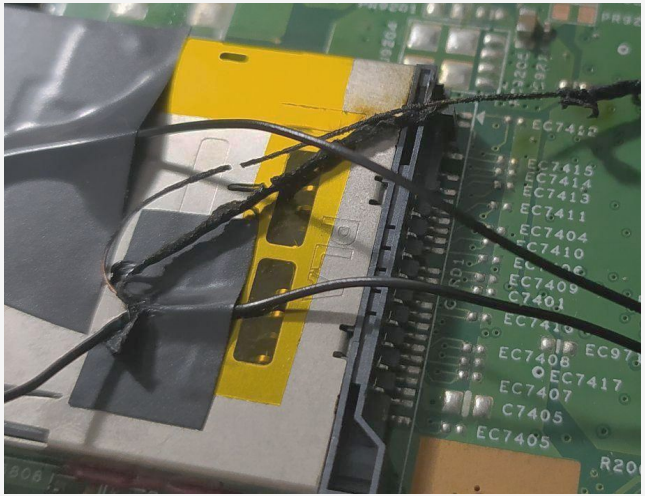
RICARDO GOMES DA SILVA

BECOMING A GAME DEVELOPER 25 YEARS TOO LATE

WHO AM I?



Software engineer during the day, tinkerer at night

I hack stuff for fun and absolutely zero profit

I have too many unfinished projects

"Projects that caught fire" counter: 1

DISCLAIMER

I am no expert in the PlayStation 2, Go and TinyGo.

Everything you see here is was done for fun, and is not related to any previous, current and possibly future employers.

No consoles were harmed in the making of this talk.

and I used ChatGPT a lot

BEFORE WE BEGIN

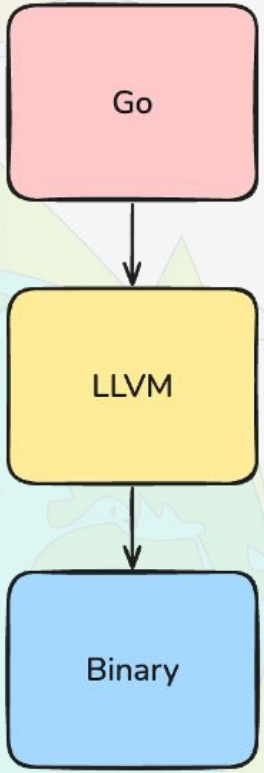
CONCEPTS

Things will only get weirder from here, so buckle up!

WHAT IS TINYGO?

“TinyGo brings the Go programming language to embedded systems and to the modern web by creating a new compiler based on LLVM.”

“The LLVM Project is a collection of modular and reusable compiler and toolchain technologies.”



TINYGO CAN RUN BAREMETAL

TinyGo doesn’t need an OS, but then you have to manage yourself:

Hardware initialization, memory allocation, threads, interrupts, sleep, clock, network, USB, everything.

The sky is the limit, as long as you build a ladder.

THE PLAYSTATION 2

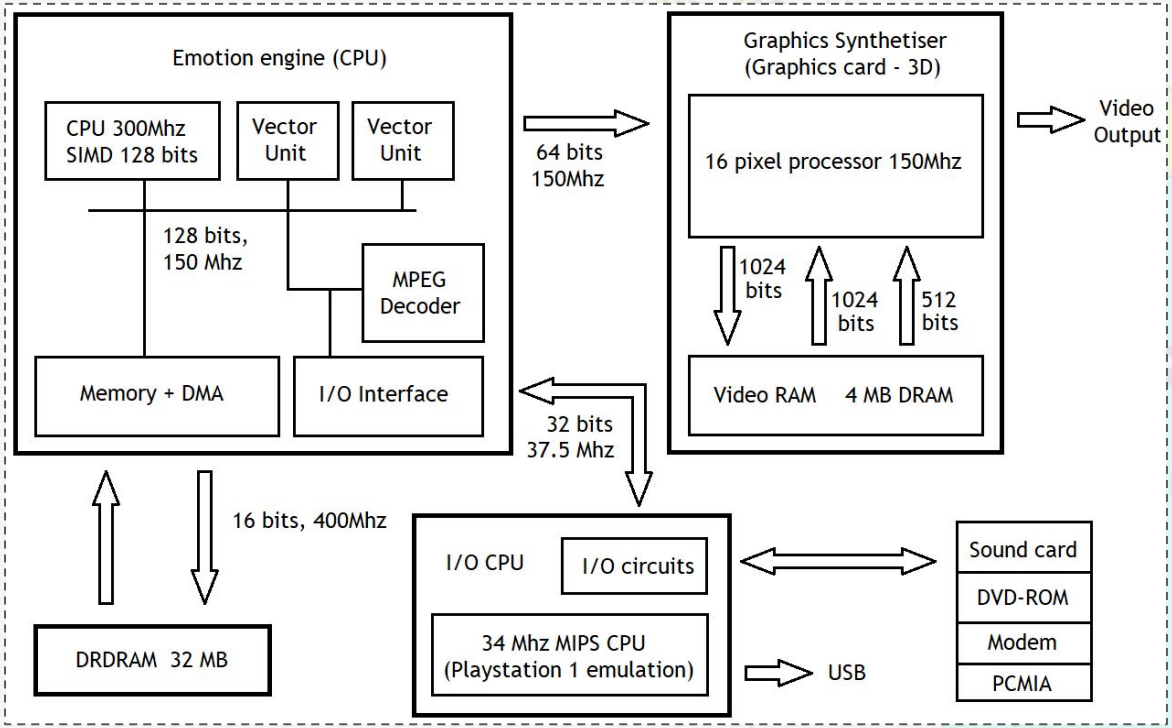
Released by Sony in 2000

160+ million units sold worldwide

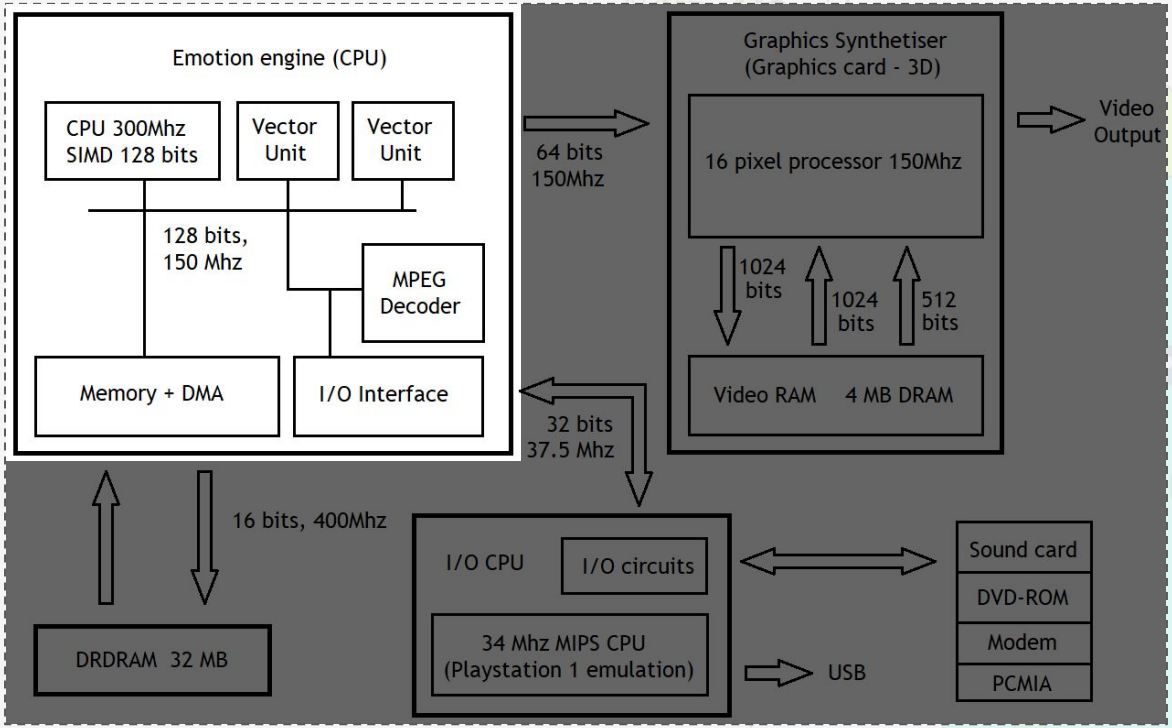
Best selling console (up this day?)

Community is still active

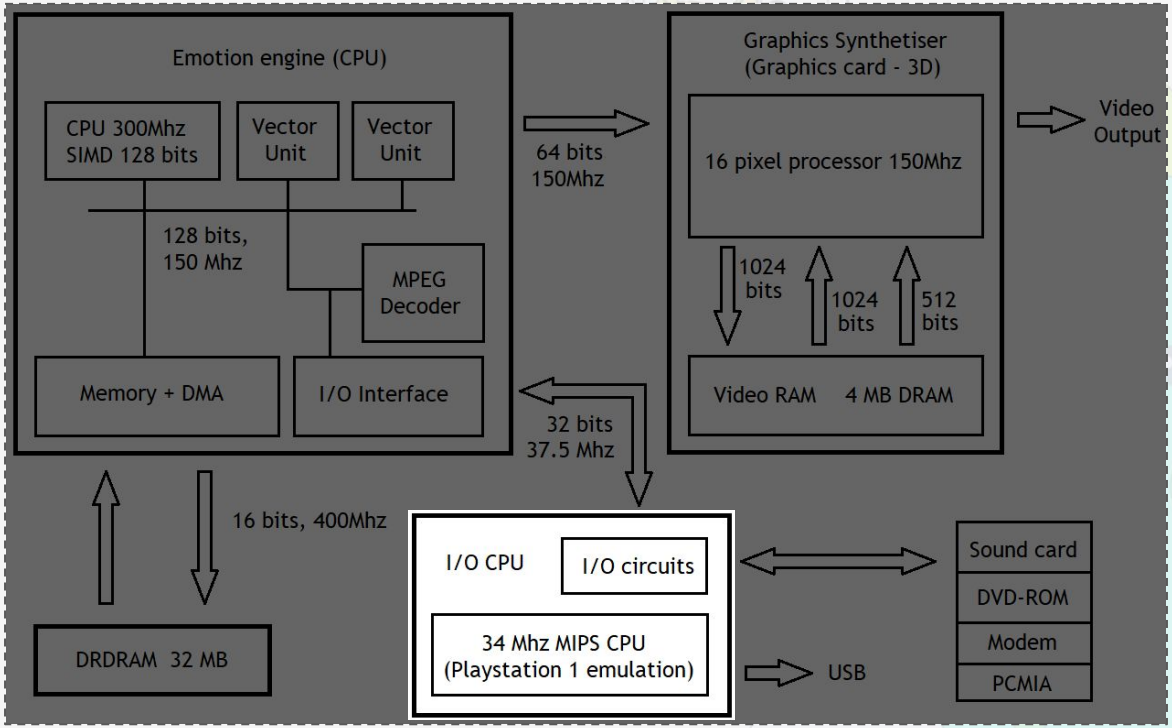
THE PLAYSTATION 2 ARCHITECTURE



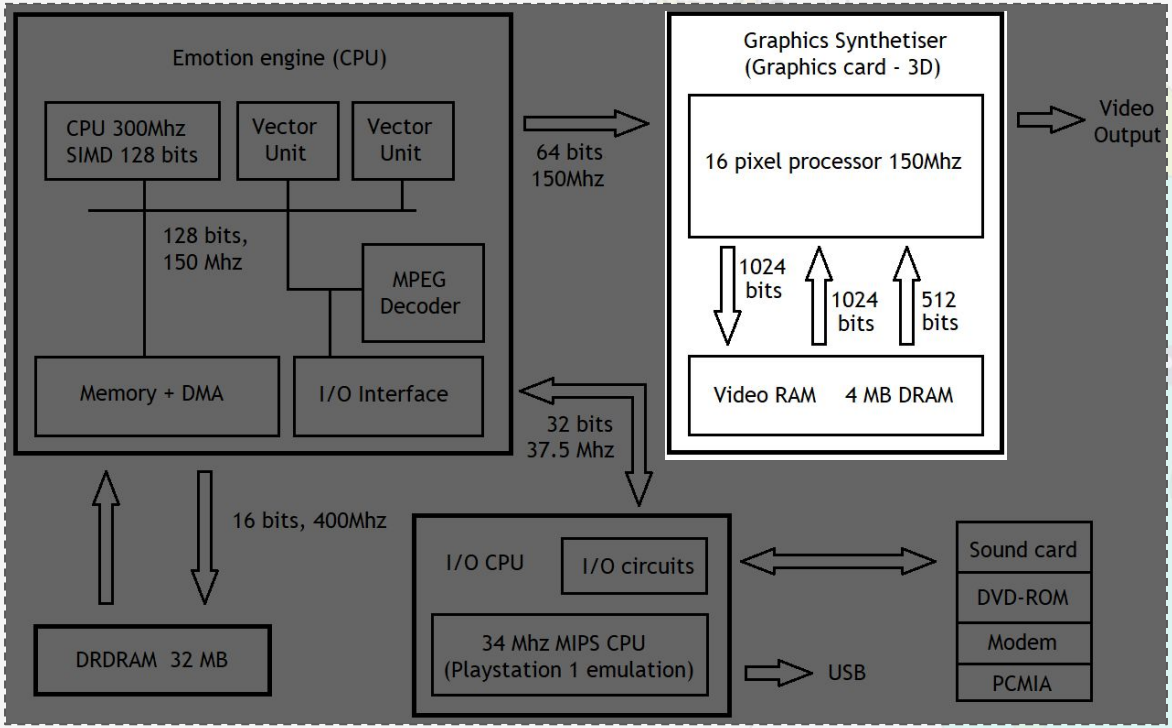
THE PLAYSTATION 2 ARCHITECTURE



THE PLAYSTATION 2 ARCHITECTURE



THE PLAYSTATION 2 ARCHITECTURE



THE PS2SDK

"PS2SDK is a collection of Open Source libraries used for developing applications on Sony's PlayStation 2® (PS2)."

BIOS and internal OS calls
Controller support
Memory card access
USB mouse and keyboard
Network stack

HDD support
CD and DVD
Sound system
POSIX compatible
…and so much more!

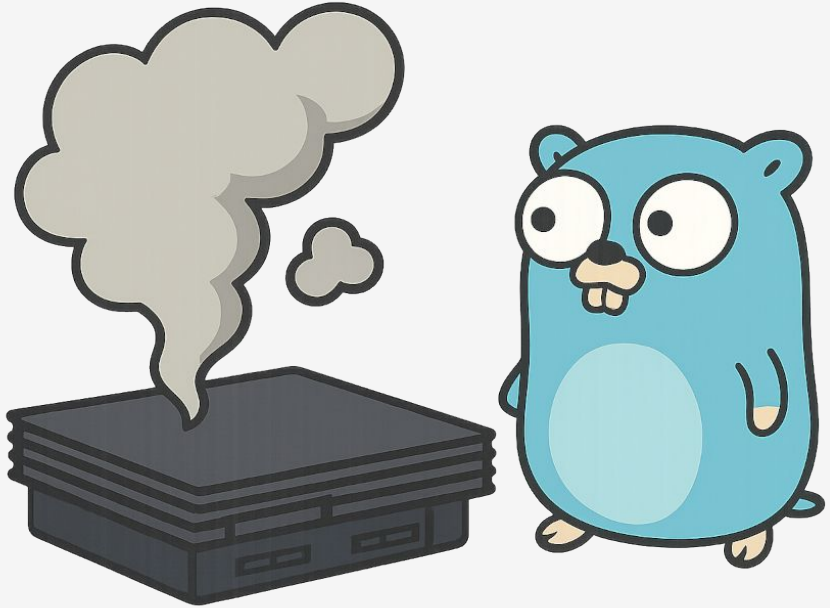
Maybe we can use this somehow? 

TINYGO
<3 PS2

Disclaimer: a lot of the next steps are the bare minimum to get it working and based on decisions I took early in this project.

Improvements are slowly coming through as I learn more and more about it :)

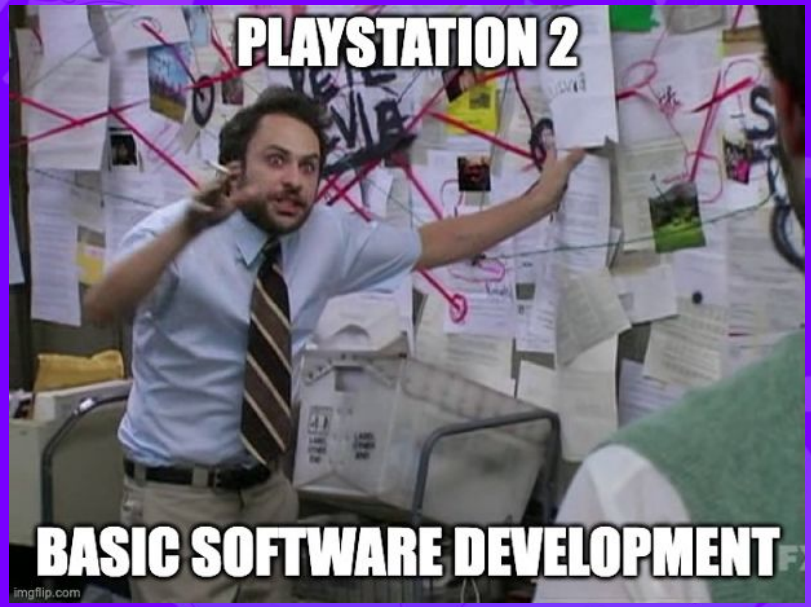
Sorry TinyGo devs for my crimes.



HELLO WORLD?

The PlayStation 2 is definitely not the simplest platform to work with.

Simple things can be complex, and often require very low-level calls, and doing that in Go is… *weird*.

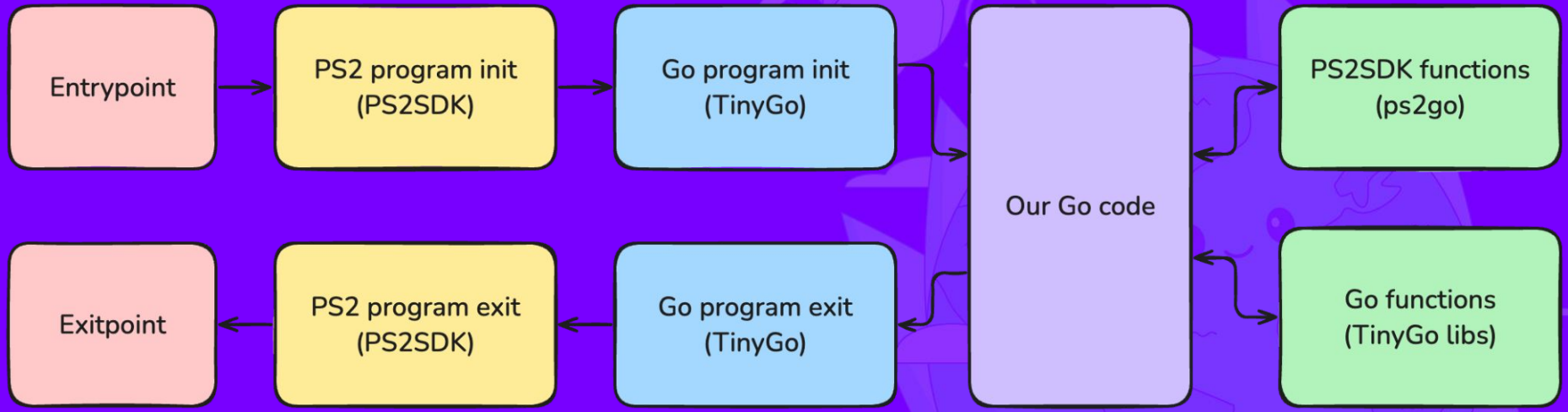


PS2SDK AS A ~~SERVICE~~ LIBRARY

PS2SDK deals with memory management and all the low-level things.

TinyGo deals with the fun side of things: games!

THE EXECUTION PLAN



THE BARE MINIMUM?

There are some requirements:

- Target definition file
- Some baremetal functions
- Some runtime functions

Plus, a lot of duct tape and hacking around!

TARGET DEFINITION

Controls how Go code is built against the target, flags, tags, what is available, what is not, etc.

Thankfully the LLVM supports our target already! (or does it?)

```
{
  "llvm-target": "mipsel-unknown-unknown",
  "cpu": "mips3",
  "features": "-noabicalls,+single-float",
  "build-tags": ["ps2", "baremetal", "mipsel"],
  "goos": "linux",
  "goarch": "mipsle",
  "linker": "ld.lld",
  "rtlib": "compiler-rt",
  "libc": "",
  "cflags": ["-mabi=n32", "-mno-ldc1-sdc1"],
  "ldflags": ["-mabi=n32"],
  // ...
}
```

BAREMETAL IS FUN

Memory layout is defined here through the heap, stack and global memory locations.

Low level functions for clock, memory allocation and basic output are also defined here, if necessary.

```
var heapStartSymbol [0]byte
var heapEndSymbol [0]byte
var globalsStartSymbol [0]byte
var globalsEndSymbol [0]byte
var stackTopSymbol [0]byte

func growHeap() bool
func runtime_putchar(c byte)
func syscall_Exit(code int)

func now() (sec int64, nsec int32, mono int64)
func AdjustTimeOffset(offset int64)
```

RUNTIME FUNCTIONS

This is where some I/O (UART), sleep and some program runtime functions are defined.

The most important part is also here: the binary **`main`** function.

```
func putchar(c byte)
func getchar() byte
func buffered() int
func sleepWDT(period uint8)
func exit(code int)
func abort()
func ticksToNanoseconds(ticks timeUnit) int64
func nanosecondsToTicks(ns int64) timeUnit
func sleepTicks(d timeUnit)
func ticks() (ticksReturn timeUnit)

//export main
func main()
```

THE MAIN FUNCTION

This is where the magic happens! Initialization will allocate the memory, while exit will free it.

Between those, TinyGo will find and call our code's **`main.main`** function.

```
func preinit() {
  // Memory allocation
}

func preexit() {
  // Memory release
}

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

FINALLY HELLO WORLD

This is what a simple Hello World looks like.

Without **ps2go**, I mean.

```
package main
/*
#define _EE
#include <stdlib.h>
#include <debug.h>
*/
import "C"
import (
  "unsafe"
)

func main() {
  C.init_scr()
  str := C.CString("Hello world")
  C.scr_printf(str)
  C.free(unsafe.Pointer(str))
  for { }
}
```

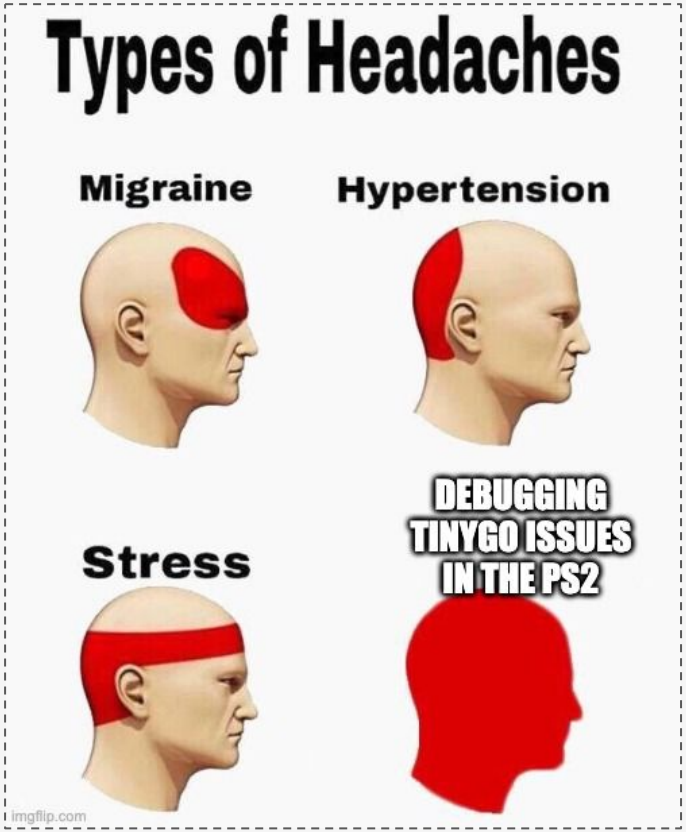
IT CAN'T BE THAT EASY

From the same creators of

"LLVM and the 9 circles of hell"

and

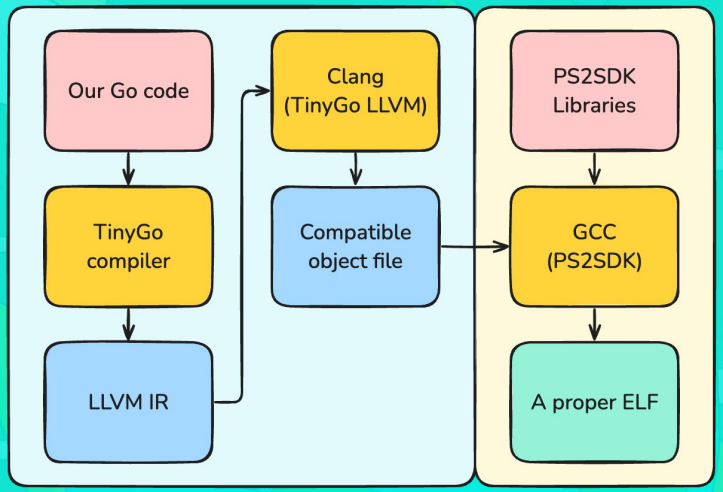
"Seriously, Sony?!"



BUILDING IS KINDA BROKEN

We can’t generate the final ELF just yet, neither a compatible object file.

But with the magic of Clang and adding a ton of steps to the process, we can make it work!



IT CAN'T BE THAT EASY

MATH IS KINDA BROKEN

Sony didn't implement **`MUL`** and **`DIV`** for 64-bit integers. Plus, the FPU is 32-bits only, so all **`float64`** math is not available as well.

Patches to both TinyGo and the LLVM are required to do these operations in *software*.

We don't about about ~~Bruno~~ performance.

```
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);
// ...
```

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

IT CAN'T BE THAT EASY

CGO IS EXPERIMENTAL

Adding custom header paths on every single wrapper is annoying.

Let's make TinyGo support **`CGO_CFLAGS`** instead.

```
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...)
}
```

IT CAN'T BE THAT EASY

COMPILER DIFFERENCES

PS2SDK uses GCC, LLVM uses Clang. Not everything is directly compatible, so we need to... "improvise".

Will this bite me back? Probably, yeah.

```c
// libpad.h
struct padButtonStatus
{
   // ...
} __attribute__((packed));
```

```go
// libpad/wrappers.go
package libpad

/*
#define __attribute__(x) // libpad doesn't play nice
with clang.
#include <libpad.h>
*/
import "C"
```

IT CAN'T BE THAT EASY

A DEFERRED CRASH

Works perfectly (?) on the emulator, but hangs on the real hardware.

This is not fixed yet.

Yet.

```go
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)
}
```

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

IT CAN'T BE THAT EASY

SONY MESSED UP TOO!

"The **`rom0:LOADFILE`** RPC service is missing support for **`LoadModuleBuffer`**, making it impossible (by default) to IOP load modules from EE RAM."

Thanks, Sony!

```
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();
}
```

THE FUN PART

With all these quirks and issues, is it actually useful?

FLAPPY GOPHER

Full game with custom textures, font, highscore and even randomized gameplay!

600+ lines of Go code

Is it quirky and buggy? Yes, but it's a fully working game in Go for the PS2!

THE UNRELEASED UPDATE

SPOILER ALERT

We have network support!

A partial network driver for TinyGo has been implemented. This will be released in the upcoming weeks!

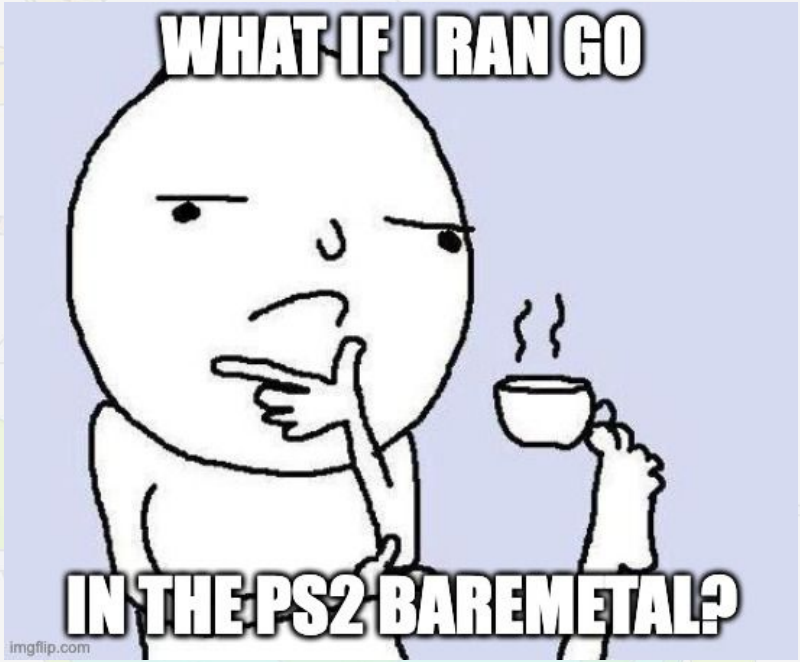
And yes, DNS was a mess to get working.

TO CONCLUDE

This project was born on a weekend because I was bored.

I'm sure it's full of bugs and problems that I'll only find in the future - but it works!

And who knows, maybe we can get this merged into TinyGo one day?

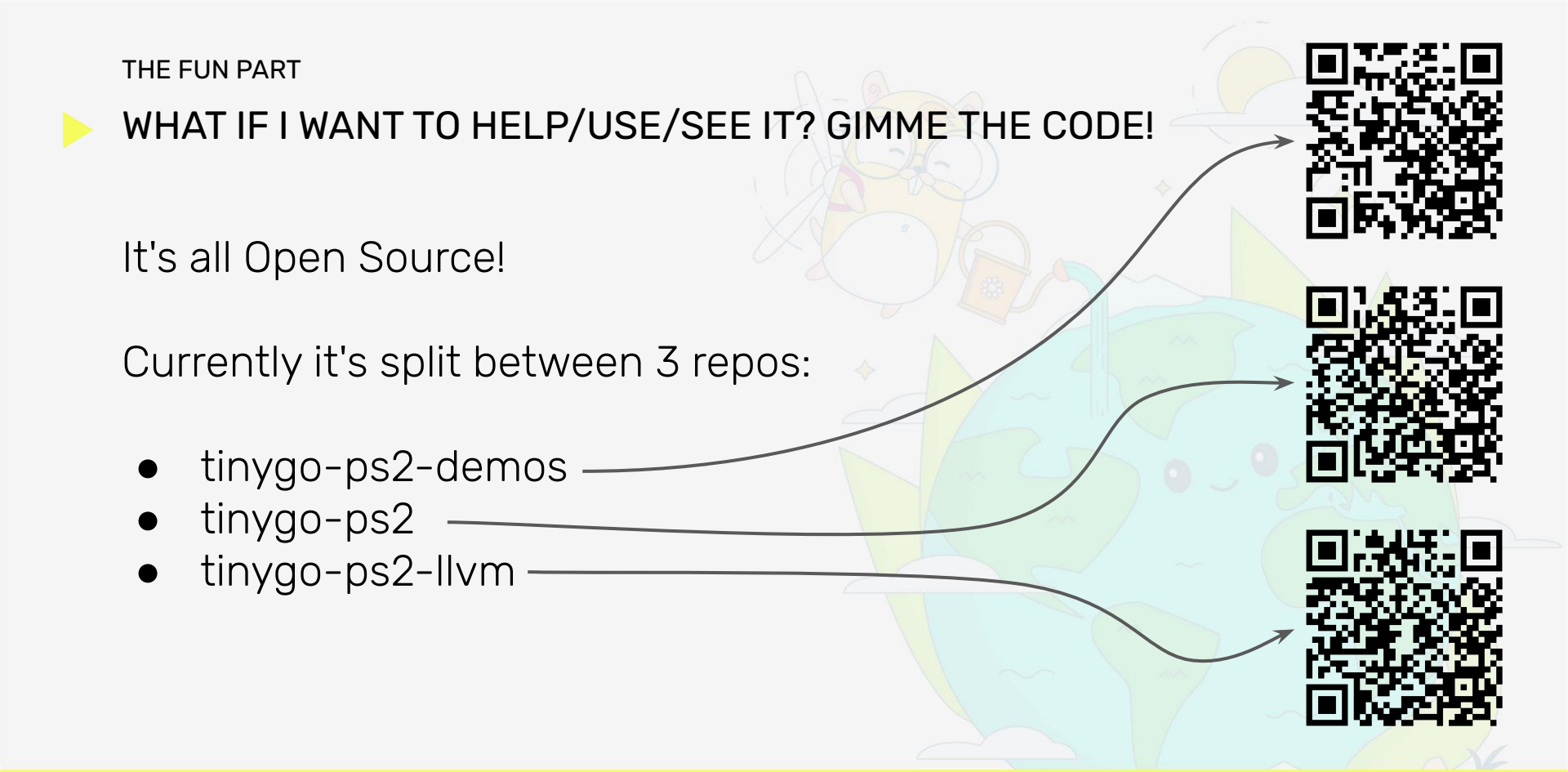


WHAT IF I WANT TO HELP/USE/SEE IT? GIMME THE CODE!

It's all Open Source!

Currently it's split between 3 repos:

- tinygo-ps2-demos
- tinygo-ps2
- tinygo-ps2-llvm



RICARDO GOMES DA SILVA

BECOMING A GAME DEVELOPER 25 YEARS TOO LATE