

# OOC

A hybrid language experiment

# Why?

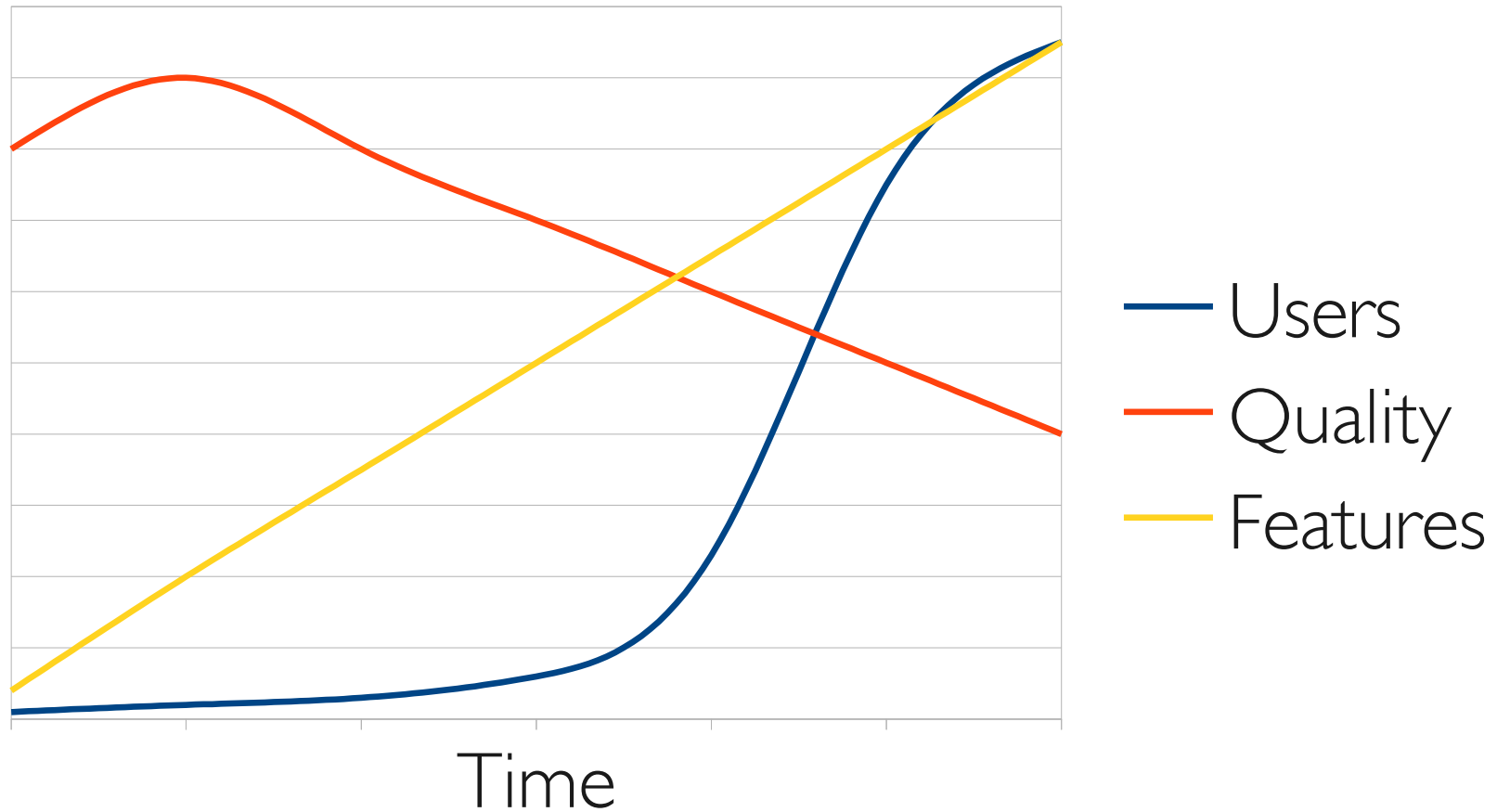
- Software sucks
  - It's unreliable
  - It's slow
  - It's too hard to develop
  - It's not modular enough
  - [insert rant here]

« The quality of a piece of software is inversely proportional to its popularity. »

— Buchheit's law

# Why?

- The fate of popular software



# Why?

- Languages suck

```
void (*signal(int, void (*fp)(int)))(int);
```

# Why?

- Languages suck

```
void (*signal(int, void (*fp)(int)))(int);
```



```
signal: Func(Int, Func(Int)) -> Func(Int)
```



# Why?

- Tools suck

```
small.cpp(17) : error C2664: 'class std::_Tree<class std::basic_string<char,struct
std::char_traits<char>,class std::allocator<char> >,struct std::pair<class
std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> >const
,int>,struct std::multimap<class std::basic_string<char,struct std::char_traits<char>,
std::allocator<char> >,int,struct std::less<class std::basic_string<char,struct
std::char_traits<char>,class std::allocator<char> > >,class std::allocator<int> >::_Kf
std::less<class std::basic_string<char,struct std::char_traits<char>,class std::alloca
> >,class std::allocator<int> > ::iterator __thiscall std::multimap<class
std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> >,int,
std::less<class std::basic_string<char,struct std::char_traits<char>,class std::alloca
> >,class std::allocator<int> >::_insert(const struct std::pair<class
std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> > const
&)' : cannot convert parameter 1 from 'const int' to 'const struct std::pair<class std
::basic_string<char,struct std::char_traits<char>,class std::allocator<char> > const ,
Reason: cannot convert from 'const int' to 'const struct std::pair<class
std::basic_string<char,struct std::char_traits<char>,class std::allocator<char> const
No constructor could take the source type, or constructor overload resolution was
ambiguous
```

# Why?

- Tools suck
  - GNU autoconf, automake, autoreconf – masochism
  - GNU ld – the kitchen sink of linkers
  - GNU make – the brainless servant
- Bad workers blame the tool
  - What if good workers do too?
  - Break with tradition

# Why?

- School assignment in C – laziness.
  - 4 months later – ooc 0.1
  - « Version 1 Sucks, But Ship It Anyway » – J. Atwood
- Finding a purpose
  - To remove obstacles
  - To encourage experimentation
  - To minimize frustration

# What?

- Classes, abstract, single inheritance, virtual by default
- Garbage collection (Boehm, opt-out)
- Covers, function overloading
- Partial type inference, more type-checking
- Arrays, pointers, manual memory management
- Generic functions, generic classes, collections
- Interfaces, operator overloading, properties
- Closures, first-class functions, map/filter/reduce

# What?

- I was gonna describe the syntax here
- But we're short on schedule so I'll just sum it up:
- « ooc syntax is *Java without bullshit* » - Anonymous
- I'm sorry if you were offended.
- Java offends me too.

# Design principles

- Build upon, extend, divide and conquer
- Re-use what makes sense, rewrite the rest as we go
- JDK – an example of how NOT to do modularity
- SDK – all you need to get started, easy to swap out
- Dependency management made easy
- Making it easy to use libs encourages good design

# Acronym fair

- DRY
- KISS
- IYFF
- YAGNI
- ~~RTFM~~RTFC
- TMTOWTDI



# Paradigm City

- Procedural

```
println("Is i+=1 deterministic?")
```

# Paradigm City

- Object-oriented

```
Window new("Vista").  
  add(  
    Button new("Buy").  
      connect("clicked", ||  
        "Seriously?" println()  
      )  
  ).  
  showAll()
```

# Paradigm City

- Generic programming

```
Cell: class <T> {  
    data: T  
    next: This<T>  
    init: func (=data) {}  
}
```

```
c := Cell new(42)
```

# Paradigm City

- Functional

```
(1..100) map(|x| x*x) reduce(|a, b| a+b)
```

# Paradigm City

- Preemptive multi-threading

```
mutex := Mutex new()  
Thread new(||  
  mutex lock()  
  // prove Fermat's last theorem  
  mutex unlock()  
) start()
```

# Paradigm City

- Communicating Sequential Processes

```
chan := make(Int)
go(||
    chan << question
    answer := ! chan
)
```

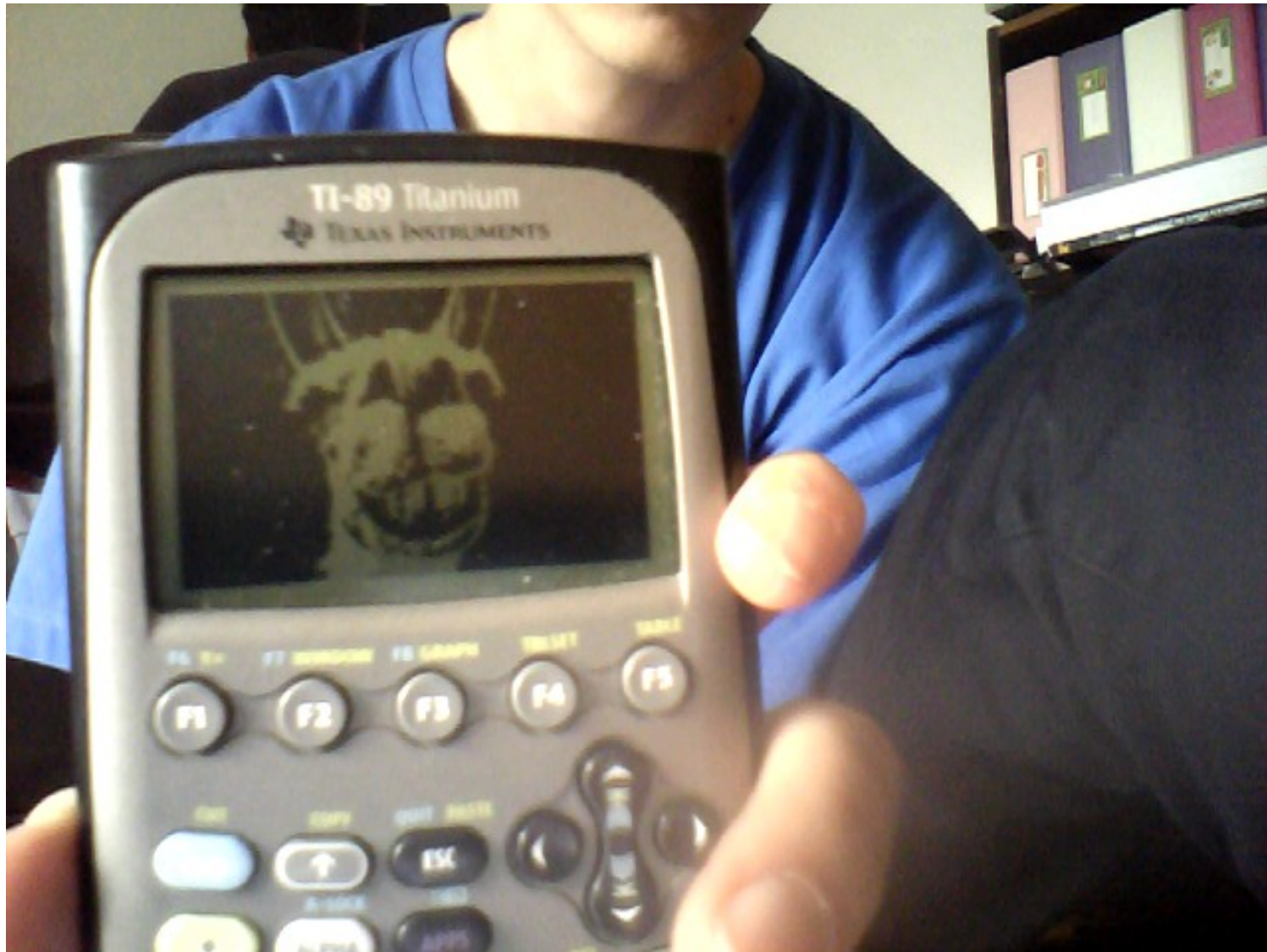
24h for a basic implementation using libcoroutine

80'000 concurrent coroutines = easy, more with tweaks

# Paradigm City

- ...is pretty much a nice walk with ooc
- Provide the necessary building blocks
- Don't enforce the « one true way »
- Politics != Technology
- High-level low-level language.

# Generating C – the perks



# Generating C – the perks

```
throw old VMExcuse("I wasn't ready!");
```

```
if name == "__main__":  
    dont_hold_your_breath()
```

# Generating C – the perks

- GCC (Gnu Compiler Collection), TI-GCC, mingw32
- TCC (TinyCC)
- ICC (Intel C++ Compiler)
- Clang (LLVM)
- PCC (Portable C compiler - with tweaks)
- No MSVC (yet?) - they're too busy with C++0x
- ...although in theory it wouldn't be that hard.

# Generating C – the perks

- Did you know that GCC -O2 did TCO?
- Turn segfault into infinite loops.
- Protip: use -O0 (the default) when debugging

# ooc - the platform

- Self-hosting
- Without a doubt the best way to generate C
- A real module system. Partial recompilation.
- The compiler as a library
- C from ooc is easy
- ooc from C is easy (**#define OOC\_FROM\_C**)
- ooc from Python is dead easy!
- ooc from X = reading JSON compiler output

# ooc – the tools

- Good compiler errors
- Valgrind, GDB
- Alleyoop, Nemiver (GTK frontends for the above)
- Callgrind, Kcachegrind, gprof
- (Pseudo-)REPL, IRC bot
- Emacs mode – flymake, tooltips with compiler errors
- Lots of C bindings

# Why use ooc?

- Because you have a ~~ham~~ syntax fetish
- As a better C/C++
- As a better Java with no VM
- Because of the tools
- Because of the community
- You want to be part of the adventure

# What's next?

- Optimizations
  - Generic calls inlining – expect huge speedups
  - (Please, Mr. Shapiro, don't burst into flames)
  - Escape analysis, stack-allocation, annotations
- An alternative to exceptions
- Typesystem improvements
  - Mixins
  - Typestate?
- Meta-programming, compile-time execution

# That's all, folks!

**Web** <http://ooc-lang.org>

**IRC** #ooc-lang on Freenode

**Twitter** @nndrylliog

**Mail** [amoswenger@gmail.com](mailto:amoswenger@gmail.com)

