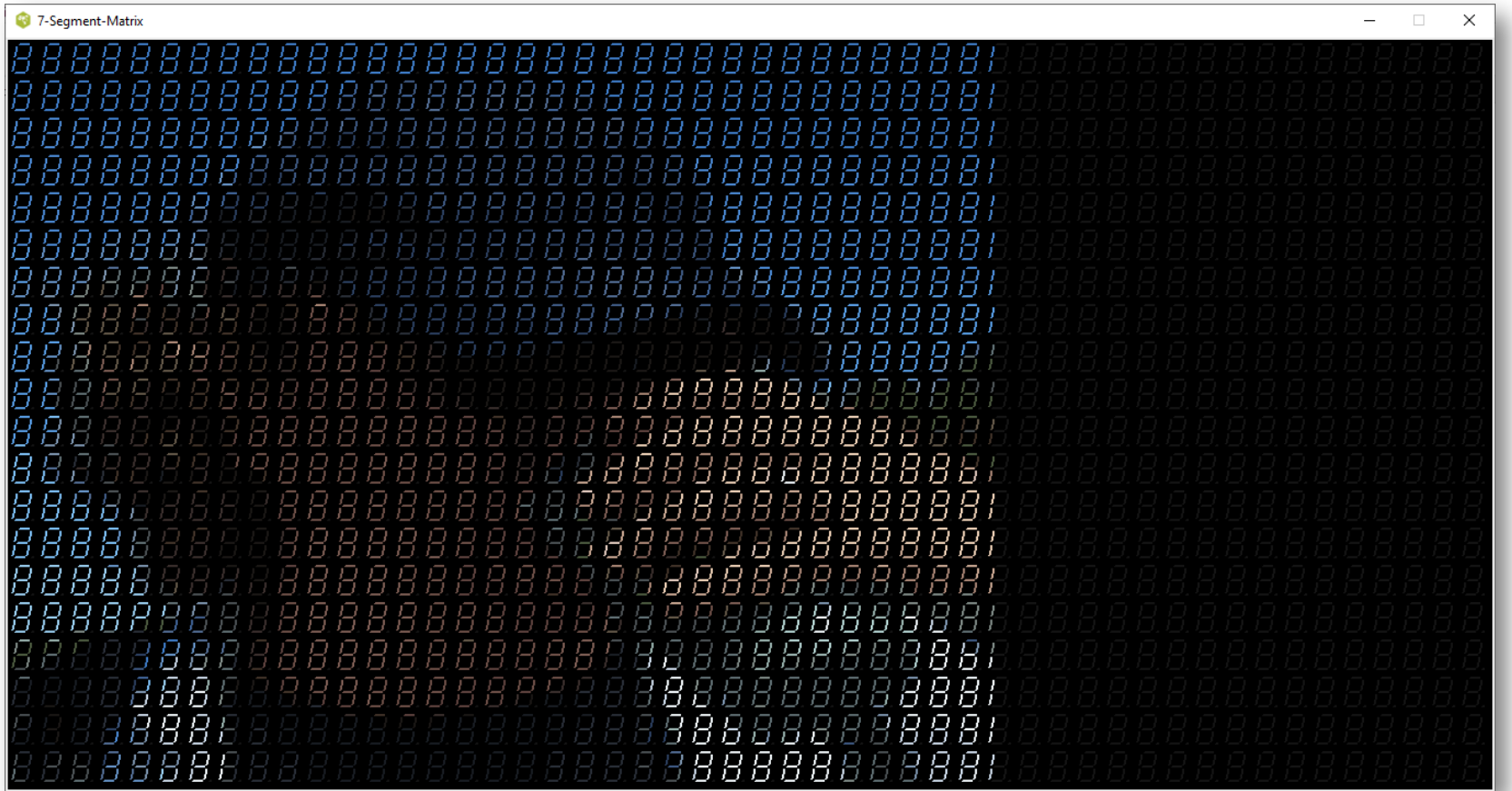
7s-Matrix-Simulator (mit V1)



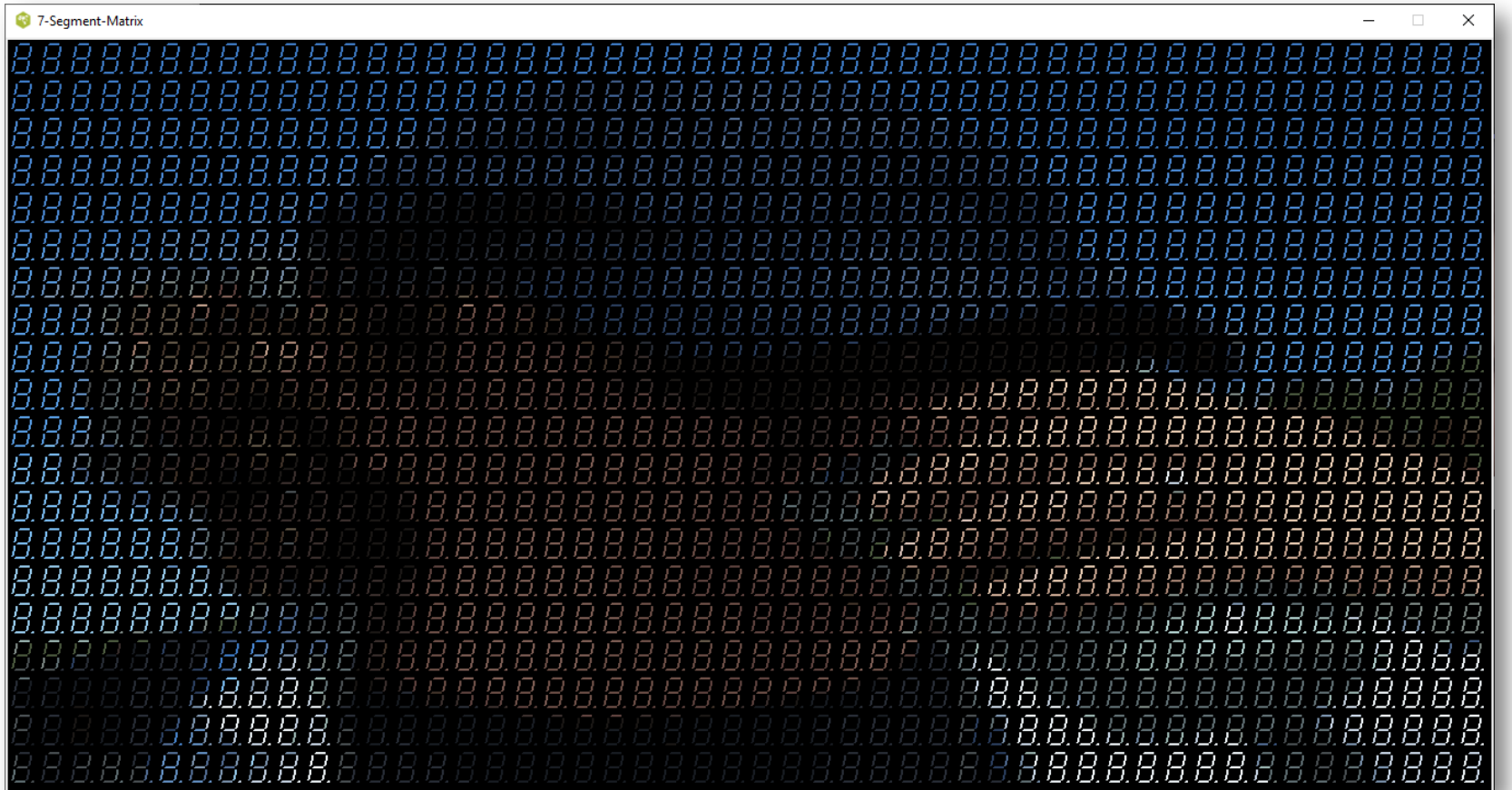
7s-Matrix-Simulator (mit V2)



7s-Matrix-Simulator (mit V1)



7s-Matrix-Simulator (mit V2)



Vergleich Varianten (Matrix-Koord.)

	Variante 1	Variante 2
Abbildbare „Pixel-Fläche“ auf einem Digit	• dx=3 • dy=5	• dx=2 • dy=5
Nicht anzeigbare Pixel pro Digit	8 (7 Segmente bei 15 Pixel; Digit-Punkt wird nicht verwendet)	2 (8 Segmente bei 10 Pixel; Digit-Punkt wird verwendet)
Pro	• Symmetrisches Abbild • Digit-Verbrauch geringer (z.B. 48x50 Pixel: 160 Digits)	• Guter Ausnutzungsgrad
Contra	• Schlechter Ausnutzungs-grad	• Unsymmetrisches Abbild • Höherer Digit-Verbrauch (z.B. 48x50 Pixel: 240 Digits)

Was kommt jetzt (vielleicht)?

- Simulator:
  - Implementierung Variante 1
  - Performance verbessern (Tcl/Tk → Python oder C…?)
  - „bewegte“ Bilder
- 7s-Matrix in Hardware...(?)…:
  - Größe z.B. 48x50 Pixel (bei V2) entsprechen 24x10 Einzel-Digits bzw. 6x10 4er-Digits
  - Ansteuerung? Graustufen via PWM?
  - Mechanischer Aufbau, Stromverbrauch, ...

Fragen?

...ansonsten Danke & Ende!