Killer Robots are a Concurrency Problem

Stephen McQuay @smcquay
Fraser Graham @twisted_weasel

I am learning Go, it's awesome.

Cool

I always wanted to write a game.

Cool

We should write a game, in Go…

LOL

“REAL”
GAMES
C
C++

“simple”
GAMES
Javascript
Objective-C
Flash😩
C#

NOT
GAMES

Java
Python
Ruby
Go
etc.

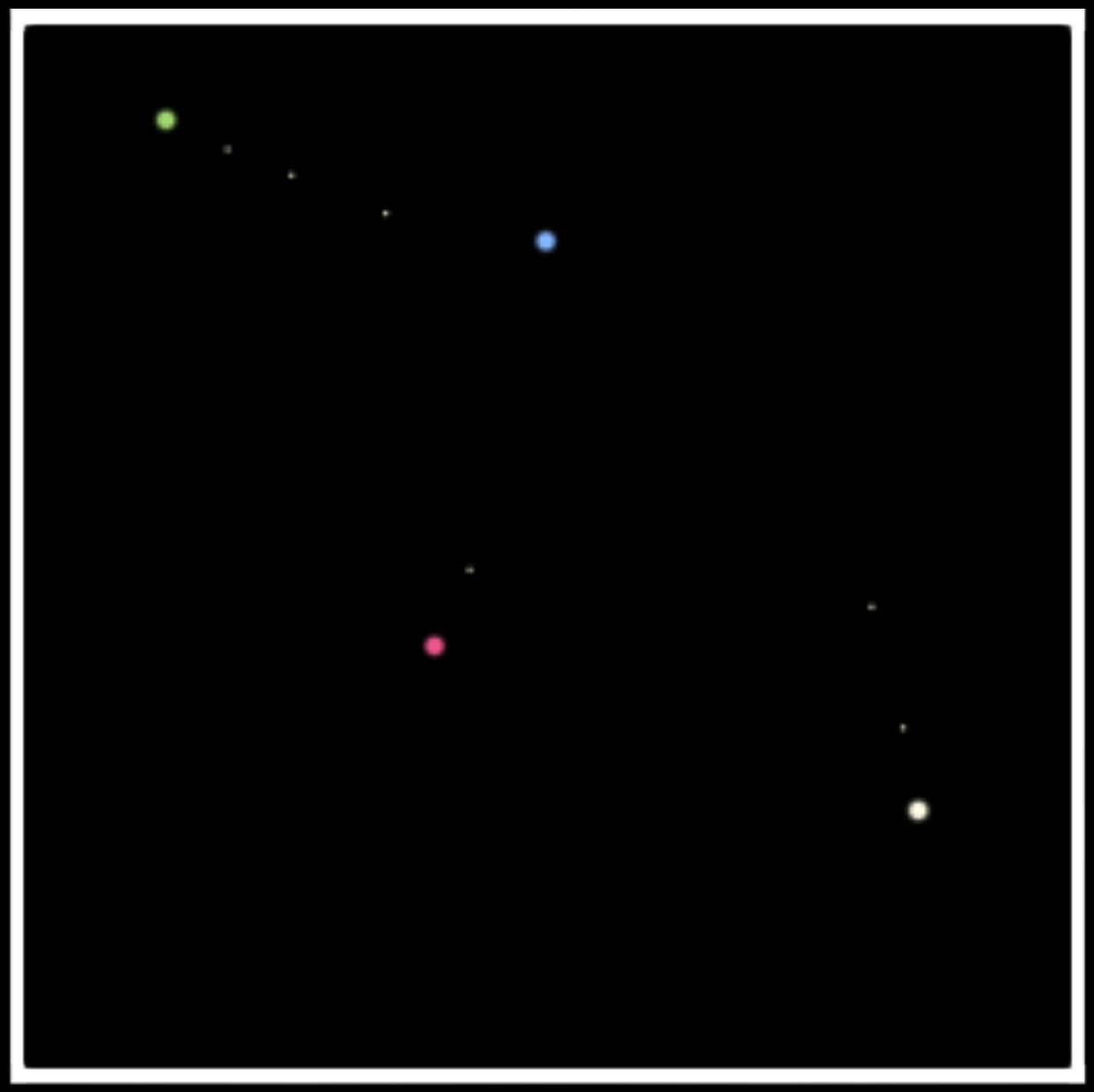
“Go is like Java, right? Best suited for server stuff? Nobody ever made a successful game in Java!”

CONCURRENCY

Let’s Make a Game!

\o/

Robots Are Cool



```
START:
MV 100, 200
FIR 300, 312
SCN 3000
JMPIF s0,
FOUND
GOTO START:
FOUND:
FIR s0
```

Screenshots are Pre-Rendered

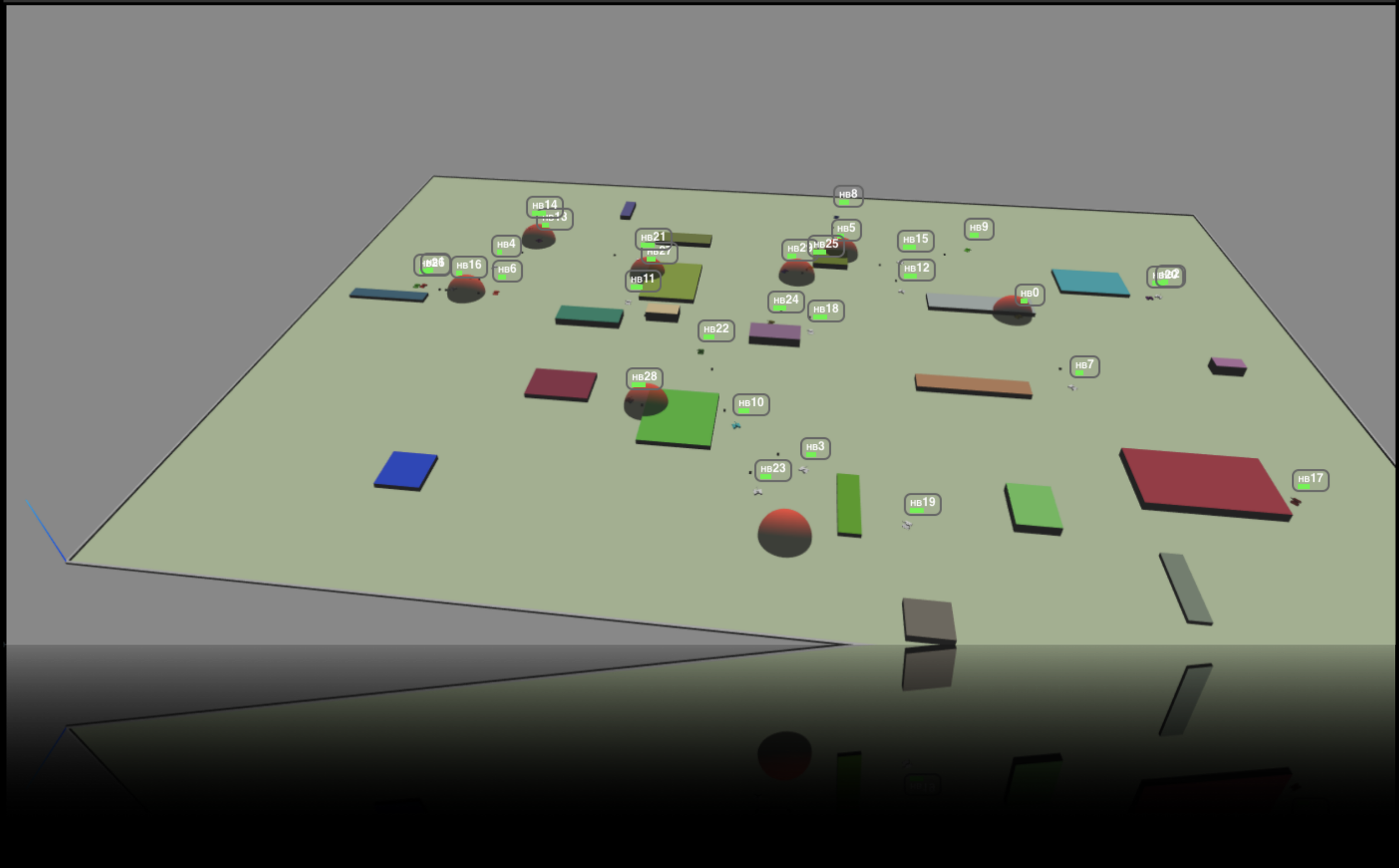
Websockets

WebGL

WEB

Game Server

05679bd4 [087] 096ac1b5 [073] 0de1b2db [074] 0f1f1a64 [060] 0fdcab75 [081] 1701b1fc [110] 2c5fae2b [090] 3b374e9f [098] 46397cc2 [110] 4b3eea92 [100] 54900cc5 [100]
5bcf92ae [070] 5cbdc877 [100] 78155f0b [087] 84b5d3f4 [090] 88ae957a [083] 8b081107 [100] 8c2b97ee [090] a0f52029 [110] abcb08a6 [110] b3582ef2 [110] c4e54d22 [110]
c76dd3cb [060] d0c80a59 [100] d2ce2b09 [087] d3d2a983 [100] e024d072 [100] e91685ec [050] ea5d1d82 [080] fa84000f [110]



Code | Arena | Raw Bot Data | Bot Data | About

Play

Watch

test

Save Bot

New Bot

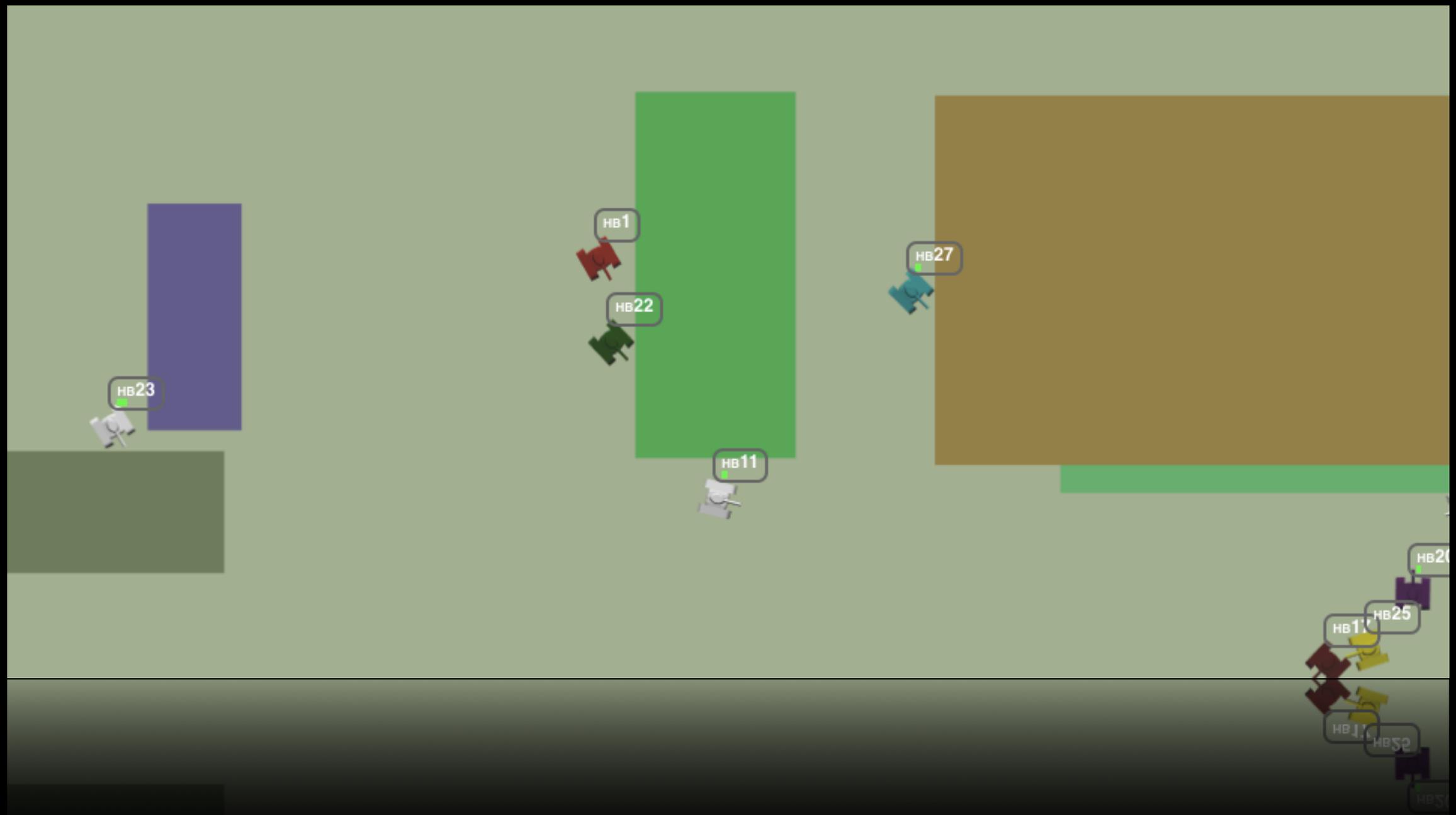
Rename Bot

```
var update = function(data, map, dbg){
    var instructions = {};
    var x,y;

    // If we have a destination and we have sent a probe for that same destination
    // and we have a result and it differs from the destination it means the
    // probe hit something, so we should change course
    if (dest &&
            data.probe &&
            data.probe_result &&
            data.probe.x == dest.x &&
            data.probe_result.x != dest.x) {
        data.collision = true;
    }

    if (dest && !data.collision){
        x = dest.x;
        y = dest.y;
        if (Math.abs(data.position.x - dest.x) < 10 &&
            Math.abs(data.position.y - dest.y) < 10){
            x = Math.floor(Math.random() * map.width);
            y = Math.floor(Math.random() * map.height);
            instructions.move_to = {"x": x, "y": y};
            dest = instructions.move_to;
        }
        else{
            instructions.move_to = {"x": x, "y": y};
        }
    }
    else{
        x = Math.floor(Math.random() * map.width);
        y = Math.floor(Math.random() * map.height);
        instructions.move_to = {"x": x, "y": y};
        dest = instructions.move_to;
    }

    }
```

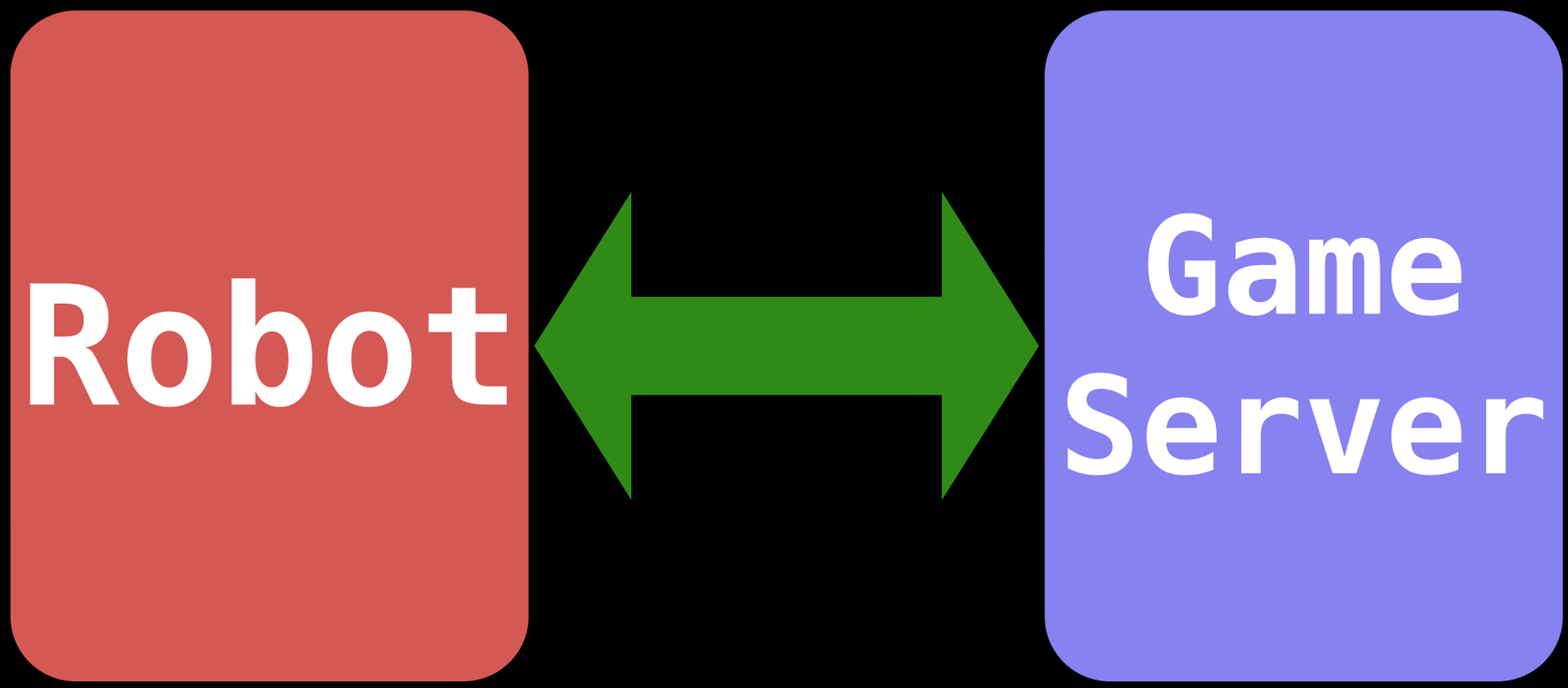

HB1
HB22
HB27
HB23
HB11
HB25

Game Architecture

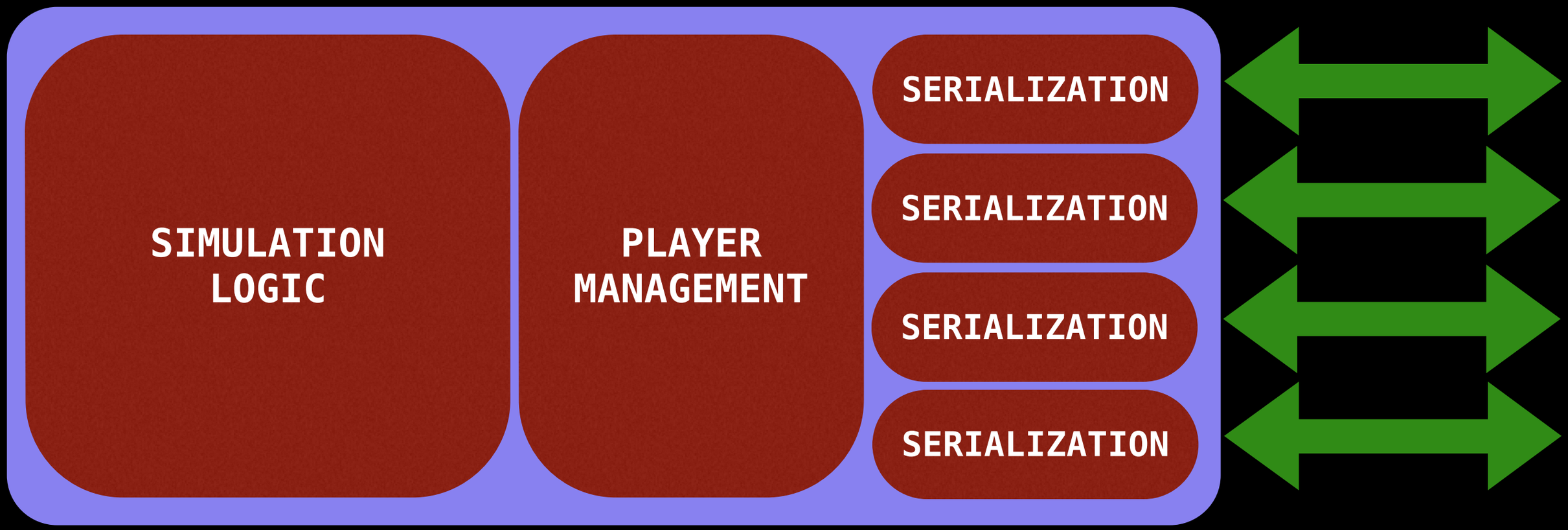
Bandwidth Management

Game Architecture

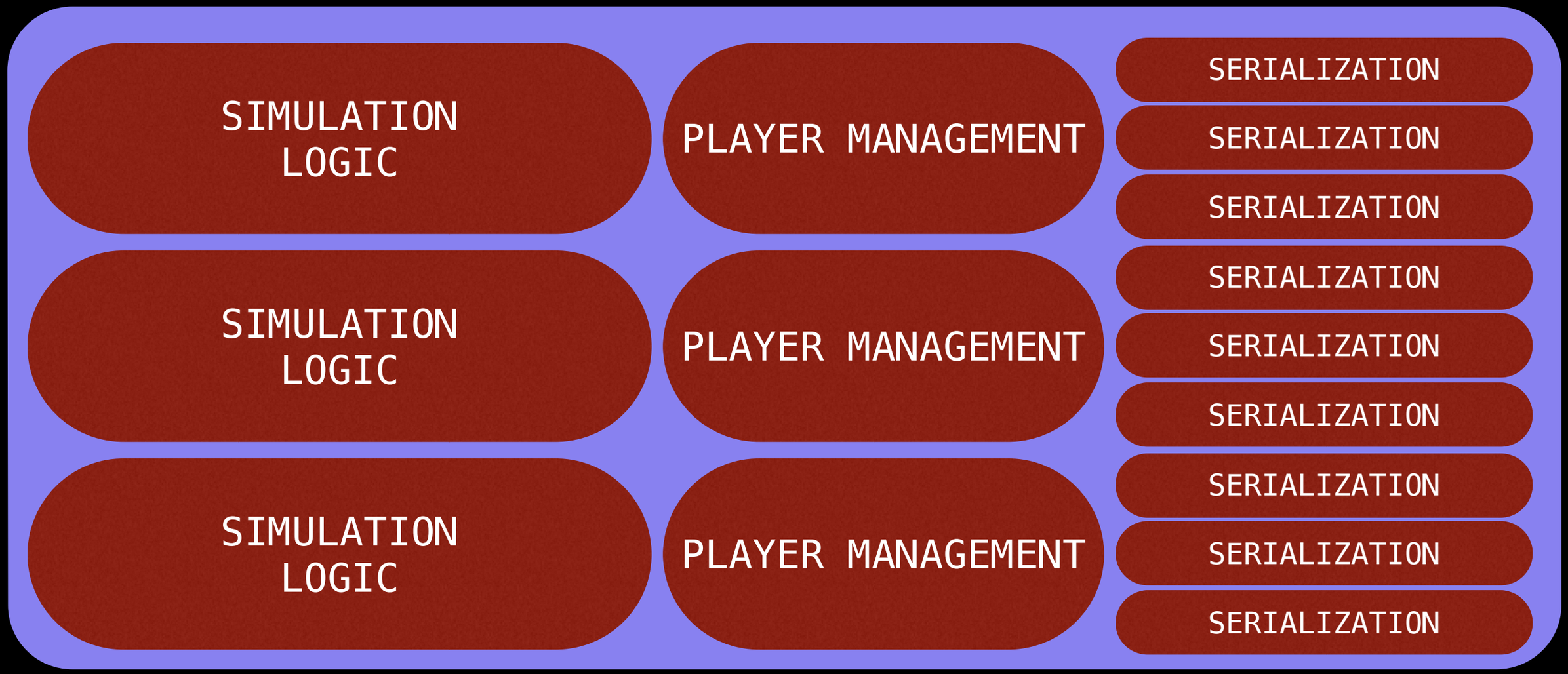
Robot Protocol



Game Server


SIMULATION LOGIC
PLAYER MANAGEMENT
SERIALIZATION
SERIALIZATION
SERIALIZATION
SERIALIZATION

Game(s) Server



SIMULATION LOGIC

PLAYER MANAGEMENT

SERIALIZATION

SERIALIZATION

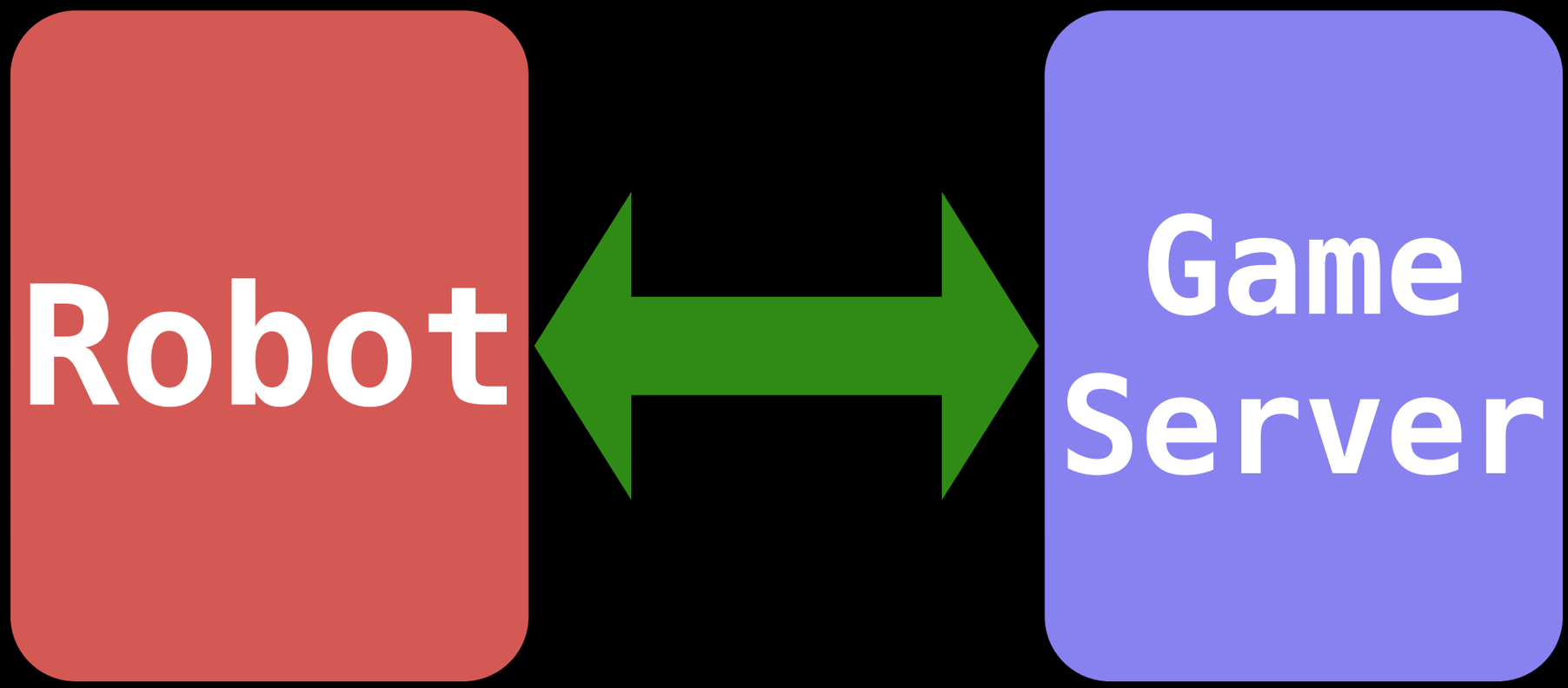
SERIALIZATION

60Hz

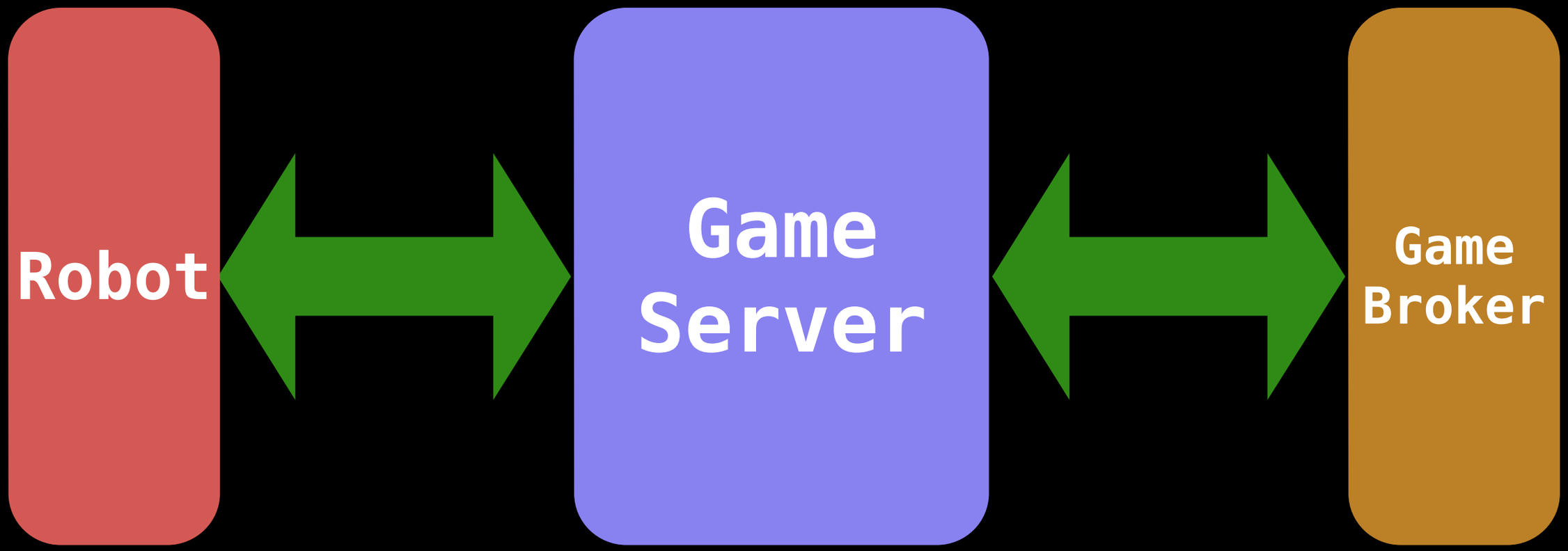
Not 60Hz

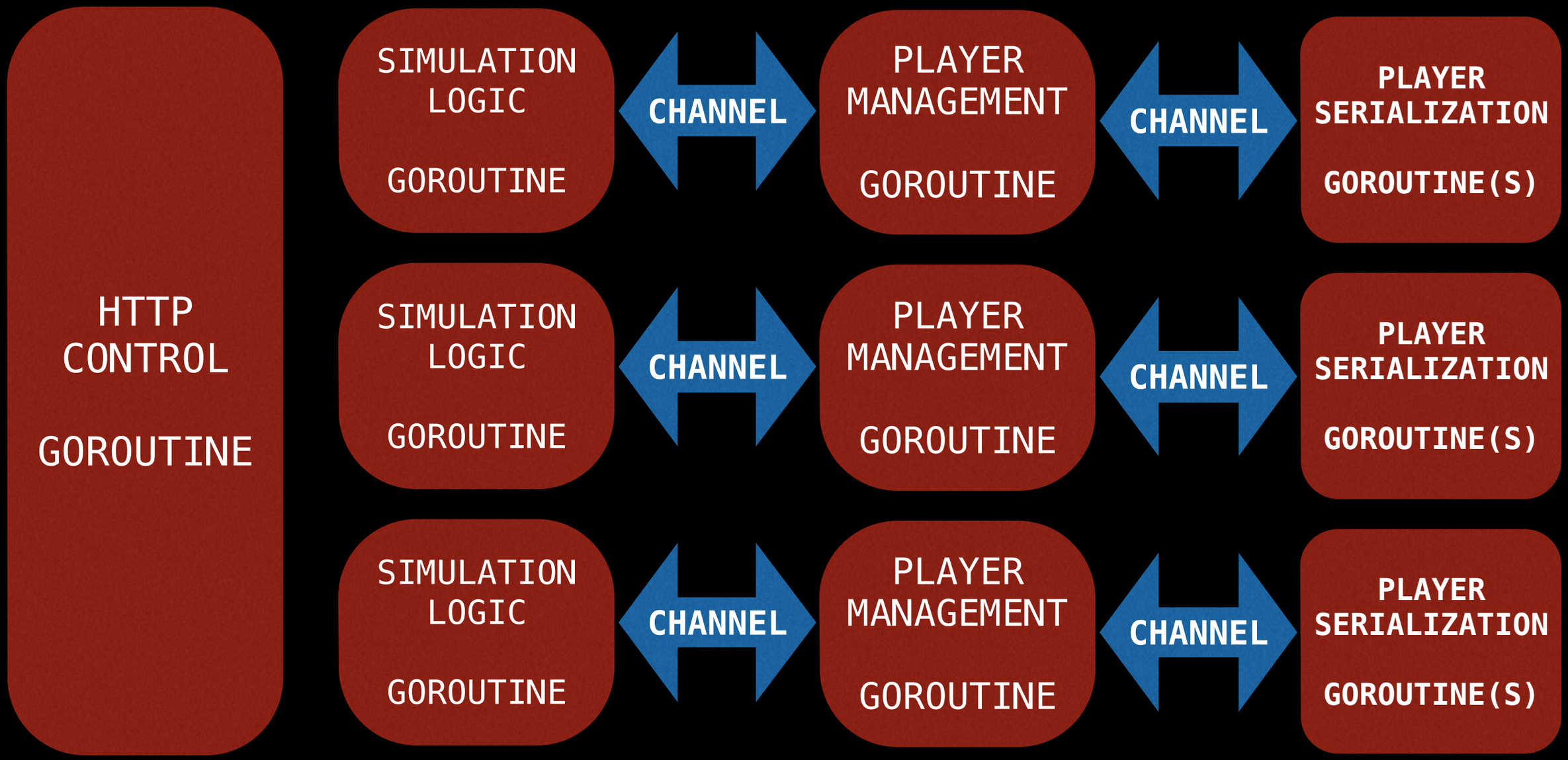

SIMULATION LOGIC
GOROUTINE
CHANNEL
PLAYER MANAGEMENT
GOROUTINE
CHANNEL
PLAYER SERIALIZATION
GOROUTINE(S)

Control, Control, You must learn Control



Control, Control, You must learn Control



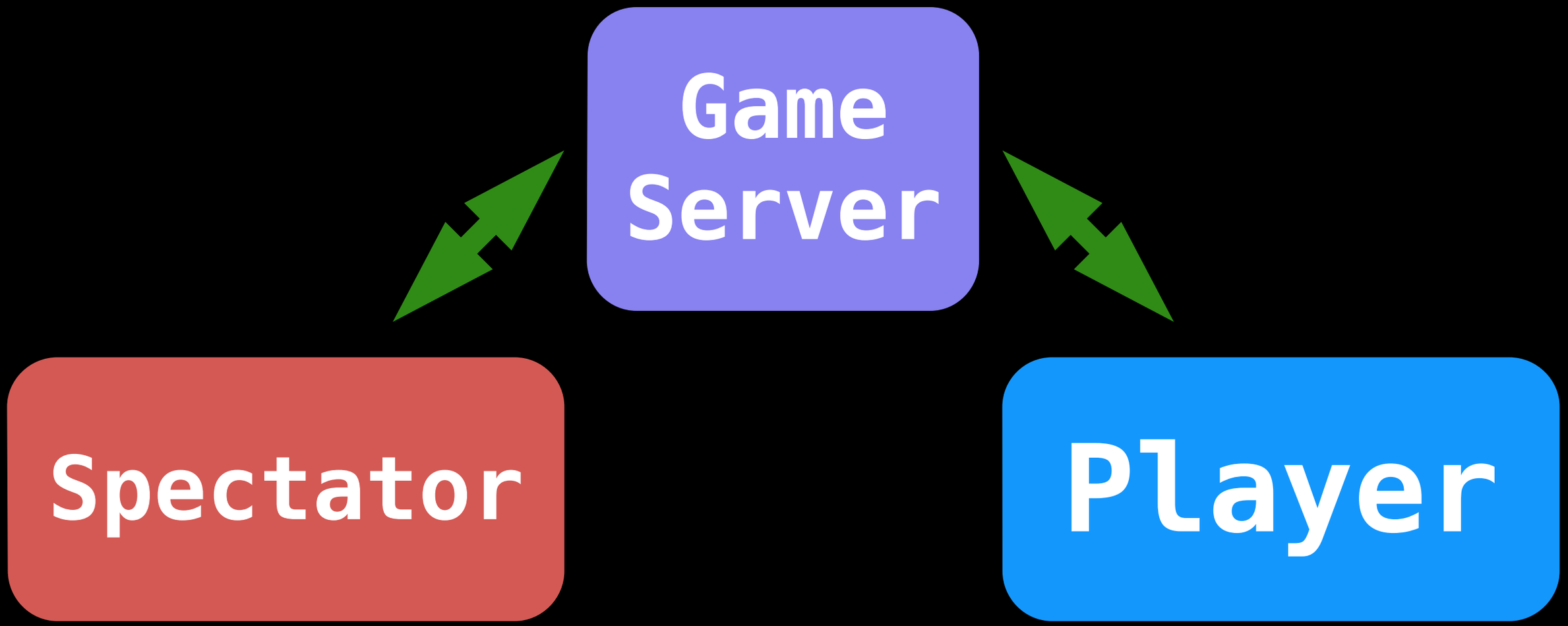

HTTP
CONTROL
GOROUTINE
SIMULATION
LOGIC
GOROUTINE
CHANNEL
PLAYER
MANAGEMENT
GOROUTINE
CHANNEL
PLAYER
SERIALIZATION
GOROUTINE(S)
SIMULATION
LOGIC
GOROUTINE
CHANNEL
PLAYER
MANAGEMENT
GOROUTINE
CHANNEL
PLAYER
SERIALIZATION
GOROUTINE(S)
SIMULATION
LOGIC
GOROUTINE
CHANNEL
PLAYER
MANAGEMENT
GOROUTINE
CHANNEL
PLAYER
SERIALIZATION
GOROUTINE(S)

Bandwidth Management

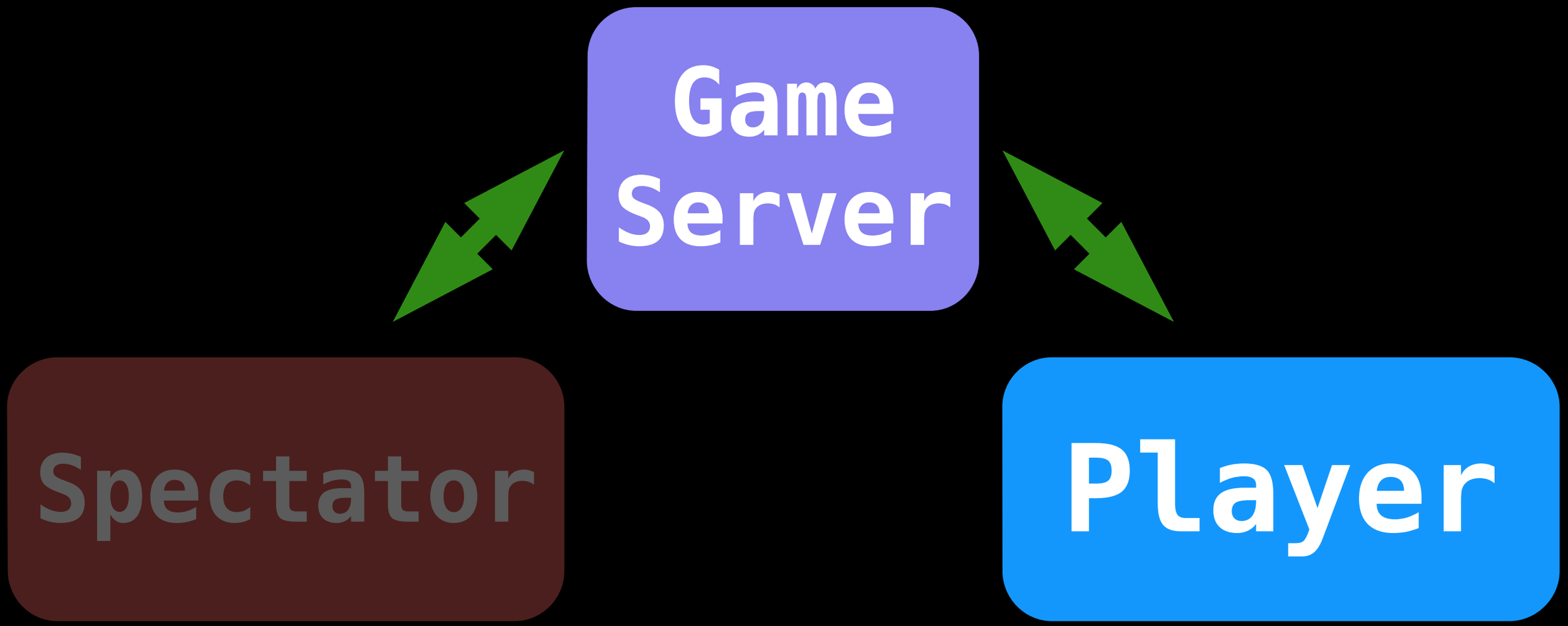
OR

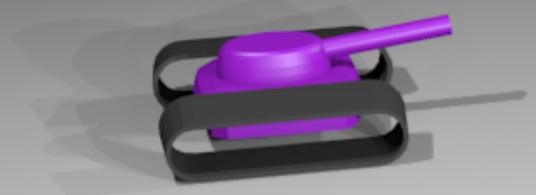
“How I Learned to Stop Worrying and Love Interfaces”

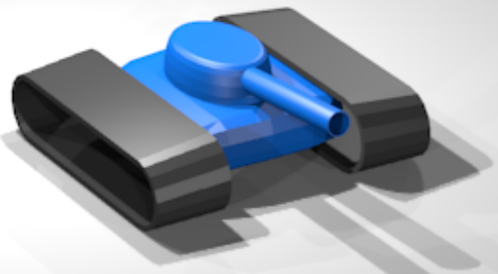
Robot Protocol

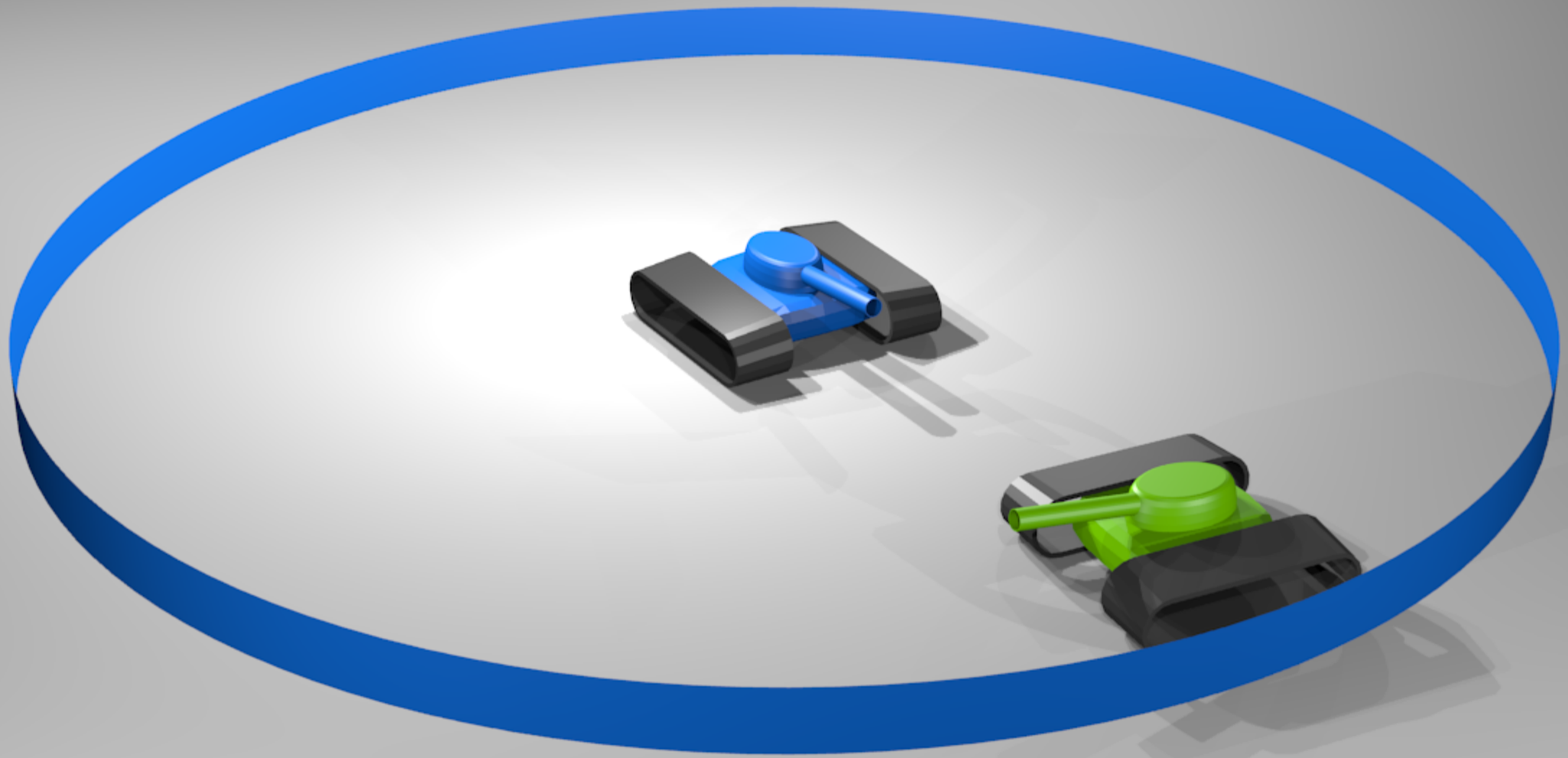


Robot Protocol

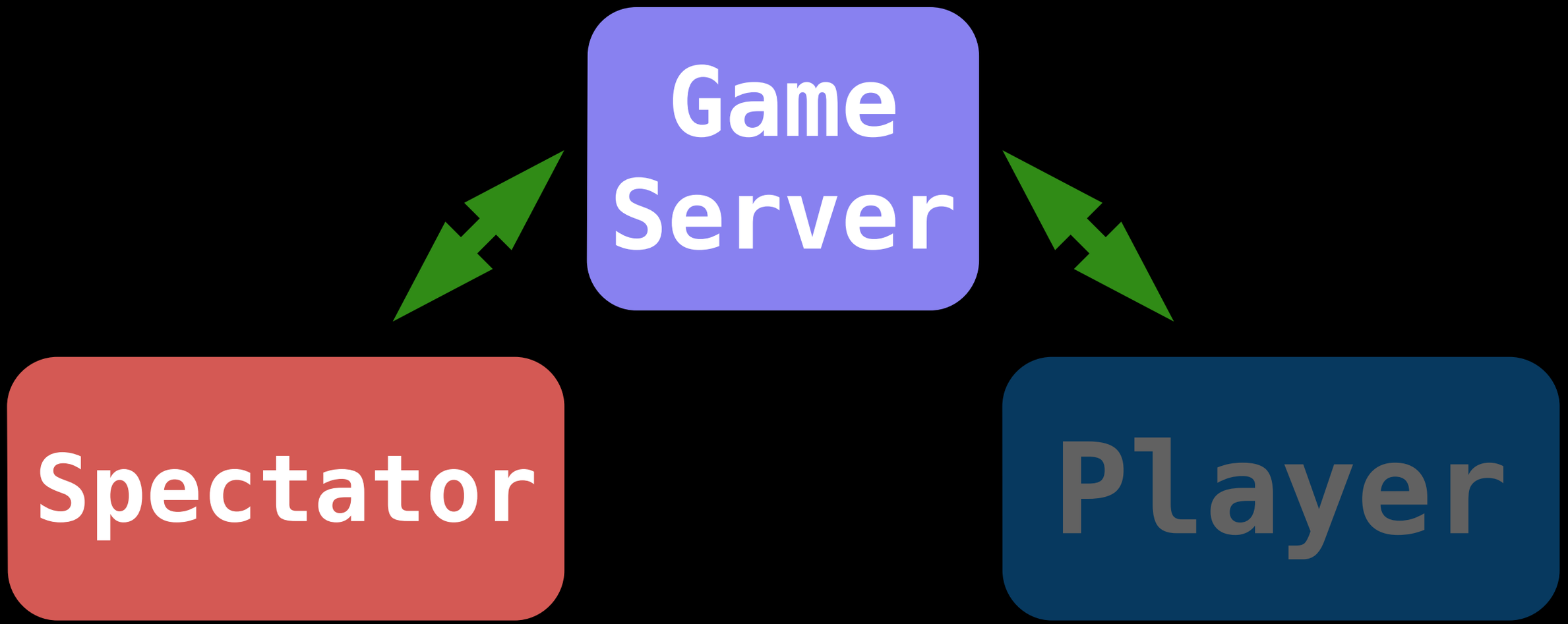








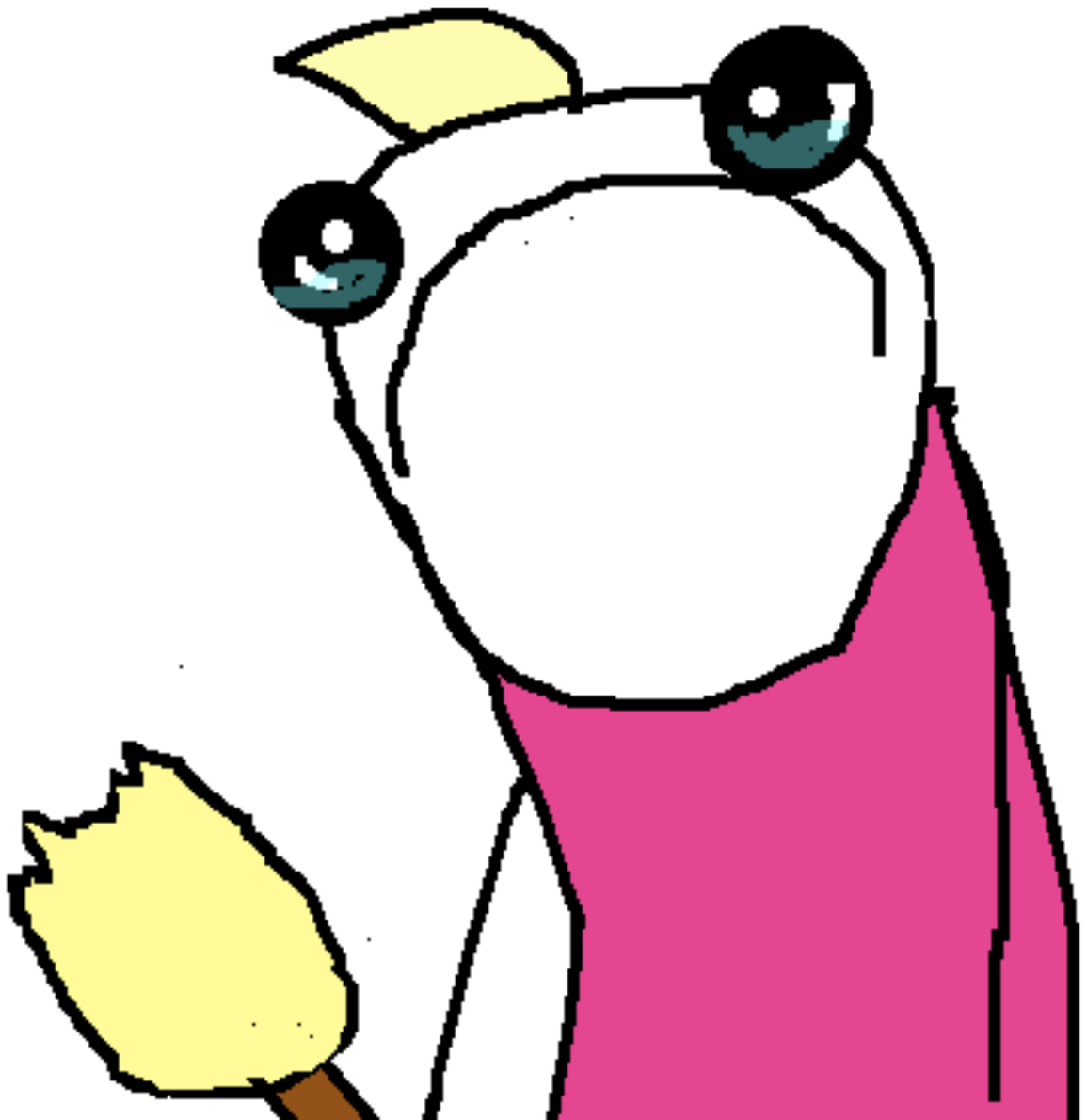
Robot Protocol



Imagine ALL THE ~~people~~!

bytes!!

send all the things?



```
type Point struct {
    X, Y float64
}

...

p := Point{x, y}
b, err := json.Marshal(p)

...

json: '{"X":23.827610293658736,"Y":
42.273774761991874}'
bytes: 47 (worst)
```

```
func (fp *Point) AsArray() []float64 {
    return []float64{fp.X, fp.Y}
}

...

p := Point{x, y}
b, err := json.Marshal(p.AsArray())

...

json: '[23.827610293658736,42.273774761991874]'
bytes: 39 (47 worst)
```

```
type Point struct {
    X, Y float32
}

func (p *Point) AsArray() []float32 {
    return []float32{p.X, p.Y}
}
```

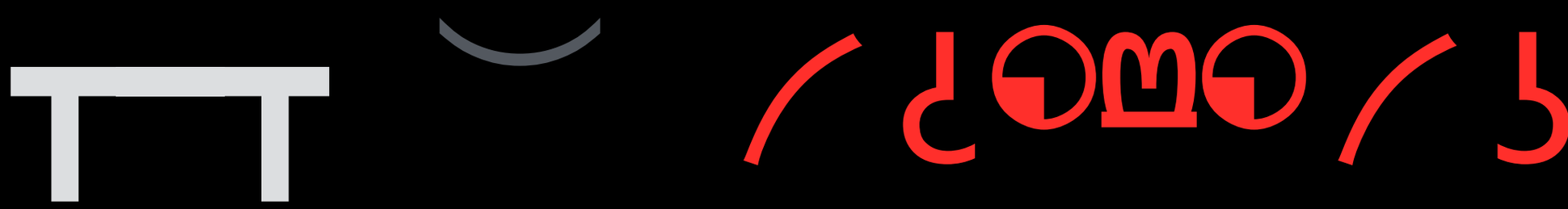
```
...

jsoned: '[23.82761,42.273773]'
bytes: 20 (47 worst)
```

ლ(ಠ益ಠლ)

but at what cost?!

```
game.go:123: invalid operation: d + f
(mismatched types float64 and float32)
```



```
type Gopher interface {
    Squeak()
}
```

```
type Point struct {
    X, Y float64
}

func (ip Point) Squeak() {
    // ...
}

...

var p Point = Point{2, 3}
var g Gopher = Point{2, 3}
```

```
func foo(g Gopher) {}
var p Point = Point{2, 3}
foo(p)
```

If an encountered value implements the **Marshaler interface** and is not a nil pointer, Marshal calls its **MarshalJSON** method to produce JSON

```
type Point struct {
    X, Y float64
}

func (p *Point) MarshalJSON() ([]byte, error) {
    coords := []float32{
        float32(fp.X),
        float32(fp.Y),
    }
    return json.Marshal(coords)
}

...

jsoned: '[23.82761,42.273773]'
bytes: 20 (47 worst)
```


GOB

```
package encoding

type BinaryMarshaler interface {
        MarshalBinary() (data []byte, err error)
}

type TextMarshaler interface {
        MarshalText() (text []byte, err error)
}
```

```
type encoder interface {
    Encode(v interface{}) error
}

if encoding == "json" {
    player.enc = json.NewEncoder(ws)
} else {
    player.enc = gob.NewEncoder(ws)
}

...

player.enc.Encode(boardState)
```

Demo

Future Work