uxn: permacomputing & roguelikes

eirikr (@d6) he/him/etc.

sun oct 20 17:00:00 pm edt 2024

objectives

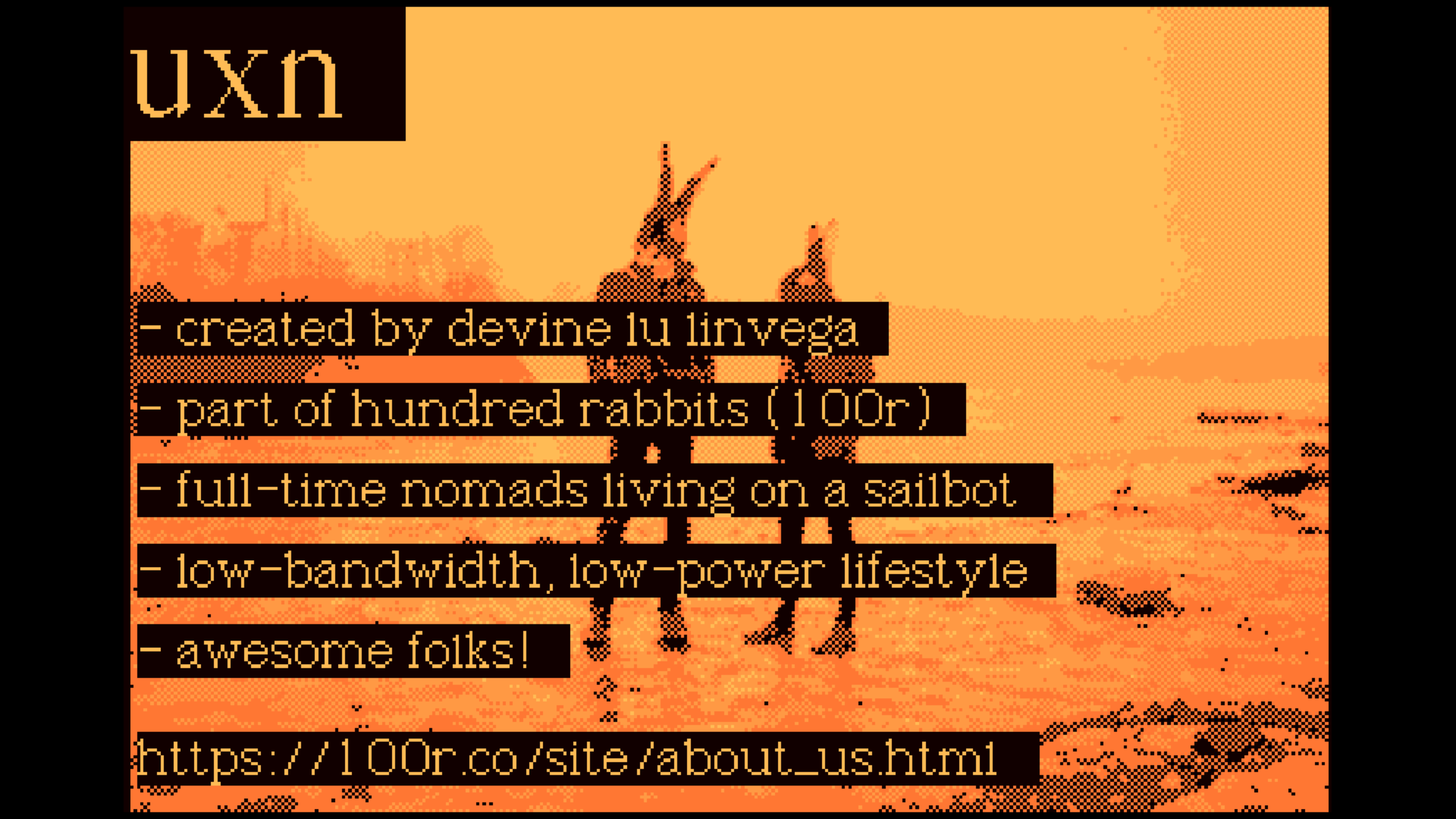
- what is uxn?
- why is it interesting?
- how can it help roguelike devs?
- (finish in under 10 minutes)

about me

- eirikr, d6, d_m, etc.
- angband dev emeritus
- c, lua, python, js, twine, inform, uxn
- not in the "games industry"

http://plastic-idolatry.com/erik

https://merveilles.town/@d6


uxn
- created by devine lu linvega
- part of hundred rabbits (100r)
- full-time nomads living on a sailbot
- low-bandwidth, low-power lifestyle
- awesome folks!
https://100r.co/site/about_us.html

uxn

- uxntal assembly language
- varvara virtual machine specification
- emulators for various platforms
- ROMs implementing programs, games, etc.

uxn

- uxntal assembly language
  * 8-bit stack-based instruction set
  * 64k addressable memory
- varvara virtual machine specification
- emulators for various platforms
- ROMs implementing programs, games, etc.

uxn

- uxntal assembly language
- varvara virtual machine specification
  * shows which ports to read/write
  * i/o, video, sound, filesystem, etc.
- emulators for various platforms
- ROMs implementing programs, games, etc.

uxn

- uxntal assembly language
- varvara virtual machine specification
- emulators for various platforms
  * SDL, X11, the web
  * linux framebuffer, x86 BIOS
  * nintendo gba, ds, 3ds, playdate, nook
  * (and more)
- ROMs implementing programs, games, etc.

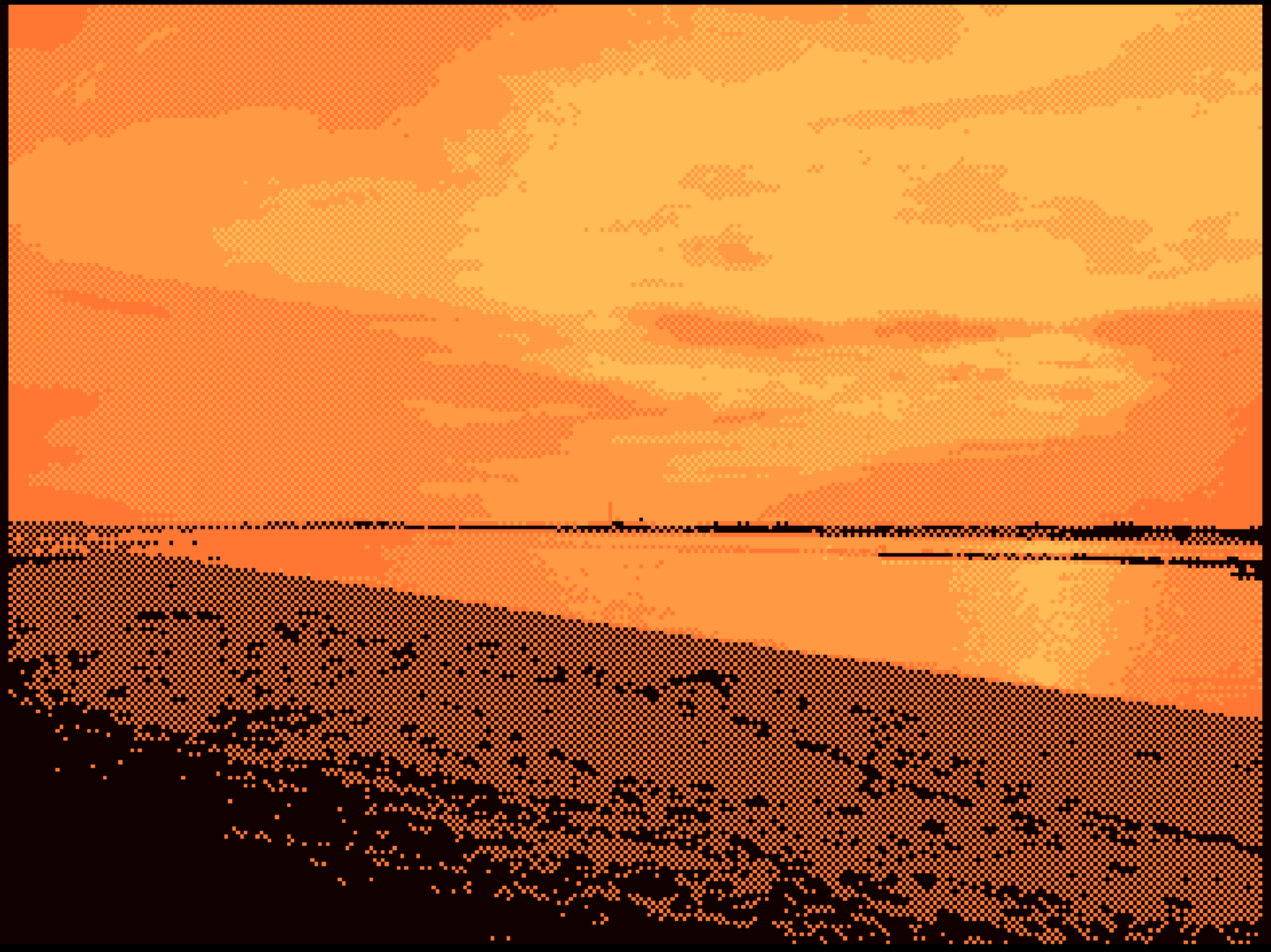
uxn

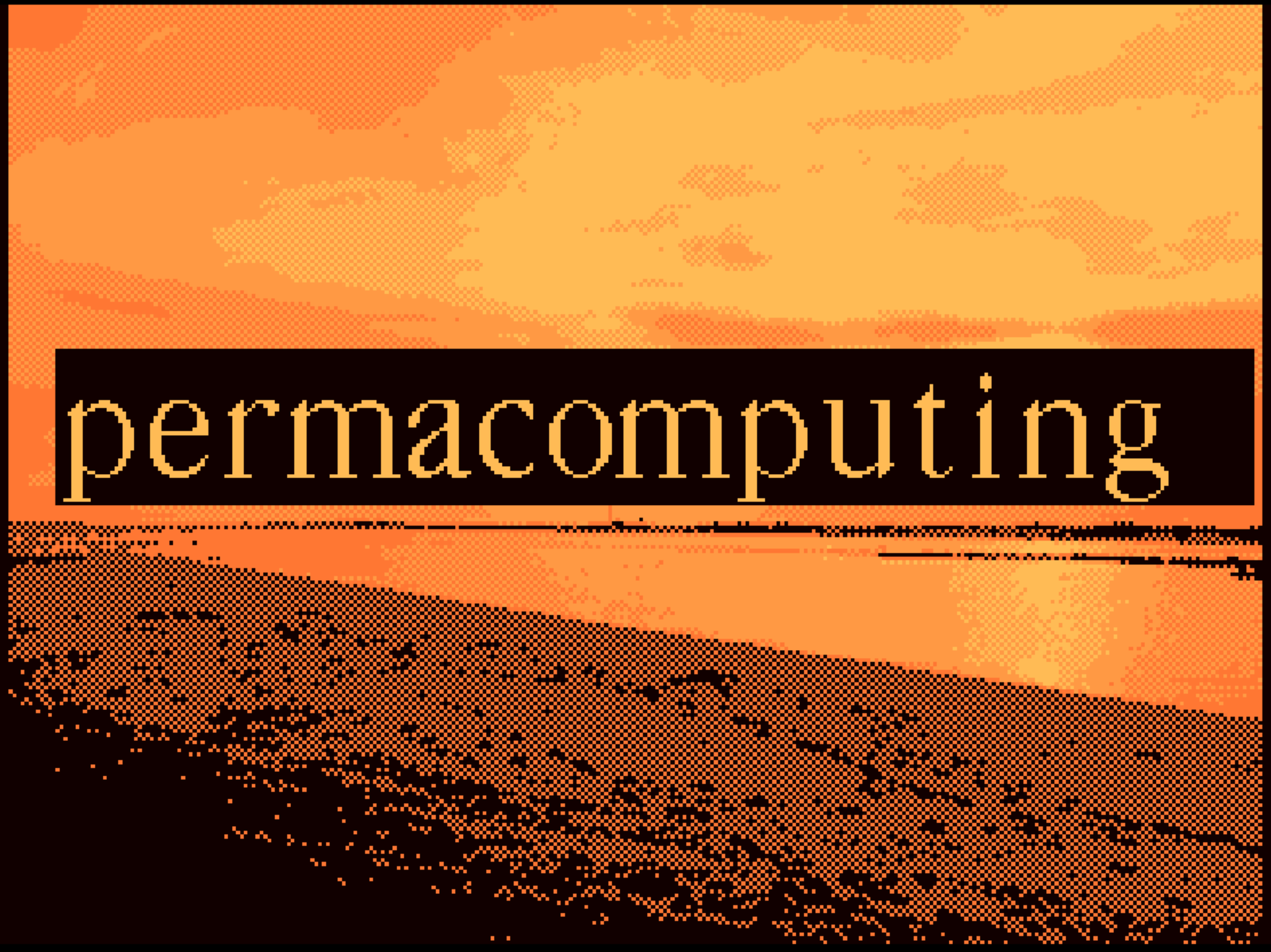
- uxntal assembly language
- varvara virtual machine specification
- emulators for various platforms
- ROMs implementing programs, games, etc.
  * 450 bytes: mandelbrot set
  * 517 bytes: julia set animation
  * 11 kilobytes: terminal emulator
  * 19 kilobytes: klondike solitaire game

more restrictions

- only 8-bit and 16-bit integers
- VM is single-threaded
- filesystem is sandboxed
- no networking support
- only 4-colors (configurable palette)


SDF
freeshell.org
public access unix system
examples
DEAD AIR




permacomputing

permacomputing

- design for older devices
- build for cultural/ecological permanence
- do more with less (better: do less with less)
- conserve energy and materials
- minimize complexity

roguelikes?

rogue

"[Rogue was] the biggest waste of CPU cycles in history."

– Dennis Ritchie

rogue

- design for older devices?
- build for cultural/ecological permanence?
- make do with less?
- conserve energy and materials?
- minimize complexity?

rogue

- design for older devices? yes
- build for cultural/ecological permanence?
- make do with less?
- conserve energy and materials?
- minimize complexity?

rogue

- design for older devices? yes
- build for cultural/ecological permanence? yes
- make do with less?
- conserve energy and materials?
- minimize complexity?

rogue

- design for older devices? yes
- build for cultural/ecological permanence? yes
- make do with less? yes
- conserve energy and materials?
- minimize complexity?

rogue

- design for older devices? yes
- build for cultural/ecological permanence? yes
- make do with less? yes
- conserve energy and materials? yes
- minimize complexity?

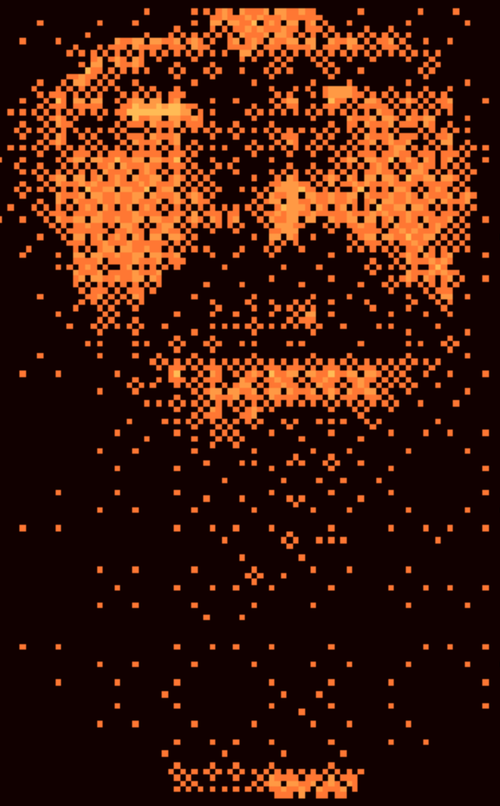
rogue

- design for older devices? yes
- build for cultural/ecological permanence? yes
- make do with less? yes
- conserve energy and materials? yes
- minimize complexity? yes

thesis

uxn is a good fit for roguelikes because it is aligned with many of the things that made rogue and roguelikes interesting and successful.

demo^W confession



varvara

varvara

- program counter (16-bit value)
    * address of the next instruction to run

varvara

- program counter (16-bit value)
  * address of the next instruction to run
- two 256-byte stacks (wst and rst)
  * passing and working with values

varvara

- program counter (16-bit value)
  * address of the next instruction to run
- two 256-byte stacks (wst and rst)
  * passing and working with values
- 64 kilobytes of main memory
  * program instructions and data

varvara

- program counter (16-bit value)
  * address of the next instruction to run
- two 256-byte stacks (wst and rst)
  * passing and working with values
- 64 kilobytes of main memory
  * program instructions and data
- device ports
  * read/write state and take actions

varvara, that's it!

- program counter (16-bit value)
  * address of the next instruction to run
- two 256-byte stacks (wst and rst)
  * passing and working with values
- 64 kilobytes of main memory
  * program instructions and data
- device ports
  * read/write state and take actions

startup

- Varvara reads ROM into memory
- starts at address 0x100 (256)
- sets program counter (pc) to 256
- executes until break (BRK) instruction

vectors

- reset: runs when emulator (re)starts
- console: runs when text is read on stdin
- screen: runs up to 60 times/second
- audio: runs when previous note ends
- mouse: runs on mouse input
- controller: runs on keyboard/controller input

each vector is an address of code to execute.

vectors continue until break (BRK) occurs.

why uxn?

why uxn?

- radical simplicity

why uxn?

- radical simplicity
- many implementations

why uxn?

- radical simplicity
- many implementations
- hardware compatibility

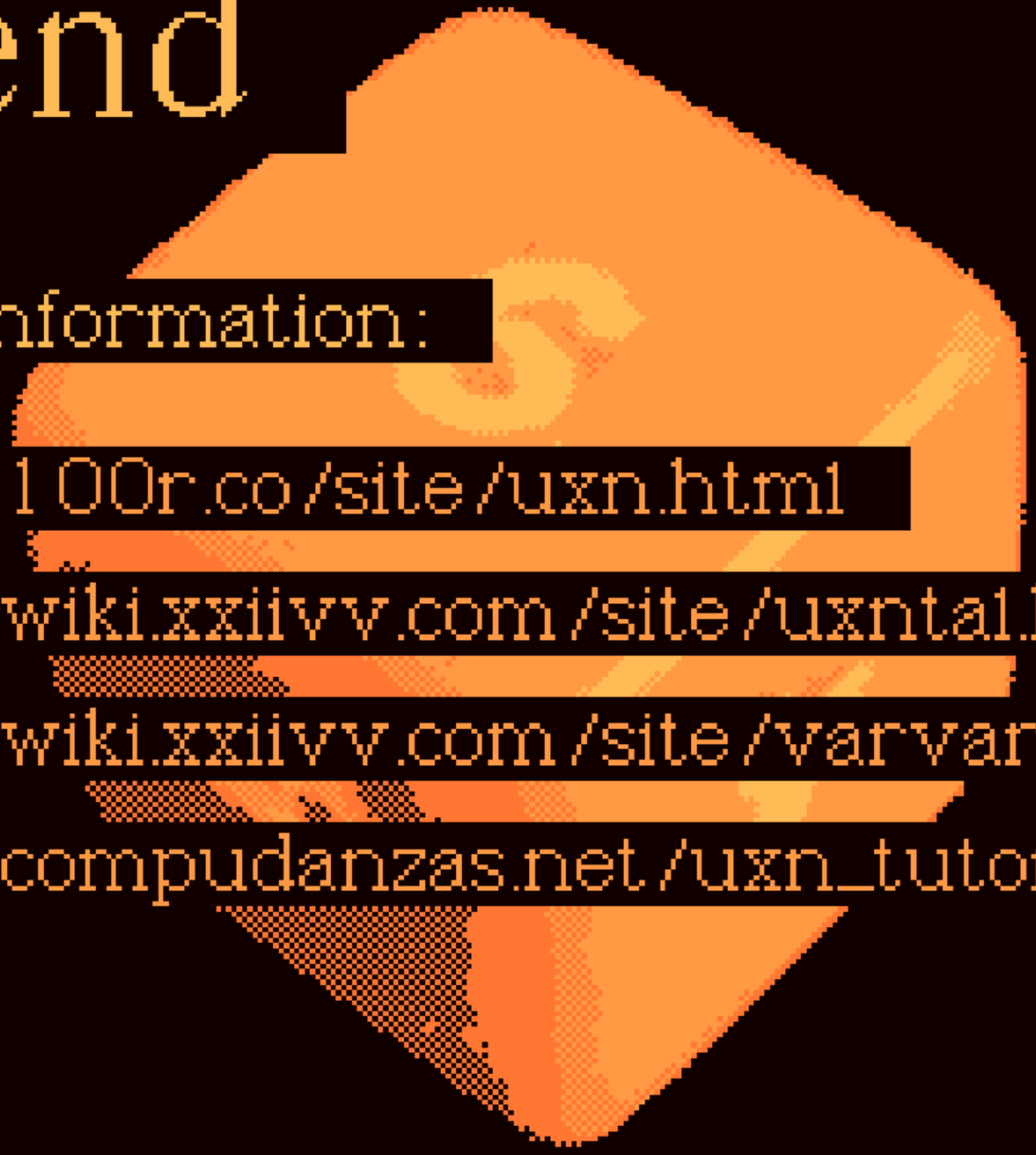
why uxn?

- radical simplicity
- many implementations
- hardware compatibility
- "choose your own adventure"

would a smaller, simpler, cozier
computing environment
be adequate or even beneficial
for your game development practice?

(...and for the world?)

the end



for more information:

https://100r.co/site/uxn.html

https://wiki.xxiivv.com/site/uxntal.html

https://wiki.xxiivv.com/site/varvara.html

https://compudanzas.net/uxn_tutorial.html