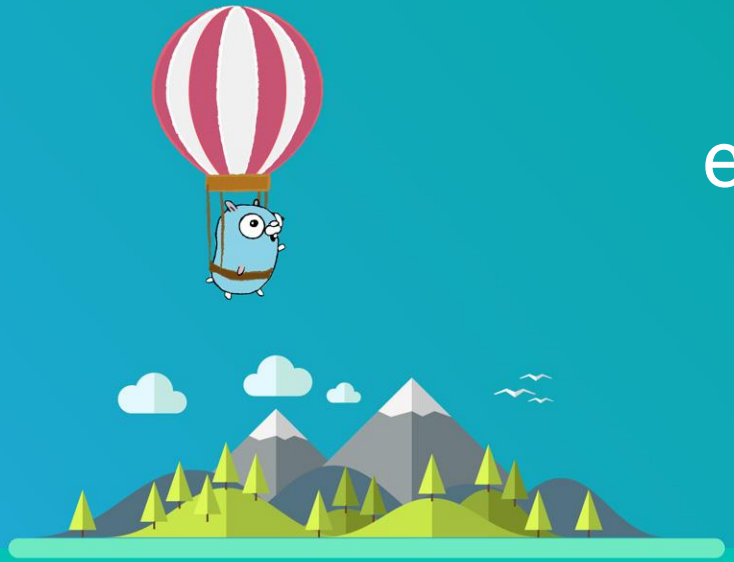
Golang Dorset

EMULATORS IN GOLANG

Dan Richards

Multiplay

@danielmrichards



“Hardware or software that enables one computer system to behave like another computer system”

What is an emulator?

Why are you doing this Dan?

- Why not?
- Learn
  - CPUs
  - System architecture
- Video games
  - Envious of friends with systems I didn’t have
  - Emulators to the rescue...eventually

First impressions

- Emulators are hard!
- My experience
  - High level languages
- Lack of exposure to memory management, CPU instructions and low level concepts
- Brush up on binary, hex and bitwise maths

Choosing a platform

- An 8-bit console?
  - NES
  - Sega Master System
- 8080 system?
- Too much for a beginner!

What is CHIP-8?

CHIP-8

- Interpreted programming language
- Developed for microcomputers in the 70's
- Runs on a virtual machine
- Designed to make development easier
- Simple
- 35 opcodes

Is this actually an emulator?

Yes (ish)

Technically an *implementation* of CHIP-8 in Golang

An *implementation* of the virtual machine not the language

Structure of an emulator

```
type chip8 struct {
    // 4K memory
    memory [4096]byte

    // CPU registers
    V [16]byte

    // 16 levels
    stack [16]uint16

    // Countdown timers
    delayTimer byte
    soundTimer byte
}
```

1st 512 bytes of memory reserved for system and fonts

Registers V0 - VE general purpose. VF used for the carry flag.

Stack stores memory addresses to power subroutines

Timers count at 60Hz

Opcodes

Two bytes long

```
// Program counter.
pc := 1
```

Fetch from memory and merge

```
// Opcode.
opc := uint16(mem[pc])<<8 | uint16(mem[pc+1])
```

Shift left by 1 byte (8 bits)

```
Hex       0x12          0x12 << 8 = 0x1200
Binary    00010010      0001001000000000
```

Bitwise OR to merge

```
0x12                    0001001000000000 |
0x34 (0x0034)                   00110100
                        -----------------
0x1234                  0001001000110100
```

Cycles

```
func (v *VM) Cycle() error {
    v.opc = uint16(v.mem[v.pc])<<8 |
uint16(v.mem[v.pc+1])

    if err := v.handle(); err != nil {
        return err
    }

    select {
    case <-v.clock.C:
        v.updateTimers()
    default:
    }
}
```

Get the opcode for this cycle

Program counter incremented by 2 each time

Handle the opcode. Map of handlers.

Decrement the timers when the clock ticks. 60 times a second.

Example Opcode Handlers

1NNN

Jump to an address

```
func (v *VM) jump() (uint16, error) {
    v.pc = v.opc & 0x0FFF
    return v.opc, nil
}
```

2NNN

Call subroutine

```
func (v *VM) callSub() (uint16, error) {
    v.sp++
    v.stack[v.sp] = v.pc
    v.pc = v.opc & 0x0FFF

    return v.opc, nil
}
```

CHIP-8 Graphics

```
func (v *VM) draw() (uint16, error) {
     x := uint16(v.v[(v.opc&0x0F00)>>8])
     y := uint16(v.v[(v.opc&0x00F0)>>4])

     for cY := uint16(0); cY < height; cY++ {
      for cX := uint16(0); cX < 8; cX++ {
           index := x + cX + ((y + cY) * 64)
           ...
           if v.disp[index] == 1 {
                v.v[0xF] = 1
           }
          ...
      }
    }
}
```

Display is a 64x32 pixel array

Opcode DXYN responsible for updating the display

Pixel at co-ordindates X,Y with a height of N updated

If pixel at the derived index was 1 update the VF register to 1. Collision detection.

Sound and Vision

- Community to the rescue!
- Pixel (https://github.com/faiface/pixel) for graphics
  - OpenGL bindings
  - 2D and 3D primitives and APIs
  - Cross platform
- Beep (https://github.com/faiface/beep) for sound
  - Cross platform
  - Multi format support
  - Embed audio with Packr

Controls

```
keys = map[byte]pixelgl.Button{
    0x1: pixelgl.Key1,
    0x2: pixelgl.Key2,
    ...
    0xF: pixelgl.KeyV
}

for i, key := range keys {
    if window.Pressed(key) {
        vm.KeyDown(i)
    }
}
```

CHIP-8 has a 16 element hex keyboard

Define a map of bytes to Pixel key bindings

On each cycle check if each key is pressed

Update the VM *keys* array at the correct index

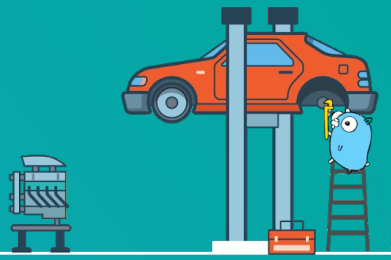
Opcodes to block or skip if keys are pressed

Demo time

Let’s play some games!

“

That’s all folks



Thank you!