



Kyiv Go Meetup, DEC 18 2018

# Writing WebGL apps in Go



---

**Ivan Danyliuk**

---

@idanyliuk

---

3dgrid125.json

3dgrid900.json

beta\_fleet\_28.json

grid10k.json

grid25.json

net100.json

net300.json

✓ smallworld200.json

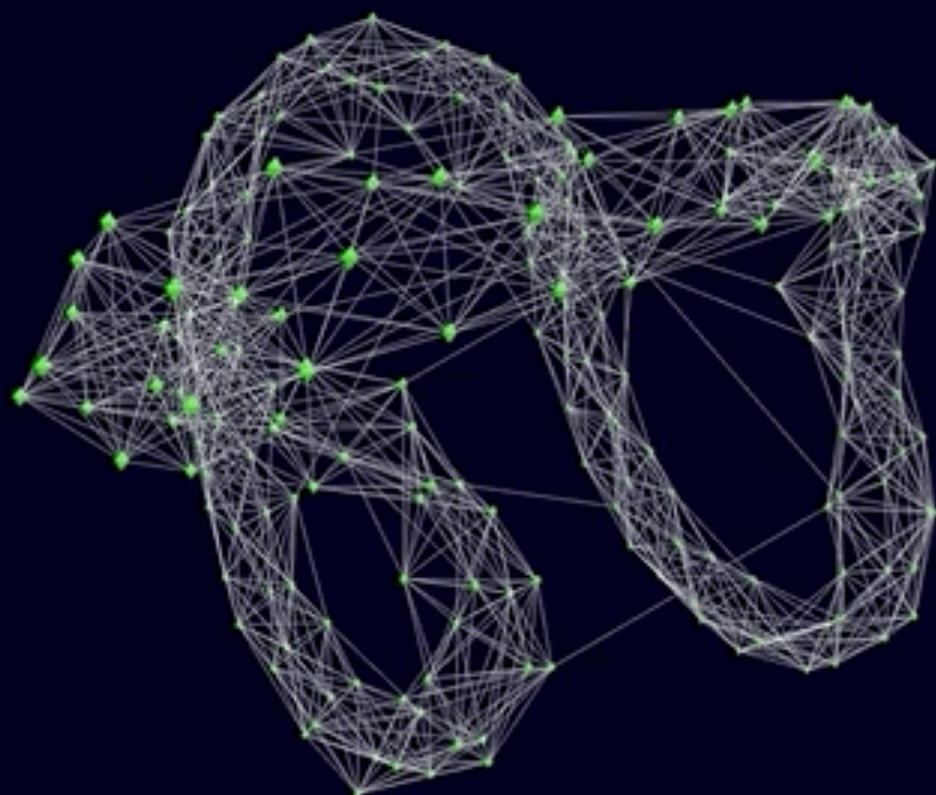
Uploaded (125 nodes)

Upload custom...

3D view

Stats view

FAQ



Graphics: +

Simulation backend:

Host:

TTL:

Start simulation

Replay

Time

Stats:

Elapsed time: 761ms

Nodes hit: 200

Links hit: 1400

but 3D visualization is worth a thousand pictures.

Within cells, DNA is organized into long structures called chromosomes. These chromosomes are duplicated before cells divide, in a process called DNA replication. Eukaryotic organisms (animals, plants, fungi, and protists) store most of their DNA inside the cell nucleus and some of their DNA in organelles, such as mitochondria or chloroplasts. [1] In contrast, prokaryotes (bacteria and archaea) store their DNA on a single chromosome in the cytoplasm. Within the chromosomes, chromatin proteins such as histones compact and organize DNA. These compact structures guide the interactions between DNA and other proteins, helping control which parts of the DNA are transcribed.



biochemical pathway that together with the other two pathways is called a nucleotide cycle. The cycle is essential for the production of nucleotides, the building blocks of DNA. The cycle is also involved in the regulation of cell growth and differentiation. The cycle is a key component of the DNA replication machinery. The cycle is a key component of the DNA replication machinery. The cycle is a key component of the DNA replication machinery.

DNA exists in many possible conformations that include A-DNA, B-DNA, and Z-DNA forms, although only B-DNA and Z-DNA have been directly observed in functional organisms.<sup>(10)</sup> The conformation that DNA assumes depends on the hydration level, DNA sequence, ionic strength, and direction of supercoiling, among other factors. The amount and direction of supercoiling, the sequence of the bases, the type and concentration of the solvent, as well as the presence of polyanions, all affect the conformation of DNA.<sup>(11)</sup>

The first published reports of A-DNA X-ray diffraction patterns—and also B-DNA used analyses based on Patterson transforms that provided only a limited amount of structural information for oriented DNA<sub>2</sub>[30][31]. An alternate analysis was then proposed by Wilkins et al in 1953, for the *in vivo* B-DNA diffraction/scattering patterns of highly hydrated fibers in terms of squares of Bessel functions.[32] Subsequently, Watson and Crick presented their molecular modeling analysis of the DNA X-ray diffraction patterns to suggest that the structure

### Native Desktop UI app:

- Compile it for each platform
- Fix issues with Windows
- Create an installer app
- Put installer app online
- User downloads the installer
- User runs the installer
- User clicks "Next" 100 times
- User launches the app

### Web UI app:

- Put the app online
- User enters URL and launches the app



## Native Desktop UI app:

- C++ / Qt
- Swift
- Java
- Python
- C
- TCL
- C#
- VB.Net
- Pascal/Delphi

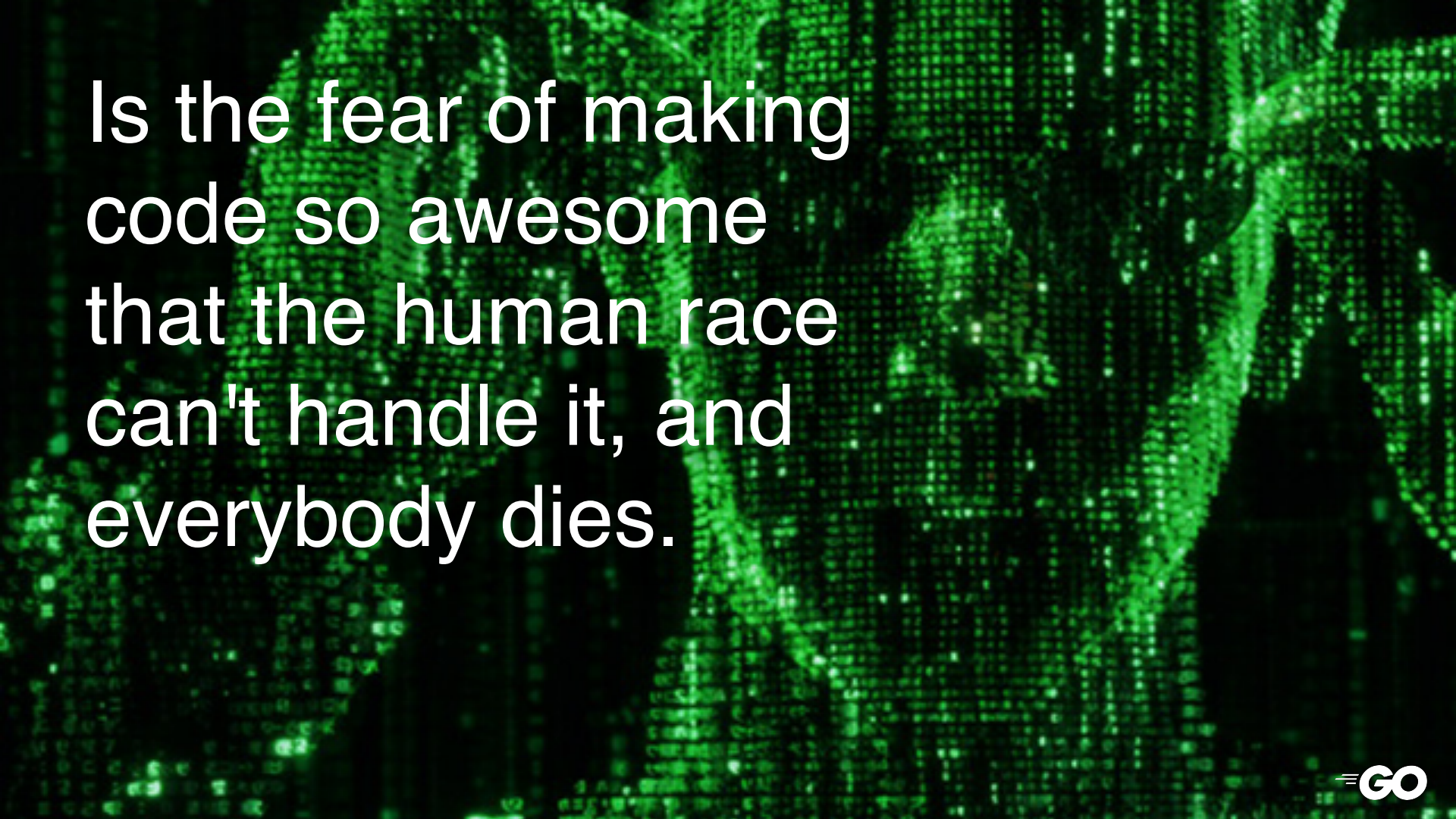
## Web UI app:

- JavaScript

# Problems that JS has



# Problems that JS **doesn't** have



Is the fear of making  
code so awesome  
that the human race  
can't handle it, and  
everybody dies.



# HTML



# WebGL



# What is WebGL?

- Rasterization engine
- Implemented in Browsers
- Knows how to talk to GPU
- Browser API for GL

## User/autogenerated JS code

WebGL  
frameworks

Emscripten/Unity

## WebGL Browser implementation

JS Engine

OpenGL

ANGLE  
(Direct3D)

GPU

# What is WebGL?



- Rasterization engine
- Implemented in Browsers
- Knows how to talk to GPU
- Browser API for GL
- JS Libraries:
  - Three.js
  - Babylon.js
  - etc

## User/autogenerated JS code

WebGL  
frameworks

Emscripten/Unity

## WebGL Browser implementation

JS Engine

OpenGL

ANGLE  
(Direct3D)

GPU

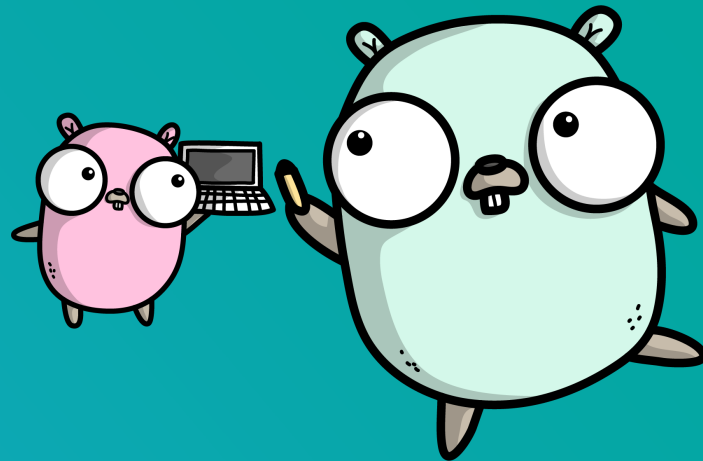
If you want to write  
interactive 3D app,  
you're almost forced  
to use JS and  
WebGL




Friends don't let friends  
write apps in JavaScript



# Can we use Go?



- Compiler from Go to JavaScript and WASM
- Started in 2013 (6 years old project!)
- Is awesome
- A lot of bindings to JS libs exist



```
$ go get -u github.com/gopherjs/gopherjs
$ gopherjs version
GopherJS 1.11-2
```

## Three.js bindings:

- <https://github.com/Lngramos/three> - three.js bindings for GopherJS

```
package three

import "github.com/gopherjs/gopherjs/js"

type DirectionalLight struct {
    *js.Object
    Position *Vector3 `js:"position"`
}

func NewDirectionalLight(color *Color, intensity float64) *DirectionalLight {
    return &DirectionalLight{
        Object: three.Get("DirectionalLight").New(color, intensity),
    }
}
```



# Three.js bindings:



## JS:

```
var scene, renderer, light, camera;
camera = new THREE.PerspectiveCamera(70, w/h, 1, 1000 );
camera.position.set( 1000, 50, 1500 );
scene = new THREE.Scene();
renderer = new THREE.WebGLRenderer();
renderer.setSize(w, h);
light = new THREE.AmbientLight( 0xffffff );
scene.add(light);
```

## Go:

```
camera := three.NewPerspectiveCamera(70, w/h, 1, 1000)
camera.Position.Set(1000, 50, 1500)
scene := three.NewScene()
renderer := three.NewWebGLRenderer()
renderer.SetSize(w, h, true)
light := three.NewAmbientLight(three.NewColor("white"))
scene.Add(light)
```



# Let's write WebGL Go "Hello, World"

## Start HTML file:



```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8>
    <title>Go WebGL app</title>
    <style>
      body { margin: 0; }
      canvas { width: 100%; height: 100% }
    </style>
  </head>
  <body>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/three.js/99/three.min.js"></script>
    <script src="go-webgl-example.js"></script>
  </body>
</html>
```



```
package main
```

```
import (  
    "github.com/divan/three"  
    "github.com/gopherjs/gopherjs/js"  
)
```

```
func main() {  
}
```



```
func main() {  
    width := js.Global.Get("innerWidth").Float()  
    height := js.Global.Get("innerHeight").Float()  
  
    renderer := three.NewWebGLRenderer()  
    renderer.SetSize(width, height, true)  
    js.Global.Get("document").Get("body").Call("appendChild",  
    renderer.Get("domElement"))  
    ...  
}
```



```
func main() {  
    ...  
    // setup camera and scene  
    camera := three.NewPerspectiveCamera(70, width/height, 1, 1000)  
    camera.Position.Set(0, 0, 500)  
  
    scene := three.NewScene()  
    ...  
}
```

```
func main() {  
    ...  
    // lights  
    light := three.NewDirectionalLight(three.NewColor("white"), 1)  
    light.Position.Set(0, 256, 256)  
    scene.Add(light)  
  
    // material  
    params := three.NewMaterialParameters()  
    params.Color = three.NewColor("blue")  
    mat := three.NewMeshLambertMaterial(params)  
    ...  
}
```

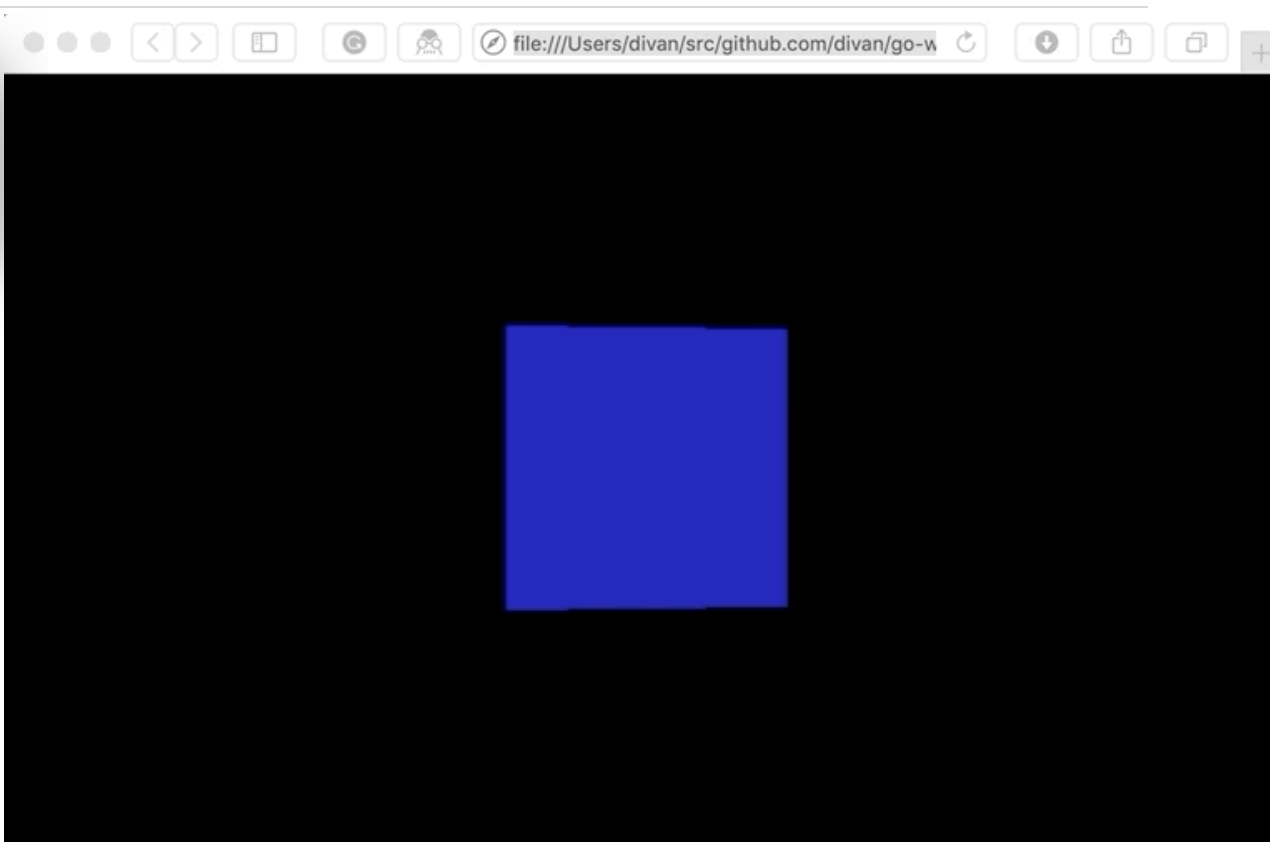
```
func main() {  
    ...  
    // cube object  
    geom := three.NewBoxGeometry(&three.BoxGeometryParameters{  
        Width: 200,  
        Height: 200,  
        Depth: 200,  
    })  
    mesh := three.NewMesh(geom, mat)  
    scene.Add(mesh)  
    ...  
}
```

```
func main() {  
    ...  
    // start animation  
    var animate func()  
    animate = func() {  
        js.Global.Call("requestAnimationFrame", animate)  
        mesh.Rotation.Set("y", mesh.Rotation.Get("y").Float()+0.01)  
        renderer.Render(scene, camera)  
    }  
    animate()  
}
```

# Hello, world



```
$ gopherjs build
$
```







We  
need more  
structure



# VECTY

Vecty - a frontend toolkit for GopherJS

<https://github.com/gopherjs/vecty>

- Write frontend app in Go
- Share frontend and backend code
- Create reusable components as Go packages

```
type MyComponent struct {  
    vecty.Core  
    // additional component fields (state or properties)  
}  
  
func (c *MyComponent) Render() vecty.ComponentOrHTML {  
    // construct DOM/HTML here  
}
```

# Vecty Hello, world:



```
package main

import (
    "github.com/gopherjs/vecty"
    "github.com/gopherjs/vecty/elem"
)

// Page is a top-level app component.
type Page struct {
    vecty.Core
    article string
}

// Render implements vecty.Component for Page.
func (p *Page) Render() vecty.ComponentOrHTML {
    return elem.Body(
        elem.Div(
            elem.Heading1(
                vecty.Text(p.article),
            ),
        ),
    )
}
```

```
package main

import "github.com/gopherjs/vecty"

func main() {
    page := &Page{article: "WebGL with Go",}
    vecty.SetTitle("Hello world")
    vecty.AddStylesheet(/* ... add your css... */)
    vecty.RenderBody(page)
}
```

```
<!DOCTYPE html>
<html>
  <head> <meta charset=utf-8>
    <title>Vecty Hello World</title>
  </head>
  <body>
    <script src="vecty-demo.js"></script>
  </body>
</html>
```

Vecty Hello, world:




```
$ gopherjs build  
$
```



Search or enter website name



# Creating WebGL apps with Go



Let's compare with  
modern React "hello  
world"



# Real project example

- P2P messaging protocols - Whisper
- No central point of observation
- No data
- No intuition about its behaviour



Centralised (A)



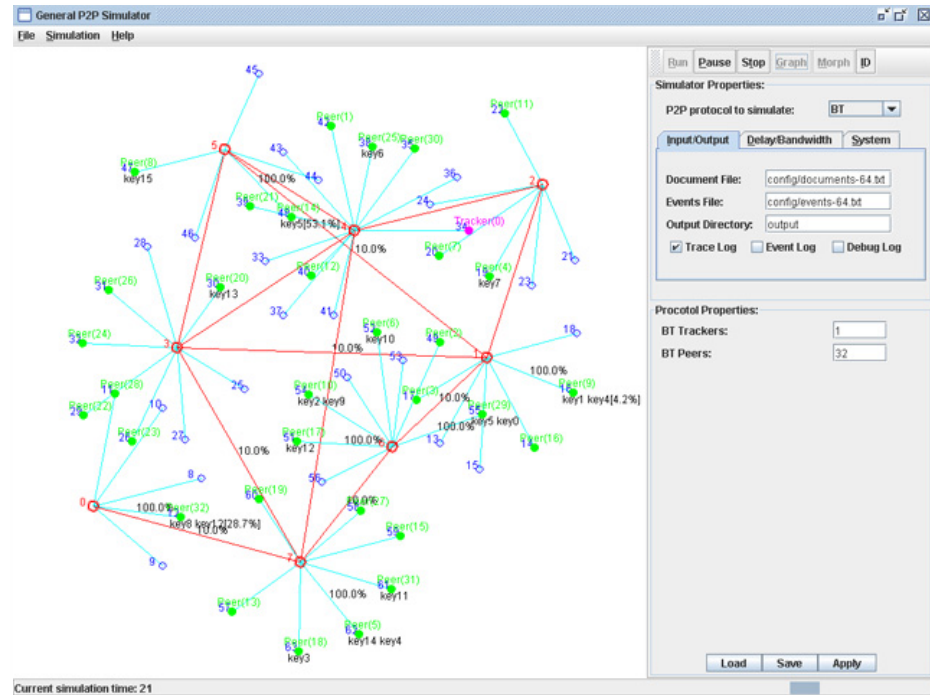
Decentralised (B)



Distributed (C)

# The problem

- You can't change the code and collect new data
- But we can run simulations in controlled network
- Existing p2p simulators require to rewrite peer's code for it



# Whisper Simulator

This simulator shows message propagation in the Whisper network.

Network graph:

beta\_fleet\_28.json ▾

Layout forces:

-

Gravity:  ?

Spring:  ?

Length:  ?

Drag:  ?

Steps:  100 ?

Apply

Graphics:

-

FPS

☒ 60 ☐ 30 ☐ 20 ☐ 15

☒ Render throttler

Blink:  236 ?

Simulation backend:

Host:  ?

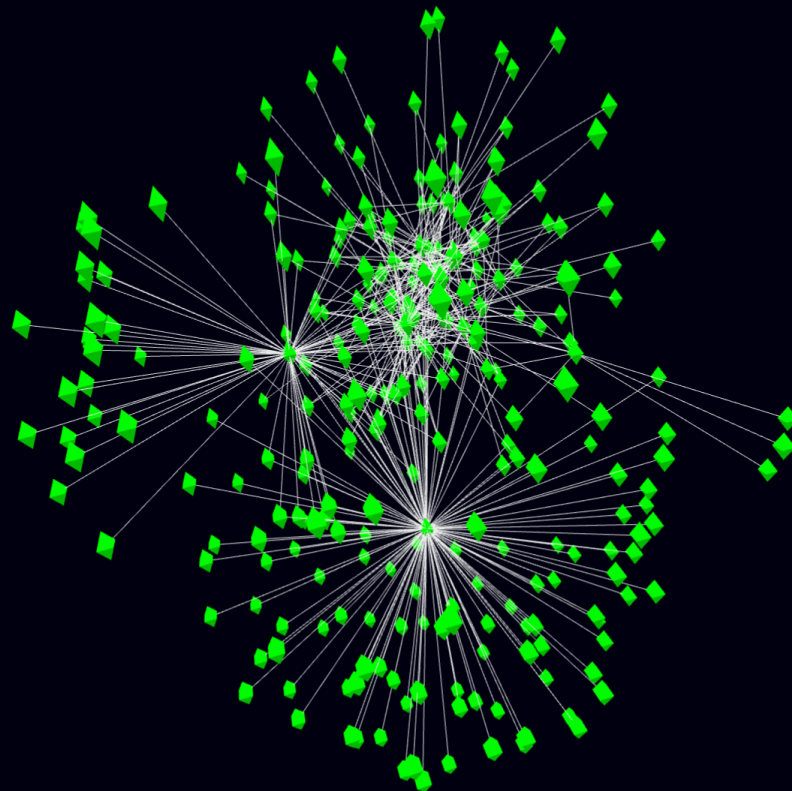
TTL:  ?

Start simulation

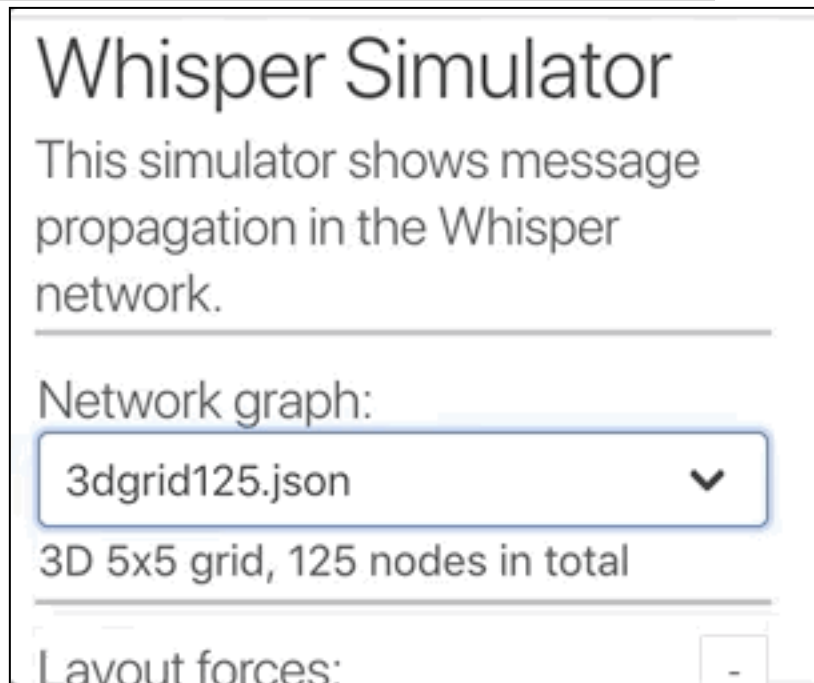
3D view

Stats view

FAQ



- UI needs number of widgets and controls
- Each one is just a vector component





```
// Render implements the vecty.Component interface for NetworkSelector.  
func (n *NetworkSelector) Render() vecty.ComponentOrHTML {  
    return Widget(  
        Header("Network graph:"),  
        elem.Div(  
            vecty.Markup(  
                vecty.Class("select", "is-fullwidth"),  
                event.Change(n.onChange),  
            ),  
            elem.Select(  
                vecty.Markup(  
                    event.Change(n.onChange),  
                ),  
                n.networkOptions(),  
                elem.Option(  
                    vecty.Markup(  
                        vecty.Property("value", "upload"),  
                        vecty.Property("selected", n.isCustom),  
                    ),  
                    vecty.Text("Upload custom..."),  
                ),  
            ),  
        ),  
        n.descriptionBlock(),  
        vecty.If(n.isCustom, n.upload),  
    )  
}
```

## Whisper Simulator

This simulator shows message propagation in the Whisper network.

Network graph:

3dgrid125.json



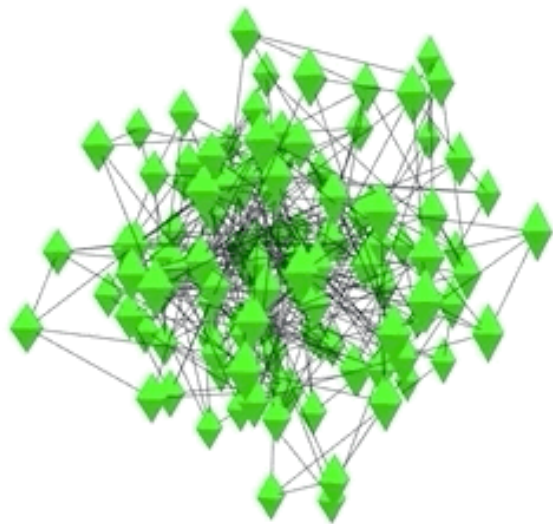
3D 5x5 grid, 125 nodes in total

Layout forces:

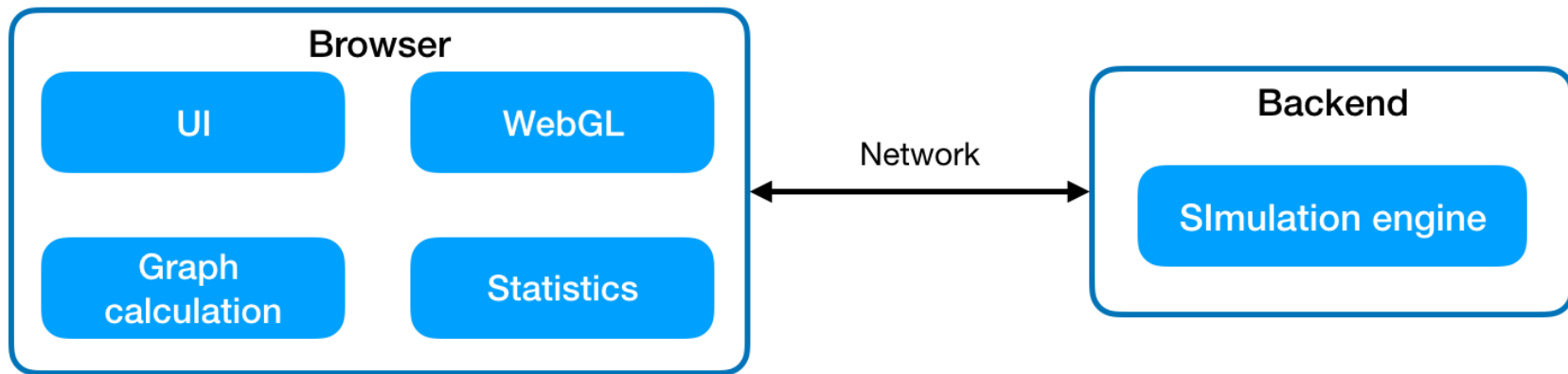




- Force-directed graph
  - place nodes pseudo-randomly
  - apply repelling force between all nodes
  - apply spring force between linked nodes
  - repeat simulation till system reaches stable energy state
- Used existing code in Go

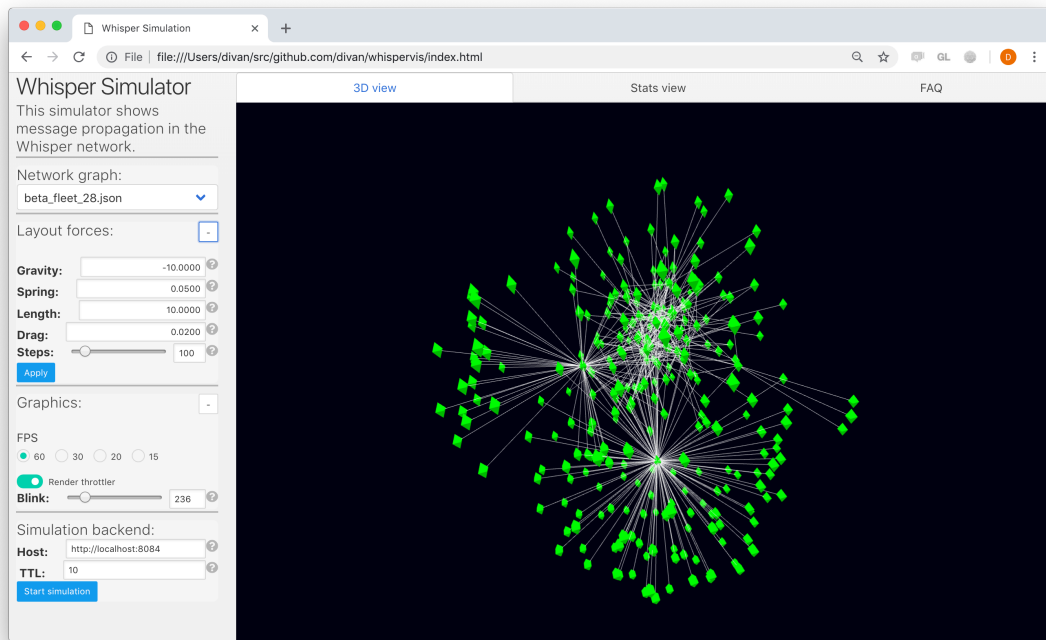


- In-memory simulation is too heavy to run in the browser
- So it's been offloaded to the "backend"
- Frontend talk to it via network and visualizes the result

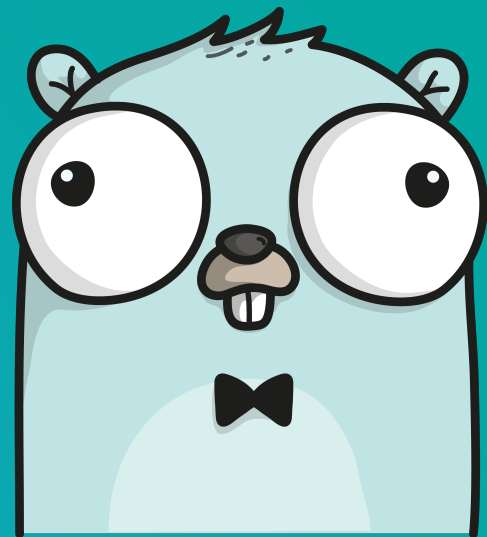


<https://www.youtube.com/watch?v=m5BwbkCxeLo>

<https://divan.github.io/whispervis/>



# Conclusions



## Conclusions:

---

- Still experimental land
- But simple frontend are 100% real with Go
- Even for WebGL ❤️
- Much simpler and easier to work with

- Of course, everybody wait for WASM
- Both GopherJS and Vecty have experimental WASM port
- [Future of GopherJS and Go in the browser](#)
- This talk [as an article](#)





# Thank you

@idanyliuk