

Live coding for Human-in-the-Loop Simulation

Ben Swift

*Research School of Computer Science
Australian National University*

live coding

the practice of writing &
editing live programs

Outline

- Extempore: a software environment for live coding
- Live demo: particle-in-Cell (PIC) simulation
- Future directions: human-in-the-loop processing in simulation, modelling & decision support

extempore

- Open-source (MIT Licence) & available for OSX, Linux & Windows (<http://github.com/digego/extempore>)
- Lisp-style syntax (Scheme-inspired)
- High-performance compiled code (LLVM JIT)
- Toll-free interop with C & Fortran
- **Hot-swappable** at a function level

programmer



```
(bind dsp1:DSP
  (λ (in time chan dat)
    (+ (fmsynth in time chan dat)
      (epad in time chan dat))))

(bind-sampler kit sampler_note_hermite_c sampler1_fx)
(sm-load-map kit *sm-NI-synthetic-kit* 0)
(sm-load-map kit *sm-NI-*sm-NI-NuPop-kit* 1)

(bind-func dsp2:DSP
  (λ (in time chan dat)
    (kit in time chan dat)))
```

TCP

extempore

(local or remote)



```
(bind-func multiply_add
  (λ (x:i32 y z)
    (+ (* x y) z)))
```

Extempore code

```
Ltmp4:
    .cfi_def_cfa_register %rbp
    movl    %edi, -4(%rbp)
    movl    %esi, -8(%rbp)
    movl    %edx, -12(%rbp)
    movl    -4(%rbp), %edx
    imull    -8(%rbp), %edx
    addl    -12(%rbp), %edx
    movl    %edx, %eax
    popq    %rbp
    retq
    .cfi_endproc
```

```
; F
def
%1 = alloca i32, align 4
%2 = alloca i32, align 4
%3 = alloca i32, align 4
store i32 %x, i32* %1, align 4
store i32 %y, i32* %2, align 4
store i32 %z, i32* %3, align 4
%4 = load i32* %1, align 4
%5 = load i32* %2, align 4
%6 = mul nsw i32 %4, %5
%7 = load i32* %3, align 4
%8 = add nsw i32 %6, %7
ret i32 %8
}
```

assembler &
LLVM IR
machine code

any function can be
redefined on-the-fly

programmer

```
(bind-func dsp1:DSP
  (lambda (in time chan dat)
    (+ (fmsynth in time chan dat)
       (epad in time chan dat))))

(bind-sampler kit sampler_note_hermite_c sampler1_fx)
(sm-load-map kit *sm-NI-synthetic-kit* 0)
(sm-load-map kit *sm-NI-*sm-NI-NuPop-kit* 1)

(bind-func dsp2:DSP
  (lambda (in time chan dat)
    (kit in time chan dat)))
```



C/Fortran

main

output



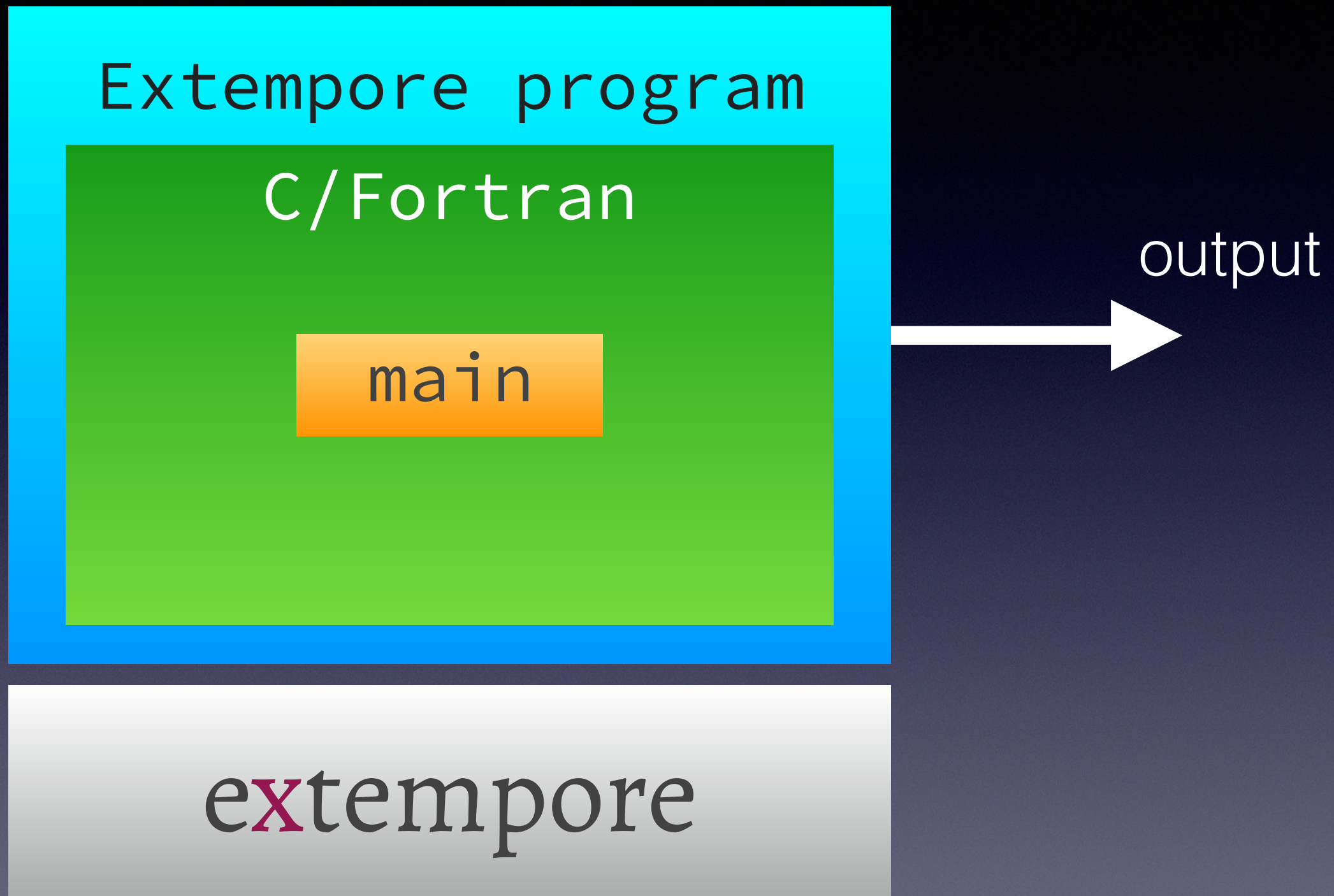
Extempore program

C/Fortran

main

output

ex**x**tempore



Extempore program

main

C/Fortran

init

push

solve

changes

output

ex**x**tempore

The diagram illustrates the architecture of the Extempore program. It features a large cyan rectangle representing the 'Extempore program'. Inside this rectangle, at the top, is an orange box labeled 'main'. Below 'main' is a green rectangle labeled 'C/Fortran'. Inside the green rectangle are three orange boxes labeled 'init', 'push', and 'solve' from left to right. To the right of the cyan rectangle, a white arrow labeled 'changes' points left towards the 'main' box. Another white arrow labeled 'output' points right away from the 'C/Fortran' box. Below the cyan rectangle is a separate light gray rectangle containing the text 'ex**x**tempore', where the 'x' is in a dark red color.

Extempore program

main

C/Fortran

init

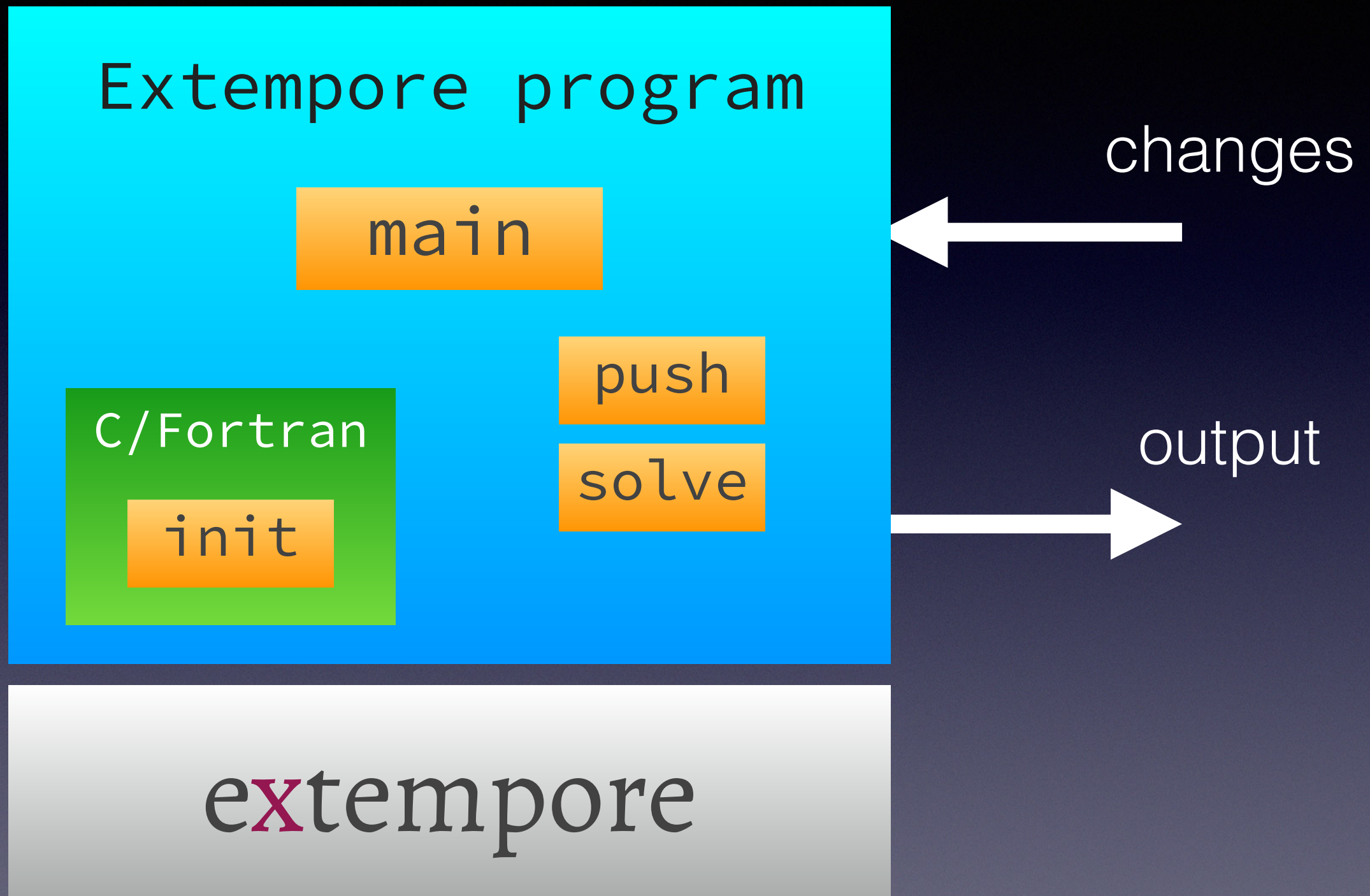
push

solve

changes

output

ex**x**tempore



Extempore program

main

C/Fortran

init

push

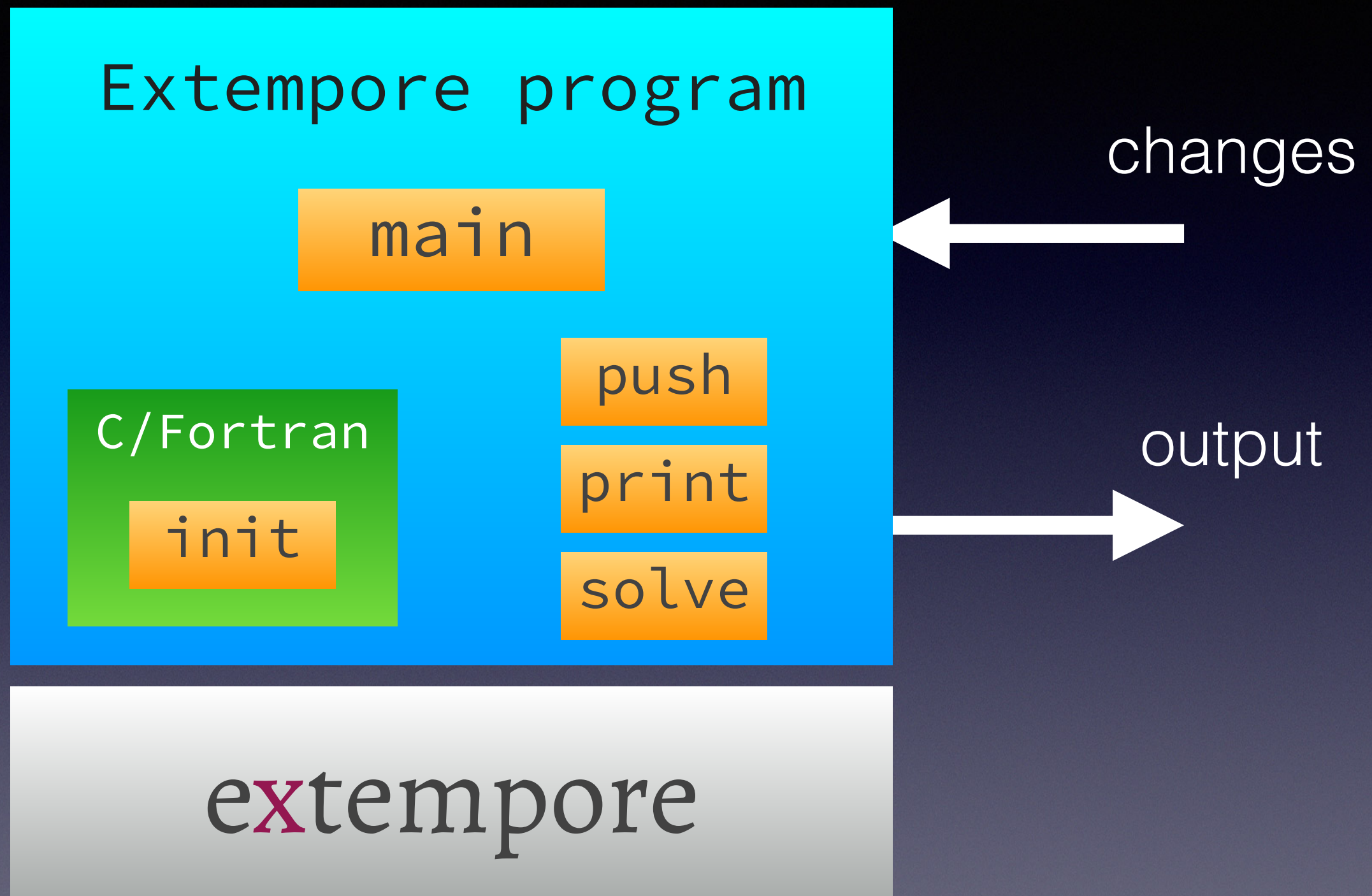
print

solve

changes

output

ex**x**tempore



Extempore program

main

C/Fortran

init

push

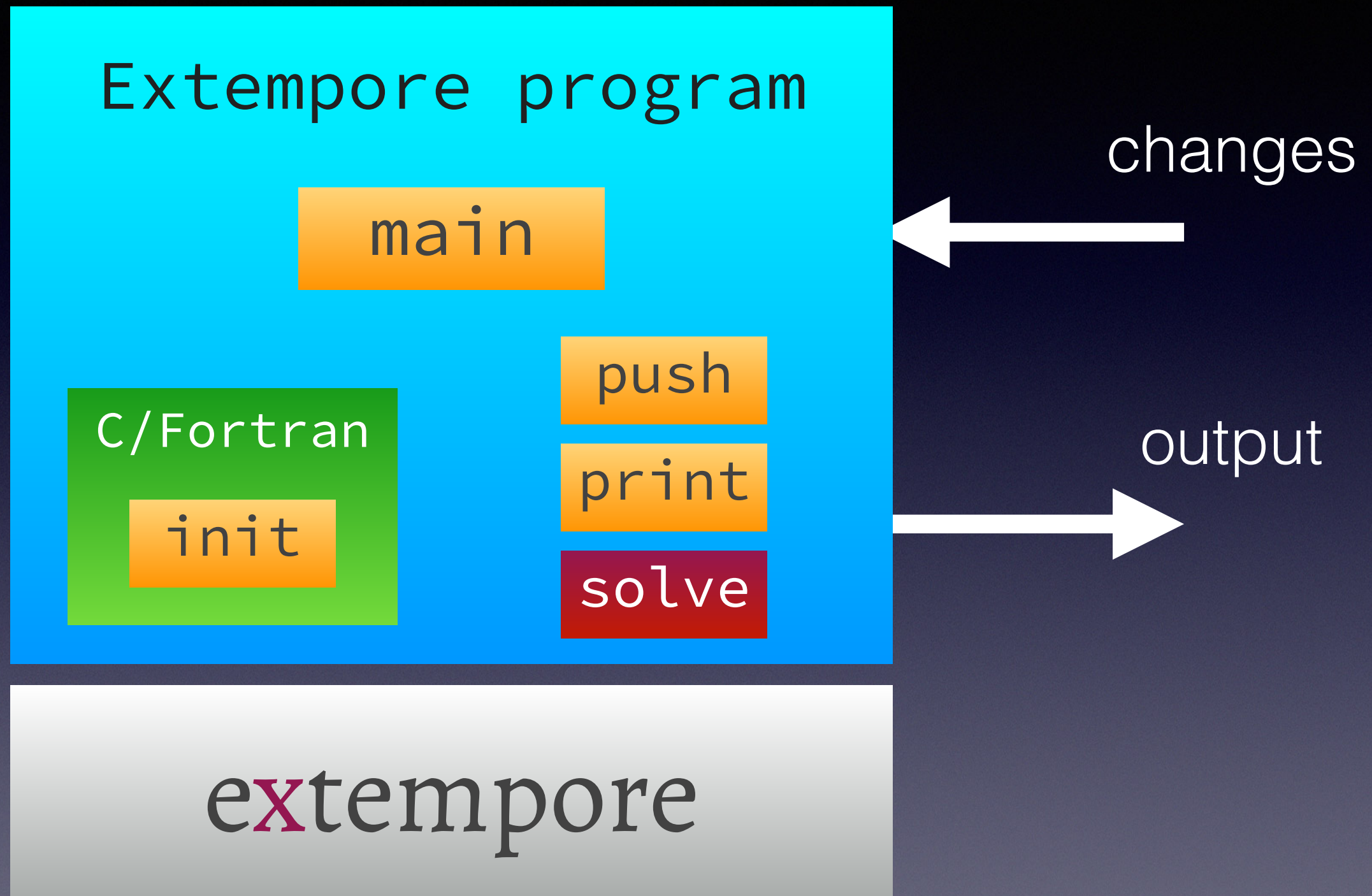
print

solve

changes

output

ex**x**tempore



Extempore program

main

C/Fortran

init

push

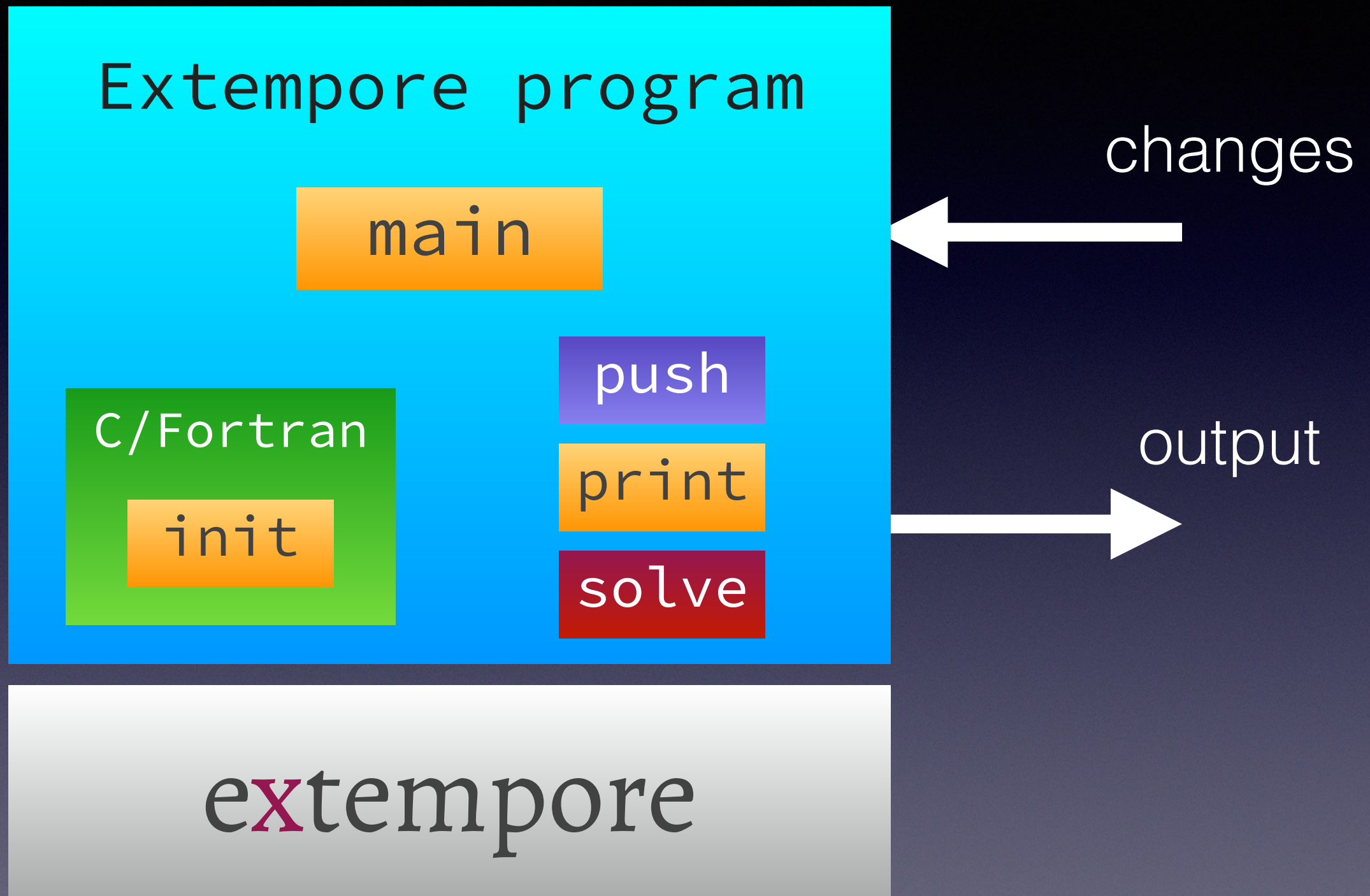
print

solve

changes

output

ex**x**tempore



Extempore program

main

C/Fortran

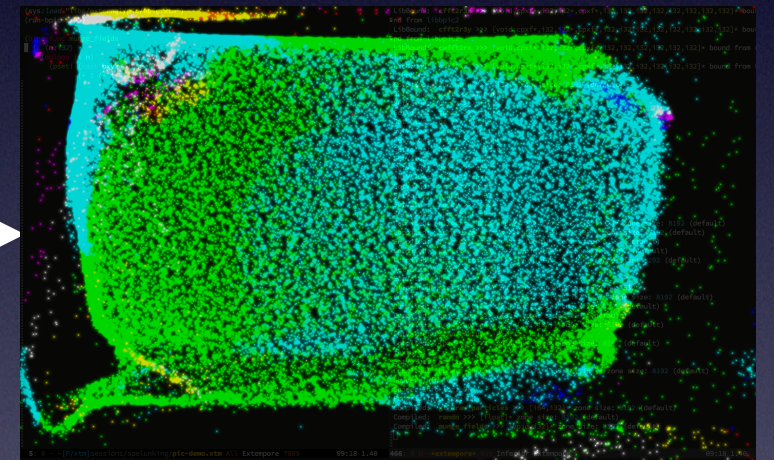
init

push

vis

solve

changes



extempore

Extempore program

main

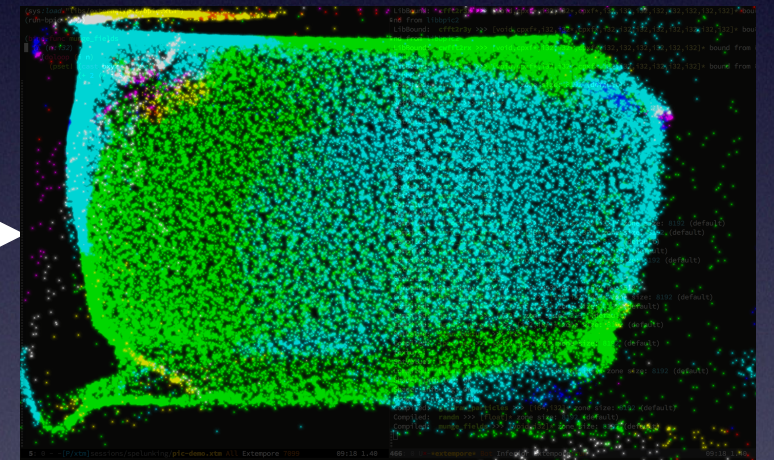
init

push

vis

solve

changes



ex**te**mpore

The power of live programming

- Change parameters on the fly, with feedback
- Interactively add debug output (no loss of state)
- Switch optimisers/solvers etc. on-the-fly (modularity in codebase & libraries helps)

*what does this have to
do with DSI?*

Similarities

- Numerical modelling (HPC-style problems)
- High-dimensional, non-linear parameter spaces
- Long-running jobs = long delays for feedback

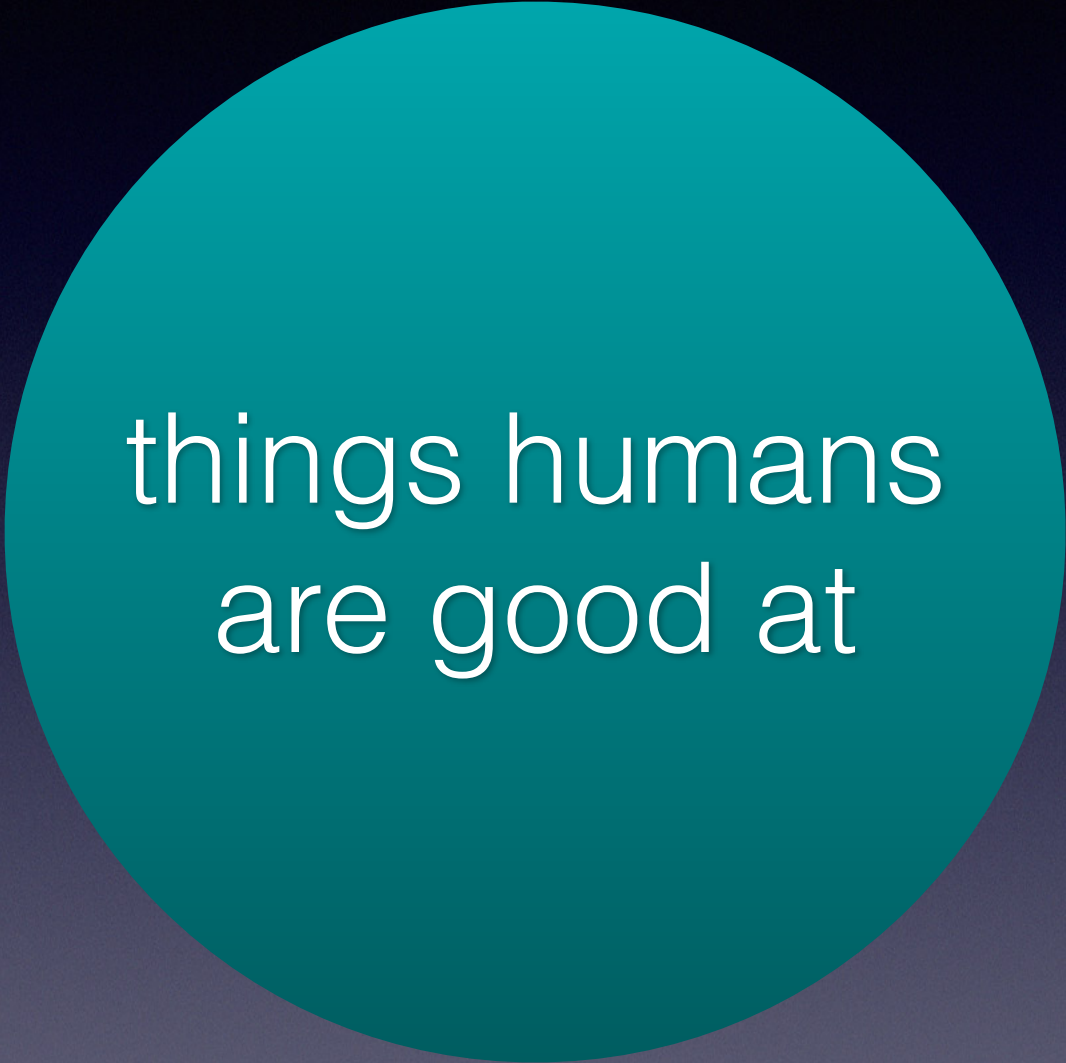
Opportunities

- Joint simulation, experimentation and wargaming (e.g. computational red-teaming)
- Human-in-the-loop decision support systems—real-time input from decision-makers
- Real-time sensor & intelligence input
- Cybersecurity



Unknown unknowns

- How to handle these unknowns?
 - contingency planning
 - adaptive/learning systems
 - **real-time human-in-the-loop intervention**

A large teal circle on a dark blue background.

things humans
are good at

A large purple circle on a dark blue background.

things computers
are good at

immature
codes

mature
codes

poor domain
understanding

good domain
understanding

qualitative
insights

quantitative
agreement



new
problems

old
problems

Extempore program

main

init

push

vis

solve

extempore

new
problem

old
problems

immature

Extempore program

main

init

push

vis

solve

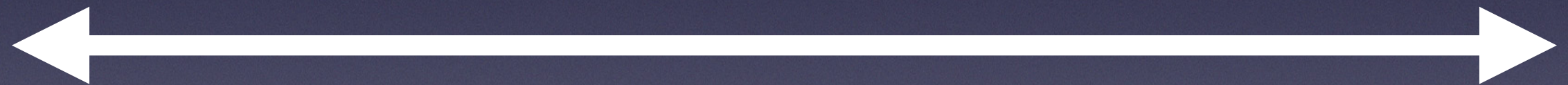
extempore

insights

mature
codes

good domain
understanding

quantitative
agreement



new
problems

old
problems

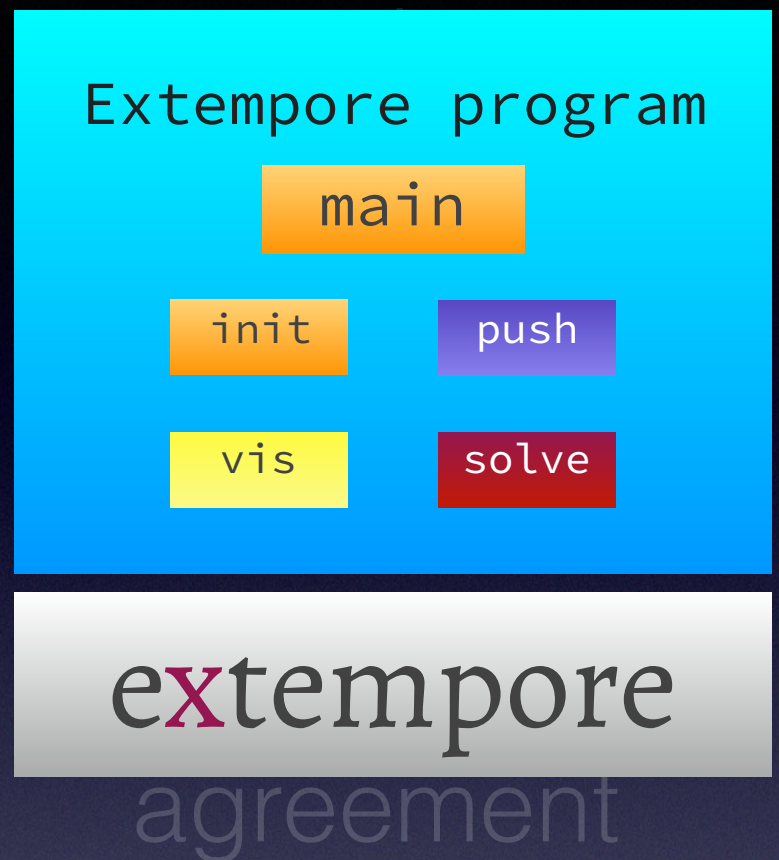
immature
codes

poor domain
understanding

qualitative
insights

new
problems

mature



old
problems

Big questions

- In what scenarios can a skilled live programmer provide useful intelligence to decision-makers, particularly in the face of unplanned events?
- How can this capability be most effectively integrated into existing techniques & processes

Next steps

- Making connections: people and problems
- We're keen to keep engaging with the DSI to figure out areas of mutual interest
 - to find out what your *care* about

expanding our capability
for simulation & modelling
in an uncertain world

What I'm **not** saying

- Everyone must be an Extempore hacker
- It works on 100 nodes—job done!
- You should let me loose to poke around in running codes on your Titan allocation

thanks

My email: **ben.swift@anu.edu.au**

Extempore Source:

<https://github.com/digego/extempore>

Docs: **<http://benswift.me/extempore-docs/>**

Mailing list:

extemporelang@googlegroups.com

IRC: **#extempore** on freenode