Command Line Interfaces

Creating High-Fidelity CLI Tools

```
$ ./start
zsh: permission denied: ./start
$ chmod +x start
$ ./start
```

What Makes a Good Command Line Interface?

- Lenient arguments
  Long/short versions, order tolerant

- Shell Citizen
  Piping, Output stdout / stderr, Exit codes, Silent by default

- Do One Thing and Do It Well
  Unix philosophy

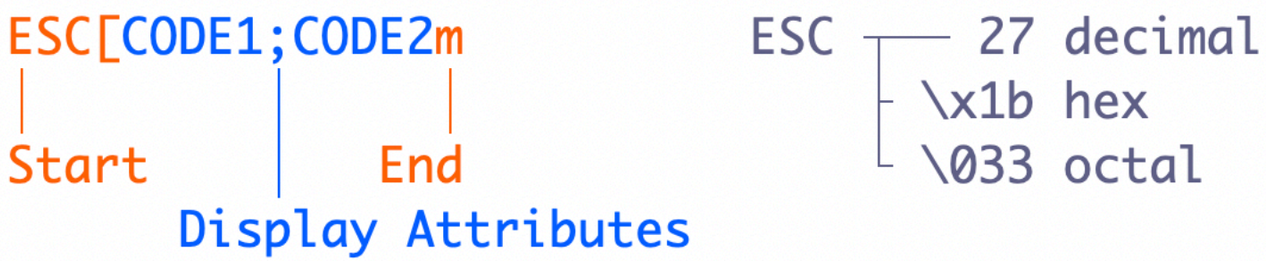
```
cat server.log | grep DEBUG | awk '{print $2}' | sed 's/[-.]/_/g'
print            search       extract             transform
```

What is this Talk About?

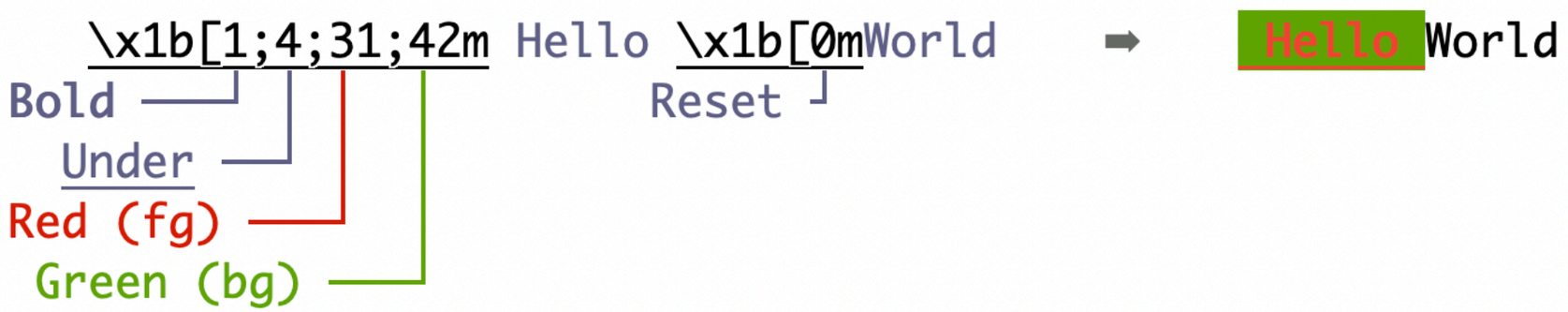
- Fun Stuff!
  - Vibrant Colors, Bold, Underlines, and… Blink
  - Drawing
  - Animation
  - <Declarative/>

How do you create these effects?

- ANSI Escape Codes (VT100 Control Codes)



- Example



More Color…

- 256 colors (More support)
  16 standard + 216 colors + 23 greyscale

```
   \x1b[38;5;128m Hello                          =>  Hello
fgcolor ┘   │   │
     8-bit ┘   └ Color
```

- 16 million colors (Limited support)
  Red, green, blue as 0-255

```
   \x1b[38;2;162;64;208m Hello                   =>  Hello
fgcolor ┘   │   │   │   └ Blue
    24-bit ┘   │   └ Green
               └ Red
```

Presentation ⌥⌘1

This is kind of tedius…

```
> npm install chalk
  github.com/chalk

  // Chain display attributes
  chalk.red.underline('Hello World');
                    // └ Hello World


chalk.bgAnsi256(128).hex('#0F0')('Use RGB')
                               // └ Use RGB
```

What else can you do?

- Gather Input from mouse/keyboard
  Arrow keys sent to STDIN as new escape sequences \x1b[D = ⬅
  (A,B,C,D)

- Move the cursor

  H3llo World\x1b[10De OR \x1b[2G
  ←10 Move to column 2

- Clear parts (or all) of the screen

  H3llo World\x1b[2K
  Clear line ┘ (doesn't move cursor, 0=after 1=before)

Will you just tell me how to make a progress indicator??

1 Draw something
2 Clear it `\x1b[2K`
3 Move cursor `\x1b[G`
4 Goto 1

17%

Components

/

\ | / –

⏯ Running

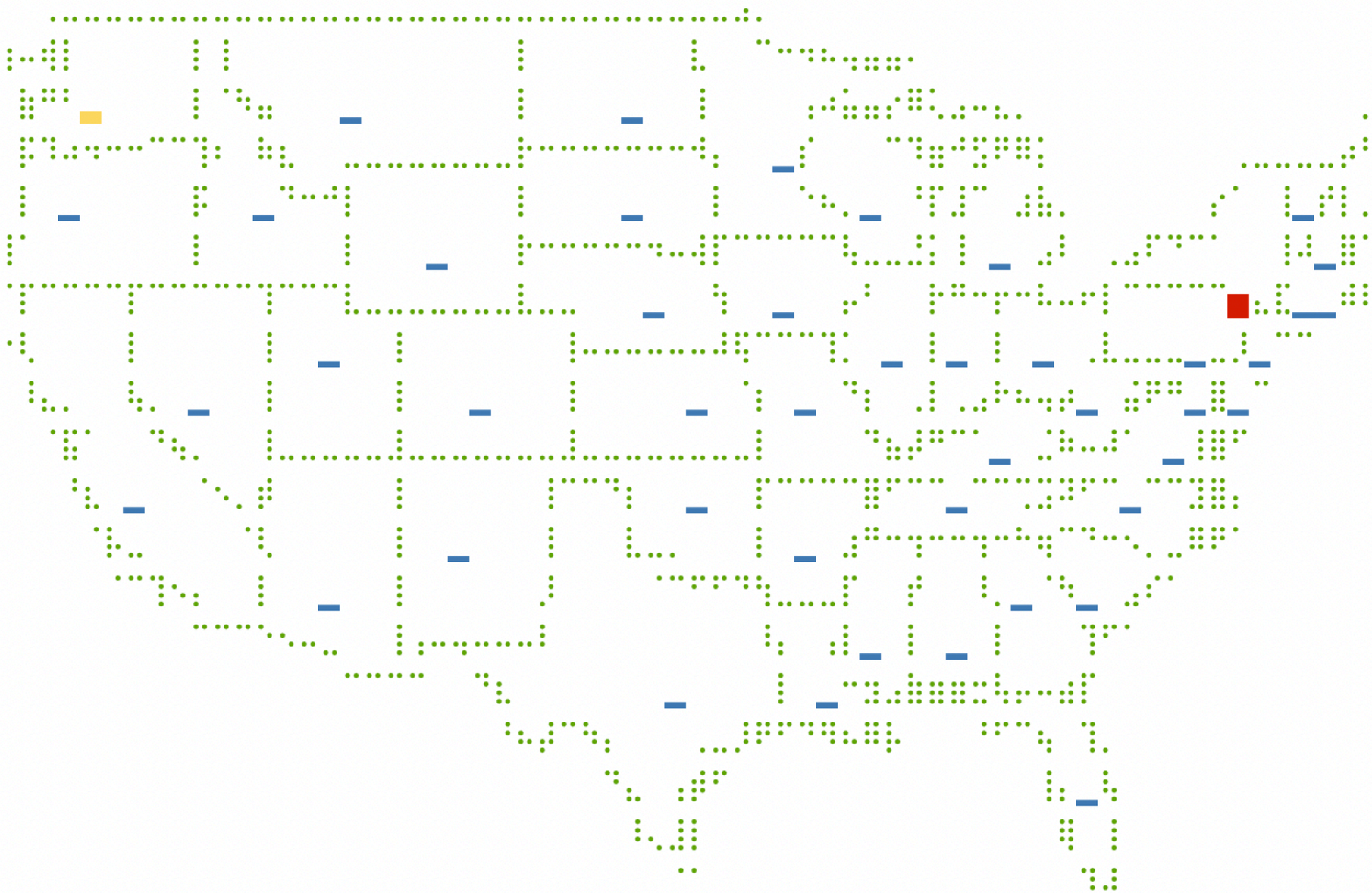
A little bit higher-level now?…

- curses Screen Updating and Cursor Movement Optimization '77 (pcurses, PDcurses, ncurses)
- Python: Blessings, Blessed github.com/jquast/blessed
- Node: Blessed github.com/chjj/blessed, neo-blessed

```
const {screen, box} from 'blessed';
screen({}).append(box({
  label: "Label",
  right:  3, bottom: 2,
  width: 12, height: 4,
  draggable: true,
  style: {
    bg: "magenta",
    transparent: true
  }
}));
```

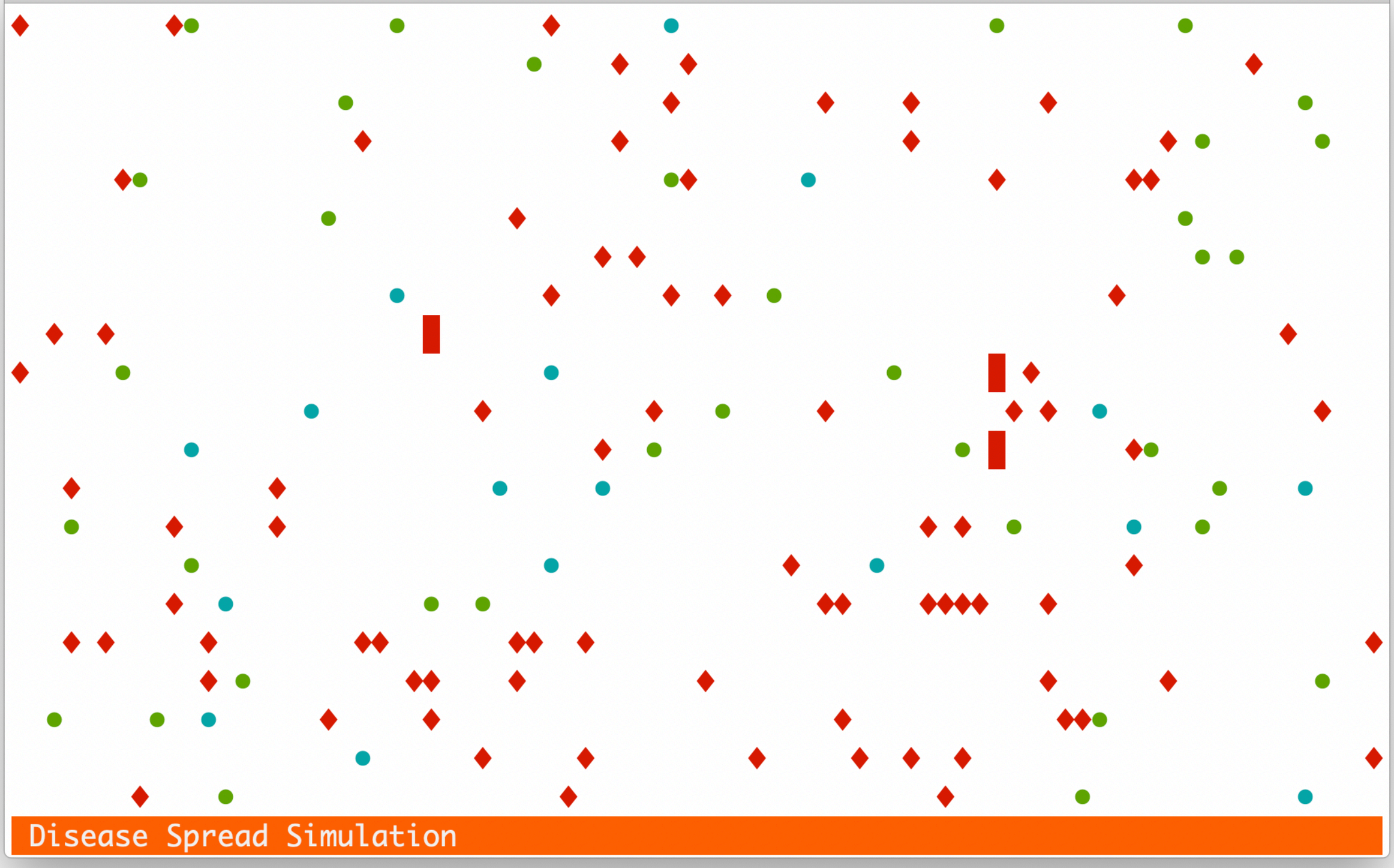
Presentation ⌥⌘1

Presentation ⌥⌘1

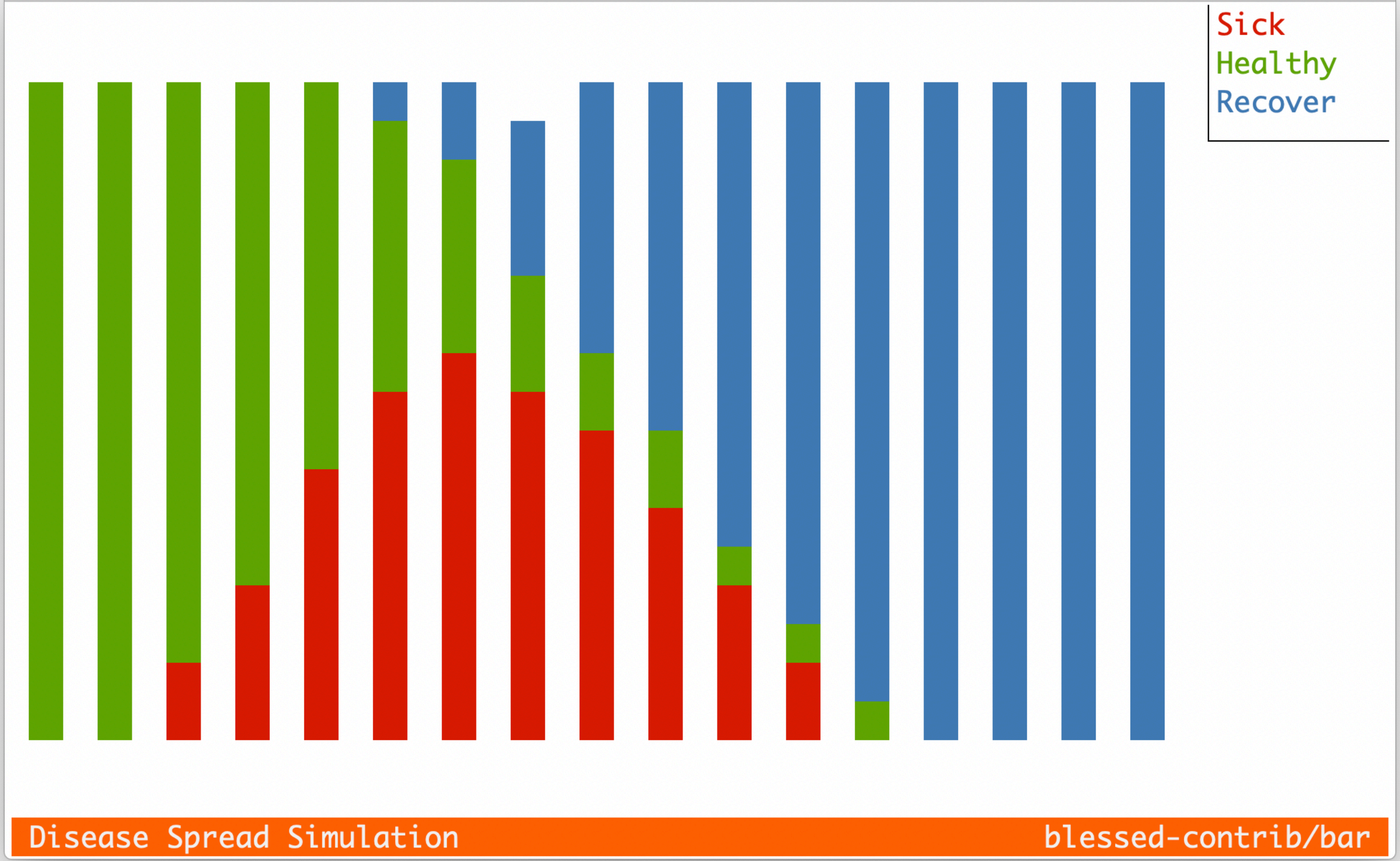


Confirmed Cases github.com/CSSEGISandData/COVID-19

Presentation ⌥⌘1

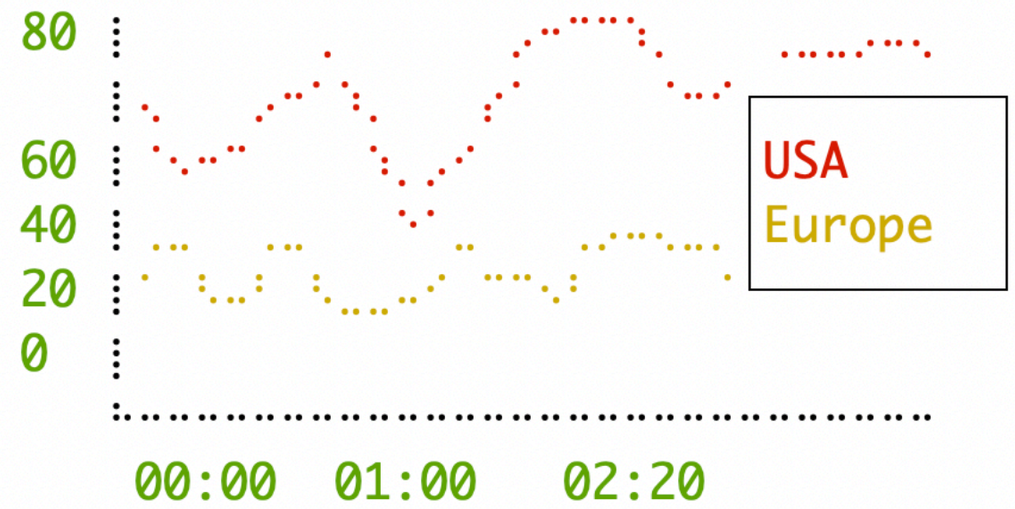


Presentation ⌥⌘1

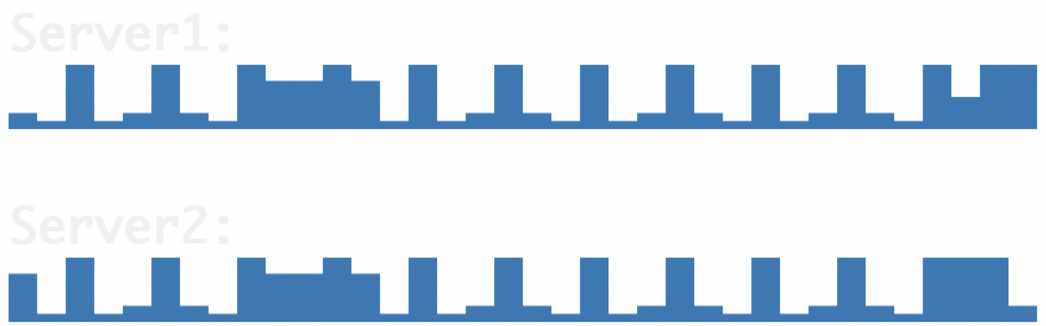


Presentation ⌥⌘1

Total Transactions



Throughput (bits/sec)



Server Log

```
terminating server US2
terminating server US2
terminating server US2
terminating server US1
terminating server EU1
```

Active Processes

Process	Cpu (%)	Memory
awk	4	60
watchdog	5	36
awk	0	88
gulp	4	12
grep	3	66
netns	3	93
awk	2	87

Imperative Programming? in 2020?

- **react-blessed** Creates a ReactFiberReconciler that maps to blessed api
  github.com/Yomguithereal/react-blessed

```
const FunkyFire = () => {
  const particles = useParticleEngine(6);

  return (
    <>
      {particles.map(particle => (
        <blessed-box key={particle.key} {...particle.box} />
      ))}
    </>
  );
}
```

Imperative Programming? in 2020?

- **react-blessed** Creates a ReactFiberReconciler that maps to blessed api
  github.com/Yomguithereal/react-blessed

```
const FunkyFire = () => {
  const particles = useParticleEngine(6);

  return (
    <>
      {particles.map(particle => (
        <blessed-box key={particle.key} {...particle.box} />
      ))}
    </>
  );
}
```