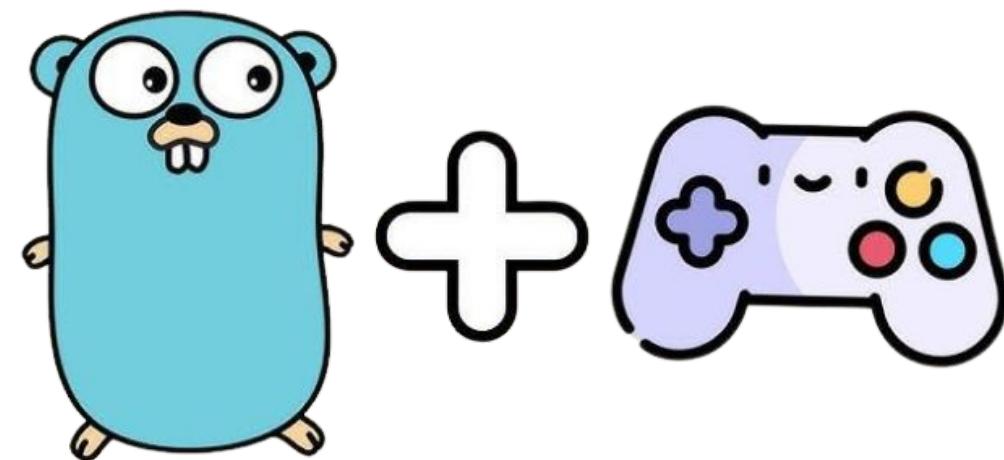


Construindo um game
clássico da década de
70 em Go com
suporte a multiplayer



Carlos Henrique

- Formado em Ciências da Computação
- Pós Graduado em Gestão de Projetos
- 18 anos inserindo bugs em produção
- 6 anos em Golang
- Atualmente Backend Developer na Coinbase
- Contribuidor em projetos Open Source



Renê Cardozo

- Especializado em transformar cafeína em código de qualidade duvidosa
- 5 anos desenvolvendo em Go
- Senior Software Engineer - TRACTIAN
- Formado em Ciência da Computação - USP



@reneepc





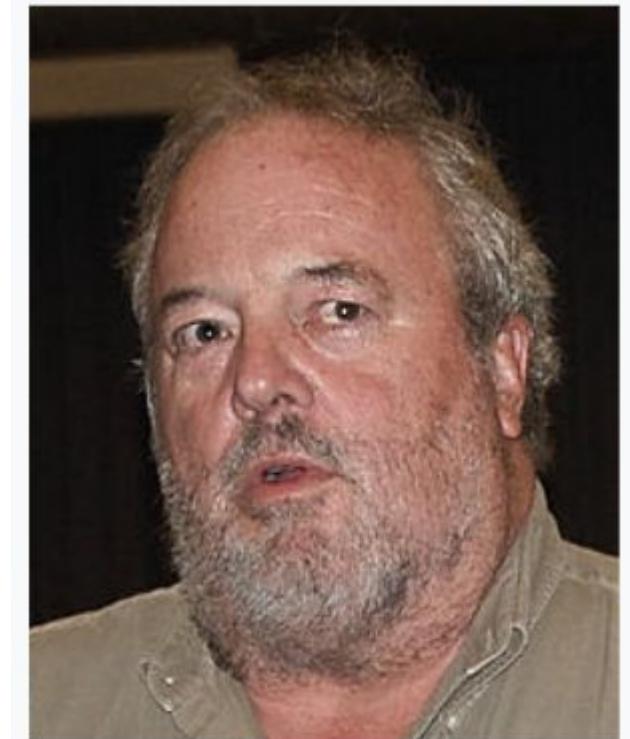
Agenda

- Um pouco de história
- Qual engine utilizar?
- Desafios
- Como detectar colisão?
- Criando uma IA simples
- Máquina de estados
- Comunicação client-server
- Goroutines, channels, context
- Demo!

Um pouco de história

- + Pong é um jogo eletrônico de esporte de arcade com temática de tênis de mesa, com gráficos bidimensionais, desenvolvido pela Atari e lançado originalmente em 1972. Foi
- + 6502 Assembly foi a linguagem escolhida pelos desenvolvedores da Atari e foi originalmente lançada em 1947.
- + Pong foi o primeiro jogo comercialmente bem-sucedido e ajudou a estabelecer a indústria de jogos eletrônicos junto com o Magnavox Odyssey.

Allan Alcorn



Um pouco de história



Qual engine utilizar?

+ **gen2brain/raylib-go**

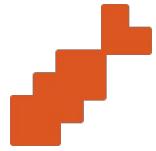
 Star 1.6k

+ **oakmound/oak**

 Star 1.5k

+ **hajimehoshi/ebiten**

 Star 10.8k



 Starred 10.8k ▾

- + **Ebitengine** é uma biblioteca open source para criação de jogos 2D. Permite criar jogos de forma rápida e simples utilizando sua API.
- + Tudo é considerado uma imagem.
- + Suporte a Gráficos 2D, Mouse, Teclado, Gamepad, Touch, Áudio.
- + Cross-Platform: Windows, macOS, Linux, Web, Mobile e até para Nintendo Switch.



收稻 - shuuaku
chill cozy harvest mini game
34.4ms
Action
[Play in browser](#)

Space Deflation
giuliano-macedo
Action
[Play in browser](#)

House Defense Operation!
Tower defense like game. Let's build barricades...
pankona
Survival
[Play in browser](#)

Villa Maravilla
Build a village and bring its community to full life...
Mikkelanglon
Action
[Play in browser](#)

Yet Another Tetris Clone
Tetris, kind of
Loig
Action
[Play in browser](#)

Totally Perfect Order to Build the Massive Space Fortress
noppilinatta
Shooter
[Play in browser](#)

Body Builder
An action maze chaser, where you... build bodies
Josswei
Action
[Play in browser](#)

Tower of Purgatory
Tower building fantasy dungeon party simulator...
kettek
Simulation
[Play in browser](#)

Crusty Crawlers
Mike
Action
[Play in browser](#)

Deck Building Deck Builder
Build a deck (of cards) to build a deck (on a hour...
hprr.dev
Card Game
[Play in browser](#)

Building Manager
mrobbit
Simulation
[Play in browser](#)



Metro Golden Builder
building metro subway jam
JohnJ255
Puzzle
[Play in browser](#)

Rum Pyramid
Prizelobby
Card Game
[Play in browser](#)

Dest
A boss-fighting shooter game with rogueuelike elements
Zyk0
Shooter
[Play in browser](#)

The Sugi Maze Building
Guide the Gopher to reach the rooftop of the Sug...
Hajime Hoshi
Puzzle
[Play in browser](#)

Untitled Castle Game
A 2D Souls-like platformer master both
Action
[Play in browser](#)

Minesweeper
minesweeper for practice
halwhite
Puzzle
[Play in browser](#)

hexslide \$2.99
Slide hexagons into matching coloured frames.
dipst
Puzzle
[Play in browser](#)

Astro Heart
Iskander (quasilyte)
Puzzle
[Play in browser](#)

Voxel demo
Zyk0
[Play in browser](#)

Openquell 1.2.3
A simple puzzle game
Bad Kraut
Puzzle
[Play in browser](#)

Abaddon
A 7drl about revenge
memmabuox
Strategy
[Play in browser](#)

Attack of the Spiders, They Are Destroying Our World
Spartan Games
Action
[Play in browser](#)



Pong Ebitengine
A simple Pong game made using Ebitengine
Tanay PrabhuDesai
[Play in browser](#)

Tiny World
A tiny, slow-paced world and colony building ga...
mlange-42
Strategy
[Play in browser](#)

Arctic Warfare
Battle tanks in the arctic rocketline
Action
[Play in browser](#)

Hacky
A real game of hacky.
humbae
Sports
[Play in browser](#)

Gopher Walk
MyFirst Game!
LidDev
Platformer
[Play in browser](#)

Bug
EHH2023 Entry
Diego Beranvides
Survival
[Play in browser](#)

Glit
A short game for Ebitengine Holiday Hack 2023
tinne26
Platformer
[Play in browser](#)

Lucky Feet
Entry for Ebitengine's Holiday Hack 2023
tinne26
Platformer
[Play in browser](#)

ebisnow
show on your desktop using ebitengine!
kettek
[Play in browser](#)

Please
A glitch library/tool for ebitengine
Zyk0
[Play in browser](#)

HAVEN
Save the world from glitched corruption in thi...
kettek
Adventure
[Play in browser](#)



Gacha Give
Keep your players happy, but not too happy.
possible
Action
[Play in browser](#)

The Passage
Small puzzle game. Reverse glyphs to create a...
Puzzles
[Play in browser](#)

RETRONAMOR
Reverse Invaders
Reverse Space Invaders where you control the...
Action
[Play in browser](#)

REVERSE TYPIST
Reverse the powers of reversing time and light...
Shozen
[Play in browser](#)

Reverse Raiders
A minimalist game about stringing eliminate...
reverse_gravity
Action
[Play in browser](#)

REVERSE RAIDERS
Reverse the goals to guide your raiders to vict...
hikaroid
Action
[Play in browser](#)

REVERSE TO THE GARAGE
Your goal is to carefully reverse your car into...
reverse_enlightened
Action
[Play in browser](#)

RED CAT RUN 2D
Red cat run 2d
Action
[Play in browser](#)

Transition
Eltengame 2023 game jam entry's
Action
[Play in browser](#)

Cute Mold
Create and evolution mold
Action
[Play in browser](#)

MAG
Music Magnet 2
Action
[Play in browser](#)

MAGNETIC SURVIVAL
Survival in magnetic fields
Action
[Play in browser](#)

Scrapyard Magnetic
Scrapyard Magnetic
Action
[Play in browser](#)

DOCTOR LECTRO
Doctor Lectro
Action
[Play in browser](#)

Electromagnet Charge
Electromagnet Charge
Action
[Play in browser](#)

Crane
Crane
Action
[Play in browser](#)

Manual LinealGame
Manual LinealGame
Action
[Play in browser](#)

Reverse defense
Tower defense reversed
Action
[Play in browser](#)

Sides
A death game where you play the NPC.
ZAKO
Platformer
[Play in browser](#)

Anaru
Death game Jam 2023
monner
[Play in browser](#)

GAME ENGINE DEVELOPMENT SIMULATOR
Game Engine Development Simulator
You're the rules, you're no longer just a game.
Hajime Hoshi
Simulation
[Play in browser](#)

Sinecord
Sines made by the power of math!
hikaroid
Simulation
[Play in browser](#)

RevDriller
An action puzzle game
yohann
Action
[Play in browser](#)

Rail
Ball hell shooter with rogue-like elements
ZAKO
Shooter
[Play in browser](#)

Freefall
Control a humanitarian airdrop parachute in t...
Siln Rovik
Adventure
[Play in browser](#)

Roboden
A hybrid between real-time strategy game ab...
hikaroid
Strategy
[Play in browser](#)

Magnet Get
Magnet Get
Action
[Play in browser](#)

Attractive Defense
Attractive Defense
Action
[Play in browser](#)

Dominos
Smooth moving dominos
Action
[Play in browser](#)

Bindless
Bindless
Action
[Play in browser](#)

Bindless
Bindless
Action
[Play in browser](#)

Bindless
Bindless
Action
[Play in browser](#)

KARTULI
START
Kartuli
Learn Georgian
Learn the Georgian alphabet as well as some...
Lolig
Educational
[Play in browser](#)

HARVESTER MADNESS
Harvester Madness
Escort Mission
Find the harvester plowed field in search of another...
hikaroid
Action
[Play in browser](#)

DECIPHERISM
Decipherism
A game where you solve the encoding...
Elies Deter
Puzzle
[Play in browser](#)

RETROHAVE CITY
Retrohave City
A retro-style game set in a futuristic city.
hikaroid
Action
[Play in browser](#)

AUTOTANKS
AUTOTANKS
A game where you control tanks with AI...
hikaroid
Action
[Play in browser](#)

Nine Sweeper Duck
Nine Sweeper Duck
A game where you sweep the floor with a broom...
hikaroid
Action
[Play in browser](#)

ebiten Tetris
ebiten Tetris
A game where you play Tetris with multiple screens and media...
hikaroid
Action
[Play in browser](#)

Flash Element TD v1.5
Flash Element TD v1.5
Action
[Play in browser](#)

Fish Fight Back
Fish Fight Back
Action
[Play in browser](#)

Sharks
Smooth moving sharks
Action
[Play in browser](#)

Diabolos Demolition
Diabolos Demolition
Action
[Play in browser](#)

Thirsty
Thirsty
Action
[Play in browser](#)

Veeve
Go high or low, cross enemies and...
hikaroid
Action
[Play in browser](#)

AAAAXY
AAAAXY
Action
[Play in browser](#)

Manual Linear Motor Car
Manual Linear Motor Car
Action
[Play in browser](#)

- + Lista de jogos <https://itch.io/c/2030581/made-with-ebitengine>

Ebitengine

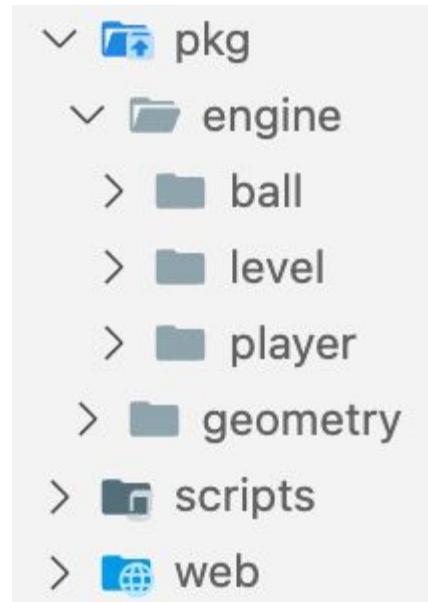
```
● ● ●  
1 package main  
2  
3 import (  
4     "github.com/hajimehoshi/ebiten/v2"  
5     "github.com/hajimehoshi/ebiten/v2/ebitenutil"  
6 )  
7  
8 type Game struct{}  
9  
10 func (g *Game) Update() error {  
11     return nil  
12 }  
13  
14 func (g *Game) Draw(screen *ebiten.Image) {  
15     ebitenutil.DebugPrint(screen, "Hello, World!")  
16 }  
17  
18 func (g *Game) Layout(outsideWidth, outsideHeight int) (screenWidth, screenHeight int) {  
19     return 320, 240  
20 }  
21  
22 func main() {  
23     ebiten.SetWindowSize(640, 480)  
24     ebiten.SetWindowTitle("Hello, World!")  
25  
26     if err := ebiten.RunGame(&Game{}); err != nil {  
27         // handle error  
28     }  
29 }
```

- 60 FPS por padrão
- Draw
- Update
- Layout

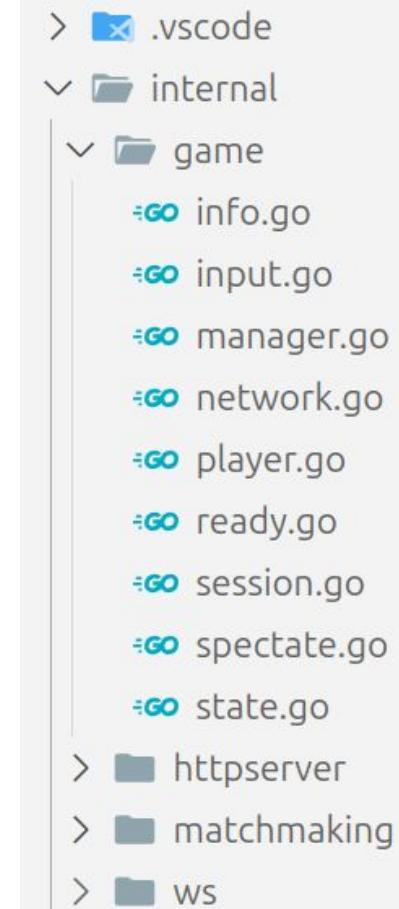
Desafios

- + Aprender a mecânica e as funções da Ebitengine
- + Criar um mecanismo de colisão
- + Exibir estatísticas tais como FPS, Ping, ...
- + Interagir com o usuário
- + Adicionar suporte ao modo multiplayer

Estrutura do Projeto



Lógica da engine
compartilhada



Como detectar colisão?



```
1 // MaxX returns the maximum X value of the rectangle.
2 func (r Rect) MaxX() float64 {
3     return r.X + r.Width
4 }
5
6 // MaxY returns the maximum Y value of the rectangle.
7 func (r Rect) MaxY() float64 {
8     return r.Y + r.Height
9 }
10
11 // Intersects returns true if the rectangle intersects with another rectangle.
12 func (r Rect) Intersects(other Rect) bool {
13     return r.X <= other.MaxX() &&
14         other.X <= r.MaxX() &&
15         r.Y <= otherMaxY() &&
16         other.Y <= r.MaxY()
17 }
```

Como detectar colisão?



```
1 func (p *player) Bounds() geometry.Rect {  
2     return geometry.Rect{  
3         X:      p.position.X,  
4         Y:      p.position.Y,  
5         Width:   p.bouncerWidth,  
6         Height:  p.bouncerHeight,  
7     }  
8 }
```



```
1 func (b *Local) checkPaddleBounce(p1Bounds, p2Bounds geometry.Rect) {  
2     if b.ball.Bounds().Intersects(p1Bounds) {  
3         b.position.X = p1Bounds.X + p1Bounds.Width + width  
4         b.bounceOffPaddle()  
5     }  
6     if b.ball.Bounds().Intersects(p2Bounds) {  
7         b.position.X = p2Bounds.X - b.width  
8         b.bounceOffPaddle()  
9     }  
10 }  
11 }
```

Criando uma IA simples



```
1 func GuessBallPosition(ballY, enemyY, enemyHeight, screenHeight, fieldBorderWidth float64) float64 {
2     delta := float64(rand.Intn(15))
3
4     if enemyY < ballY-delta {
5         enemyY += cpuSpeed // Move down
6     }
7
8     if enemyY > ballY+delta {
9         enemyY -= cpuSpeed // Move up
10    }
11
12    return keepInBounds(enemyY, enemyHeight, screenHeight, fieldBorderWidth)
13 }
```

Lógica de Menu

```
game
  ball.go
  field.go
  game.go
  player.go
  score.go
  winner.go
menu
  instructions.go
  menu.go
```

```
// nolint:gocyclo
func (m *Menu) Update() {
    // key down
    if inpututil.IsAnyKeyJustPressed(ebiten.KeyDown) {
        switch m.state {
        case mainMenu:
            switch m.selectedOption {
            case 0: // local mode
                m.selectedOption = 1
            case 1: // multiplayer mode
                m.selectedOption = 2 // instructions
            }
        case localMode:
            switch m.selectedOption {
            case 0: // one player
                m.selectedOption = 1
            case 1: // two players
                m.selectedOption = 2 // back
            }
        case levelSelection:
            switch m.selectedOption {
            case 0: // easy
                m.selectedOption = 1
            case 1: // medium
                m.selectedOption = 2
            case 2: // hard
                m.selectedOption = 3 // back
            }
        }
        return
    }
    // ...
}
```

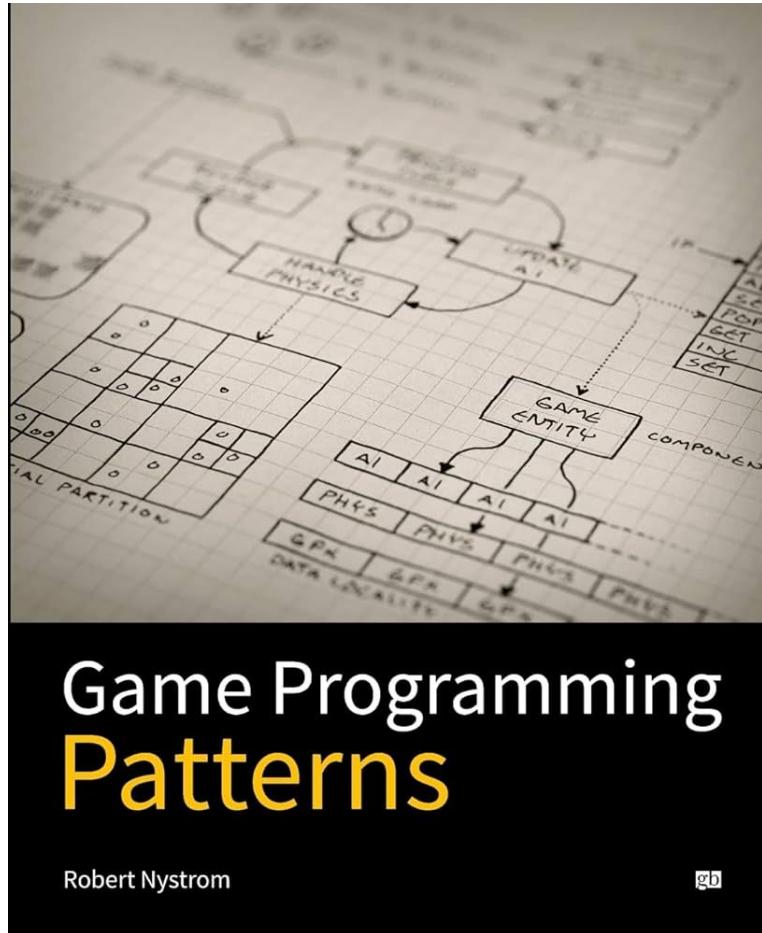
Spaghetti

- Apenas 3 menus já tornam a lógica de seleção enorme
- Alto acoplamento entre componentes

```
1 // nolint:gocyclo
2 func (m *Menu) Update() {
3     // key down
4     if inpututil.IsAnyKeyJustPressed(ebiten.KeyDown) {
5         switch m.state {
6             case mainMenu:
7                 switch m.selectedOption {
8                     case 0: // local mode
9                         m.selectedOption = 1
10                case 1: // multiplayer mode
11                    m.selectedOption = 2 // instructions
12                }
13            case localMode:
14                switch m.selectedOption {
15                    case 0: // one player
16                        m.selectedOption = 1
17                    case 1: // two players
18                        m.selectedOption = 2 // back
19                }
20            case levelSelection:
21                switch m.selectedOption {
22                    case 0: // easy
23                        m.selectedOption = 1
24                    case 1: // medium
25                        m.selectedOption = 2
26                    case 2: // hard
27                        m.selectedOption = 3 // back
28                }
29        }
30    }
31    return
32 }
33 // ...
34 
```

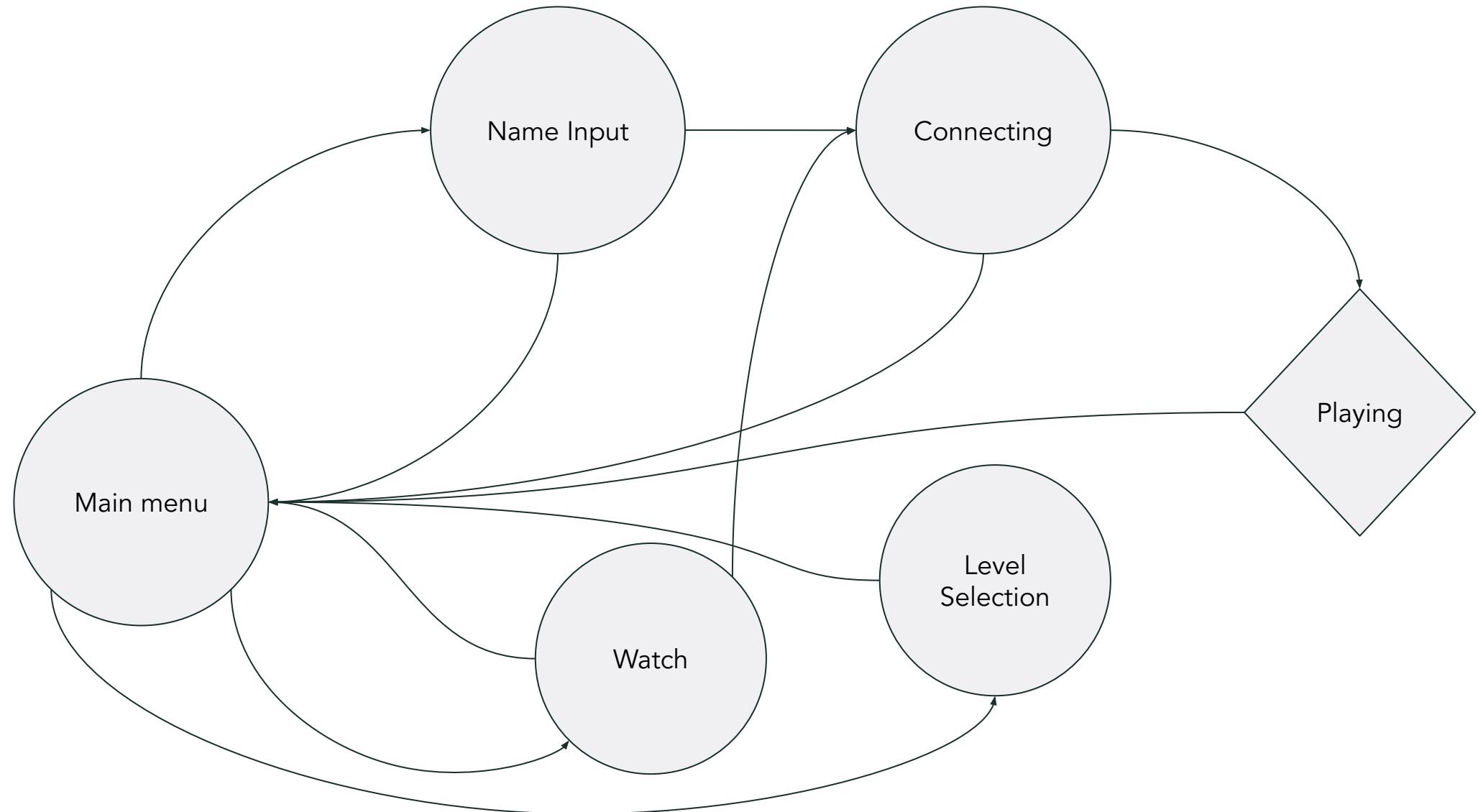
```
1 // nolint:gocyclo
2 func (m *Menu) Update() {
3     // key down
4     if inpututil.IsAnyKeyJustPressed(ebiten.KeyDown) {
5         switch m.state {
6             case mainMenu:
7                 switch m.selectedOption {
8                     case 0: // local mode
9                         m.selectedOption = 1
10                    case 1: // multiplayer mode
11                        m.selectedOption = 2 // instructions
12                    }
13            case localMode:
14                switch m.selectedOption {
15                    case 0: // one player
16                        m.selectedOption = 1
17                    case 1: // two players
18                        m.selectedOption = 2 // back
19                    }
20            case levelSelection:
21                switch m.selectedOption {
22                    case 0: // easy
23                        m.selectedOption = 1
24                    case 1: // medium
25                        m.selectedOption = 2
26                    case 2: // hard
27                        m.selectedOption = 3 // back
28                    }
29        }
30    }
31    return
32 }
33 // key up
34 if inpututil.IsAnyKeyJustPressed(ebiten.KeyUp) {
35     switch m.state {
36         case mainMenu:
37             switch m.selectedOption {
38                 case 0: // instructions
39                     m.selectedOption = 1
40                 case 1: // local mode
41                     m.selectedOption = 0 // local mode
42                 case 2: // multiplayer mode
43                     m.selectedOption = 0 // local mode
44             case localMode:
45                 switch m.selectedOption {
46                     case 0: // back
47                     m.selectedOption = 1
48                     case 1: // two players
49                     m.selectedOption = 0 // one player
50                 }
51             case levelSelection:
52                 switch m.selectedOption {
53                     case 0: // back
54                     m.selectedOption = 2
55                     case 1: // medium
56                     m.selectedOption = 1
57                     case 2: // easy
58                     m.selectedOption = 0
59                 }
60         }
61     }
62 }
63 // key enter
64 if inpututil.IsAnyKeyJustPressed(ebiten.KeyEnter) {
65     switch m.state {
66         case mainMenu:
67             switch m.selectedOption {
68                 case 0: // local mode
69                     m.state = localMode
70                     m.selectedOption = 0
71                 case 1: // multiplayer mode
72                     m.state = multiplayerMode
73                     m.selectedOption = 0
74                 case 2: // instructions
75                     m.state = levelSelection
76                     m.level = level.Medium // default level for multiplayer
77                     m.readyToPlay = true
78             case localMode:
79                 switch m.selectedOption {
80                     case 0: // Instructions
81                     m.state = instructions
82                     m.selectedOption = 0
83                 }
84             case multiplayerMode:
85                 switch m.selectedOption {
86                     case 0: // single player
87                     m.gameMode = onePlayer
88                     m.state = levelSelection
89                     m.selectedOption = 0
90                 case 1: // two players
91                     m.gameMode = twoPlayers
92                     m.state = levelSelection
93                     m.selectedOption = 0
94                 }
95             case levelSelection:
96                 switch m.selectedOption {
97                     case 0: // back
98                     m.state = mainMenu
99                     m.gameMode = undefined
100                    m.readyToPlay = false
101                    m.selectedOption = 0
102                }
103            }
104        }
105    }
106 }
107 // key esc
108 if inpututil.IsAnyKeyJustPressed(ebiten.KeyEscape) {
109     switch m.state {
110         case mainMenu:
111             // exit the game
112             m.state = undefined
113             m.readyToPlay = true // just to force exit menu
114             case localMode, levelSelection, instructions:
115                 m.selectedOption = 0
116                 m.gameMode = undefined
117                 m.readyToPlay = false
118             case mainMenu:
119                 m.gameMode = undefined
120                 m.readyToPlay = false
121                 m.selectedOption = 0
122             }
123 }
124 }
125 }
126 }
127 // key esc
128 if inpututil.IsAnyKeyJustPressed(ebiten.KeyEscape) {
129     switch m.state {
130         case mainMenu:
131             // exit the game
132             m.state = undefined
133             m.readyToPlay = true // just to force exit menu
134             case localMode, levelSelection, instructions:
135                 m.selectedOption = 0
136                 m.gameMode = undefined
137                 m.readyToPlay = false
138             }
139 }
140 }
```

State Pattern

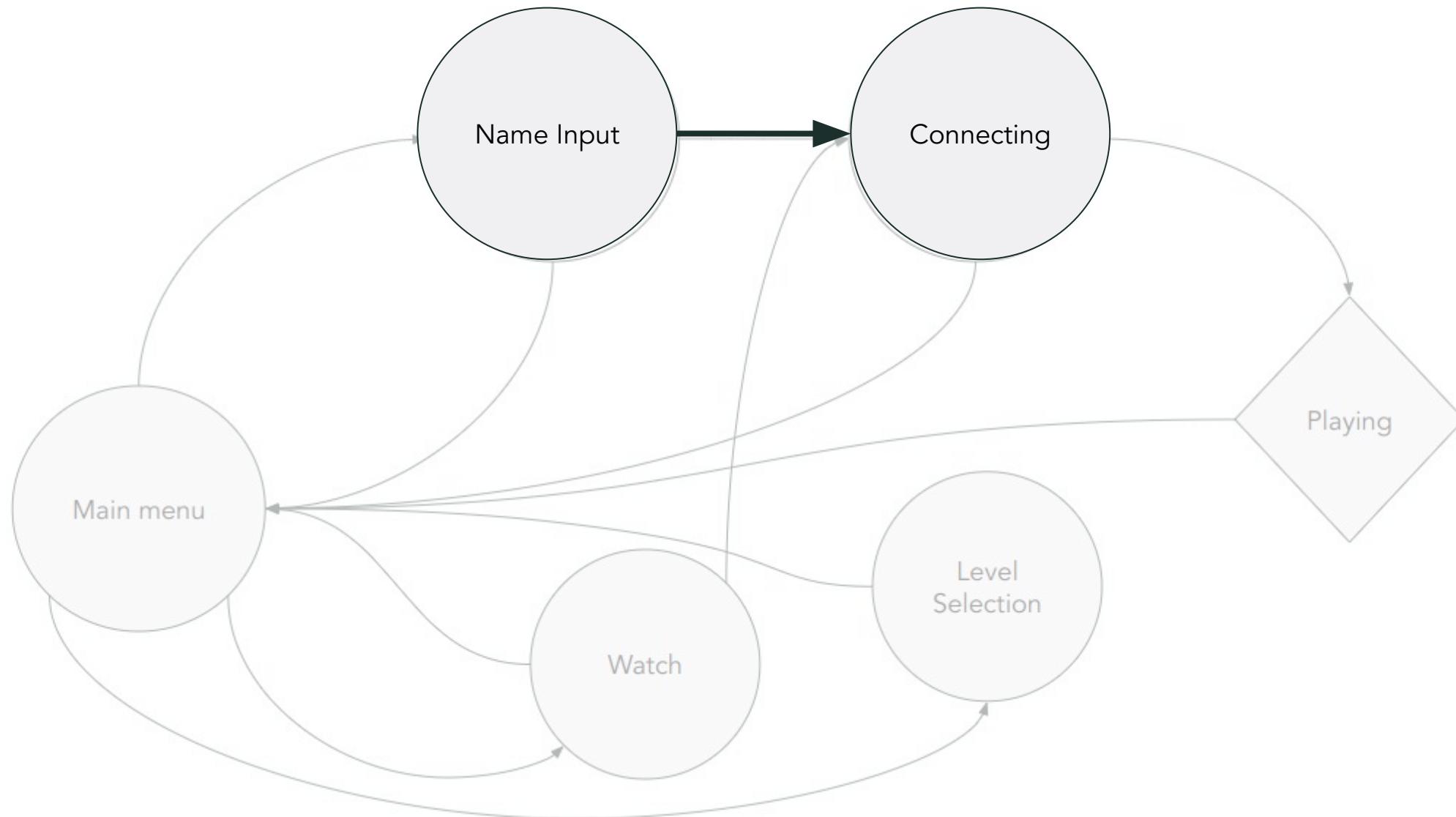


Robert Nystrom

Menu State



Menu State



State Pattern

```
● ● ●  
1 // New creates a new game menu.  
2 func New(font *font.Font, screenWidth, screenHeight int) *Menu {  
3     menu := &Menu{  
4         font:        font,  
5         gameMode:    Undefined,  
6         screenWidth: screenWidth,  
7         screenHeight: screenHeight,  
8         states:      make(map[string]state),  
9     }  
10  
11     menu.ChangeState(newMainMenuState(menu))  
12  
13     return menu  
14 }  
15  
16 // ChangeState changes the current state of the menu.  
17 func (m *Menu) ChangeState(state state) {  
18     if st, ok := m.states[state.String()]; ok {  
19         m.currentState = st  
20         return  
21     }  
22  
23     m.states[state.String()] = state  
24     m.currentState = state  
25 }
```

State Pattern

```
● ● ●  
1 // New creates a new game menu.  
2 func New(font *font.Font, screenWidth, screenHeight int) *Menu {  
3     menu := &Menu{  
4         font:        font,  
5         gameMode:    Undefined,  
6         screenWidth: screenWidth,  
7         screenHeight: screenHeight,  
8         states:      make(map[string]state),  
9     }  
10  
11     menu.ChangeState(newMainMenuState(menu))  
12  
13     return menu  
14 }  
15  
16 // ChangeState changes the current state of the menu.  
17 func (m *Menu) ChangeState(state state) {  
18     if st, ok := m.states[state.String()]; ok {  
19         m.currentState = st  
20         return  
21     }  
22  
23     m.states[state.String()] = state  
24     m.currentState = state  
25 }
```



```
● ● ●  
1 // Update updates the menu.  
2 func (m *Menu) Update() {  
3     m.currentState.Update()  
4 }  
5  
6 // Draw draws the menu.  
7 func (m *Menu) Draw(screen *ebiten.Image) {  
8     m.currentState.Draw(screen)  
9 }
```

Base menu

```
● ○ ●
1 type baseState struct {
2     selectedOption int
3     options        []string
4     menu           *Menu
5 }
6
7 func (s *baseState) navigateOptions(maxOptions int) {
8     if inpututil.IsAnyKeyJustPressed(ebiten.KeyDown) {
9         if s.selectedOption < maxOptions-1 {
10             s.selectedOption++
11         }
12     }
13
14     if inpututil.IsAnyKeyJustPressed(ebiten.KeyUp) {
15         if s.selectedOption > 0 {
16             s.selectedOption--
17         }
18     }
19 }
```

Base menu

```
● ● ●

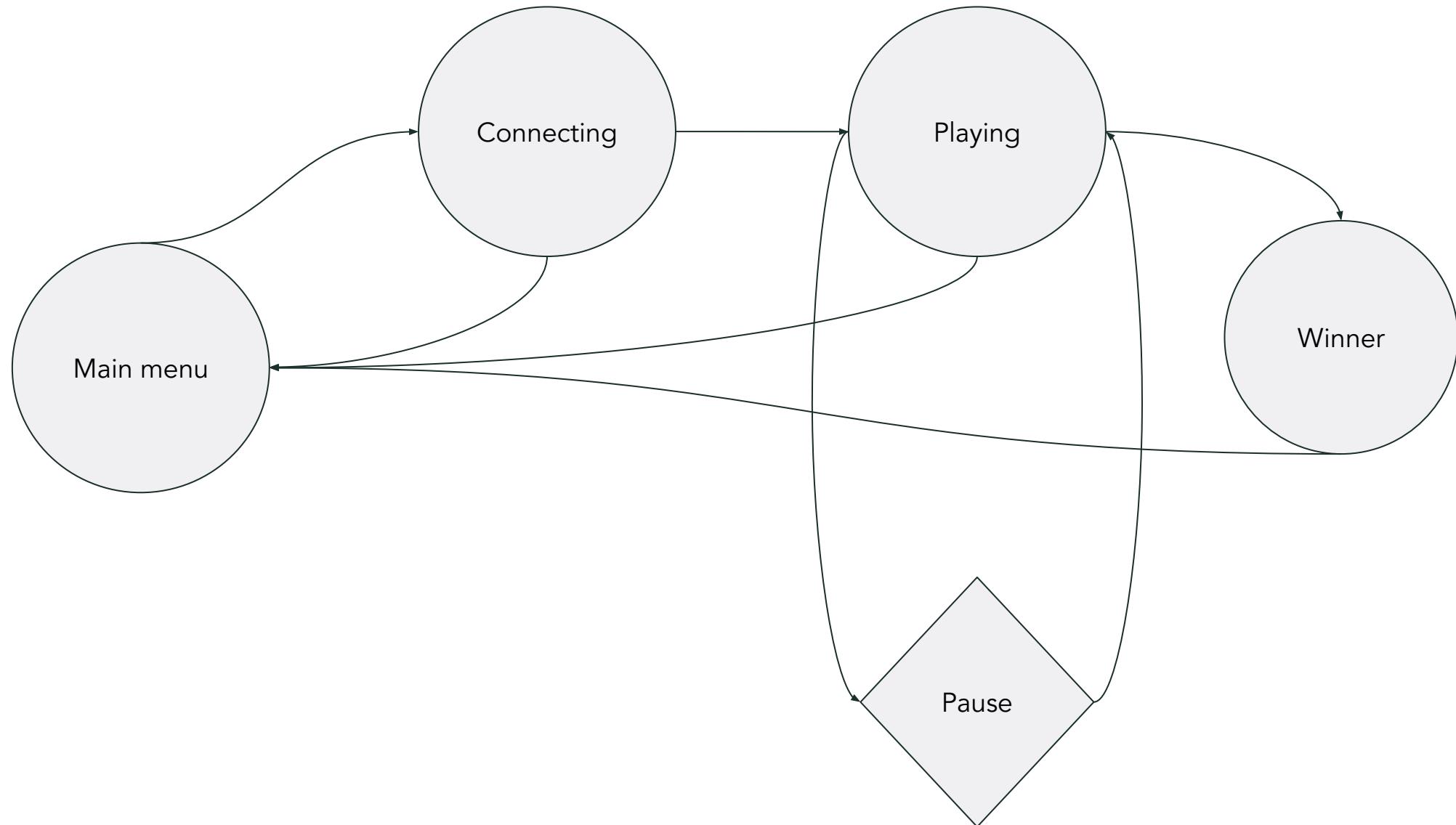
1 type mainMenuState struct {
2     *baseState
3 }
4
5 func newMainMenuState(menu *Menu) *mainMenuState {
6     return &mainMenuState{
7         baseState: &baseState{
8             menu: menu,
9             options: []string{
10                 localModeStr,
11                 multiplayerStr,
12                 spectateStr,
13                 instructionsStr},
14             },
15         }
16 }
```

Desacoplamento

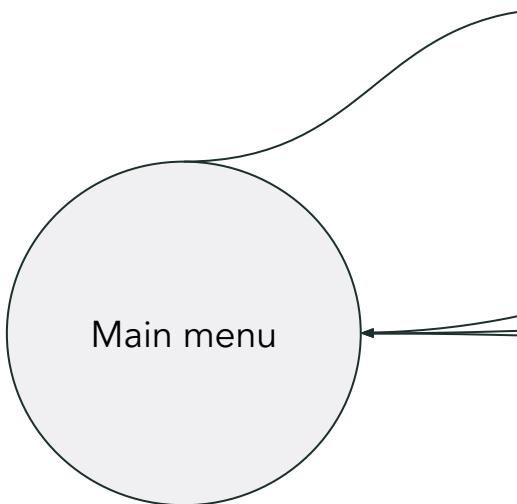
```
● ● ●

1 func (s *mainMenuState) Update() {
2     s.navigateOptions(len(s.options))
3
4     if inpututil.IsAnyKeyJustPressed(ebiten.KeyEnter) {
5         switch s.selectedOption {
6             case 0:
7                 s.menu.ChangeState(newLocalModeState(s.menu))
8             case 1:
9                 s.menu.ChangeState(newInputNameState(s.menu))
10            case 2:
11                s.menu.ChangeState(newSpectateSessionsState(s.menu))
12            case 3:
13                s.menu.ChangeState(newInstructionsState(s.menu))
14        }
15    }
16
17    // return to quit the game
18    if inpututil.IsAnyKeyJustPressed(ebiten.KeyEscape) {
19        s.menu.gameMode = Undefined
20        s.menu.readyToPlay = true
21    }
22 }
```

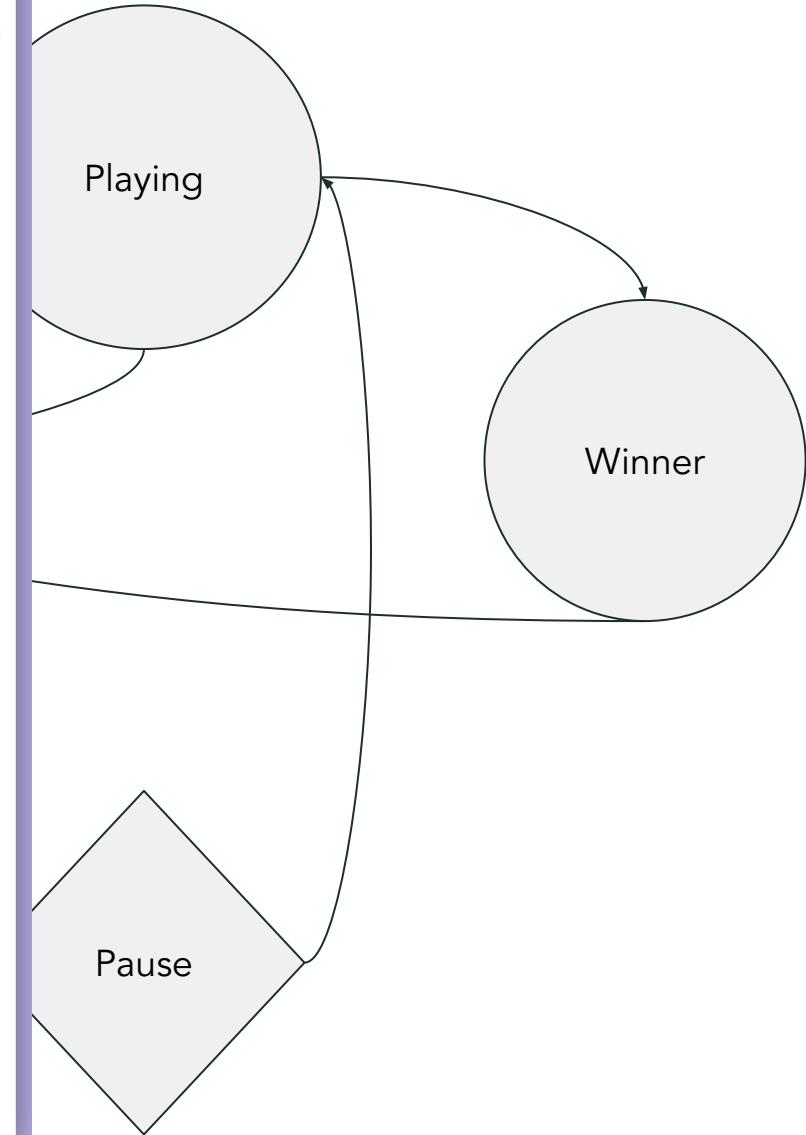
Game State



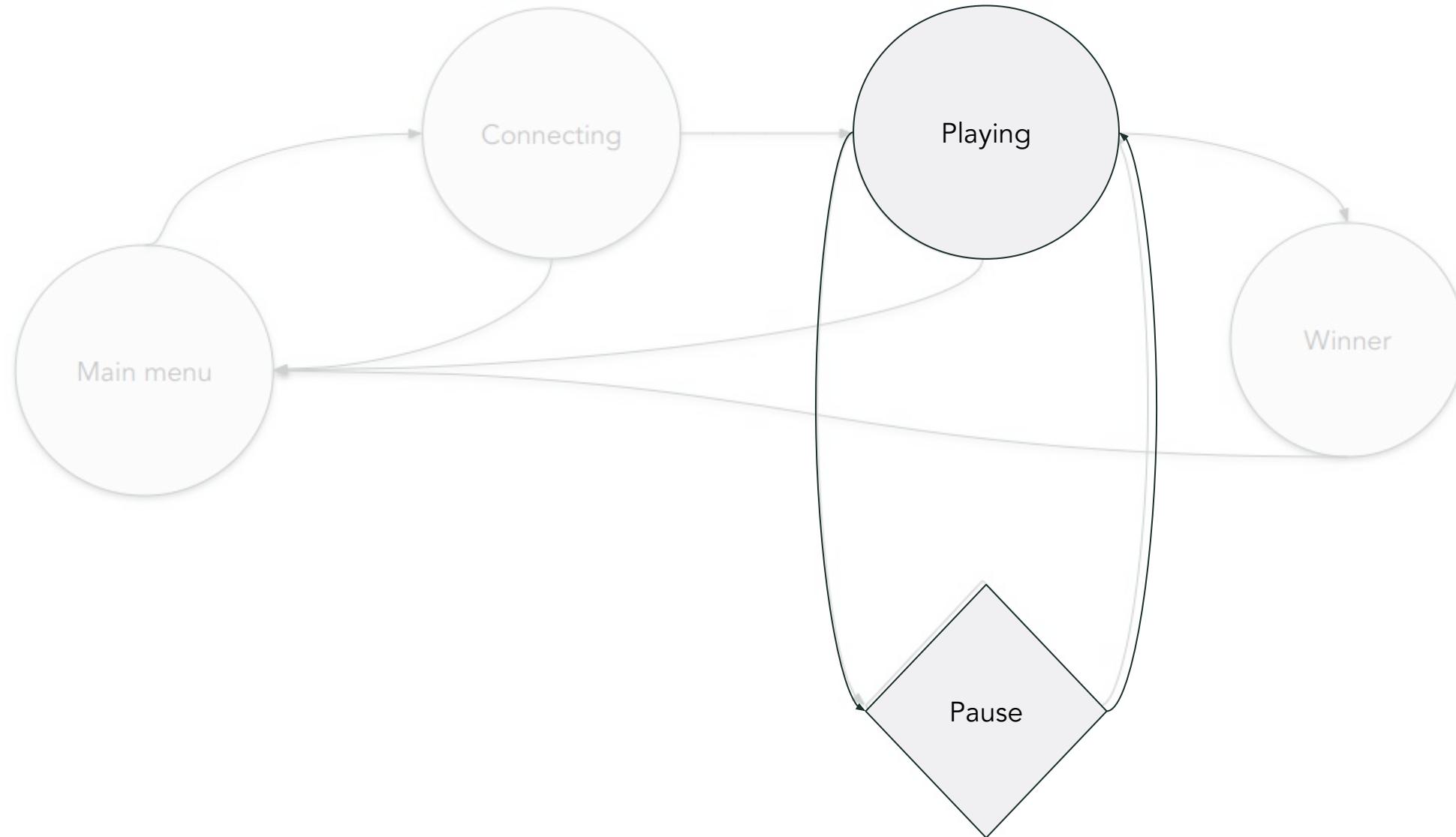
Game State



```
1 // Update updates the game.
2 func (g *Game) Update() error {
3     switch g.state {
4     case notReady:
5         // if the game is not ready to play, update the menu
6         if !g.menu.IsReadyToPlay() {
7             g.menu.Update()
8         }
9         return nil
10    }
11
12    // it means to exit the game
13    if g.menu.GameMode() == menu.Undefined {
14        if runtime.GOOS != "js" {
15            return ebiten.Termination
16        }
17        os.Exit(0)
18    }
19
20    // if game mode is multiplayer, then change the state to connecting and wait for the connection
21    if g.menu.GameMode() == menu.Multiplayer {
22        g.state = connecting
23        return nil
24    }
25
26
27    // if the game is ready to play, change the state to playing
28    g.state = playing
29
30    case connecting:
31        if g.networkClient == nil {
32            g.networkClient = network.NewClient(g.ctx, g.cancel, network.BaseURL)
33            if err := g.networkClient.Connect(); err != nil {
34                return fmt.Errorf("failed to connect to the server: %w", err)
35            }
36
37            // TODO: get player name from input
38            if err := g.networkClient.SendPlayerInfo(network.GameInfo{
39                PlayerName: "Player 1",
40                Level:      int(g.menu.Level()),
41                ScreenWidth:ScreenWidth,
42                ScreenHeight:ScreenHeight,
43                MaxScore:   maxScore,
44                FieldBorderWidth:int(fieldBorderWidth),
45            }); err != nil {
46                return fmt.Errorf("failed to send player info: %w", err)
47            }
48
49            g.networkClient.ReceiveGameState(g.networkGameStateCh)
50
51            go func() {
52                // wait for the connection to be established
53                <-g.networkGameStateCh
54
55                // TODO: check if start returns an error
56                _ = g.start()
57                g.state = playing
58            }()
59        case playing:
60            if g.menu.GameMode() == menu.Multiplayer {
61                gameState := <-g.networkGameStateCh
62
63                // update the game state
64                if gameState.CurrentPlayer.Side == geometry.Left {
65                    g.player1.SetPosition(gameState.CurrentPlayer.PositionY)
66                } else {
67                    g.player2.SetPosition(gameState.CurrentPlayer.PositionY)
68                }
69
69                if gameState.OpponentPlayer.Side == geometry.Left {
70                    g.player1.SetPosition(gameState.OpponentPlayer.PositionY)
71                } else {
72                    g.player2.SetPosition(gameState.OpponentPlayer.PositionY)
73                }
74
75                g.ball.SetAngle(gameState.Ball.Angle)
76                g.ball.SetBounces(gameState.Ball.Bounces)
77                g.ball.SetPosition(gameState.Ball.Position)
78
79                g.score1.value = gameState.CurrentPlayer.Score
80                g.score2.value = gameState.OpponentPlayer.Score
81
82                g.pingCurrentPlayer = gameState.CurrentPlayer.Ping
83                g.pingOpponent = gameState.OpponentPlayer.Ping
84
85            }
86
87            // call the update to do the game logic
88            if err := g.update(); err != nil {
89                return fmt.Errorf("failed to update the game: %w", err)
90            }
91
92            // if ESC is pressed, then show/hide metrics
93            if inpututil.IsAnyJustPressed(ebiten.KeyTab) {
94                g.showMetric = !g.showMetric
95            }
96        case ended:
97            if inpututil.IsAnyJustPressed(ebiten.KeyEnter) {
98                g.reset()
99            }
100       }
101
102    return nil
103 }
```



Game State

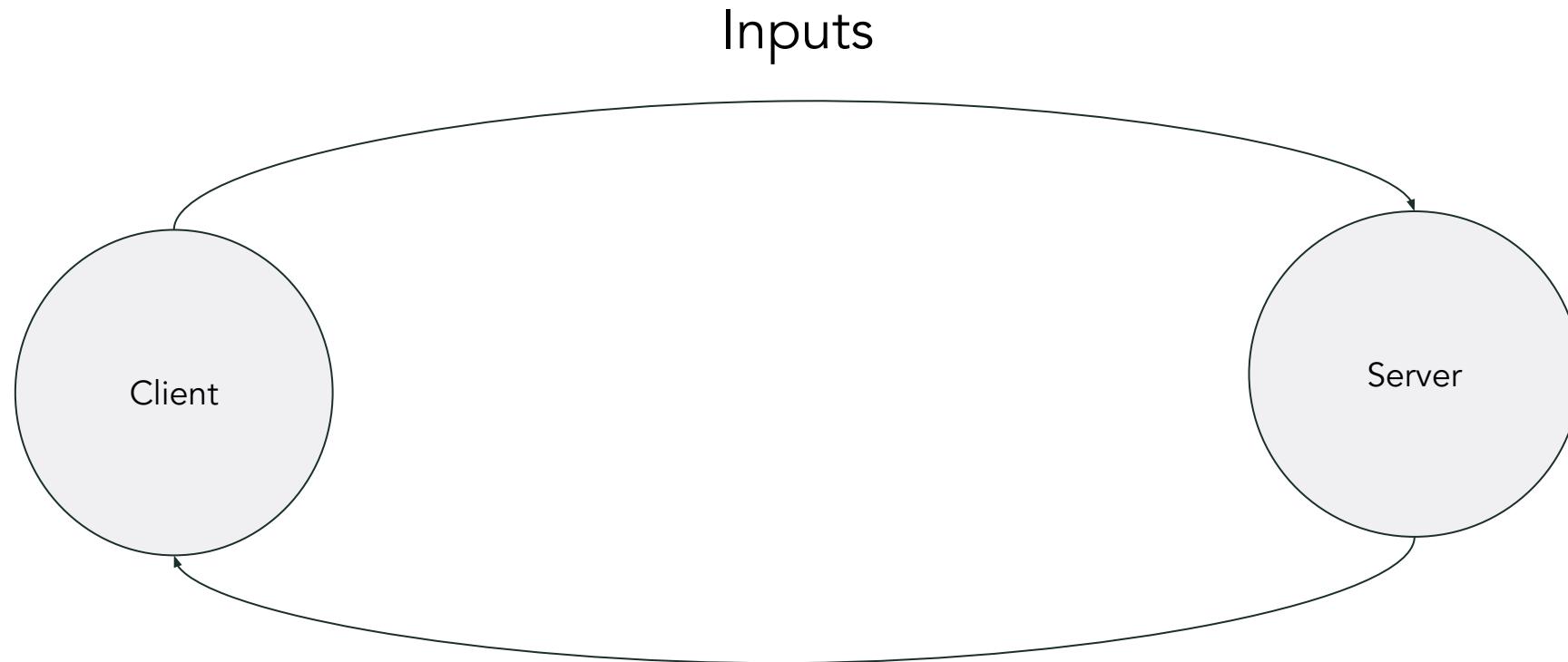


Desacoplamento

```
✓ game
-:GO ball.go
-:GO base.go
-:GO connecting.go
-:GO field.go
-:GO game.go
-:GO main_menu.go
-:GO multiplayer.go
-:GO one_player.go
-:GO pause_menu.go
-:GO player.go
-:GO score.go
-:GO spectator.go
-:GO state.go
-:GO two_players.go
-:GO winner.go
```

```
✓ menu
-:GO base.go
-:GO input_name.go
-:GO instructions.go
-:GO level.go
-:GO local.go
-:GO main.go
-:GO menu.go
-:GO spectate_sessions.go
-:GO spectator_connecting.go
-:GO state.go
-:GO two_player_instructions.go
```

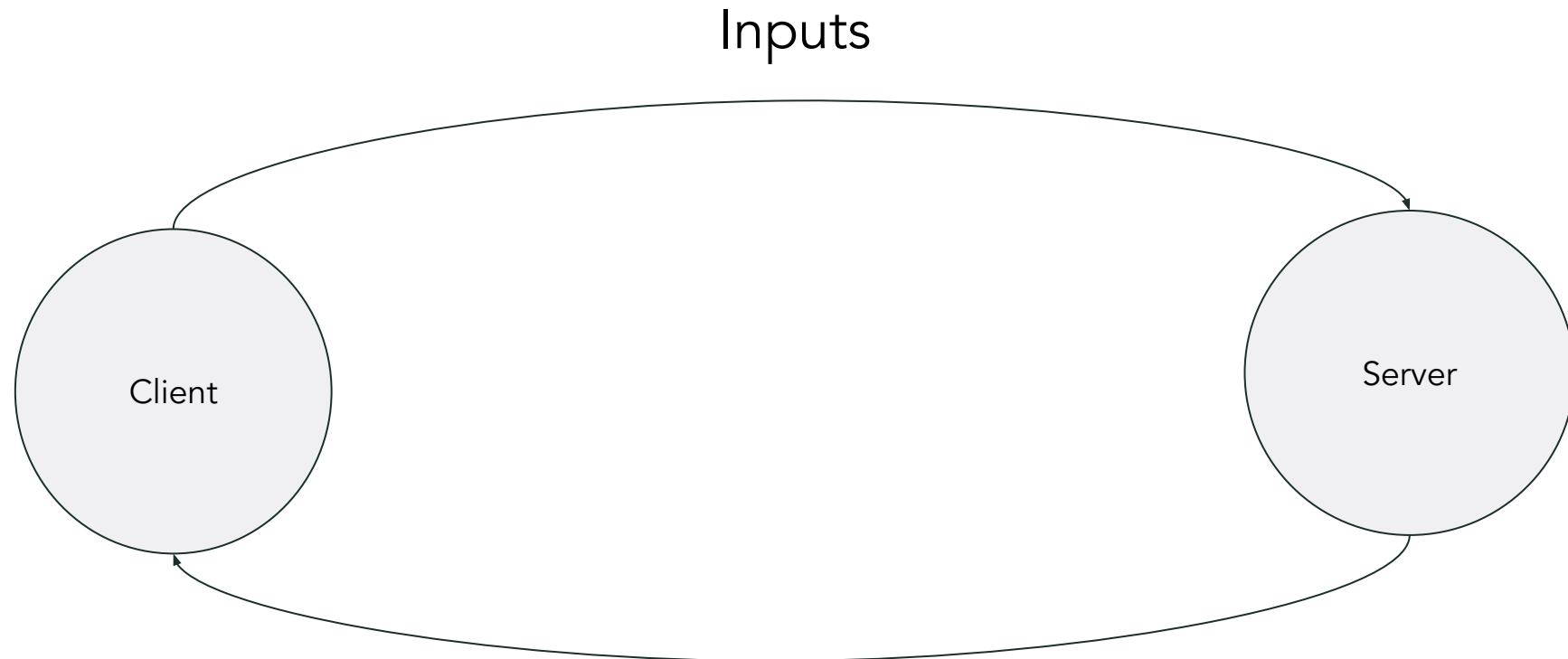
Multiplayer



- Overhead mínimo para envio de mensagens

Estado do mundo

Multiplayer

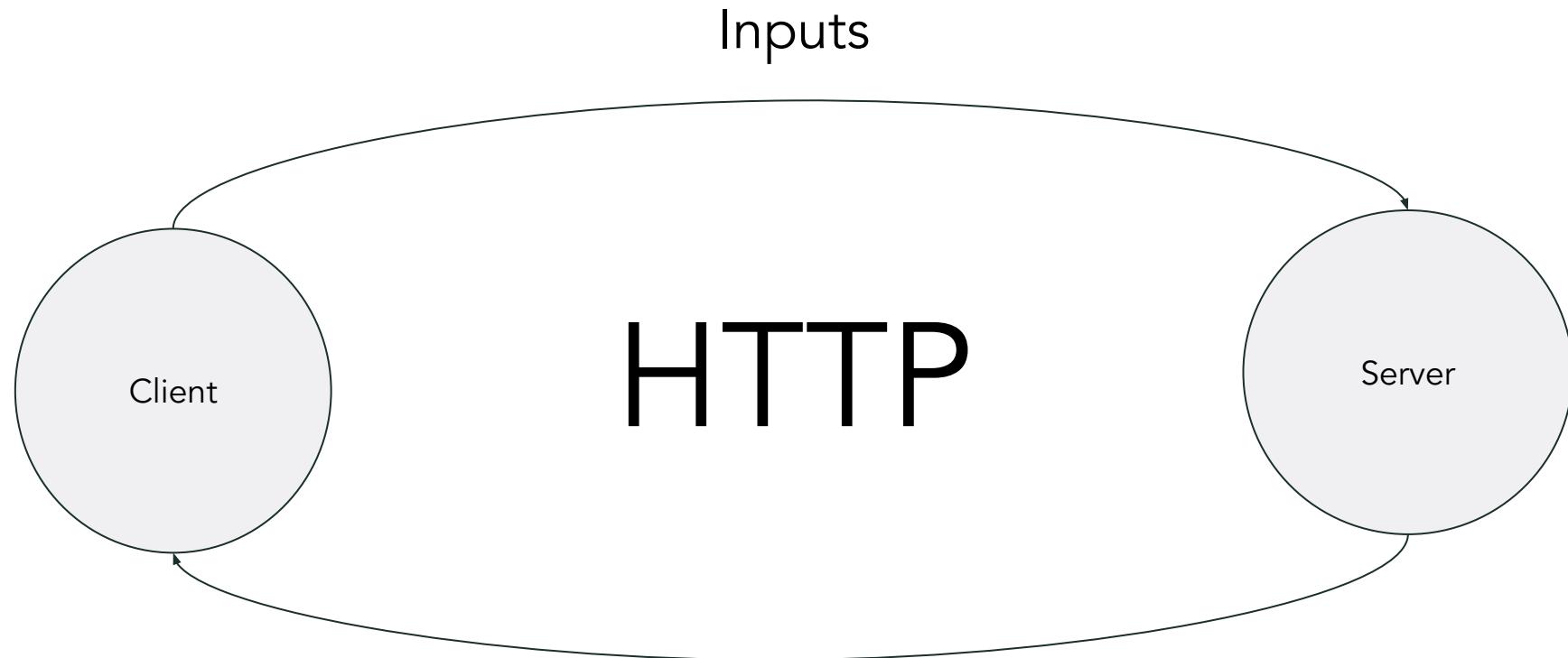


- Overhead mínimo para envio de mensagens

Estado do mundo

- Full duplex
- Real-time

Multiplayer (HTTP)

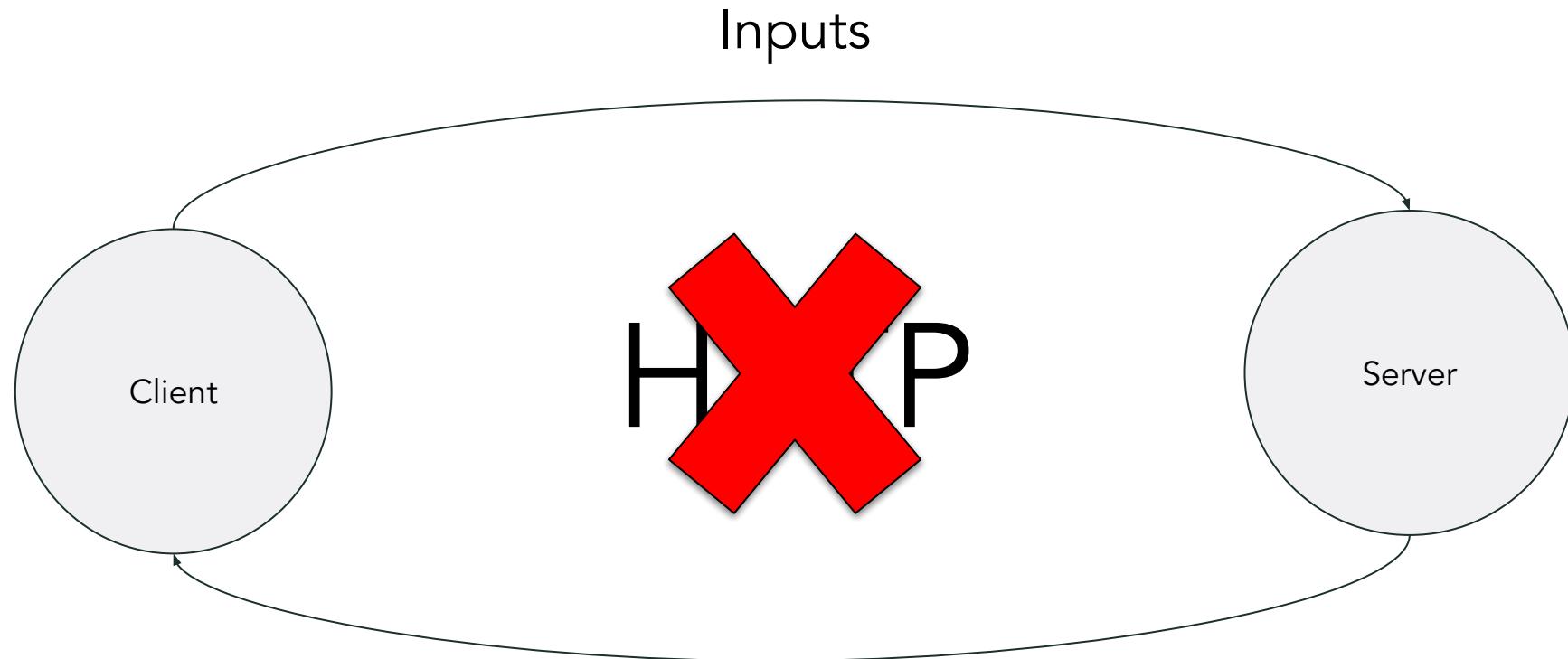


- Overhead mínimo para envio de mensagens

Estado do mundo

- Full duplex
- Real-time

Multiplayer (HTTP)

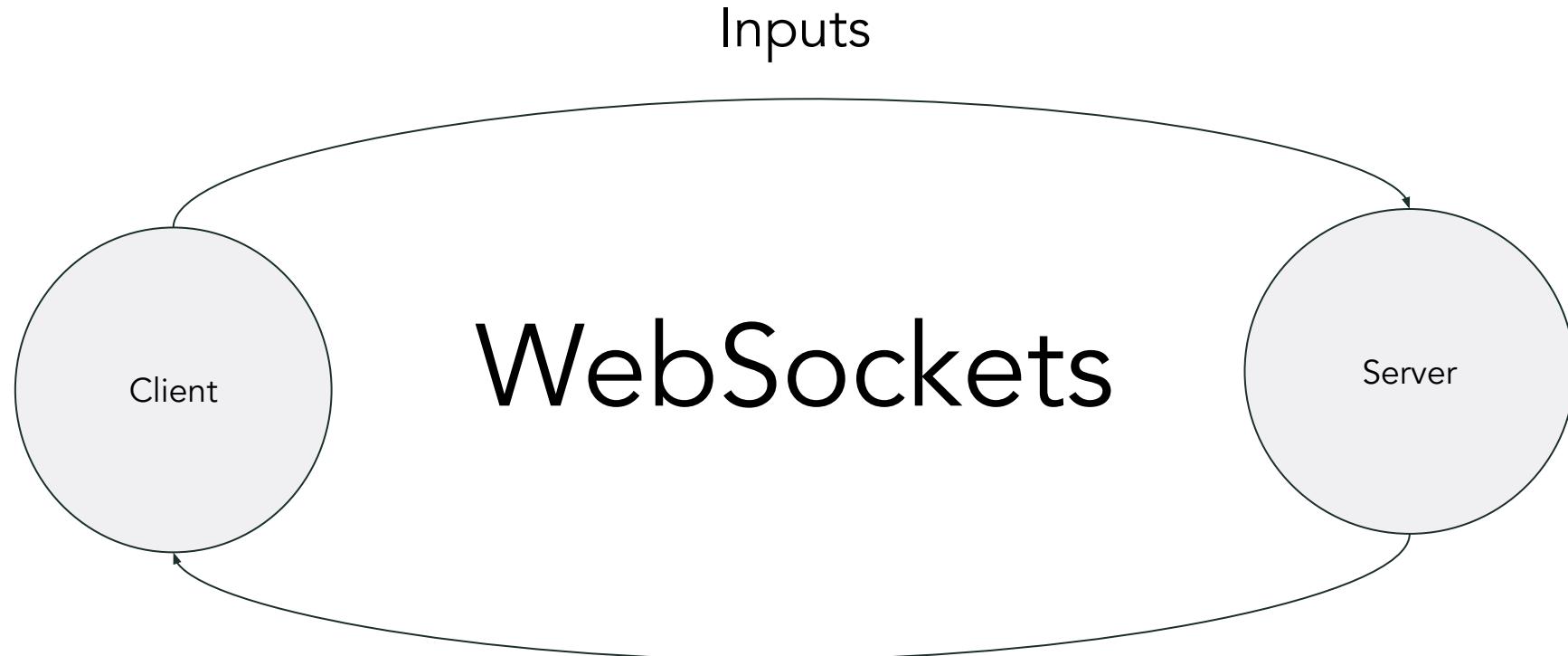


- Overhead mínimo para envio de mensagens

Estado do mundo

- Full duplex
- Real-time

Multiplayer (WebSockets)



- Overhead mínimo para envio de mensagens

Estado do mundo

- Full duplex
- Real-time

gorilla/websocket



```
1 http.HandleFunc("/multiplayer", wsServer.HandleConnections)
2 http.HandleFunc("/spectate", wsServer.HandleSpectatorConnections)
3 http.HandleFunc("/sessions", s.handleSessions)
```

gorilla/websocket

```
● ● ●  
1 func (s *Server) HandleConnections(w http.ResponseWriter, r *http.Request) {  
2     // Upgrade HTTP connection to WebSocket  
3     conn, err := s.upgrader.Upgrade(w, r, nil)  
4     if err != nil {  
5         slog.Error("Failed to upgrade connection", slog.Any("error", err))  
6         return  
7     }  
8  
9     // Wait for initial player info  
10    var info game.GameInfo  
11    if err := conn.ReadJSON(&info); err != nil {  
12        //...  
13    }
```

coder/websocket

- WASM support
- O acesso de rede em WebAssembly é restrito
 - Erro de DNS, dial, etc.
- Implementa build tags para detectar quando está rodando no browser e utiliza APIs específicas.



Goroutines (latência)

```
● ● ●  
1 func sendPingMessages(player *game.Network) {  
2     for {  
3         select {  
4             case <-player.Ctx.Done():  
5                 // Cleanup ...  
6                 return  
7  
8             case <-time.After(5 * time.Second):  
9                 player.Ping()  
10            }  
11        }  
12    }  
13  
14 go sendPingMessage(...)
```

Goroutines (input)

```
● ● ●
1 for {
2     select {
3         case <-player.Network.Ctx.Done():
4             return
5         default:
6             var input PlayerInput
7             if err := player.Network.Conn.ReadJSON(&input); err != nil {
8                 slog.Error("Error reading player input", slog.Any("error", err))
9                 player.Terminate()
10            return
11        }
12    }
13
14    if !input.Up && !input.Down {
15        continue
16    }
17
18    slog.Info("Received input", slog.Any("input", input))
19
20    player.inputQueue <- input
21}
22}
```

User Network Context

```
● ● ●  
1 func sendPingMessages(player *game.Network) {  
2     for {  
3         select {  
4             case <-player.Ctx.Done():  
5                 // Cleanup ...  
6                 return  
7             case <-time.After(5 * time.Second):  
8                 player.Ping()  
9             }  
10        }  
11    }  
12 }  
13  
14 go sendPingMessage(...)
```

Game loop

- Ticker (60hz)

```
● ● ●

1 func (session *GameSession) Start() {
2     session.ticker = time.NewTicker(time.Second / 60)
3     defer session.ticker.Stop()
4
5     session.ready()
6
7     for {
8         select {
9             case <-session.Player1.Network.Ctx.Done():
10                 session.handleDisconnection(session.Player1)
11                 return
12             case <-session.Player2.Network.Ctx.Done():
13                 session.handleDisconnection(session.Player2)
14             case <-session.ticker.C:
15                 session.update()
16
17                 session.broadcastGameState()
18
19                 session.broadcastToSpectators(
20                     session.currentGameState()
21                 )
22
23                 if session.gameEnded() {
24                     session.ticker.Stop()
25                     session.endGame()
26                 }
27             }
28         }
29 }
```

Frame update



```
1 func (session *GameSession) update() {
2     session.Player1.ProcessInputs()
3     session.Player2.ProcessInputs()
4
5     session.ball.Update(
6         session.Player1.basePlayer.Bounds(),
7         session.Player2.basePlayer.Bounds()
8     )
9
10    if scored, goalSide := session.ball.CheckGoal(); scored {
11        session.handleScore(goalSide)
12    }
13 }
```

Read Input



```
1 func (p *Player) ProcessInputs() {
2     for {
3         select {
4             case input := <-p.inputQueue:
5                 p.basePlayer.Update(player.Input{
6                     Up:    input.Up,
7                     Down: input.Down,
8                 })
9             default:
10                return
11            }
12        }
13 }
```

State Broadcast



```
1 case <-session.ticker.C:  
2     session.update()  
3  
4 session.broadcastGameState()
```

Game State

```
● ● ●  
1 type GameState struct {  
2     Ball     BallState `json:"ball"  
3     Current  PlayerState `json:"current"  
4     Opponent PlayerState `json:"opponent"  
5 }  
6  
7 type BallState struct {  
8     Angle    float64      `json:"angle"  
9     Bounces   int         `json:"bounces"  
10    Position  geometry.Vector `json:"position"  
11 }  
12  
13 type PlayerState struct {  
14     Name      string      `json:"name"  
15     PositionY float64     `json:"position_y"  
16     Side      geometry.Side `json:"side"  
17     Score     int8        `json:"score"  
18     Ping      int64       `json:"ping"  
19     Winner    bool        `json:"winner,omitempty"  
20 }
```

Live streaming



```
1 session.broadcastToSpectators(session.currentGameState())
```

Spectator (Live streaming)

```
● ● ●  
1 type GameSession struct {  
2     ID      string  
3     Player1 *Player  
4     Player2 *Player  
5     ball    ball.Ball  
6     level   level.Level  
7     ticker  *time.Ticker  
8  
9     // Spectate  
10    spectators []Network  
11    spectatorMutex sync.Mutex  
12 }
```

Melhorias

- + Command Pattern (permite rebind)
- + Streaming das sessões
- + Replay das sessões
- + Utilizar custom UDP ao invés de TCP
- + Melhorias de performance
 - channels vs Iterators
 - generics vs interface
- + Não enviar mudança de estado quando não houver necessidade
- + Tocar som ao interagir

PONGO

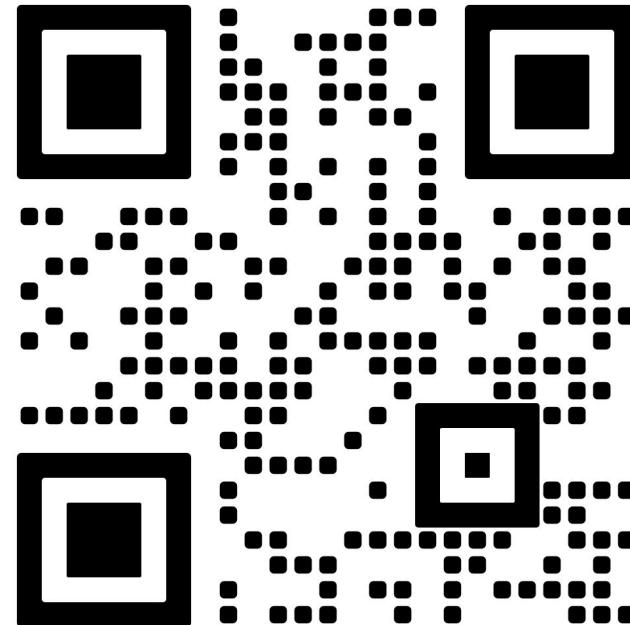


■ One Player

Two Players

Back

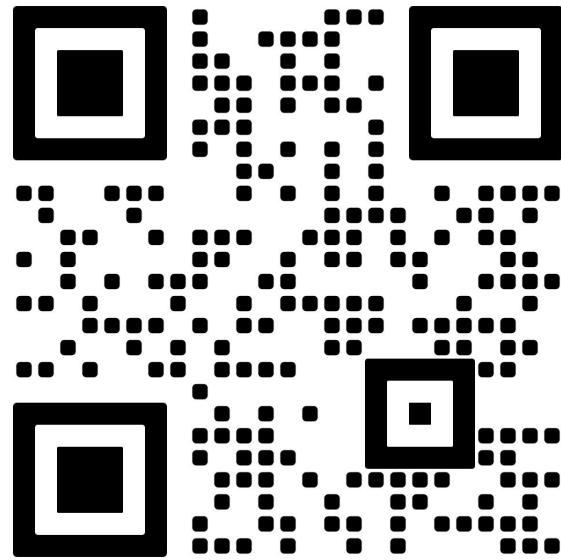
Demo!!



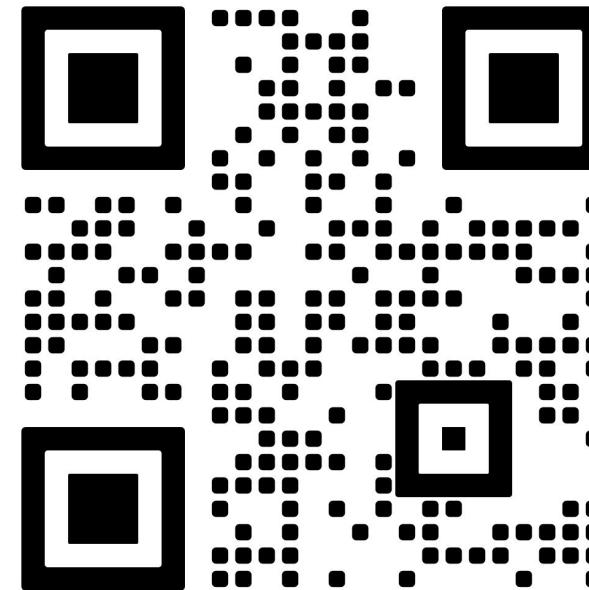
pong.go-go.dev

It's only a demo sometime shit happens

Repositórios



pong-multiplayer-go
(game)

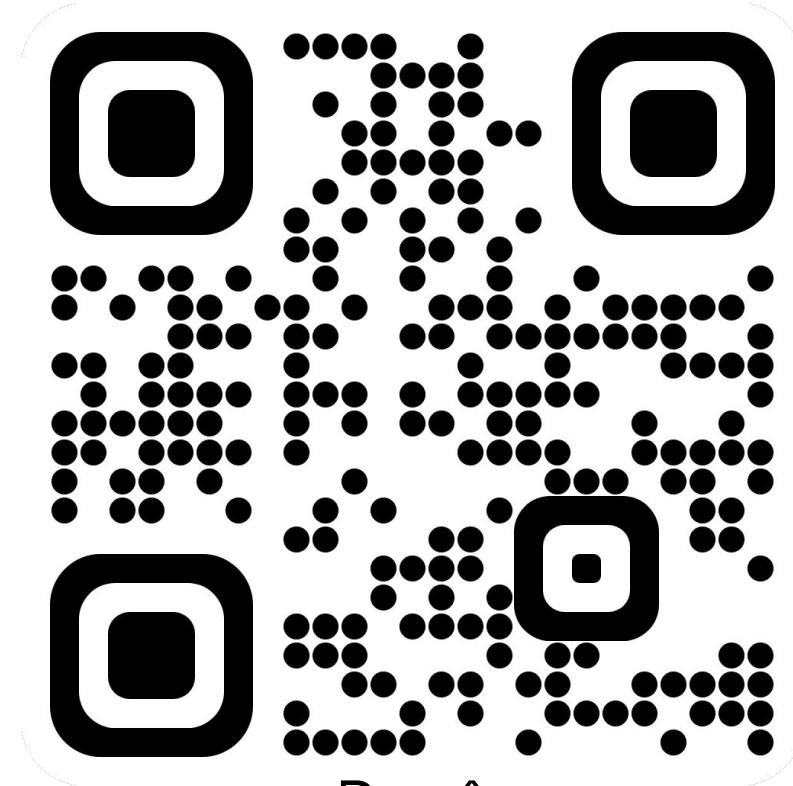


pongo-server
(server)

Obrigado



Carlos



Renê