The Polygons of Another World

Interactive graphics in R

mike
@cool but useless

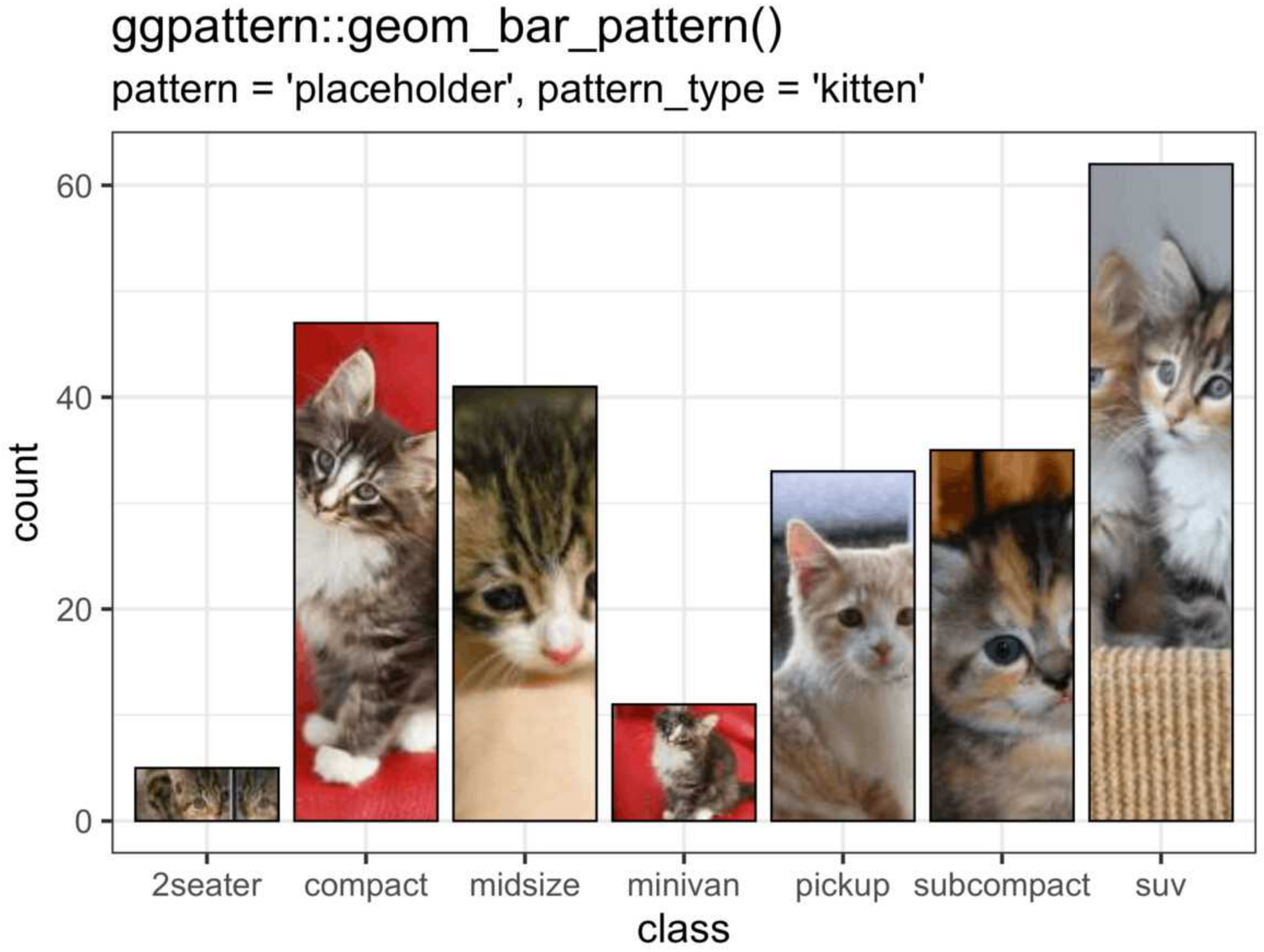
About Me

@coolbutuseless

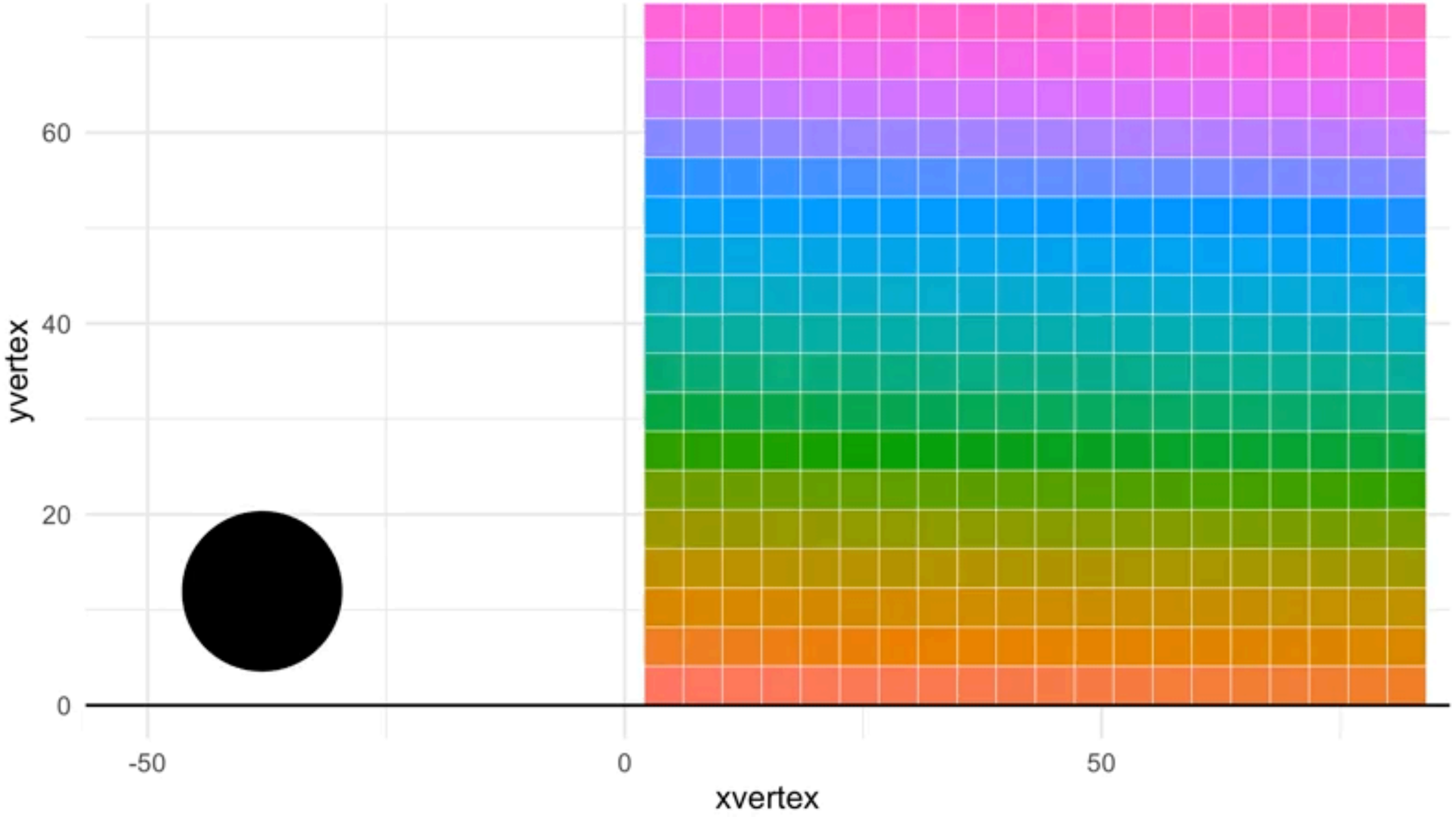
I code in R for a living.

I believe R can be fun.

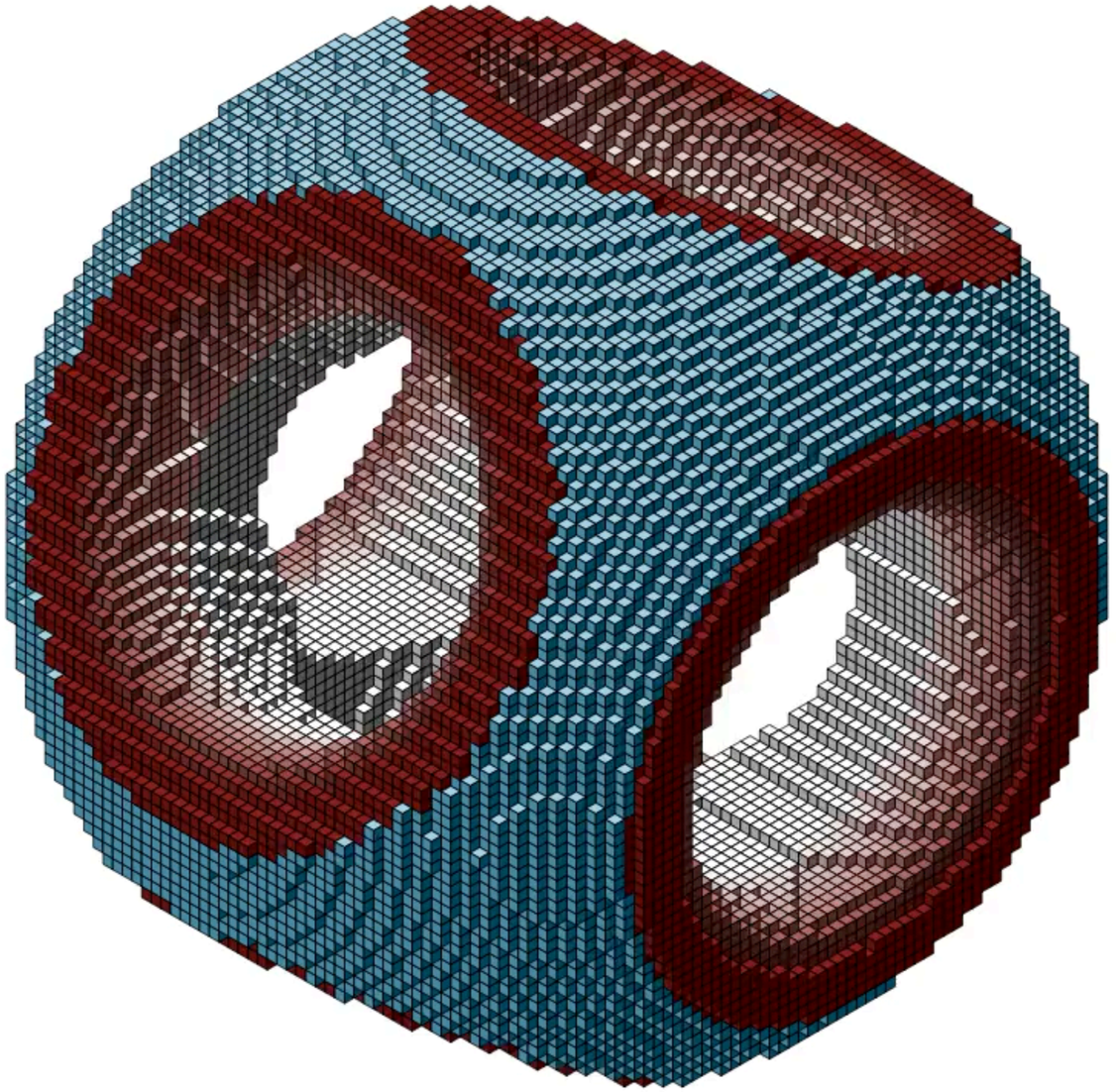
{ggpattern}


ggpattern::geom_bar_pattern()
pattern = 'placeholder', pattern_type = 'kitten'
count
60
40
20
0
2seater
compact
midsize
minivan
pickup
subcompact
suv
class

{chipmunk}



{isocubes}



With careful use of base R,

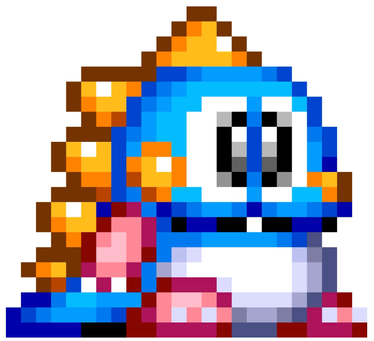
interactive graphics are possible,

opening the way for new visualisations!

With careful use of base R,

interactive graphics are possible,

opening the way for ~~new visualisations~~.



Games!

The Summer* of Wordle

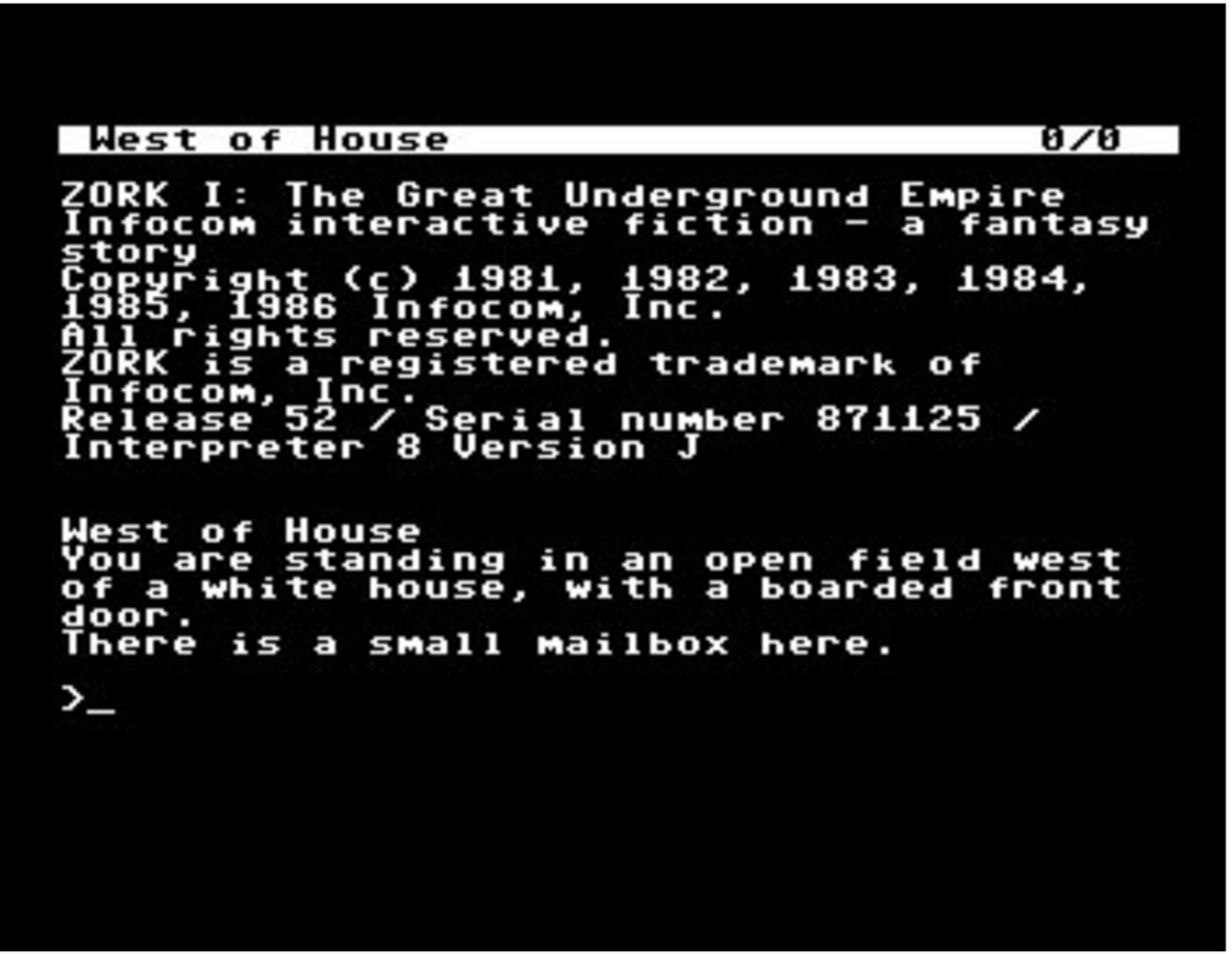
December 2021

* Southern Hemisphere

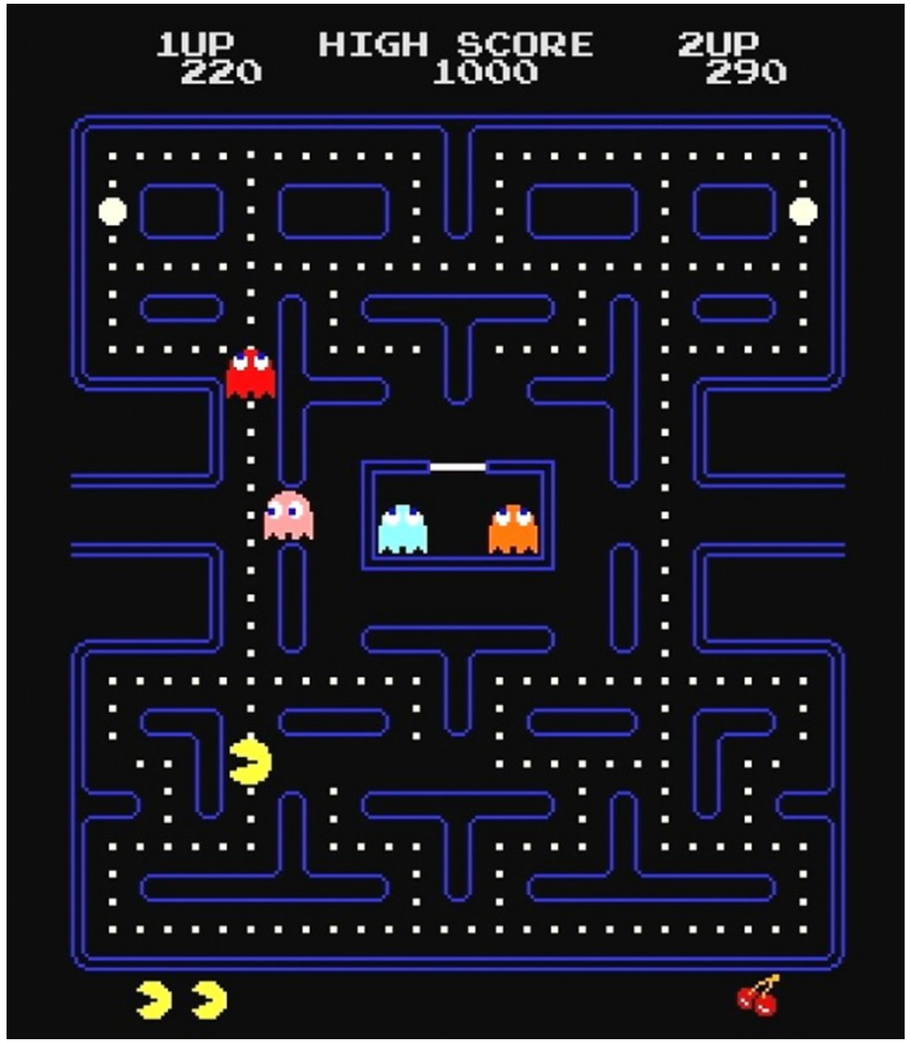
Wordle @ NYT

Play {wordle} in R

I have this …



but I want this!



Moonshot

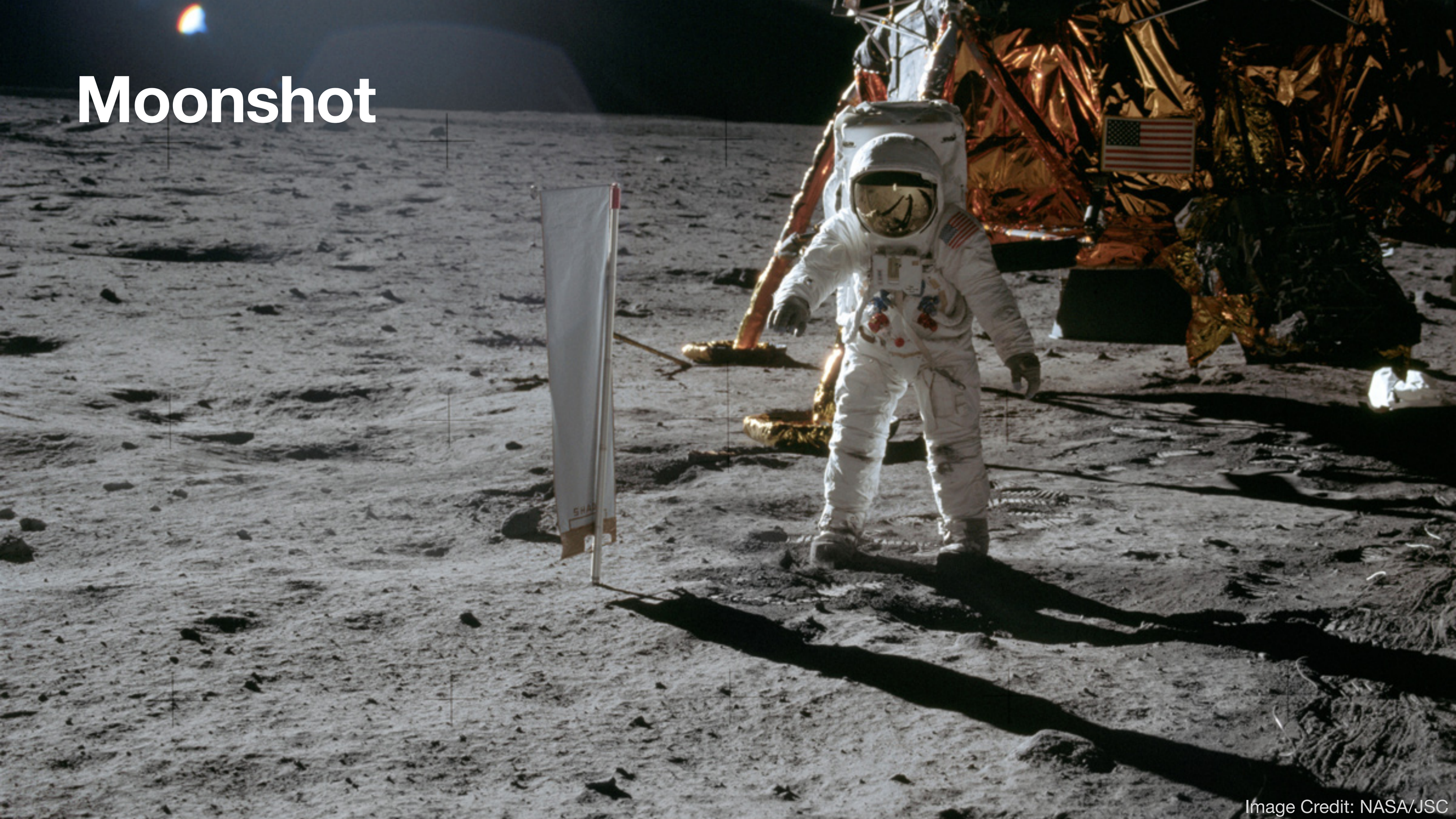


Image Credit: NASA/JSC

Moonshot

“I will commit myself to achieving the goal, before this year is out, of writing and playing a video game in R.”
Me, 2022

R + ??? = Games

1. Fast graphics device
2. Fast way of drawing
3. Mouse & keyboard interaction

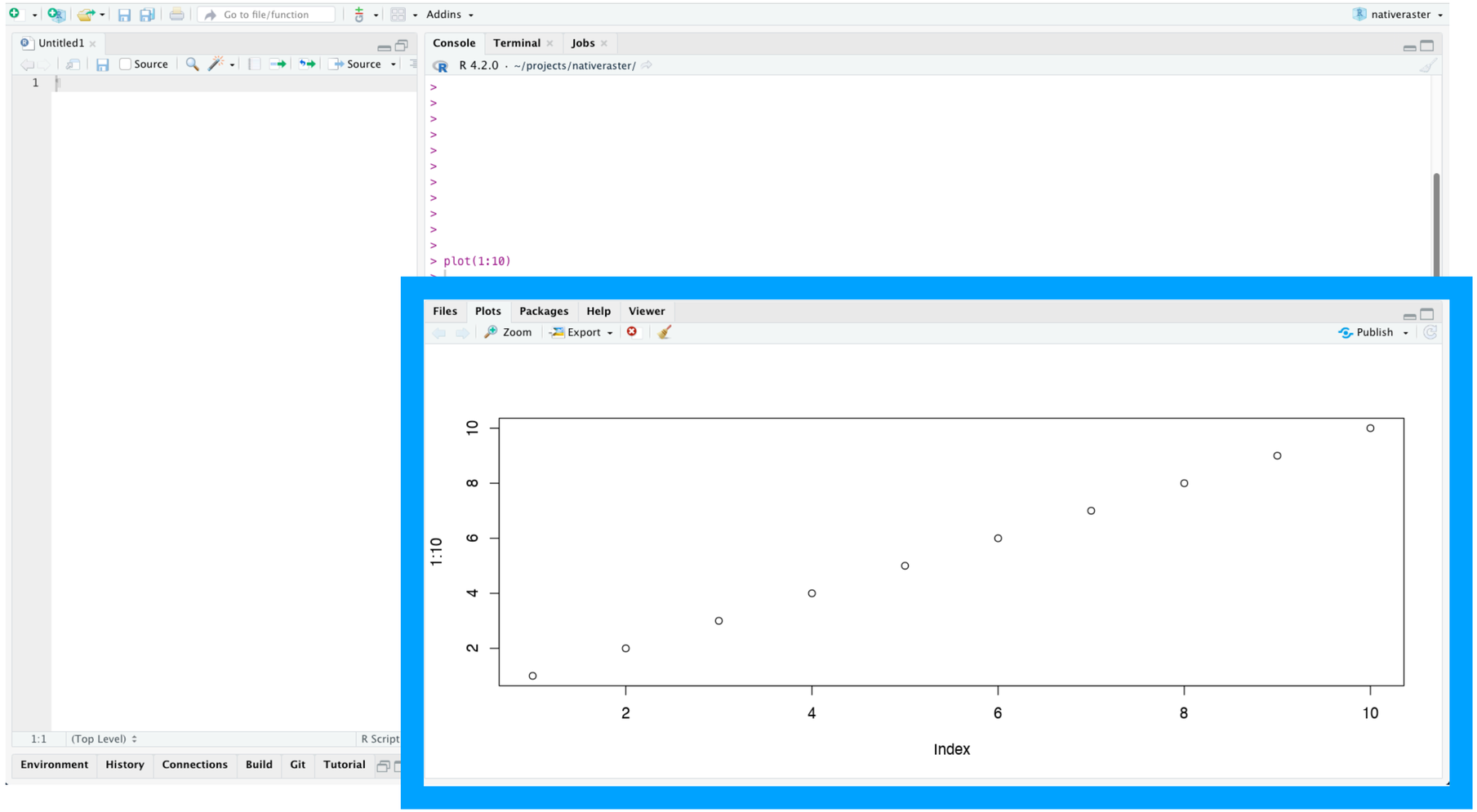
1. Fast graphics device
2. Fast way of drawing
3. Mouse & keyboard interaction

Fast graphics device

Fast = 30 fps

fps = frames per second

Fast graphics device



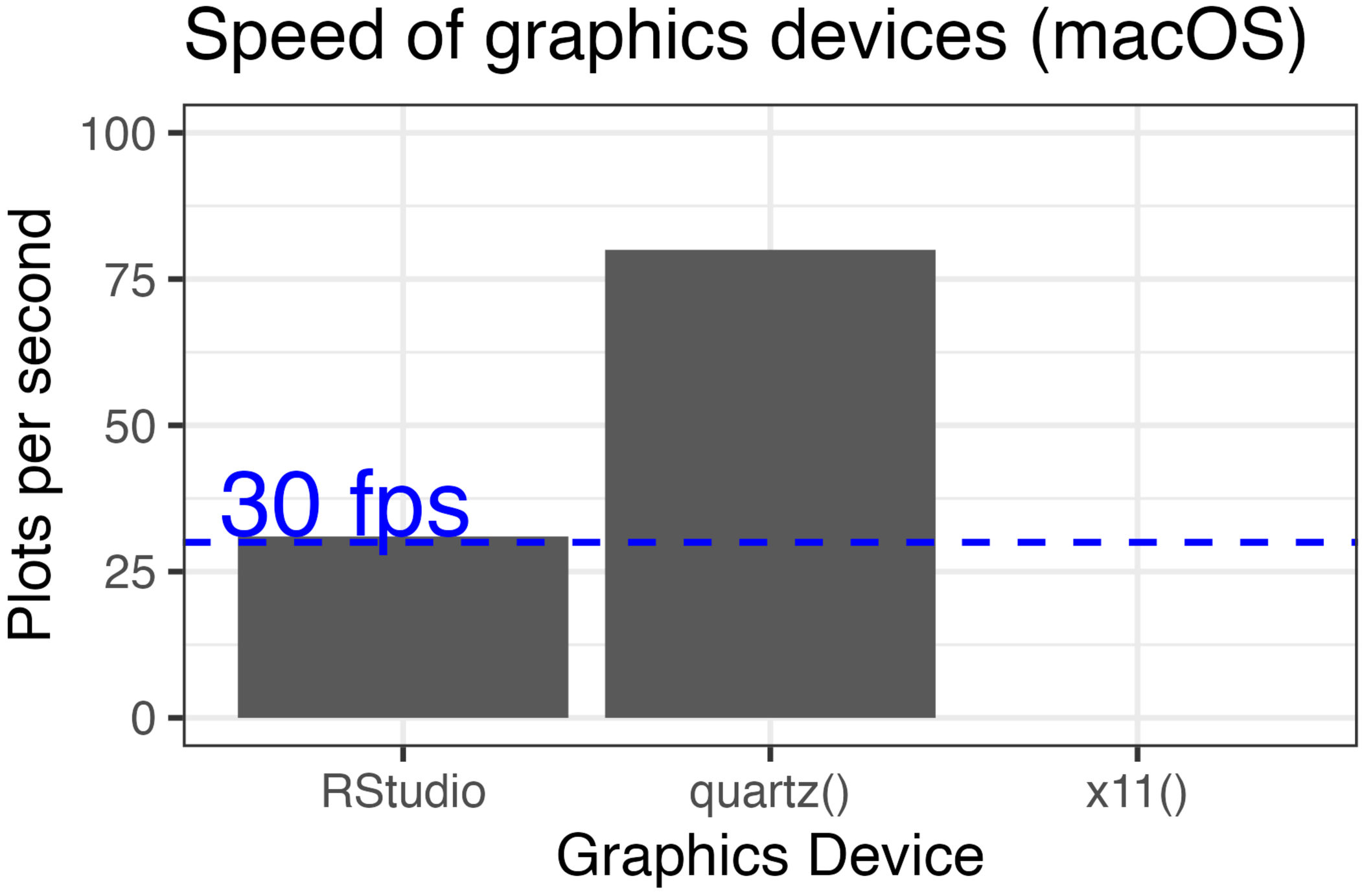
Fast graphics device

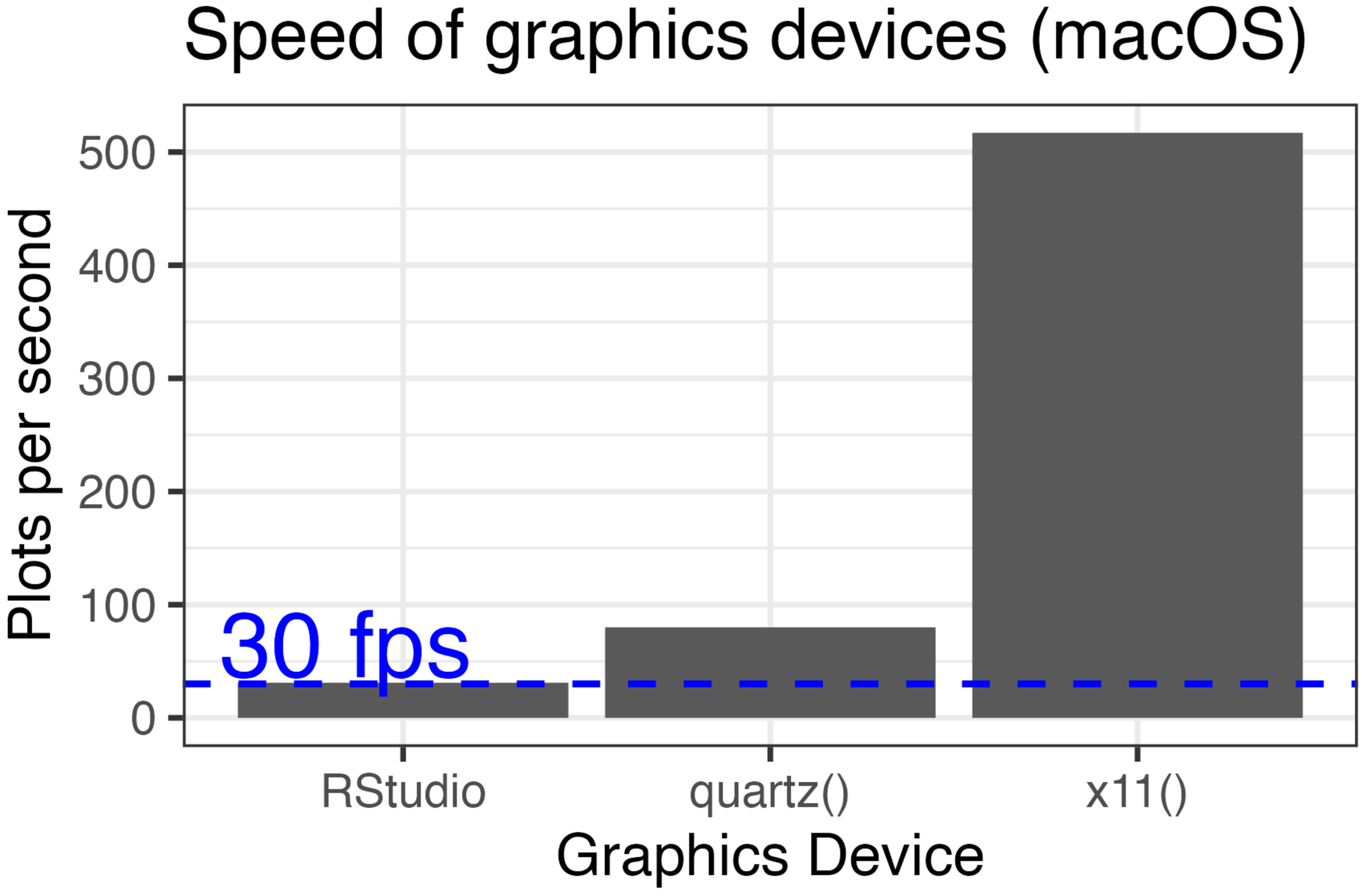
On macOS

- RStudio
- quartz()
- x11()

Graphics device benchmark

```
for (i in 1:100) {
  plot(1:10)
}
```


Speed of graphics devices (macOS)
Plots per second
100
75
50
25
0
30 fps
RStudio
quartz()
x11()
Graphics Device


Speed of graphics devices (macOS)
Plots per second
500
400
300
200
100
0
30 fps
RStudio
quartz()
x11()
Graphics Device

Fast graphics device? ✅

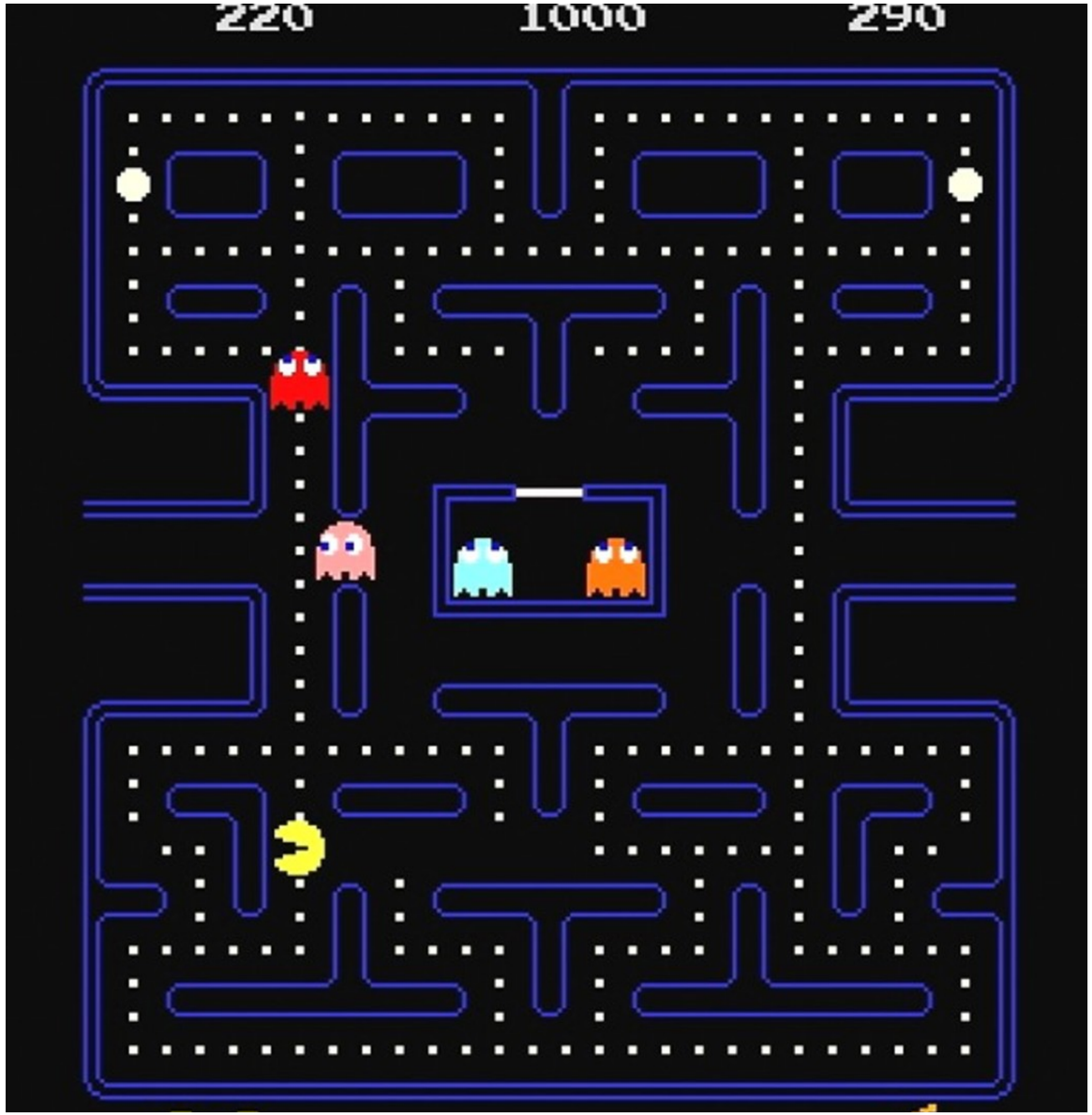
- All graphics devices can respond fast enough
- Some graphics devices are faster than others
- The ‘x11’ device is ridiculously fast.

1. Fast graphics device

2. Fast way of drawing

3. Mouse & keyboard interaction

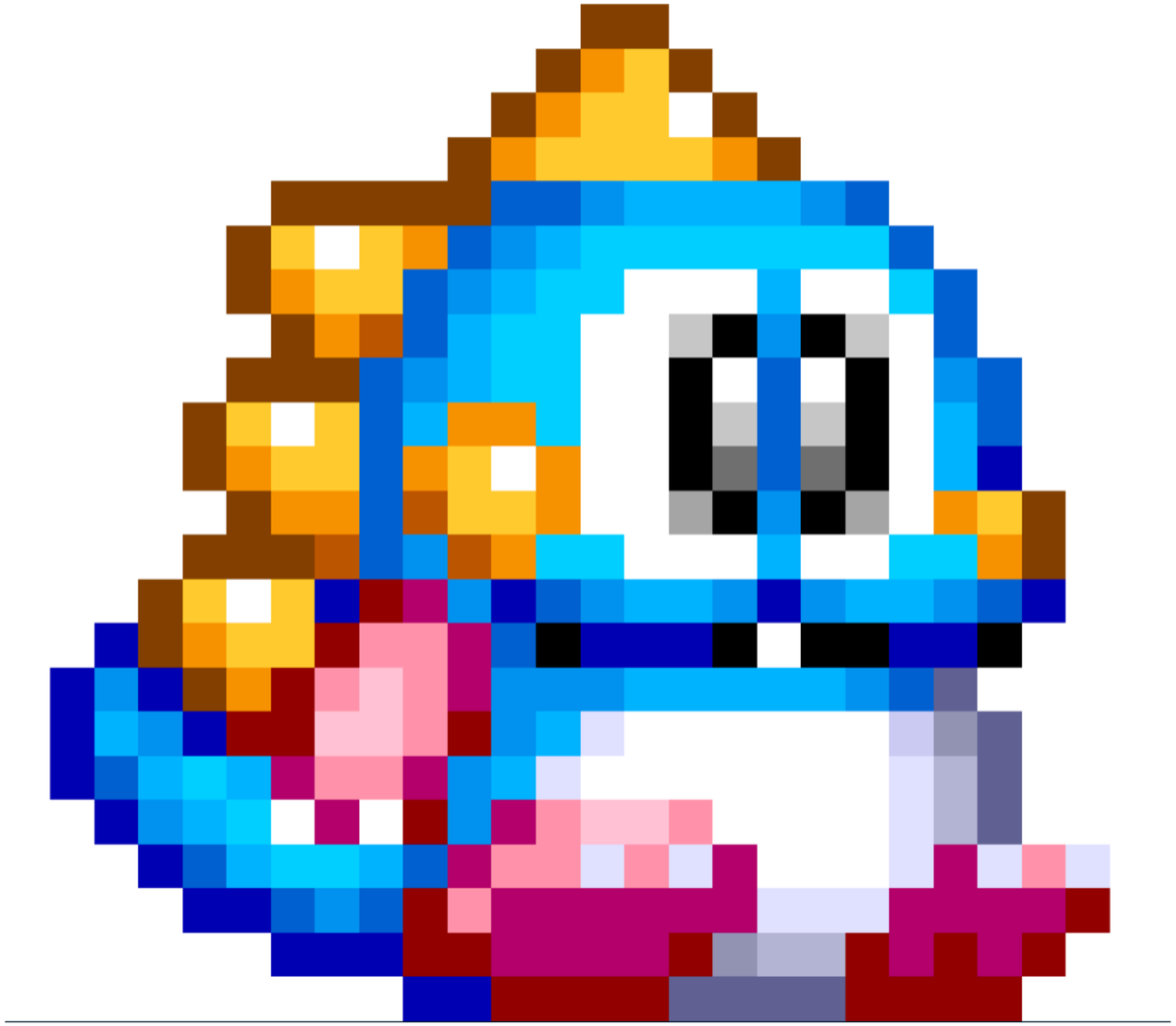
Pixel Graphics



Draw Pixels

x	y	colour
12	24	yellow
12	25	red
13	15	blue
...	...	...

```
grid.rect(pixels)
```



Live Rendering

Double-buffered rendering

Create buffer in R

For every frame

1. Draw game into buffer
2. Copy buffer to graphics device

Buffer = R matrix ?

blue	blue	grey	grey	grey	grey	blue	blue
blue	blue	grey	green	green	grey	blue	blue
blue	blue	grey	grey	grey	grey	blue	blue
blue	blue	grey	grey	blue	blue	blue	blue
blue	blue	grey	green	grey	blue	blue	blue
blue	blue	grey	green	green	grey	blue	blue

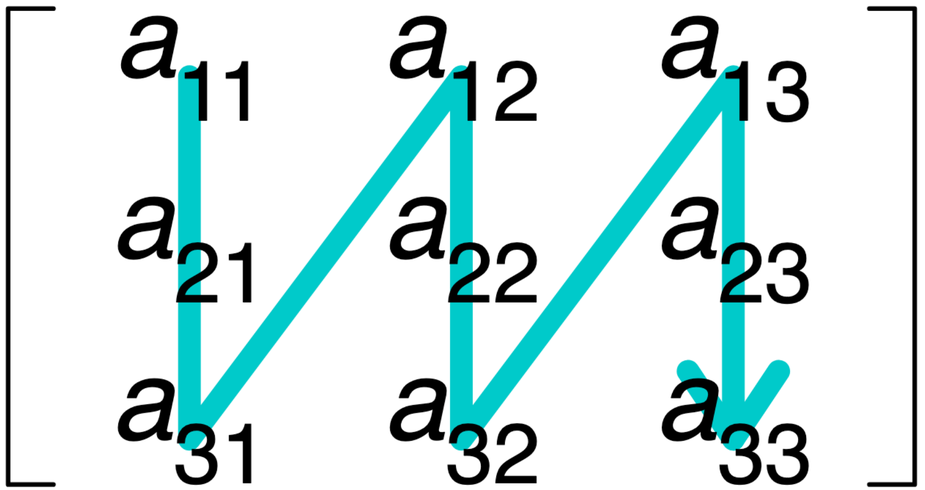
Buffer = R matrix ?

Two challenges

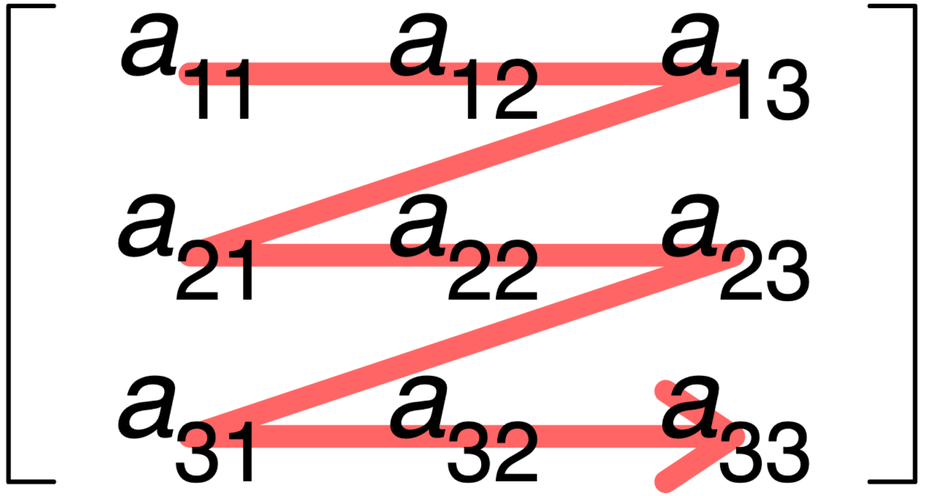
- data ordering
- colour representation

Challenge 1: Data ordering

R matrix



Graphics device



Challenge 2: Colour representation

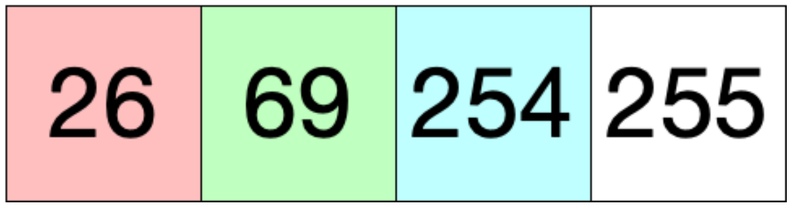
R matrix

String hex colour

“#1a45feff”

Graphics device

Four bytes RGBA



Buffer = ~~R matrix~~

- Matrix data layout does not match graphics device.
- Data conversion for every frame => slow!

Buffer = nativeRaster

Buffer = nativeRaster

- Built-in data structure in R
- Rarely used
- Difficult to manipulate from R
- Highly compatible with graphics devices

Buffer = nativeRaster

	matrix	array	native raster	graphics device
Four-byte colour	❌	❌	✅	✅
Row-major data order	❌	❌	✅	✅

Double-buffered rendering

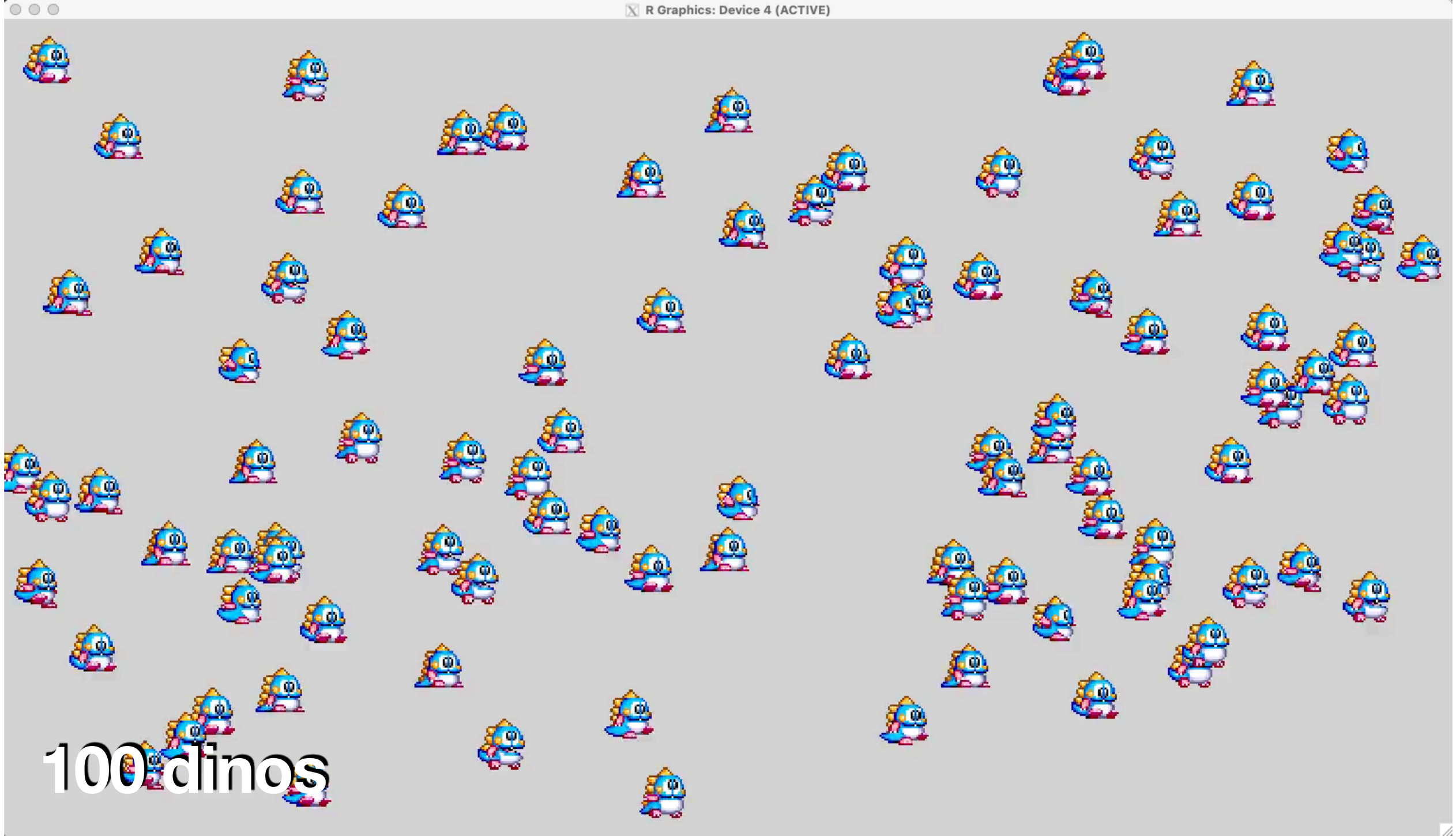
Create ***native raster*** buffer in R

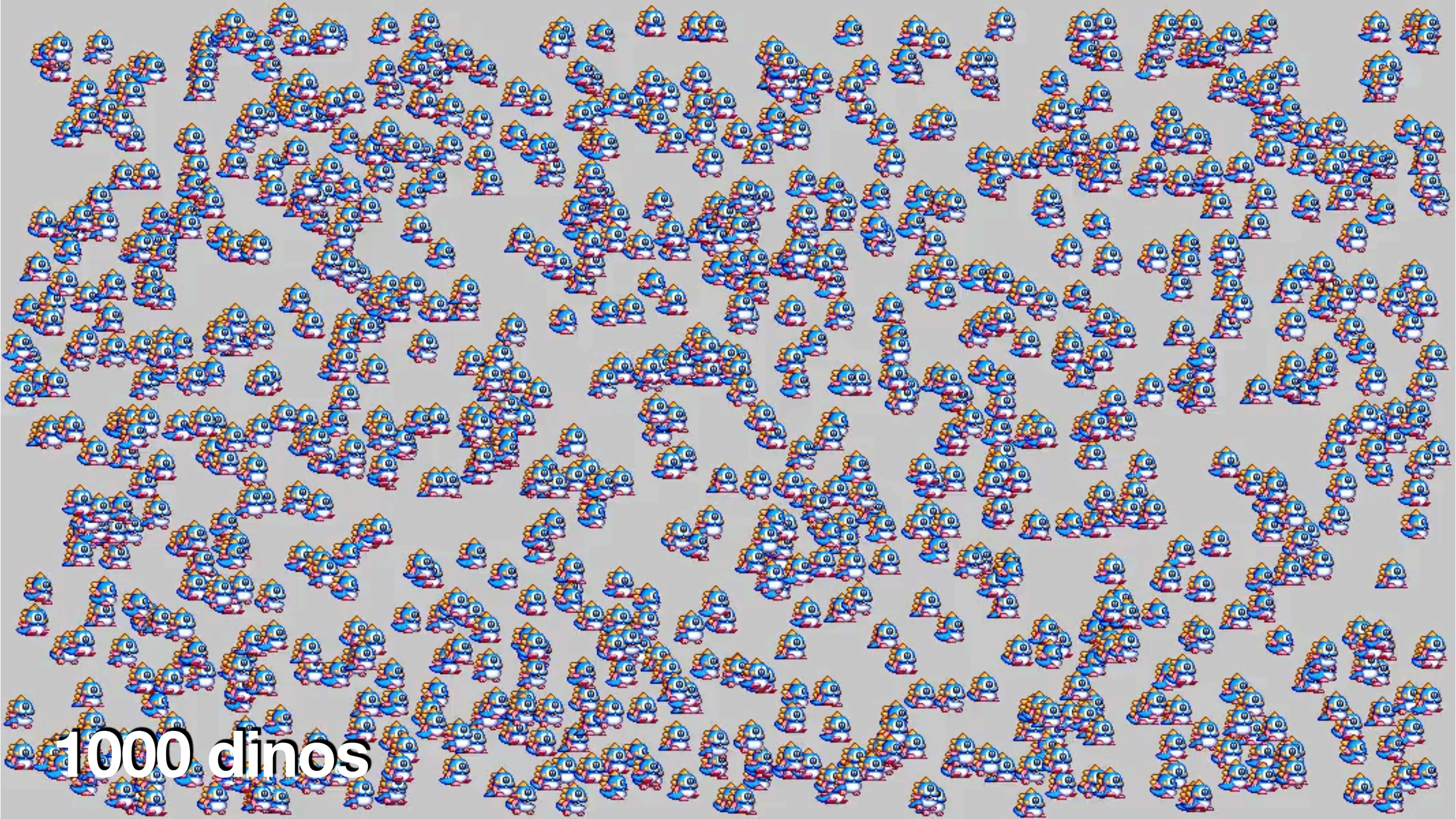
For every frame

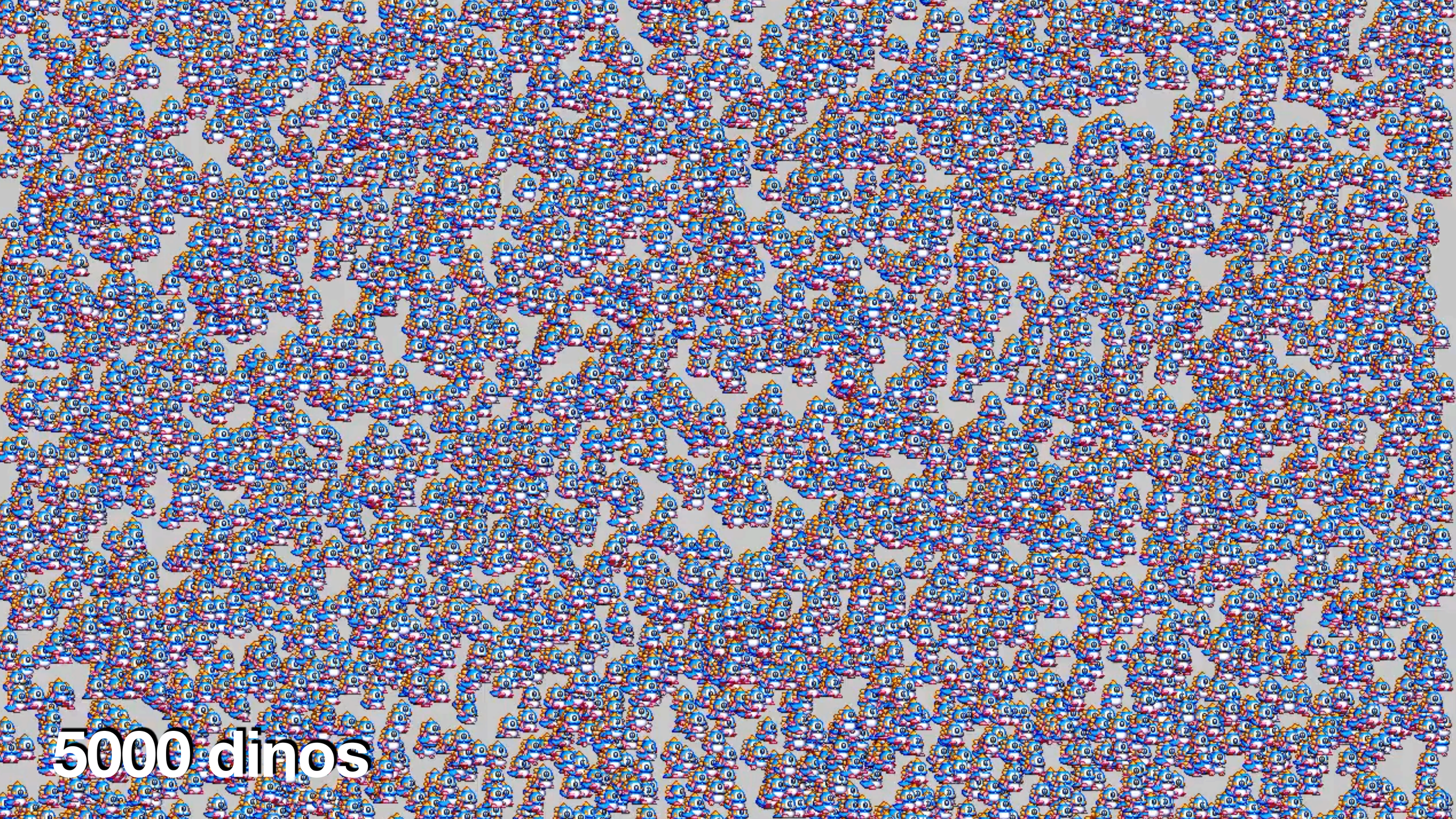
1. Draw game into buffer
2. Copy buffer to graphics device

```
grid.raster(buffer)
```

3 dinos


R Graphics: Device 4 (ACTIVE)
100 dinos


1000 dinos


5000 dinos

Fast way of drawing? ✅

- Double buffering
- Use nativeRaster as a drawing buffer
- Copy buffer to graphics device

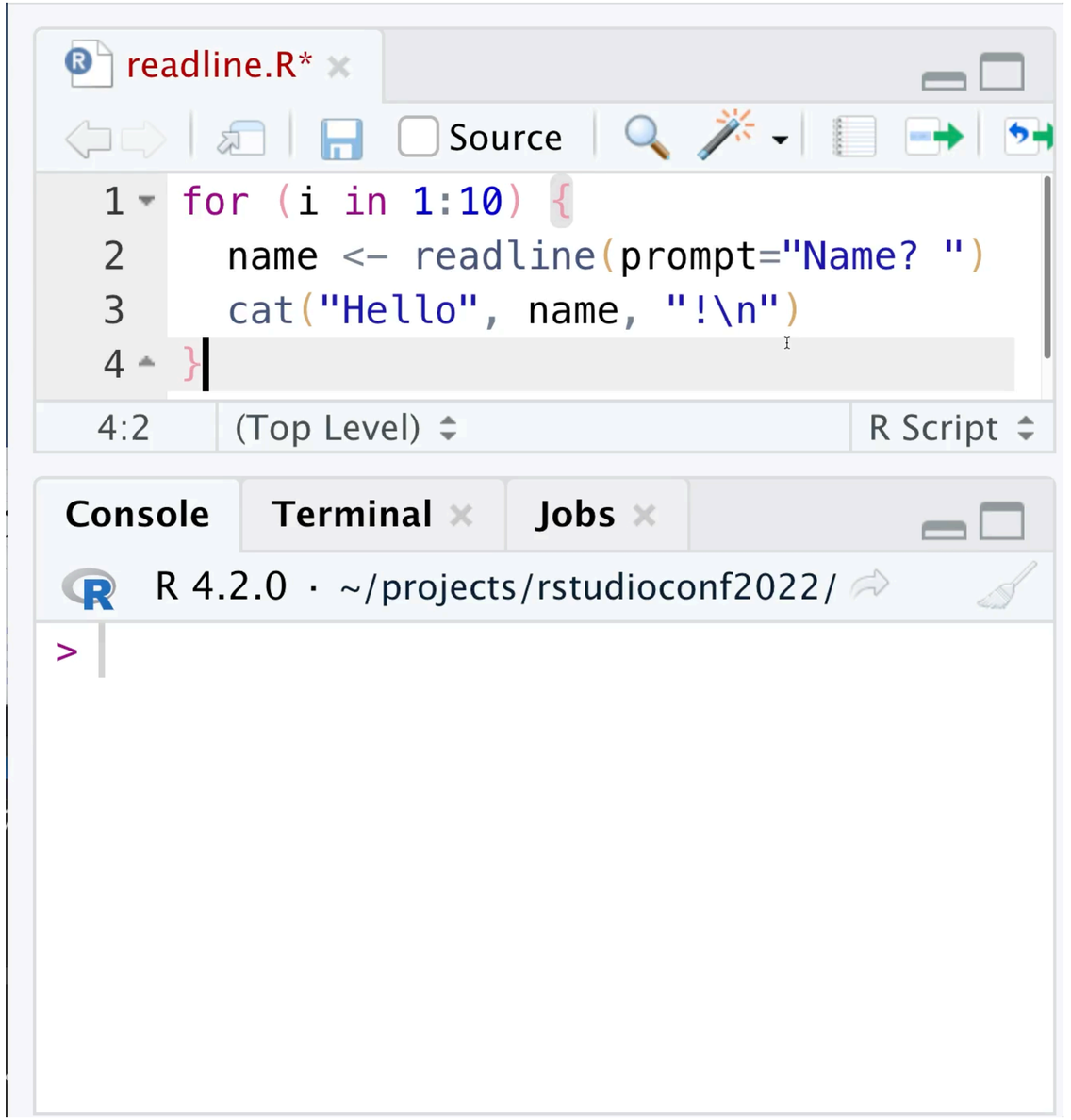
1. Fast graphics device

2. Fast way of drawing

3. Mouse & keyboard interaction

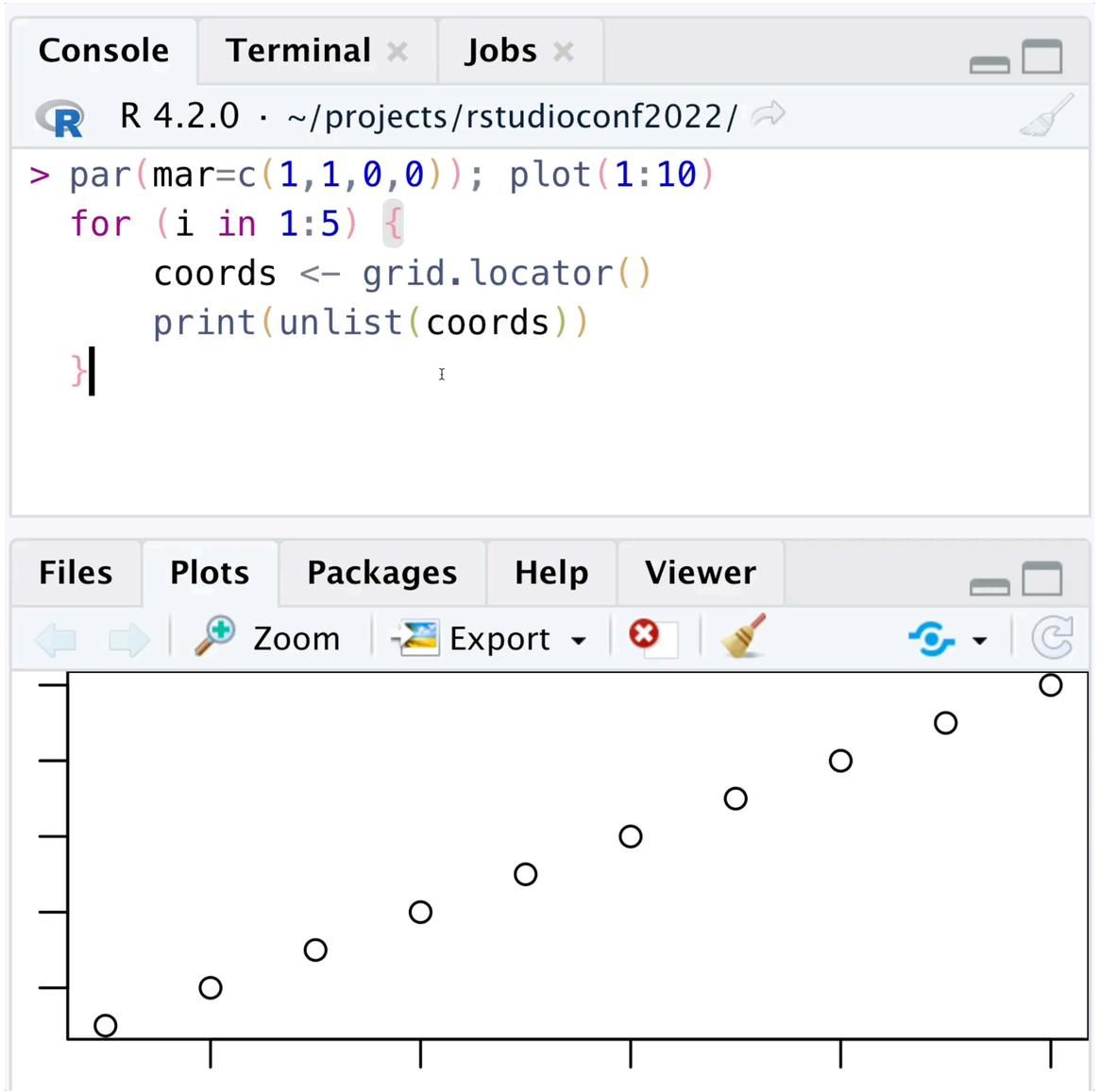
Keyboard Interaction

readline()



Mouse Interaction

grid.locator()



Rendering is blocked!

```
# Not good for games

while (TRUE) {
  readline()
  update_graphics()
}
```

Event-driven interaction

Ask the system:

When [x] happens, please do [y]

Event-driven interaction - Shiny

Ask Shiny:

> When button is clicked, run my function.

```
observeEvent(event, func)
```

Graphics device events

Ask graphics device:

When mouse is moved, run my function.

```
grDevices::setGraphicsEventHandlers(
      onMouseMove = my_function
)
```

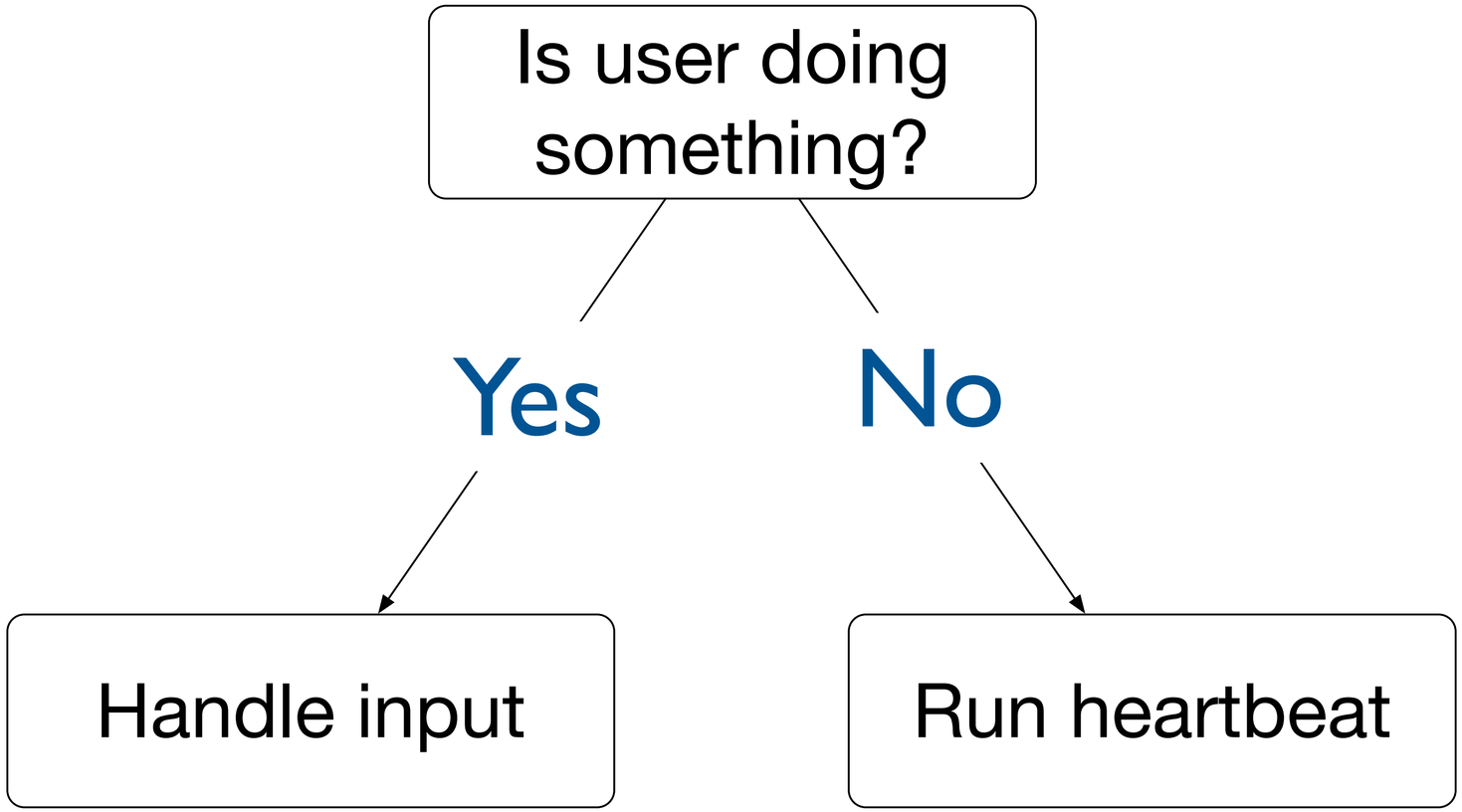
Graphics device events

Event	
Mouse moved	x, y
Mouse button	button number
Keyboard	which key

Graphics device events

Event	
Mouse moved	x, y
Mouse button	button number
Keyboard	which key
Idle	**???**

‘idle’ event = heartbeat



Event-driven application

```
grDevices::setGraphicsEventHandlers(
        onMouseMove = update_position,
        onKeybd     = check_for_quit,
        onIdle      = render_next_frame
)
```

TR-808

Electronic Drum Machine

{tr808r}

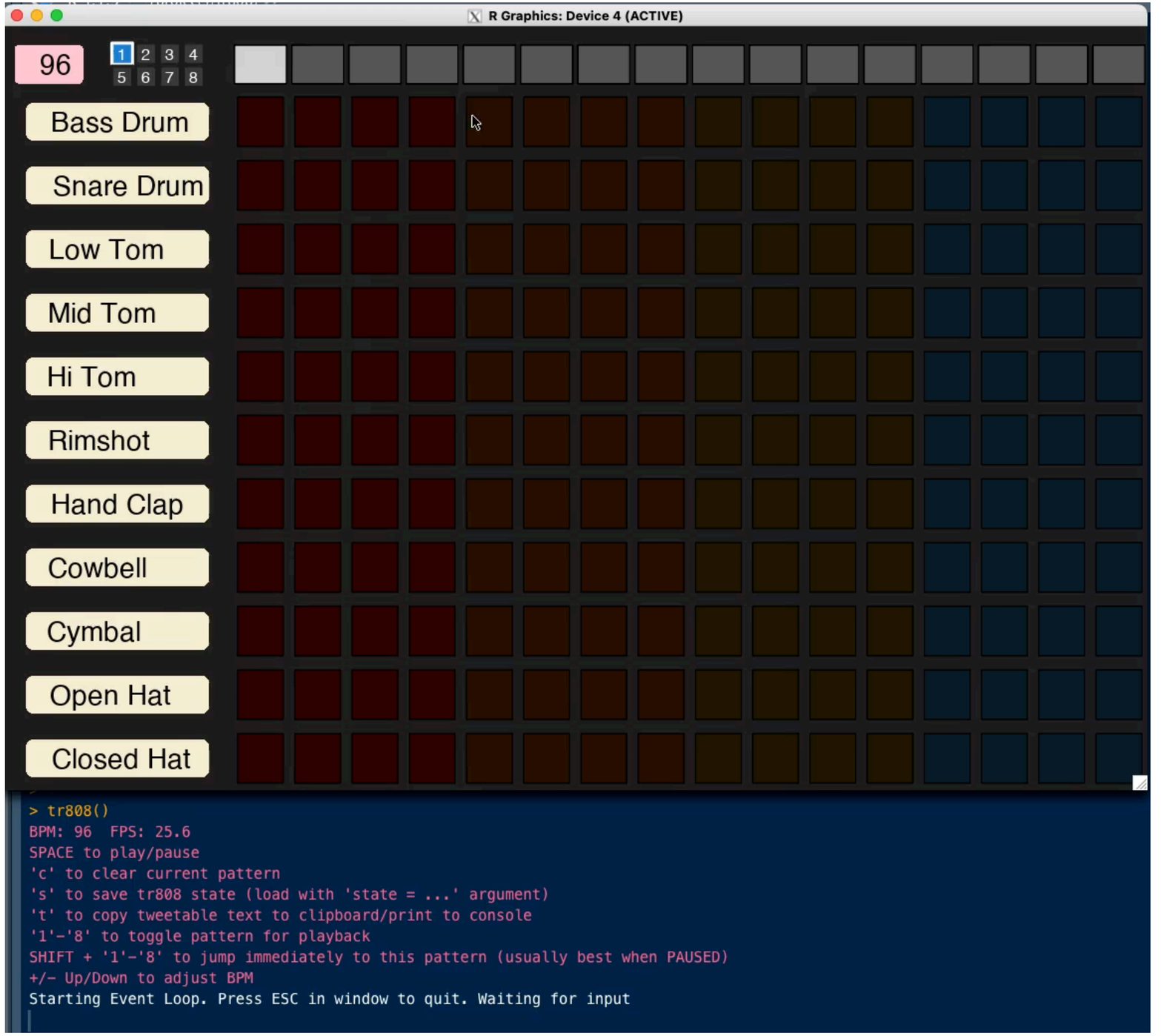
Ask device:

When mouse is pressed

- Toggle the button

When user is idle

- Play the next beat



Mouse & Keyboard? ✅

- Event-driven interaction
- Simultaneously render graphics while reacting to user input

1. Fast graphics device ✅

2. Fast way of drawing ✅

3. Mouse/keyboard interaction ✅

1. Fast graphics device ✅

2. Fast way of drawing ✅

3. Mouse/keyboard interaction ✅

4. Write a game


ANOTHER WORLD

“Another World”

- Action/Adventure
- Side-scrolling platform
- Released 1991
- Amiga, AtariST, MSDOS
- Cinematic cut-scenes
- Novel virtual machine architecture
- Free demo available


Another World
Polygon Engine


Another World
Polygon Engine

R version of Another World

- Pure R implementation of the Virtual Machine
  - Based upon Javascript version
    - Based upon the C version
      - Based upon the MSDOS x86 assembly version
        - Based upon the Amiga 68k assembly version (original)

Event-driven game loop

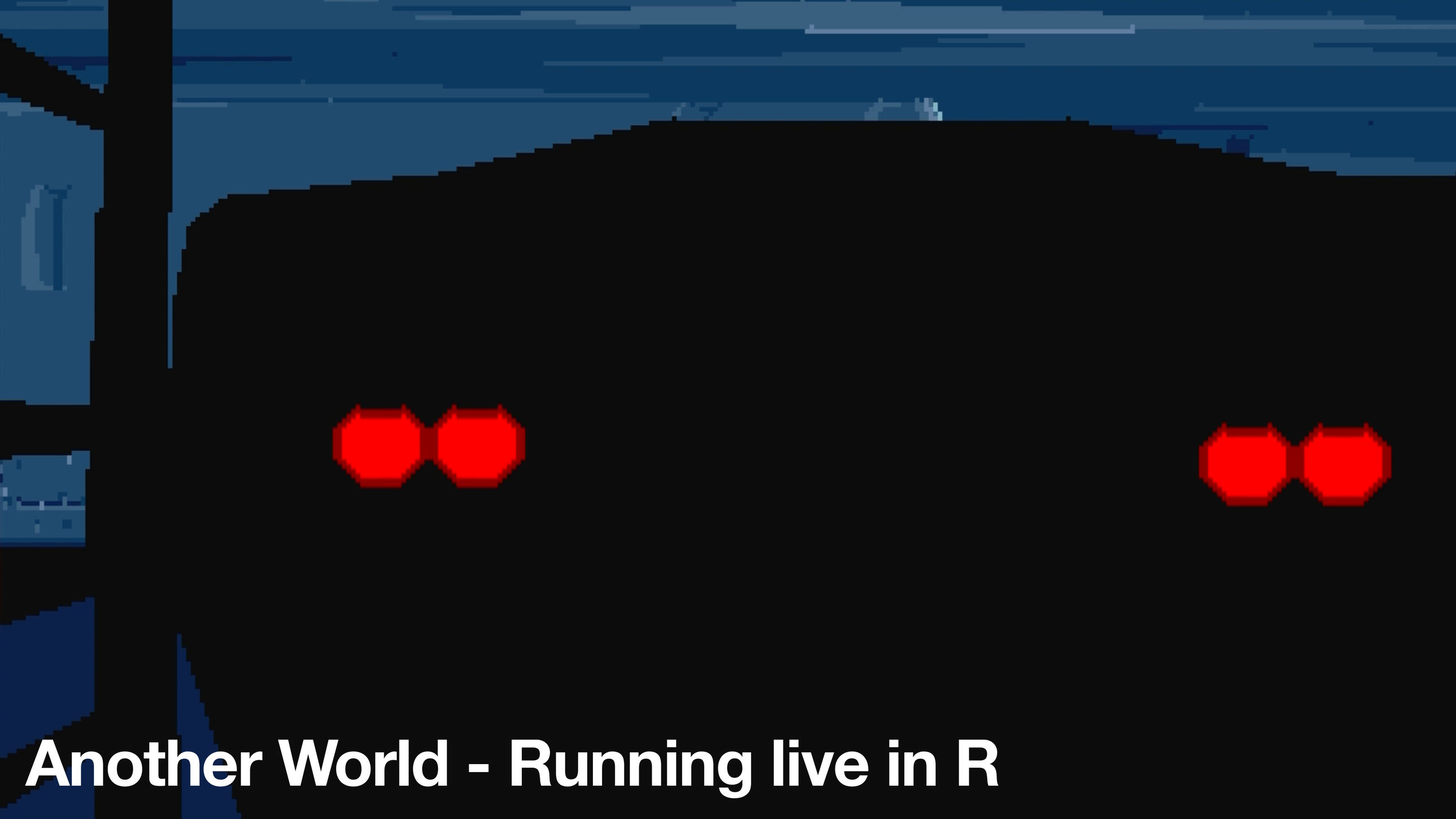
Ask the graphics device:

When mouse/keyboard used:

- Update variables in the Virtual Machine

When user is idle:

- Execute the next instruction e.g.
  - draw polygon
  - play sound, etc



Another World - Running live in R

Another World - Keyboard controls swaying

Moonshot: 90% complete

- Complete the virtual machine!
- Release {AnotherWorld}

What’s Next?

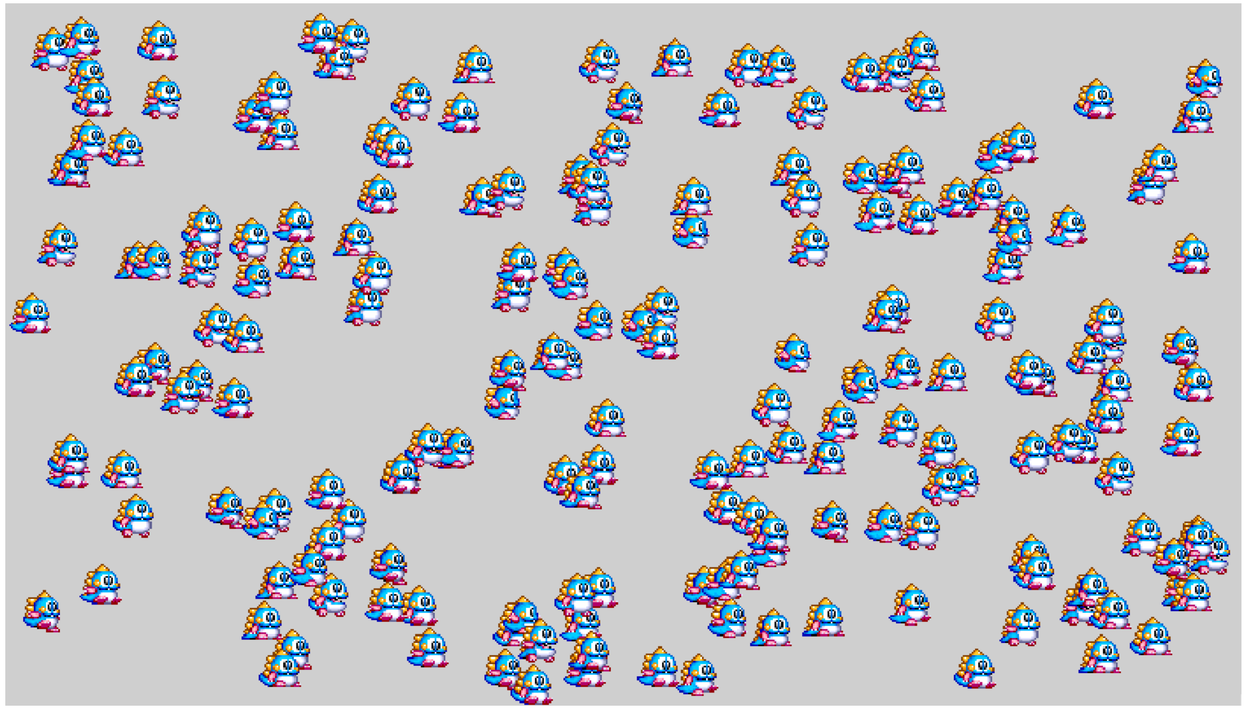
With careful use of base R,

interactive graphics are possible,

opening the way for new visualisations!

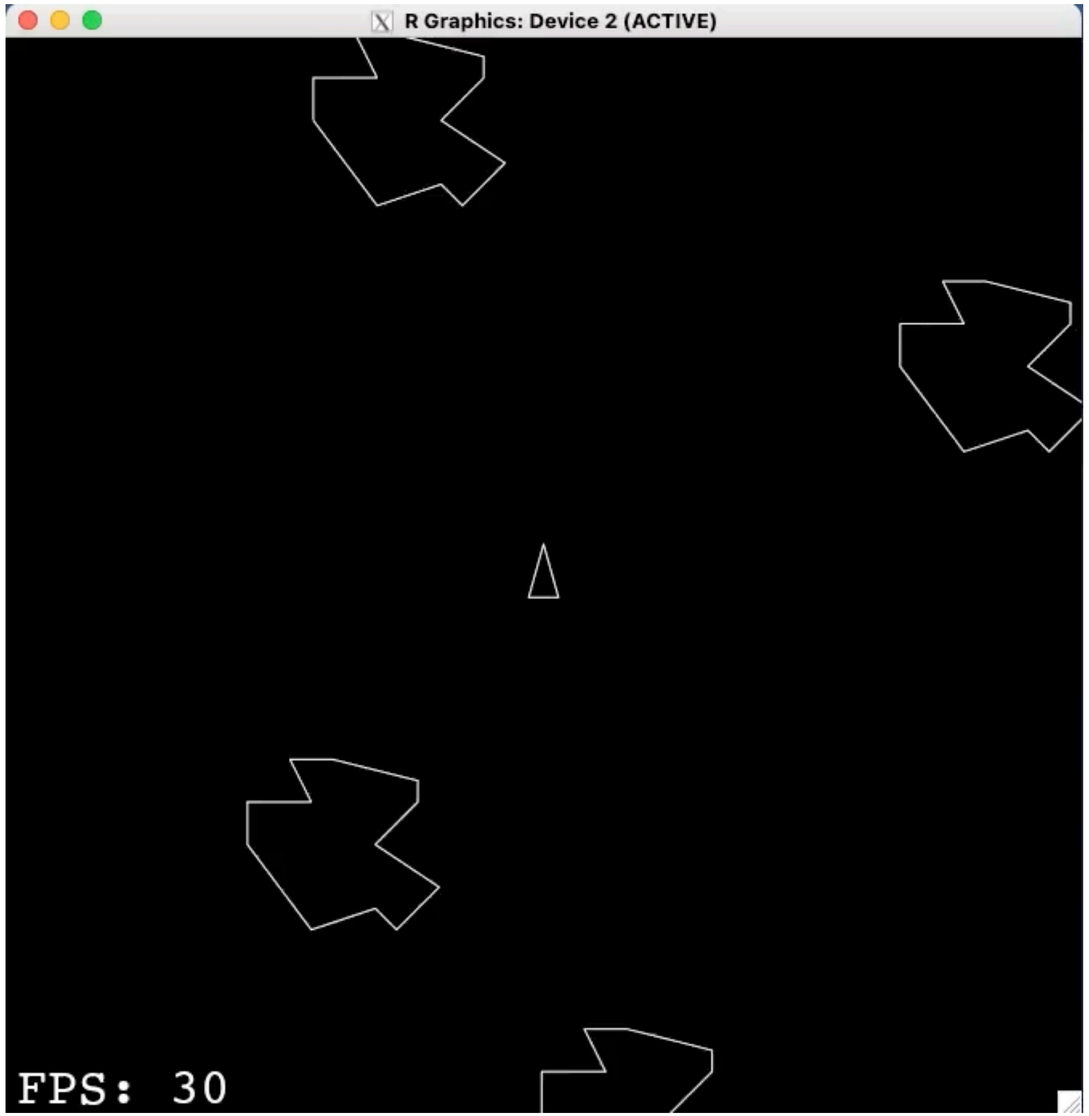
{nara}

Manipulation of native raster images



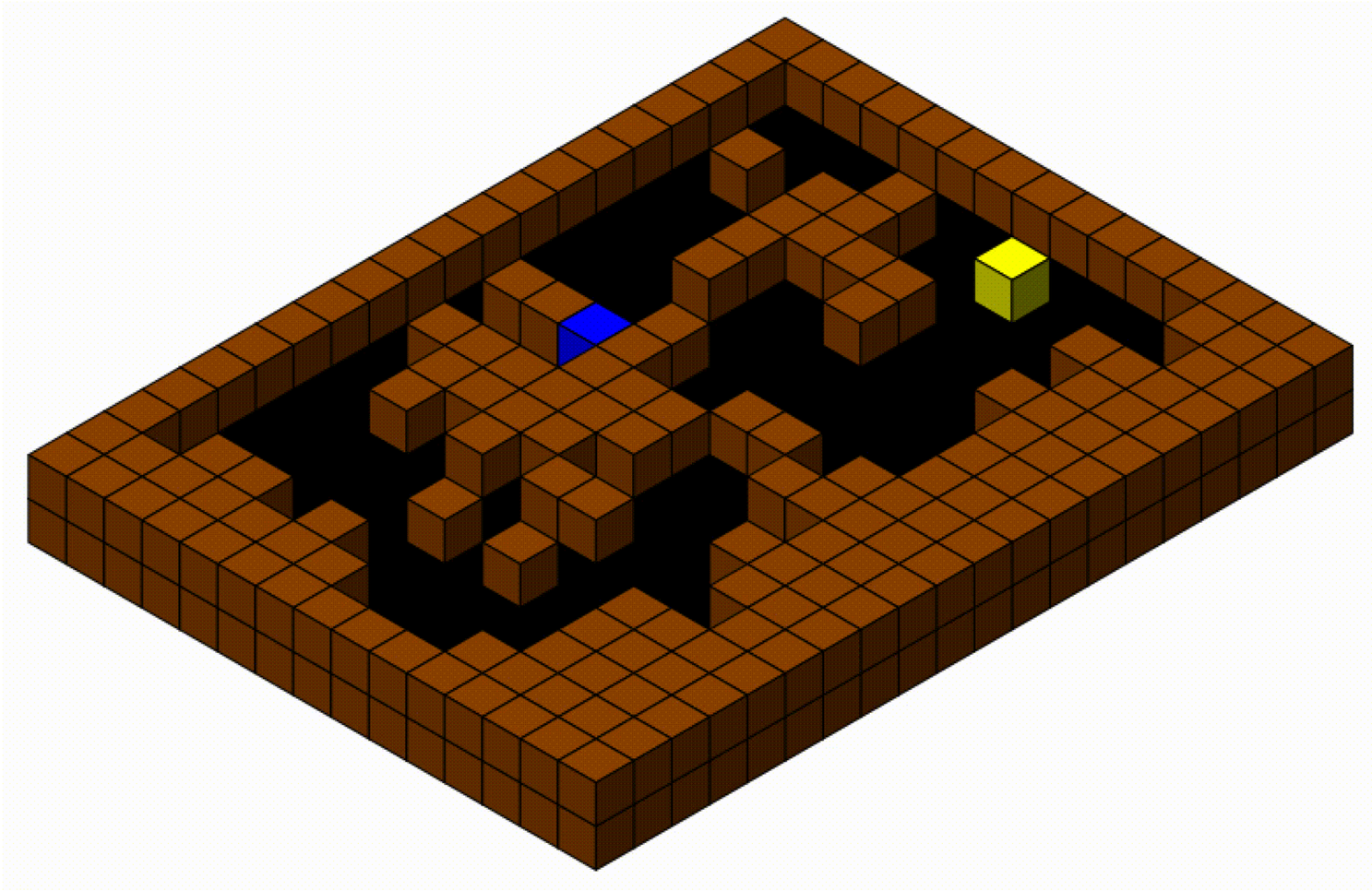
{eventloop}

Friendly wrapper around event callbacks



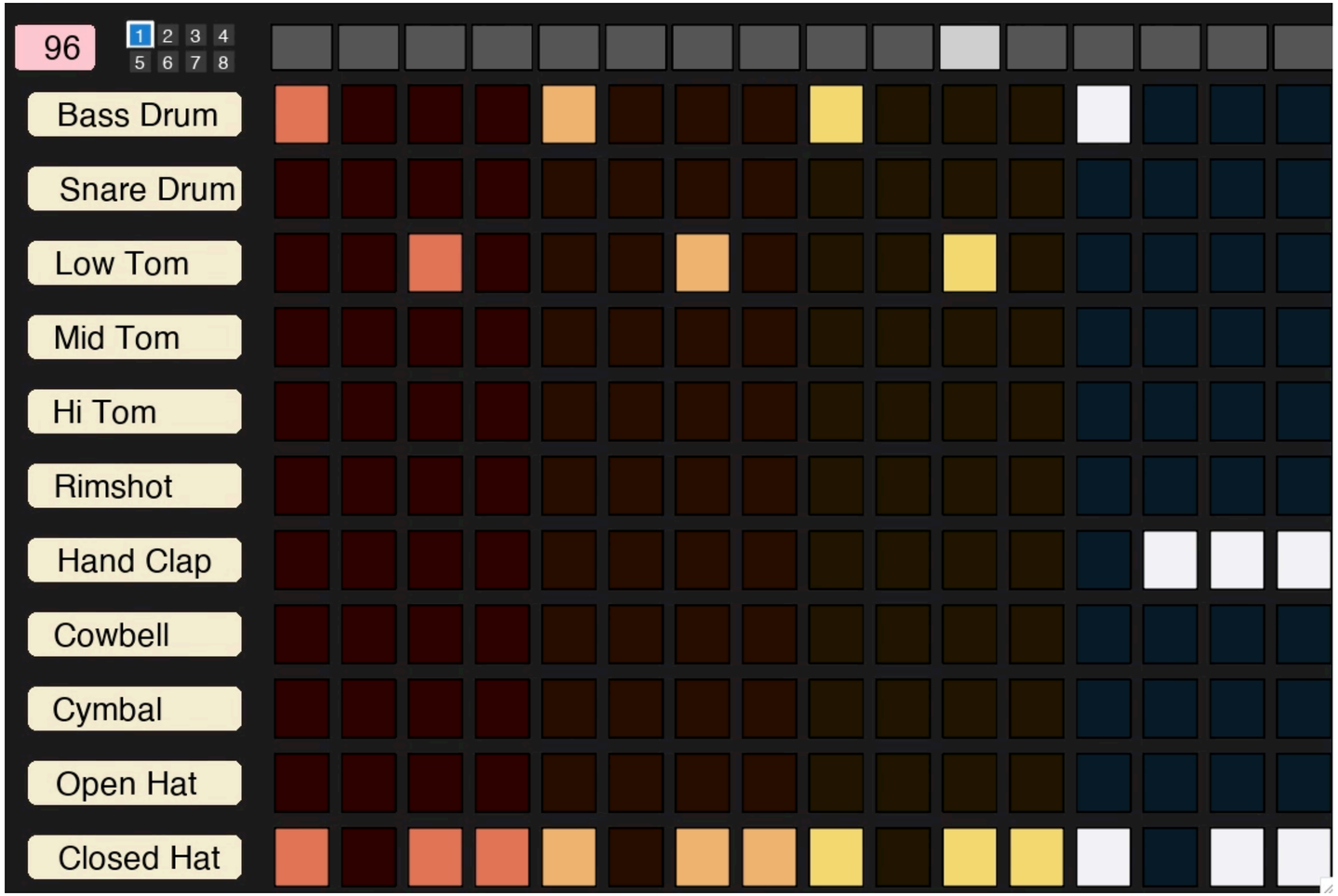
{eventloop}

@mattdray's dungeon crawler



{tr808r}

TR-808 Drum Machine



{another world}

Package coming soon!

An invitation to write a game!

Game on!

@coolbutuseless

https://github.com/coolbutuseless