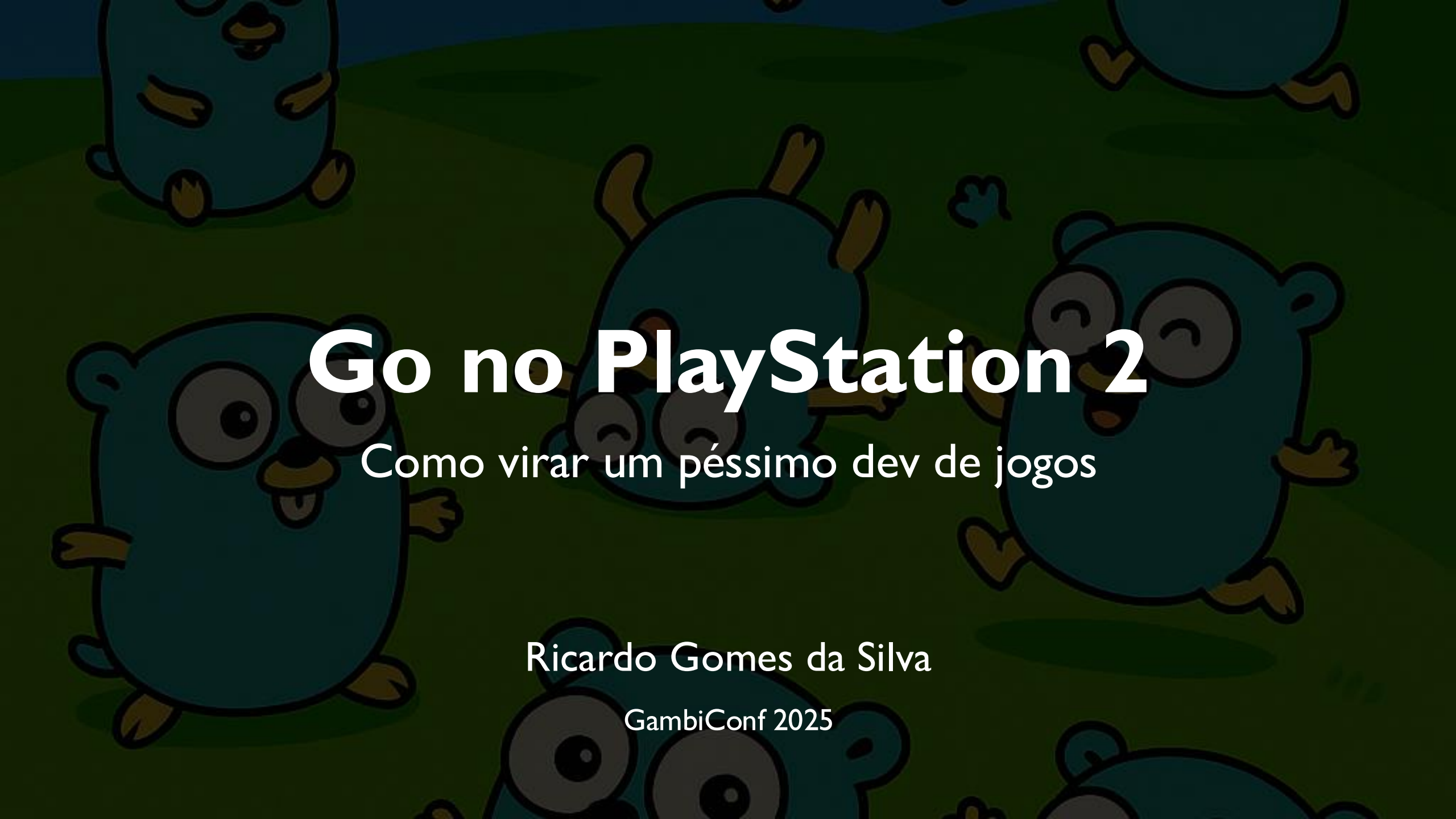


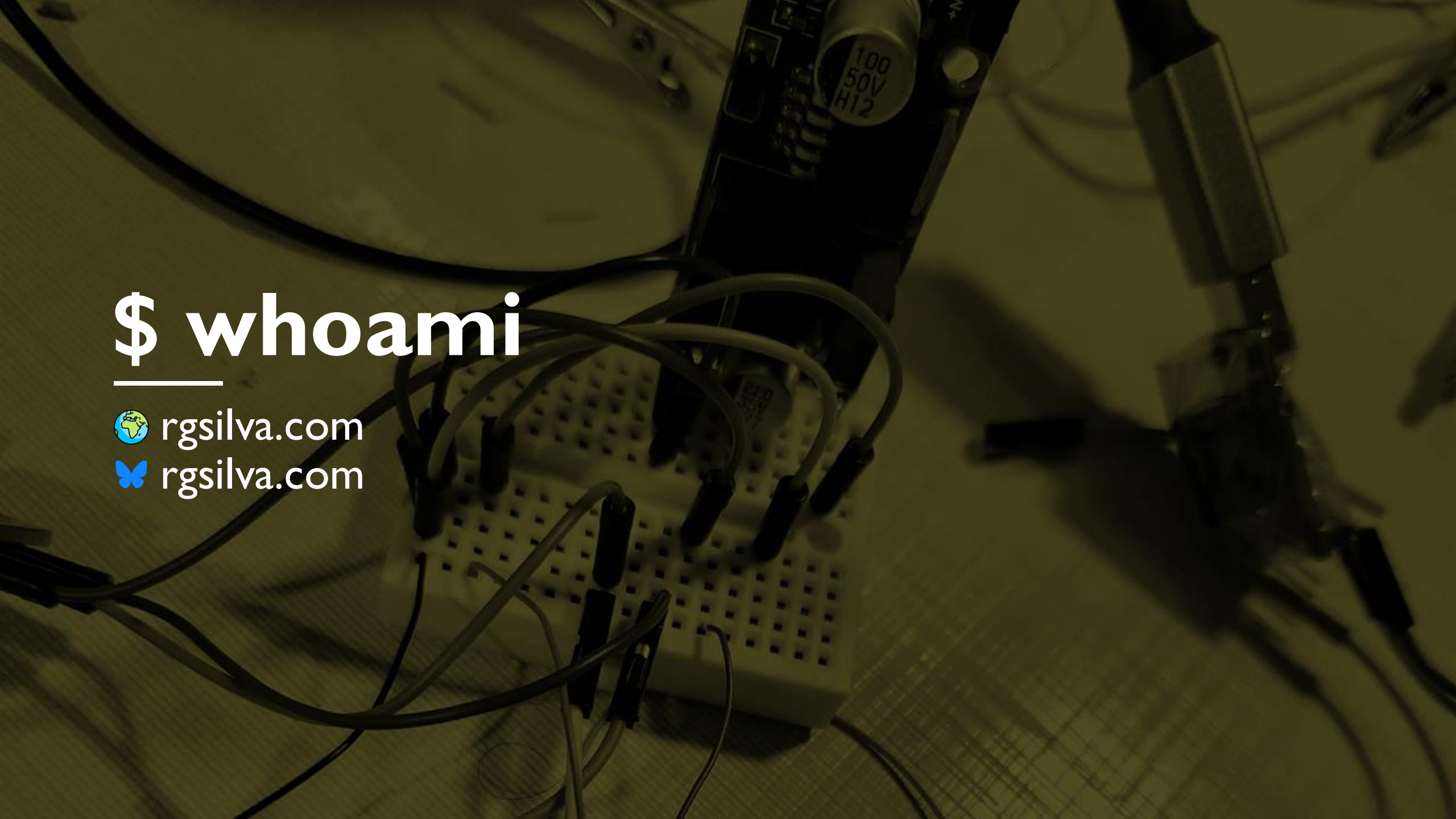
Go no PlayStation 2

Como virar um péssimo dev de jogos

Ricardo Gomes da Silva

GambiConf 2025


"MAS O PRINTF FUNCIONA?"
imgflip.com



$ whoami

🌍 rgsilva.com
🦋 rgsilva.com

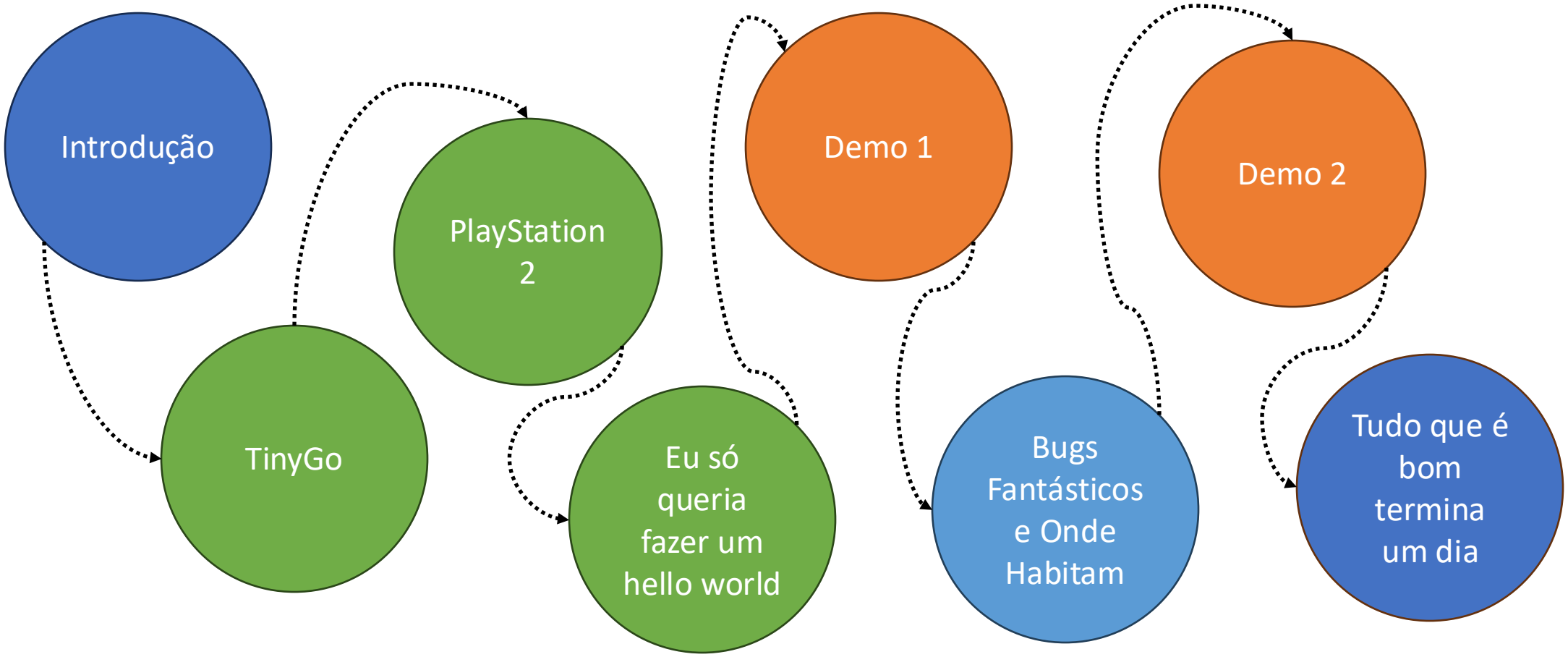
Disclaimer

Não sou expert no PlayStation 2, TinyGo ou mesmo Go.
Sequer sou desenvolvedor de jogos!

Tudo que será mostrado aqui foi feito pela diversão, meme e/ou matar o tédio.

Nenhum console foi danificado nesta talk (ainda).

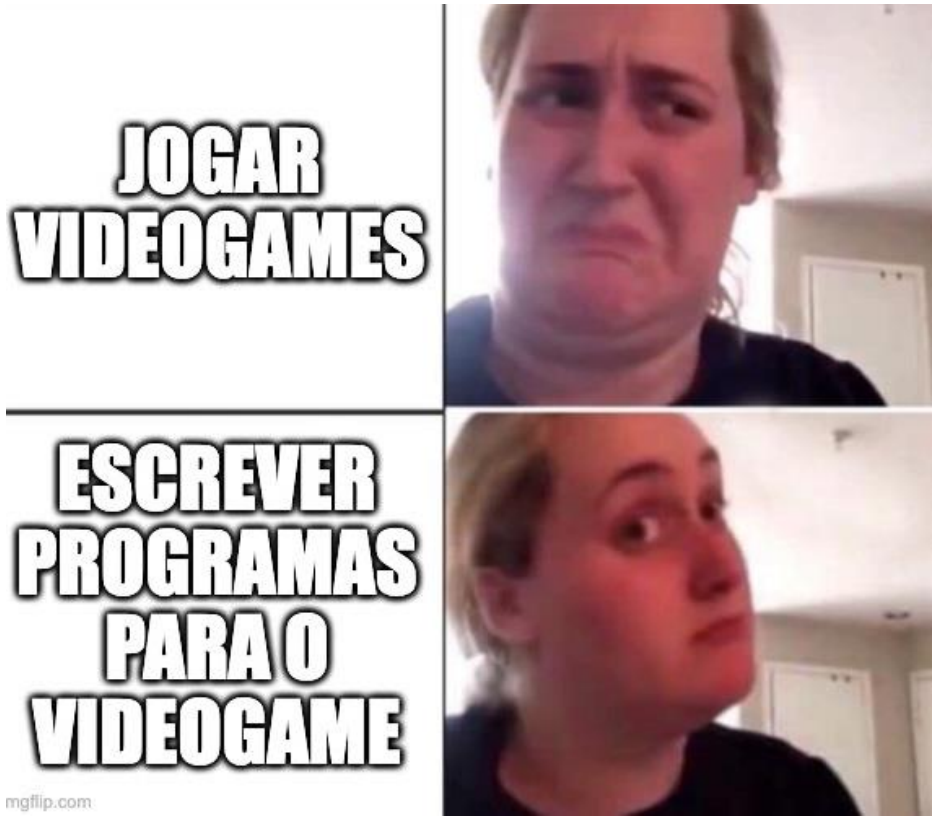
Programação para hoje



Mas antes, uma enquete!

Jogar ou hackear, eis a questão!

- Jogar é divertido
  - ... mas hackear o console é mais!
- Videogames vai além de jogos
  - O hardware é muito mais capaz do que as pessoas imaginam!
- E porque não fazer isto no PlayStation 2?



Objetivos

1. Escrever um programa em Go que rode no PlayStation 2
    1. Não utilizar Linux no PS2 sob hipótese alguma – queremos baremetal!
2. Permitir que outros desenvolvam programas também
3. Utilizar o ~~maior~~ menor número de gambiarras possíveis
4. Fama e glamour!

TinyGo

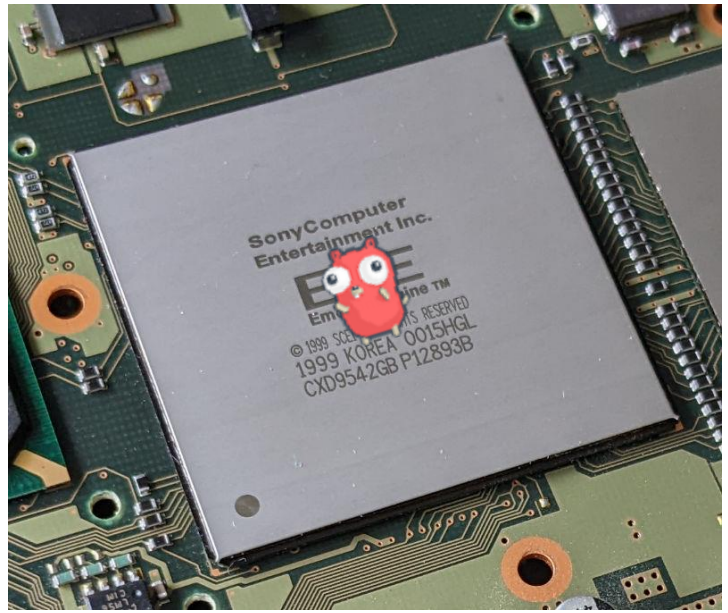
O primeiro passo (de muitos)

Pera, Go já não roda em tudo?

- Go roda em muitas plataformas
  - Linux, Windows, Mac, BSD...
- E em muitas arquiteturas
  - Intel, ARM, MIPS, PowerPC...

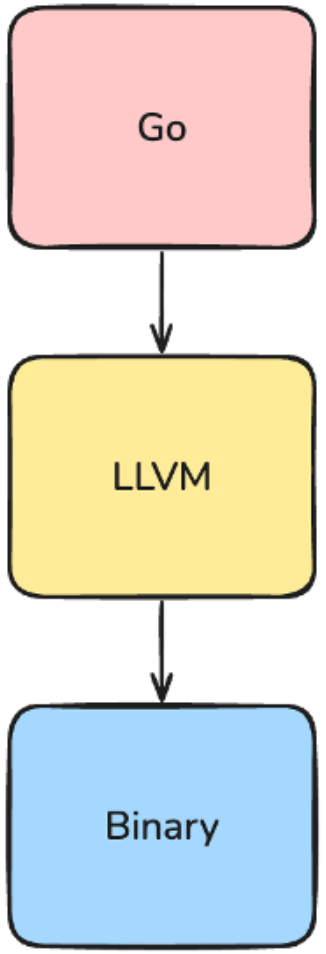
Porém queremos mais – baremetal!

- Baremetal não tem um sistema operacional
  - Microcontroladores!
- Tudo roda *diretamente* no hardware
  - Não existem syscalls, não existem libs pra chamar, threads, malloc, nada.
- Temos total controle de *tudo*
  - Porém temos que fazer *tudo* na mão também!



Eis que surge o TinyGo

- Compilador Go para sistemas embarcados
  - Suporta *basicamente* tudo* que o Go suporta
  - Código gerado não depende de um OS (baremetal!) 
- Suporta diversas plataformas (até mesmo o Gameboy!)
  - Utiliza a LLVM para fazer a mágica acontecer!
  - Porém não o PlayStation 2 – *ainda*!



O PlayStation 2

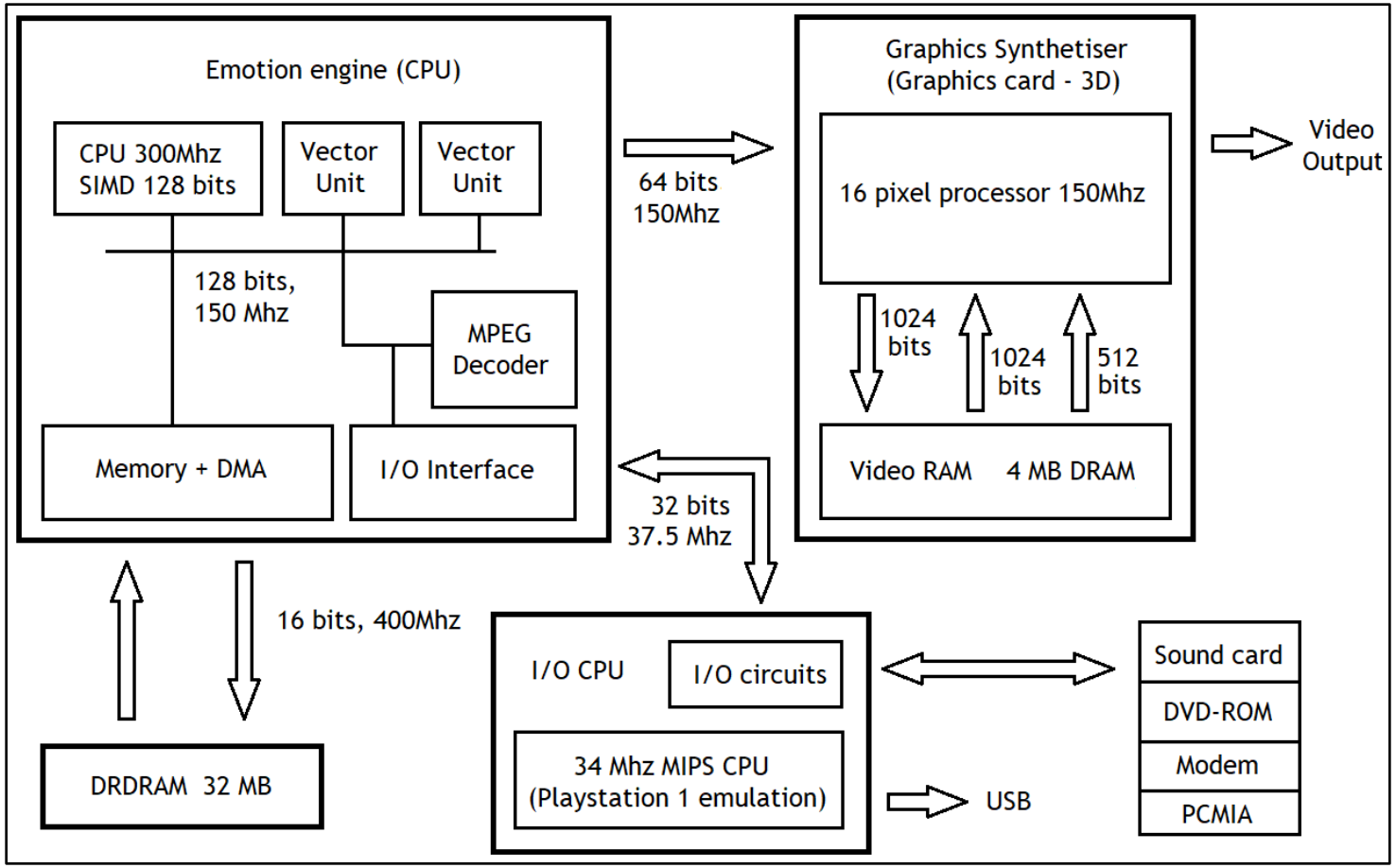
Ou como a Sony fez eu pirar por uns meses

que eu inclusive roubei da minha irmã!

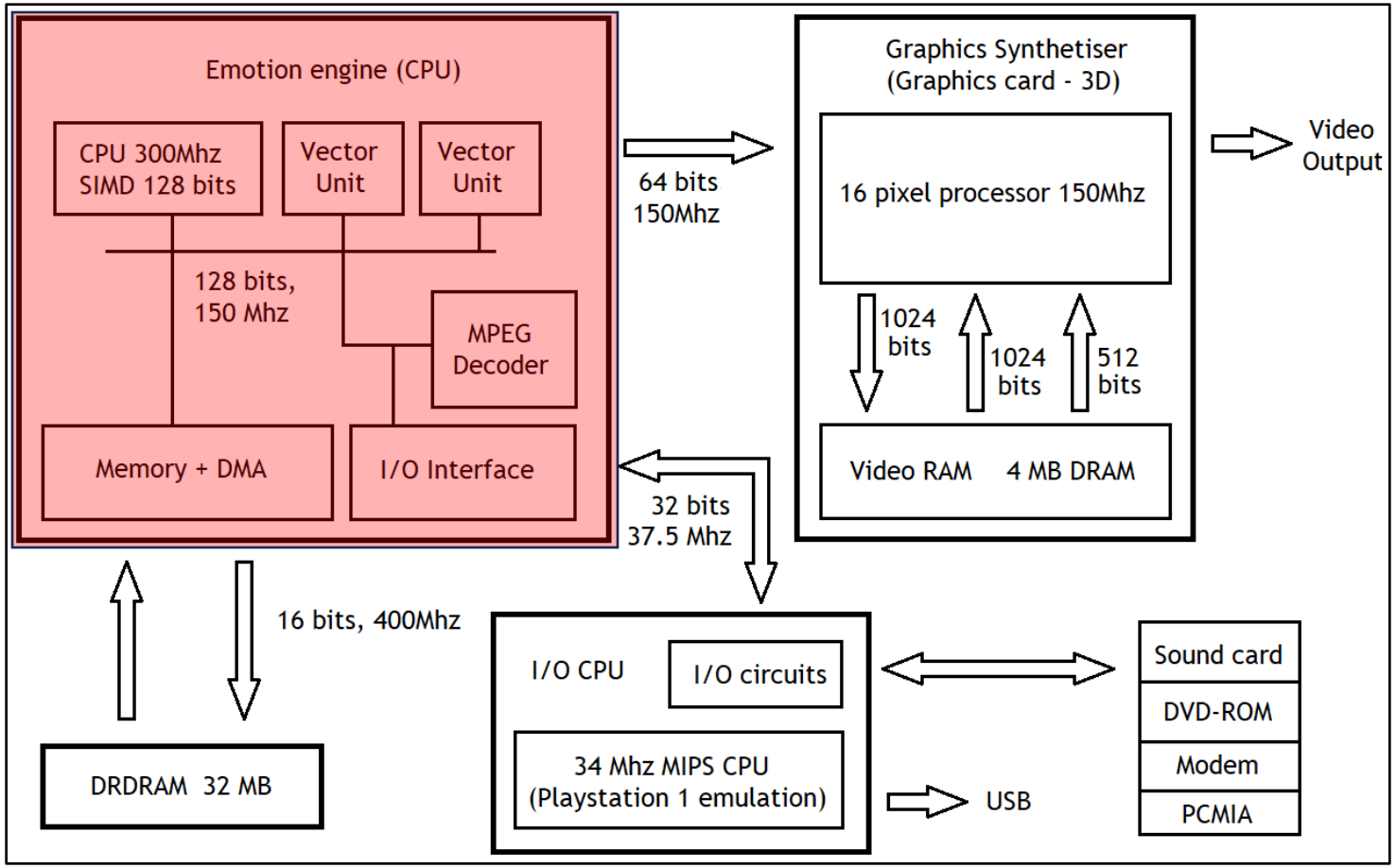
O PlayStation 2

- Lançado no ano 2000
  - Época do Windows Millenium, The Sims e Gladiador!
- Vendeu 160+ *milhões* de unidades
  - Ainda é o console mais vendido no mundo
- Comunidade segue firme e forte

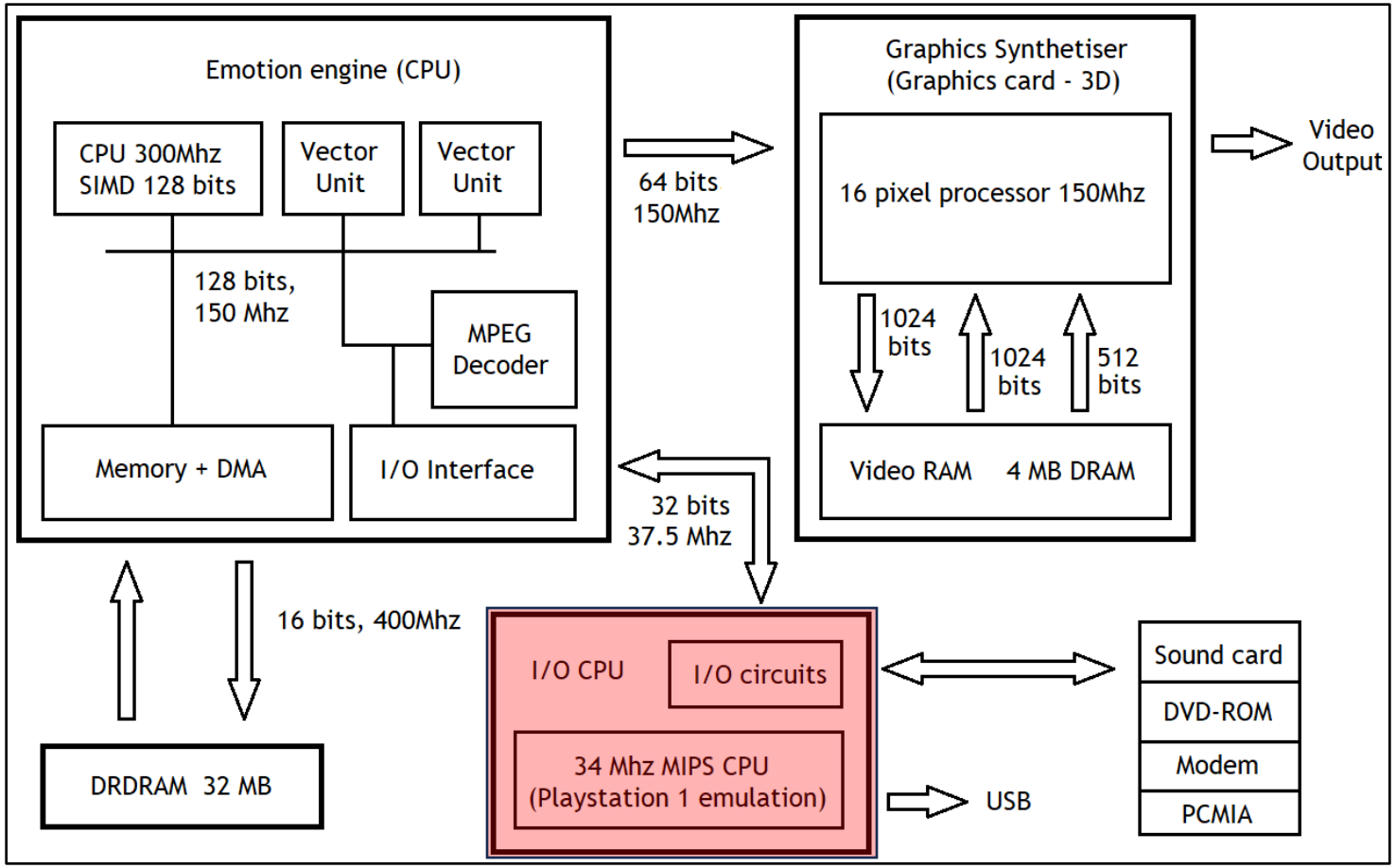
Arquitetura do PlayStation 2



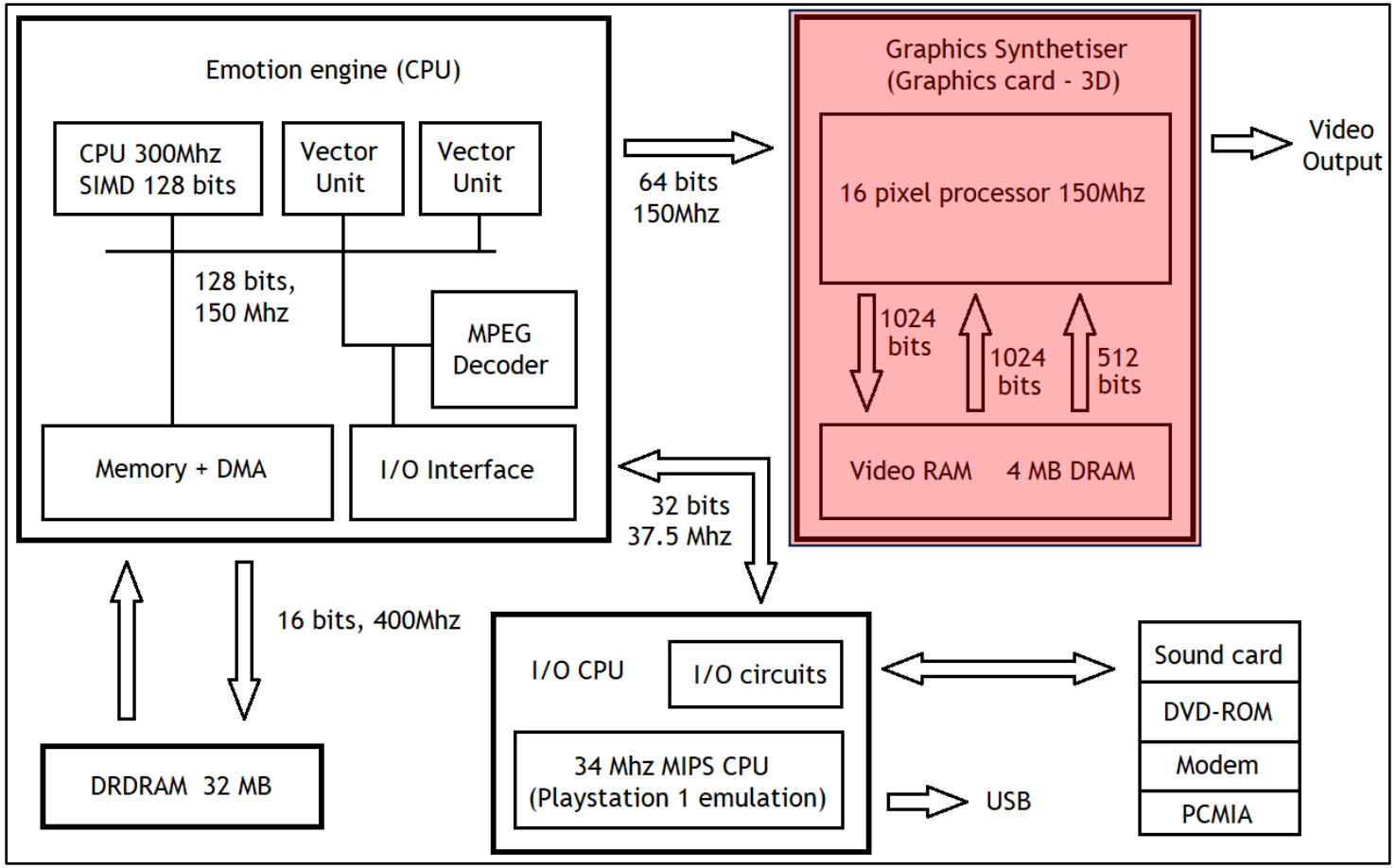
Arquitetura do PlayStation 2



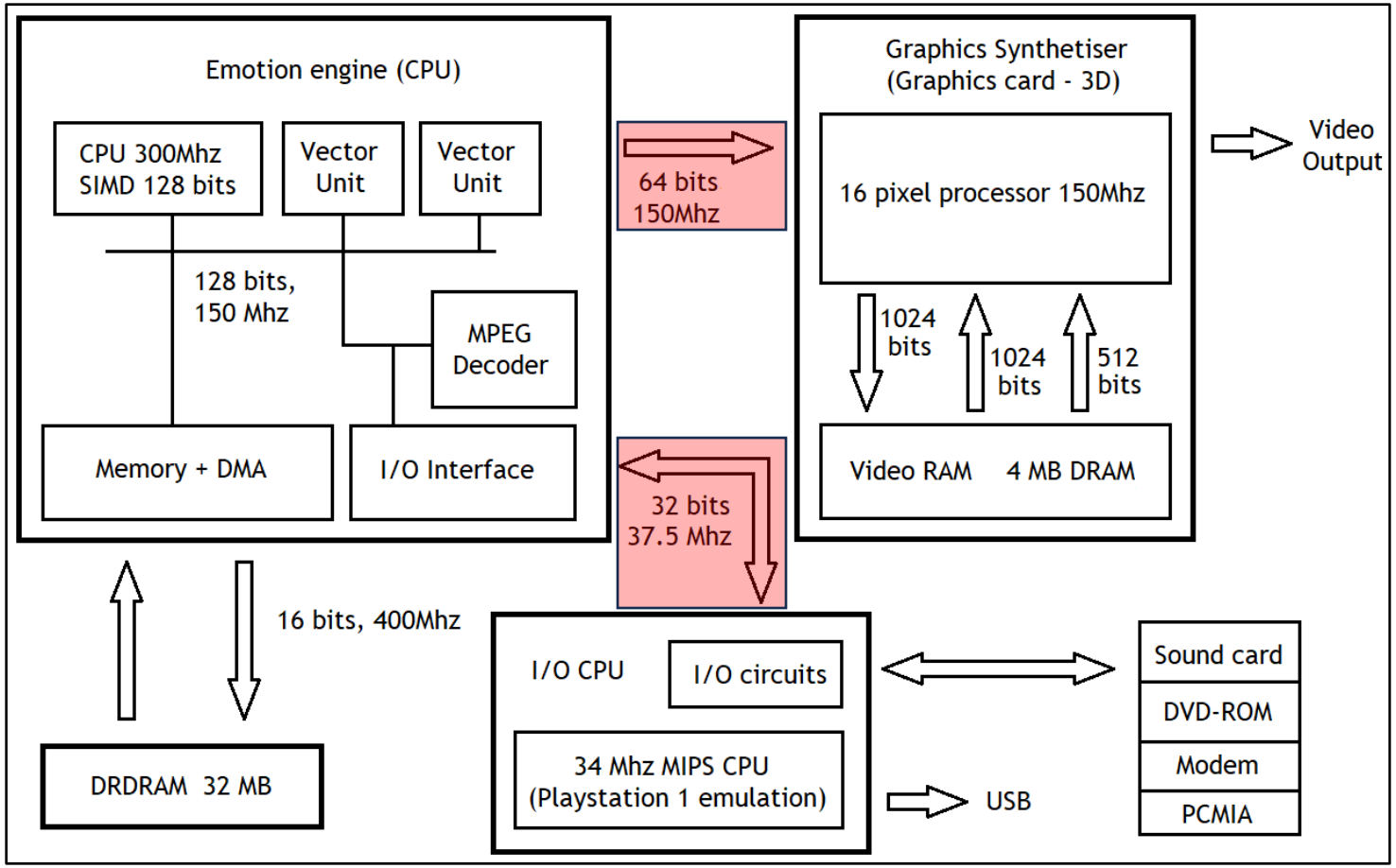
Arquitetura do PlayStation 2



Arquitetura do PlayStation 2



Arquitetura do PlayStation 2



PS2SDK – um kit completo

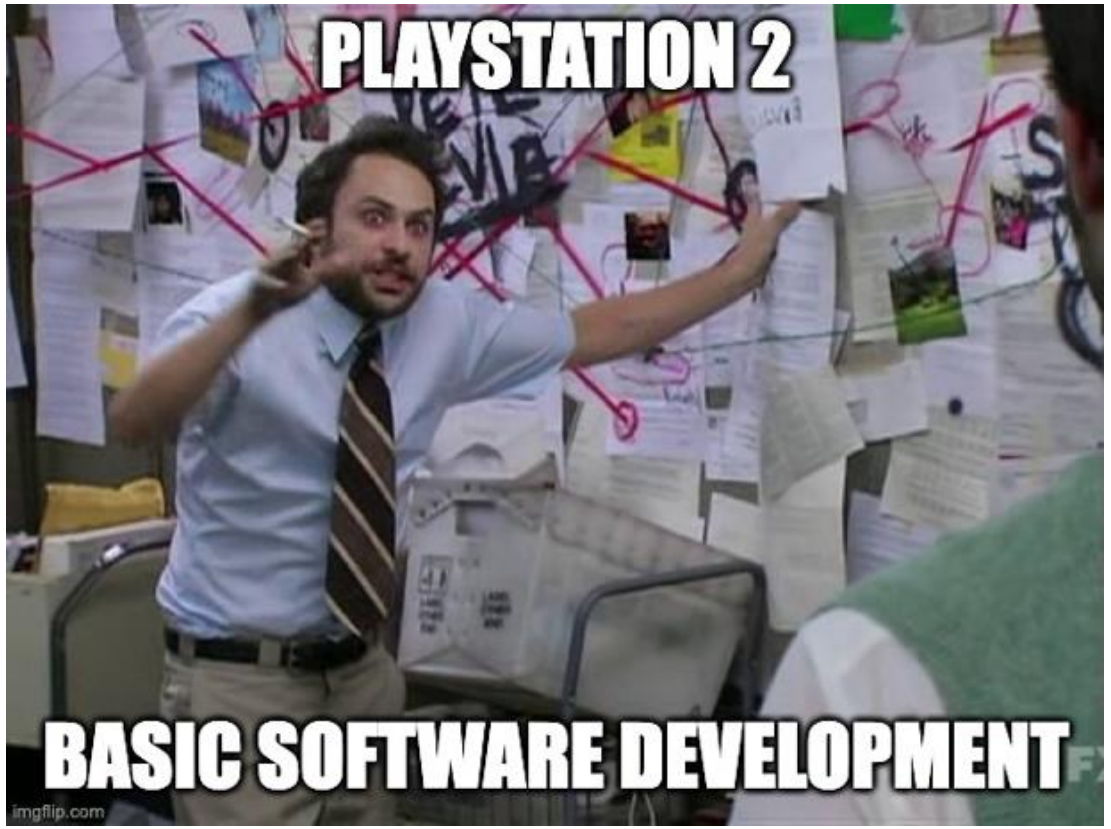
- Conjunto de bibliotecas para desenvolvimento de jogos e aplicações para o PS2
  - Escrito em C e assembly e totalmente opensource
  - Suporte a *tudo*, basicamente: CD/DVD, Memory Card, USB, Ethernet, HD, controle, som, POSIX, e por aí vai!

- E se a gente usasse isso de alguma forma? 


Hello world?
Que comecem as gambiarras!

O PlayStation 2 é meio... complexo

- Tudo é baixíssimo nível
  - Assembly
  - DMA, SIF RPC
  - Interrupts
- Go não foi feito pra isso
  - Go é bonito, é cheiroso, é legal!
  - Go é fofinho!



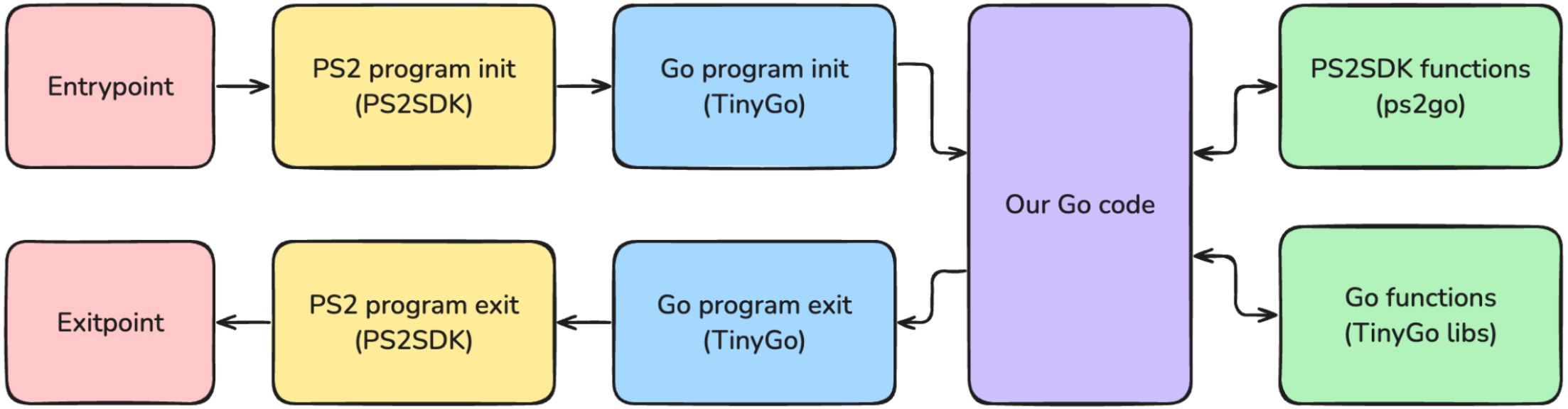
Uma combinação inesperada

- Uma pitada de PS2SDK para toda a parte complicada
  - Um monte de funções em C, estruturas, ponteiros e um lindo caos.
- Uma porção de Go para toda a parte divertida
  - Lógica do programa, bibliotecas modernas, programação em alto nível
- Ódio e sal do Himalaia a gosto!

Cgo – uma gambiarra oficial

** altamente experimental no TinyGo (yey!)*

Um ~~sonho~~ plano



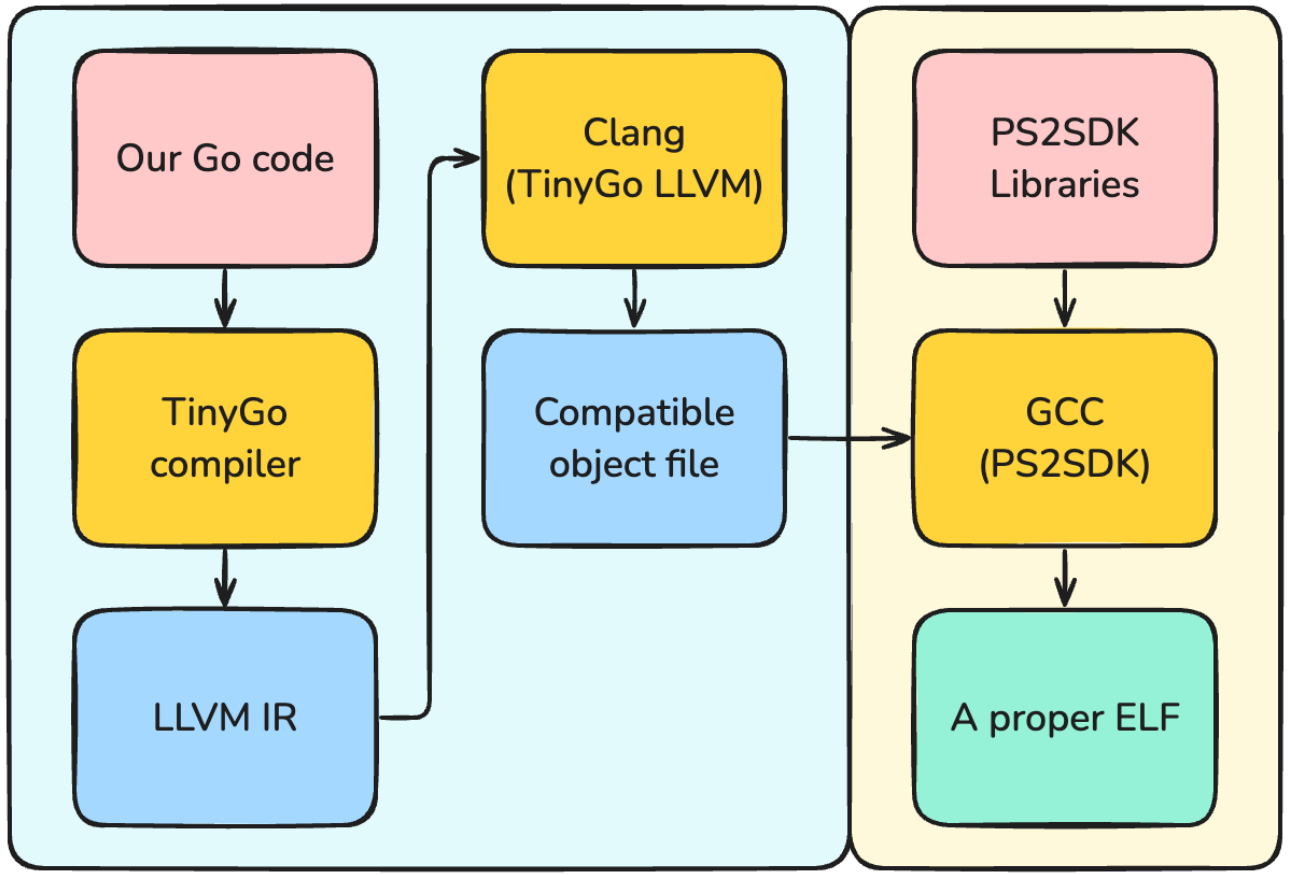
$ make helloworld

Será que agora vai?

Fácil assim?

Uma saga de bugs e gambiarras

Nem compilar é fácil



Brilho eterno de um PS2 sem memória

» A luta do ano:
  » De um lado, Go com sua heap!
  » Do outro, PS2SDK com ... alguma coisa?

» O prêmio:
  » Acesso total e irrestrito aos 32MB do EE!

» *Fight!*

Brilho eterno de um PS2 sem memória

```
const memSize = uint(24 * 1024 * 1024)

func preinit() {
  goMemoryAddr = uintptr(unsafe.Pointer(
    C.malloc(C.uint(memSize))))
  heapStart = goMemoryAddr
  heapEnd = goMemoryAddr + uintptr(memSize)
}

func preexit() {
  C.free(unsafe.Pointer(heapStart))
}
```

```
//export main
func main() {
  preinit()
  run()
  preexit()
  exit(0)
}
```

Conte até 10

» O que eu esperava:

```
-32, -31, -30, -29, -28, -27, -26, -25, -24, -23, -22, -21, -20, -19, -18, -17,
-16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 00, 01, 0
2, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
2, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32,
```

» O que eu fiz:

```
-0P, -8, -8, -8, -8, -8, -8, -8, -8, -8, -8, -8, -8, -8, -8, -8, -8, -8, -8,
-8, -8, -8, -8, -8, -8, -8, -9, -8, -7, -6, -5, -4, -3, -2, -1, 00, 01, 0
2, 03, 04, 05, 06, 07, 08, 09, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8
, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8
```

Sony e sua divina genialidade

- Na hora de formatar, a gente usa `int` aos montes
  - Em Go, `int` por padrão é `int64`

- Porém faltam instruções do MIPS III!
  - Quem precisa multiplicar e dividir inteiros 64bits né?
- E ainda por cima a FPU é estranha
  - Segue uma variação do padrão IEEE 754
  - Só aceita 32bits e não suporta NaN e infinito?!

Software to the rescue!

```
setOperationAction(ISD::MUL, MVT::i64, LibCall);
setOperationAction(ISD::SDIV, MVT::i64, LibCall);
setOperationAction(ISD::UDIV, MVT::i64, LibCall);
setOperationAction(ISD::SREM, MVT::i64, LibCall);
setOperationAction(ISD::UREM, MVT::i64, LibCall);
setOperationAction(ISD::SDIVREM, MVT::i64, LibCall);
setOperationAction(ISD::UDIVREM, MVT::i64, LibCall);
setOperationAction(ISD::SMUL_LOHI, MVT::i64, LibCall);
setOperationAction(ISD::UMUL_LOHI, MVT::i64, LibCall);
setOperationAction(ISD::MULHS, MVT::i64, LibCall);
setOperationAction(ISD::MULHU, MVT::i64, LibCall);
```

Essa nem eu entendi ¯_(ツ)_/¯

```
case "mips":
// ...
  if !strings.Contains(b.Features, "+soft-float")
     && !strings.Contains(b.Features, "+single-float") {
       constraints += ",~{$f0},~{$f1},~{$f2}..."
  }
// ...
```

Cgo é experimental

- TinyGo suporta Cgo
  - Porém é experimental
  - E algumas features podem estar faltando
- Mas nada que um pequeno "ajuste" no código não resolva!

```
cGoCFlags := os.Getenv("CGO_CFLAGS")
if cGoCFlags != "" {
    flags, err := shlex.Split(cGoCFlags)
    if err != nil {
        panic(err.Error())
    }
    p.makePathsAbsolute(flags)

    p.cflags = append(p.cflags, flags...)
}
```

A saga dos compiladores

- Os headers do PS2SDK são compartilhados:
  - Nosso código usa para saber as funções e estruturas disponíveis (via Cgo)
  - O PS2SDK usa para sua compilação
- Porém nem todos os compiladores são iguais:
  - Nós usamos o Clang da LLVM do TinyGo
  - PS2SDK usa o GCC
- Qual a chance disso dar ruim?

Mais um pequeno “ajuste”

```c
// libpad.h

struct padButtonStatus {
  unsigned char ok;
  unsigned char mode;
  unsigned short btns;
  // ...
} __attribute__((packed));
```

```go
// libpad/wrappers.go

package libpad

/*
// __attribute__ doesn't play nice with clang
#define __attribute__(x)

#include <libpad.h>
*/
import "C"
```

Execute agora ou cale-se para sempre!

- O `defer` permite jogar uma execução para o final da função (quando ela retornar)

- Porém não funciona no PS2 ainda... exceto no emulador?!

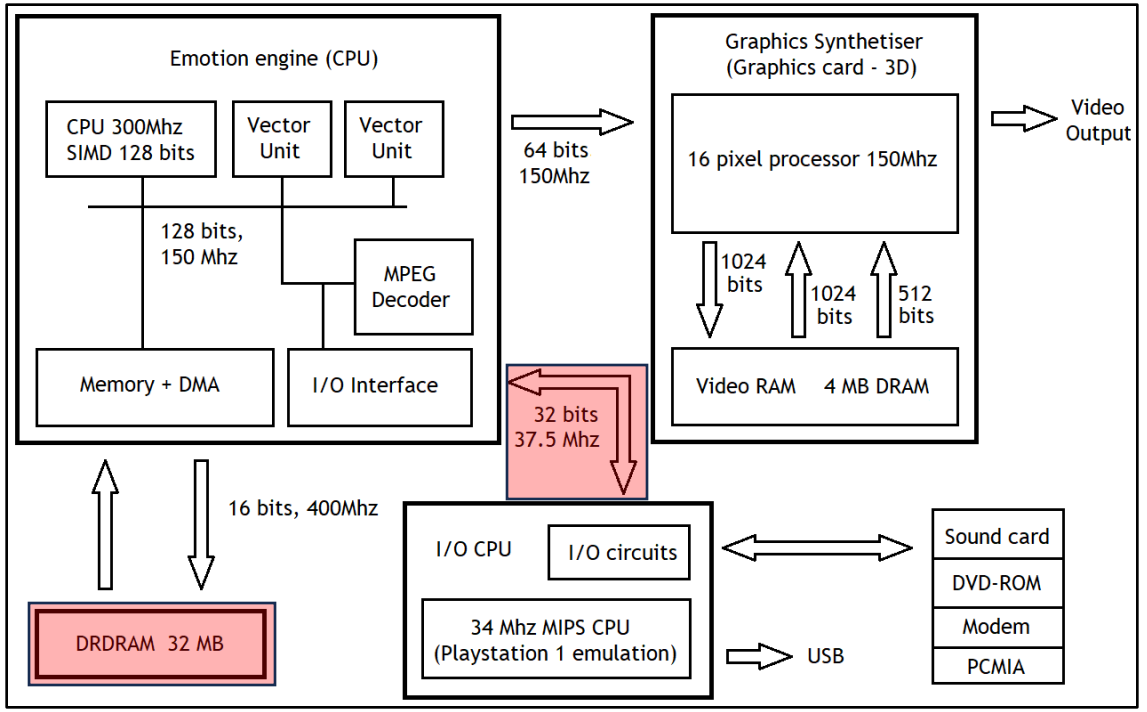
```
func Printf(format string, args ...interface{}) {
    formatted := fmt.Sprintf(format, args...)
    str := C.CString(formatted)
    defer C.free(unsafe.Pointer(str))
    C.scr_printf(str)
}
```

```
func Printf(format string, args ...interface{}) {
    formatted := fmt.Sprintf(format, args...)
    str := C.CString(formatted)
    C.scr_printf(str)
    C.free(unsafe.Pointer(str))
}
```

A Sony também erra

» Carregar código no IOP via EE requer uma transferência de memória entre as CPUs

» Porém a Sony esqueceu de implementar a função pra isso



Gambiarra oficial da Sony

```c
static int resetAndPatchIOP() {
    SifInitRpc(0);
    while(!SifIopReset("", 0)){};
    while(!SifIopSync()){};
    SifInitRpc(0);

    int ret = sbv_patch_enable_lmb();
    if (ret != 0) {
        return ret;
    }
    return sbv_patch_disable_prefix_check();
}
```

```go
func ResetAndPatchIOP() {
  C.resetAndPatchIOP()
}

func main() {
  // Fix whatever Sony broke
  sifrpc.ResetAndPatchIOP()

  // Make the magic happen!
}
```

$ make flappygopher

O primeiro jogo em Go para o PS2!

e se os deuses de fazer demo ao vivo permitirem, no console!

Conclusão

Tá quase acabando, prometo!

Um gambiarra de final de semana

- Todo esse projeto surgiu de uma ideia maluca em um final de semana
  - (obviamente levou muito mais que 2 dias para chegar neste ponto)
- Um hobby/meme que tornou uma proporção inesperada
- Até os devs do TinyGo ficaram entusiasmados com isso!

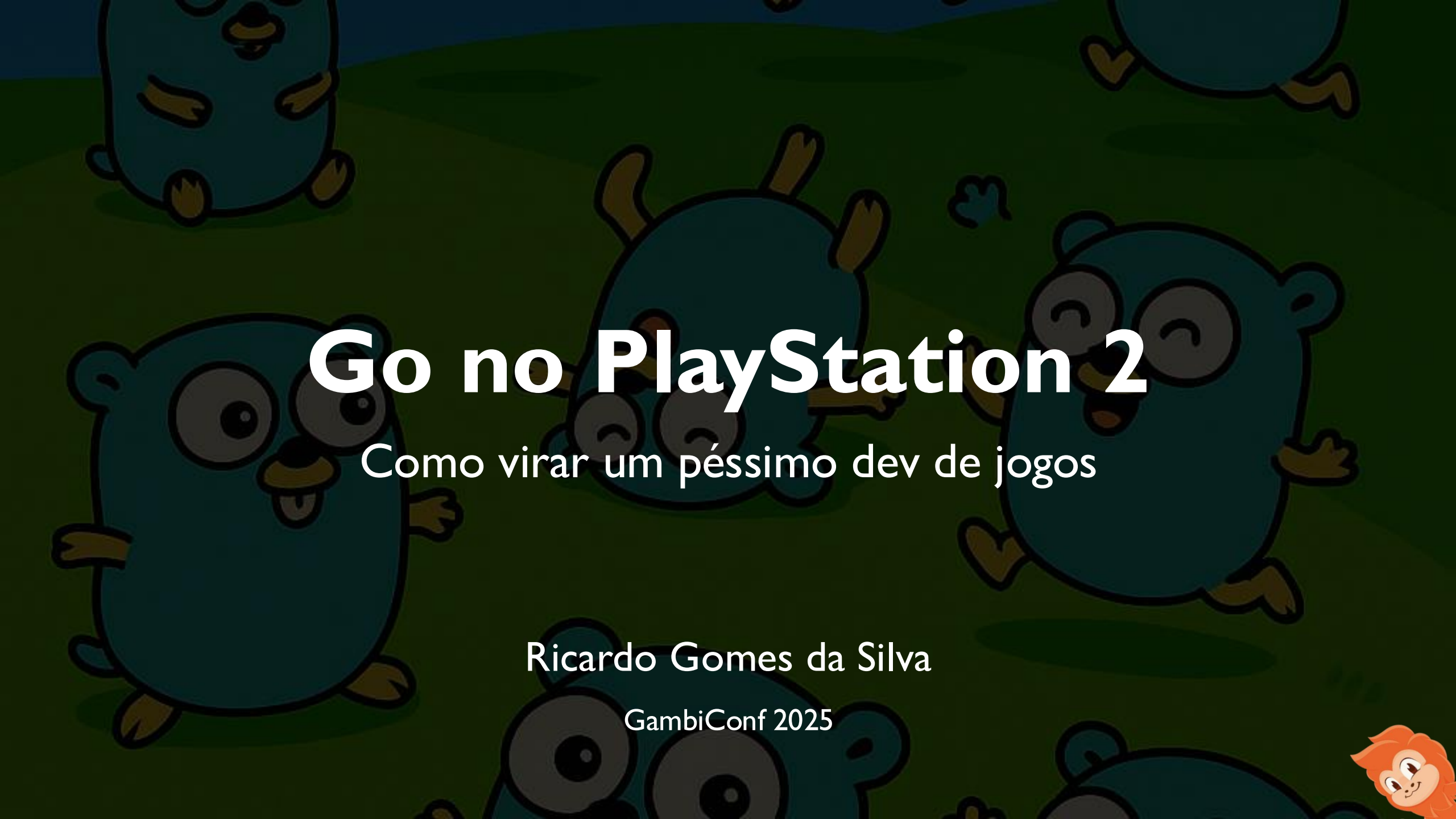
- Porém nada é perfeito...

A lista de bugs ainda é grande

- Não compila o object file diretamente (muito menos o ELF)
- Driver de rede ainda está incompleto
- DNS sempre retorna lixo de memória
- Requests HTTP funcionam até 8 vezes, depois trava o EE ou IOP
- O defer não funciona
- LLVM é toda hackeada pra funcionar e cheia de gambiarras
- Compilador tem gambiarra
- A parte de transparência no GS não quer funcionar direito
- A memória não deveria ser pré-alocada e sim alocada dinamicamente
- O GC não coleta o lixo
- O GC crasha com um ponteiro nulo

... e *por aí vai!*

Mas é open source!



Go no PlayStation 2

Como virar um péssimo dev de jogos

Ricardo Gomes da Silva

GambiConf 2025