acrionlua

Presentation on October 10, 2022

who we are

Stefan Zipproth

Managing director

✉ s.zipproth-at-acrion.ch

Alexander Braun

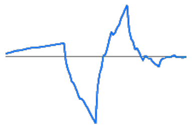
Managing director

✉ a.braun-at-acrion.ch

personal background

- studied computer science at Technical University of Munich
- freelancer since 1999, worked for 17 large companies in Germany and Switzerland, mainly embedded systems in C++
- own products:

Straton
image segmentation used by astro photographers



jetpix
image compression technology

Aristarch
chess engine

previous product

$$\begin{tikzcd}
\boxed{\text{GUI (Windows, C\#, Window Forms)}} \arrow[d] \\
\boxed{\text{.NET Assembly (C++)}}
\end{tikzcd}$$

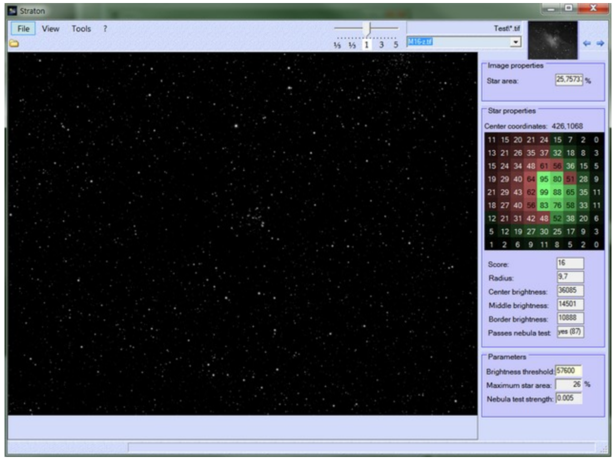


Image properties

Star area: 25,7573 %

Star properties

Center coordinates: 426,1068

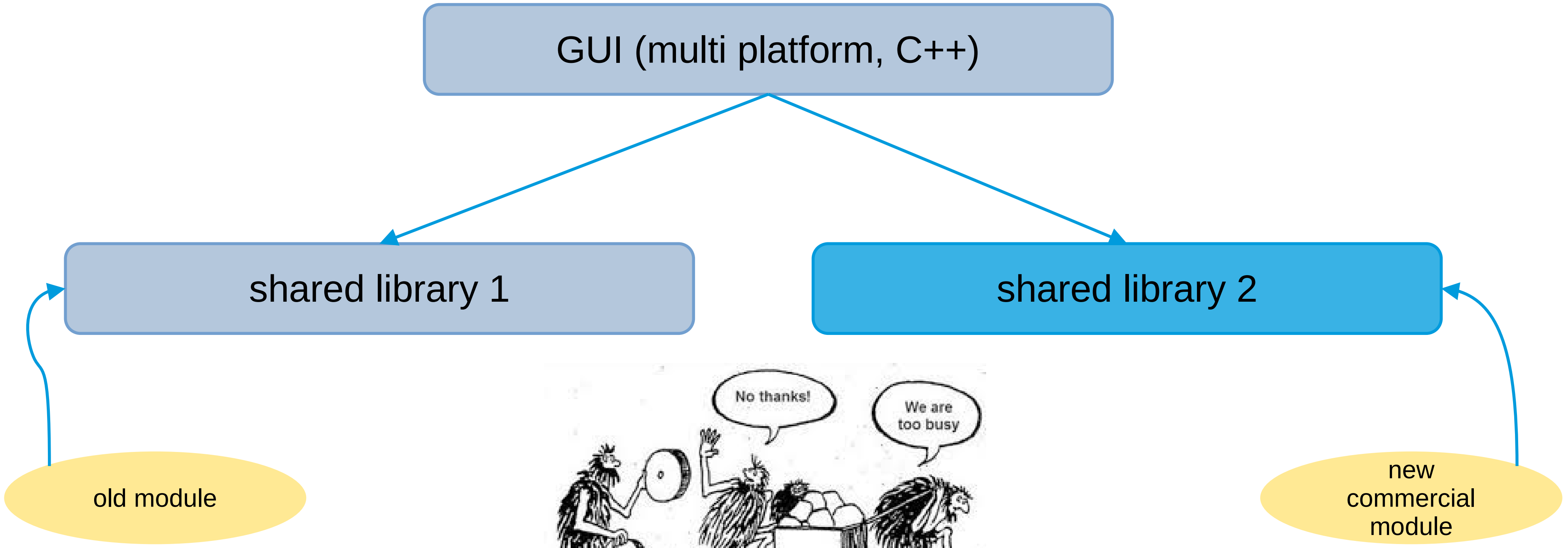
11	15	20	21	24	15	7	2	0
13	21	26	35	37	32	18	8	3
15	24	34	48	61	56	36	15	5
19	29	40	64	95	80	51	28	9
21	29	43	62	99	88	65	35	11
18	27	40	56	83	76	58	33	11
12	21	31	42	48	52	38	20	6
5	12	19	27	30	25	17	9	3
1	2	6	9	11	8	5	2	0

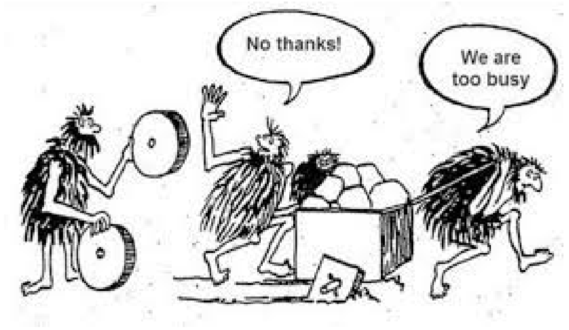
Score: 16
Radius: 9,7
Center brightness: 36085
Middle brightness: 14501
Border brightness: 10888
Passes nebula test: yes (87)

Parameters

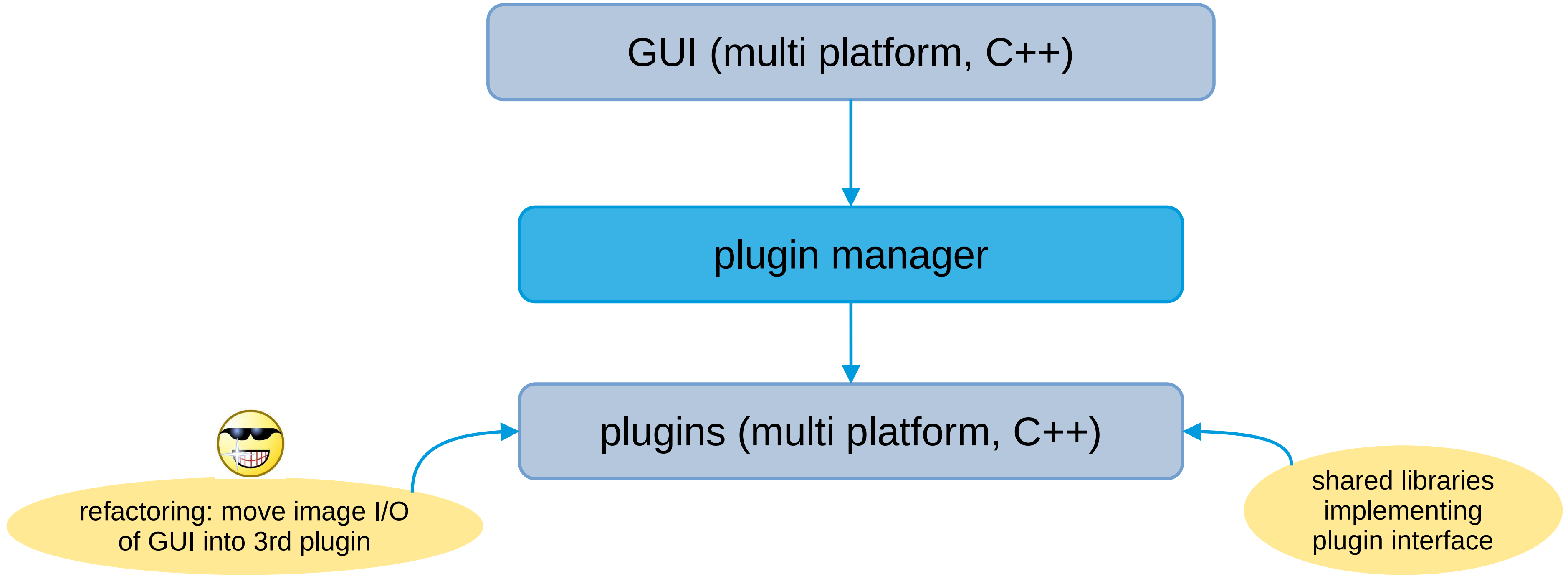
Brightness threshold: 57600
Maximum star area: 26 %
Nebula test strength: 0.005

new product


GUI (multi platform, C++)
shared library 1
shared library 2
old module
new commercial module

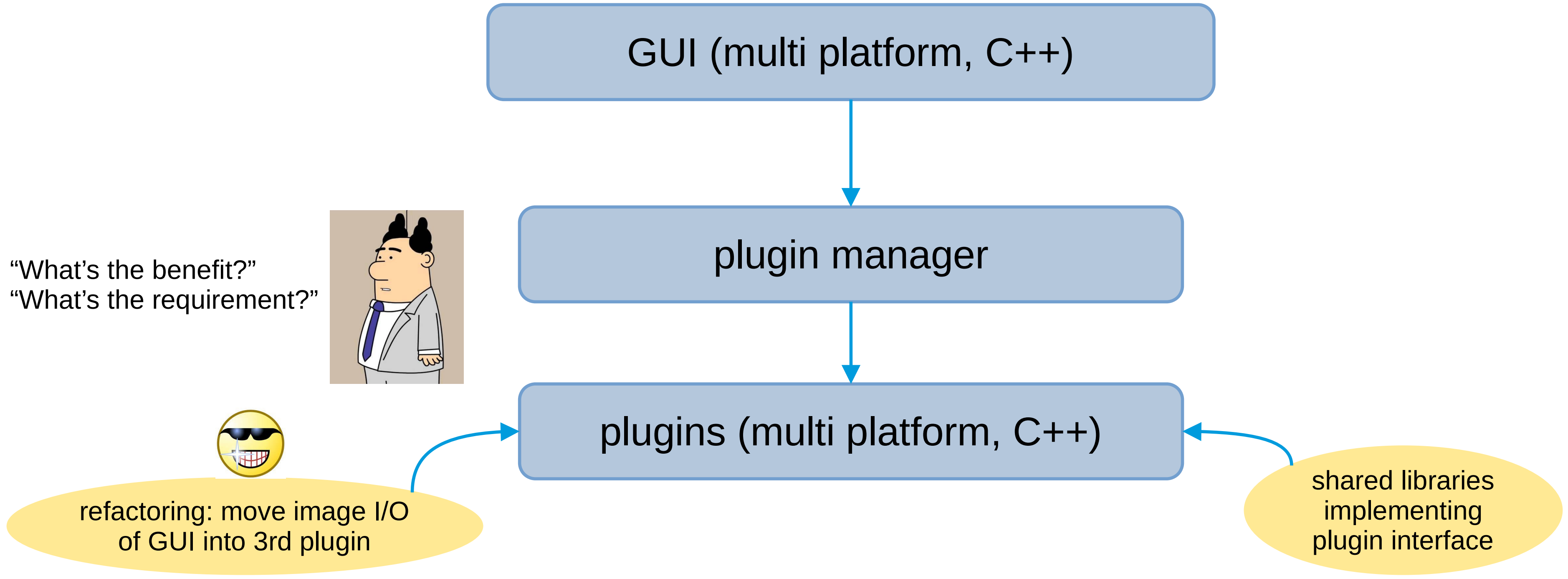

No thanks!
We are too busy

better use generic plugin interface?

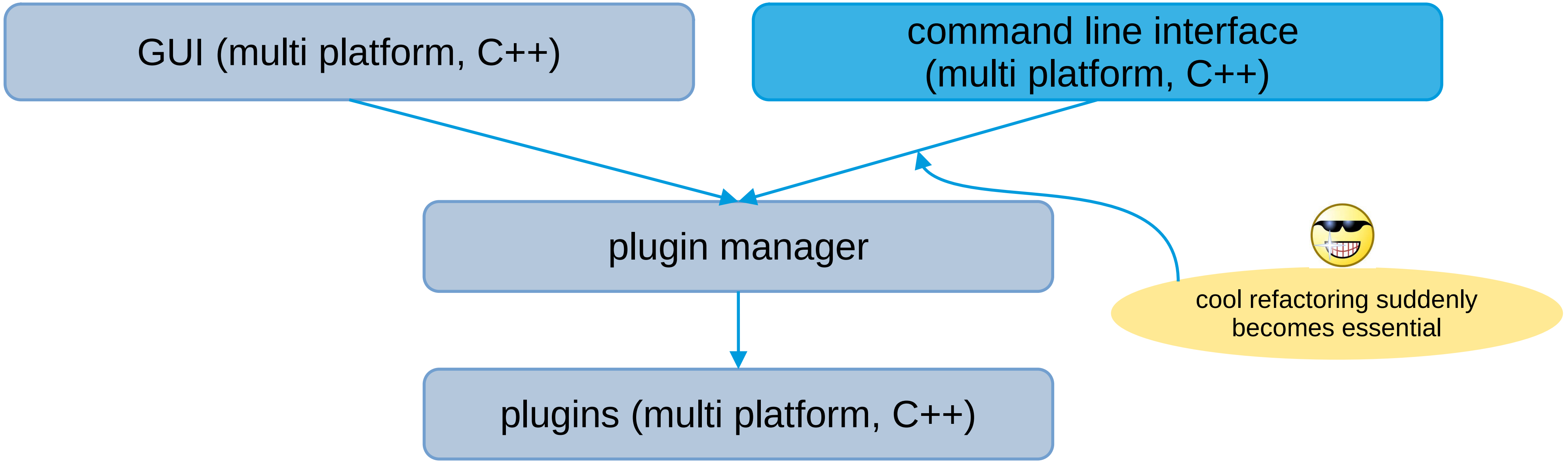

GUI (multi platform, C++)
plugin manager
plugins (multi platform, C++)
refactoring: move image I/O of GUI into 3rd plugin
shared libraries implementing plugin interface

better use generic plugin interface?

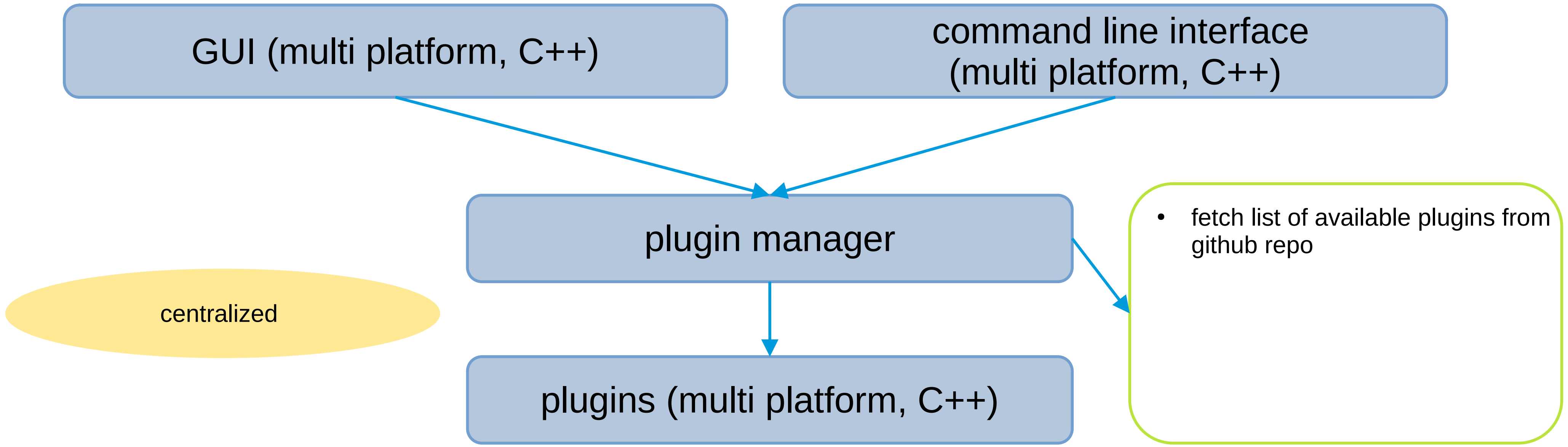
“What’s the benefit?”
“What’s the requirement?”


GUI (multi platform, C++)
plugin manager
plugins (multi platform, C++)
refactoring: move image I/O of GUI into 3rd plugin
shared libraries implementing plugin interface

make users want batch processing

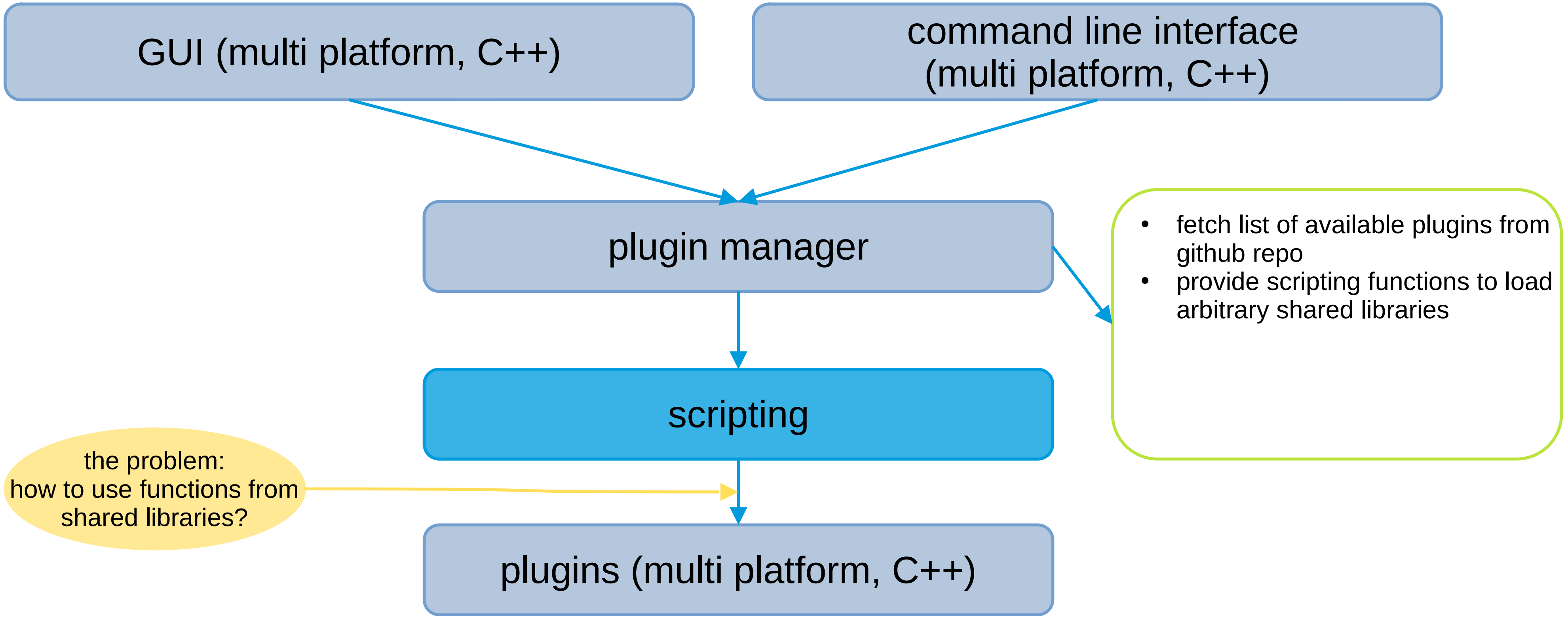

GUI (multi platform, C++)
command line interface (multi platform, C++)
plugin manager
plugins (multi platform, C++)
cool refactoring suddenly becomes essential

first task of the plugin manager


GUI (multi platform, C++)
command line interface (multi platform, C++)
plugin manager
centralized
plugins (multi platform, C++)

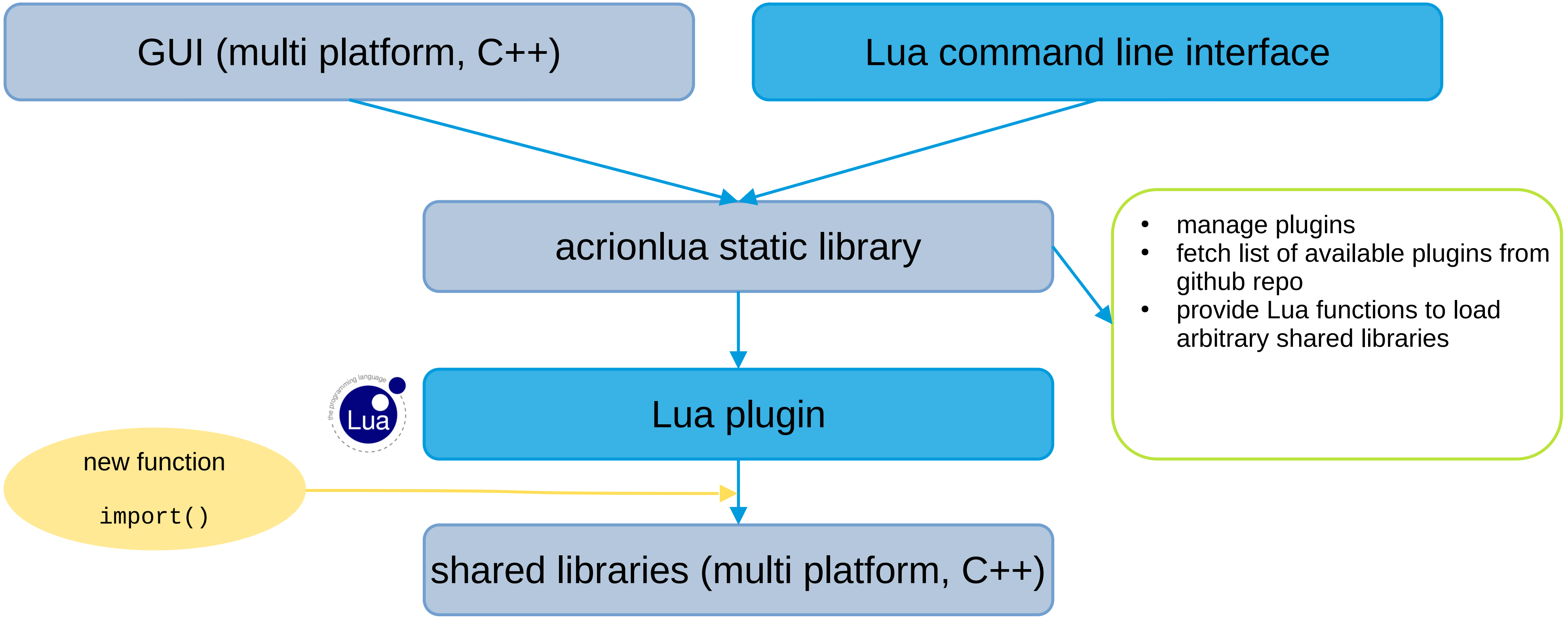
- fetch list of available plugins from github repo

we also want scripting


GUI (multi platform, C++)
command line interface (multi platform, C++)
plugin manager
scripting
plugins (multi platform, C++)
the problem: how to use functions from shared libraries?

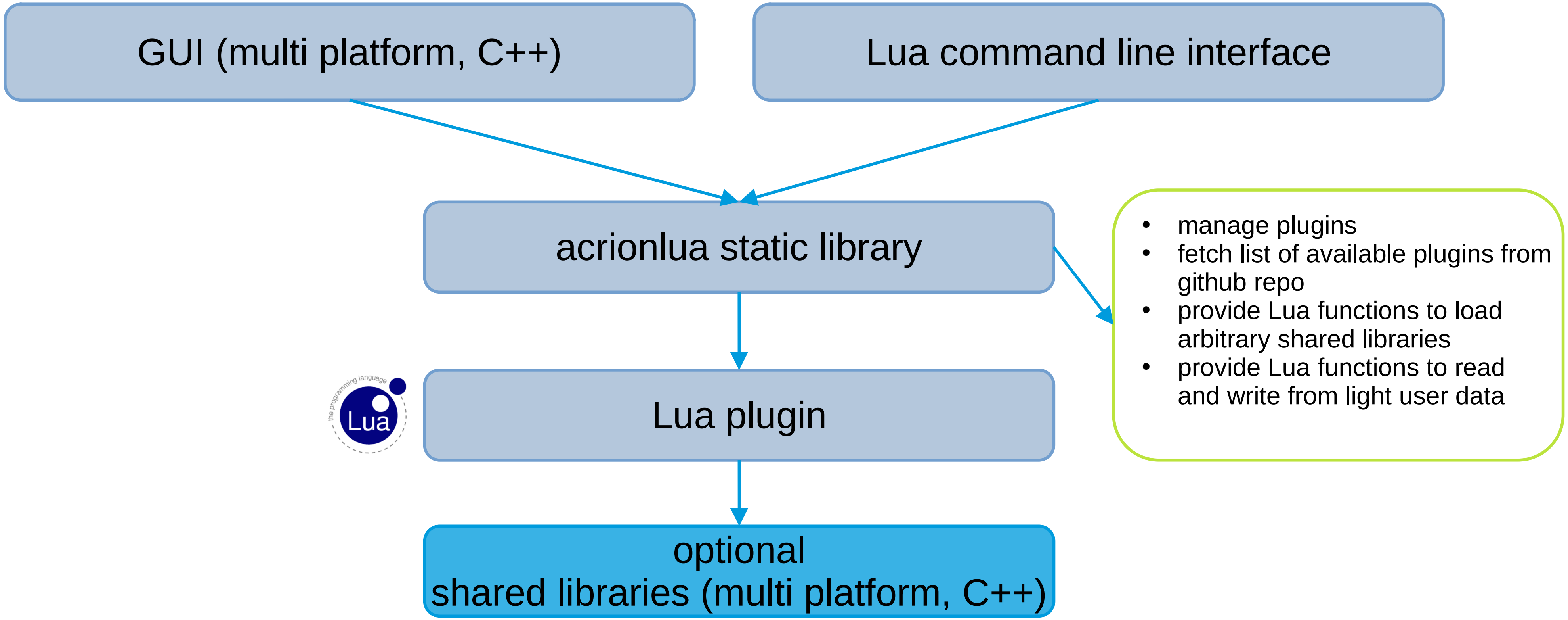
- fetch list of available plugins from github repo
- provide scripting functions to load arbitrary shared libraries

let’s use Lua!


GUI (multi platform, C++)
Lua command line interface
acrionlua static library
Lua plugin
shared libraries (multi platform, C++)
new function
import()
the programming language
Lua

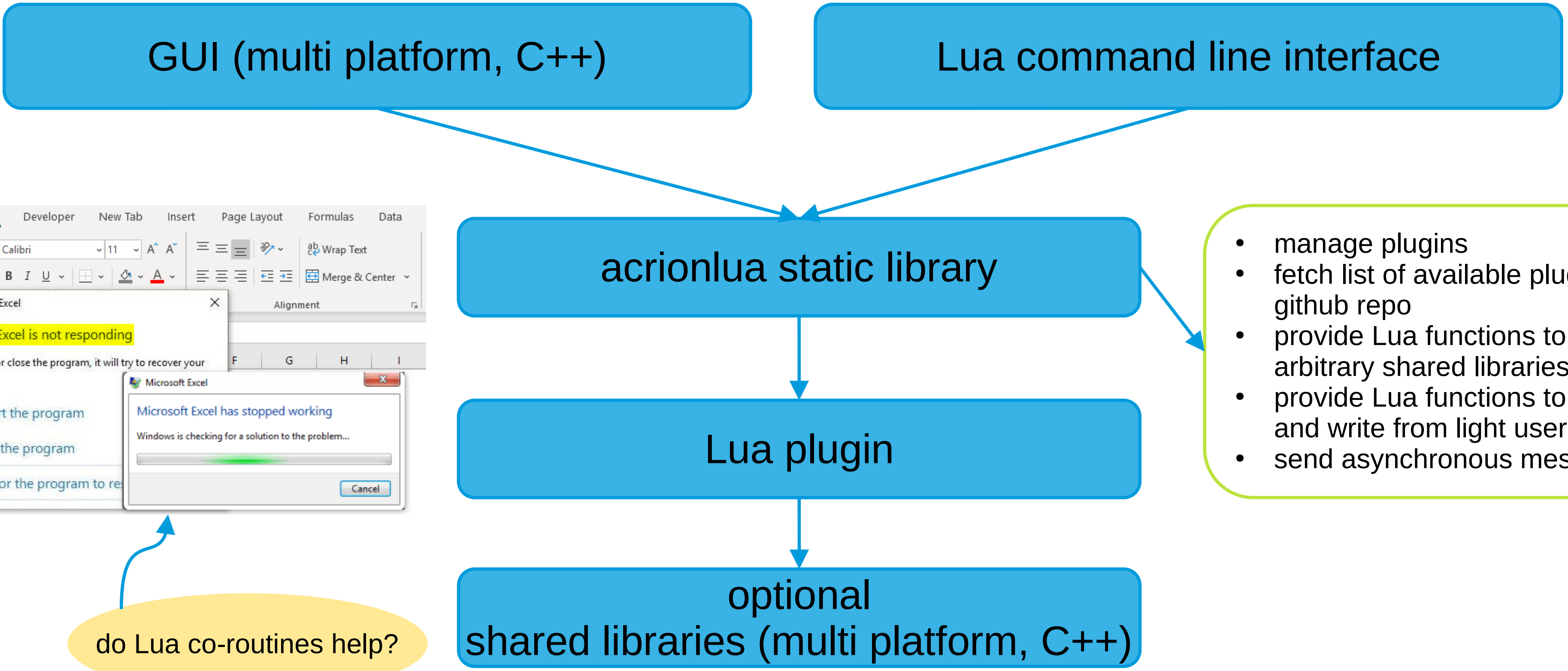
- manage plugins
- fetch list of available plugins from github repo
- provide Lua functions to load arbitrary shared libraries

make pure Lua plugins possible

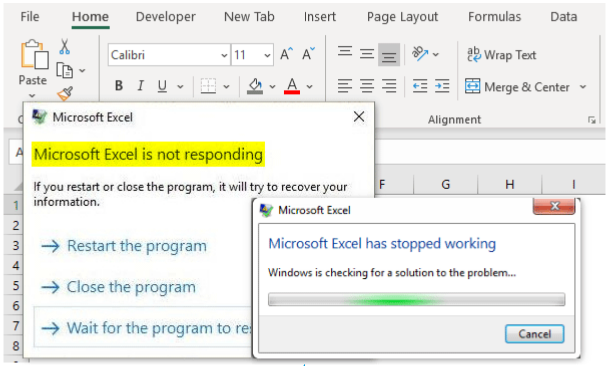

GUI (multi platform, C++)
Lua command line interface
acrionlua static library
Lua plugin
the programming language
Lua
optional
shared libraries (multi platform, C++)

- manage plugins
- fetch list of available plugins from github repo
- provide Lua functions to load arbitrary shared libraries
- provide Lua functions to read and write from light user data

take care of a common usability problem


GUI (multi platform, C++)
Lua command line interface
acrionlua static library
Lua plugin
optional
shared libraries (multi platform, C++)

- manage plugins
- fetch list of available plugins from github repo
- provide Lua functions to load arbitrary shared libraries
- provide Lua functions to read and write from light user data
- send asynchronous messages



Microsoft Excel

Microsoft Excel is not responding

If you restart or close the program, it will try to recover your information.

→ Restart the program

→ Close the program

→ Wait for the program to re

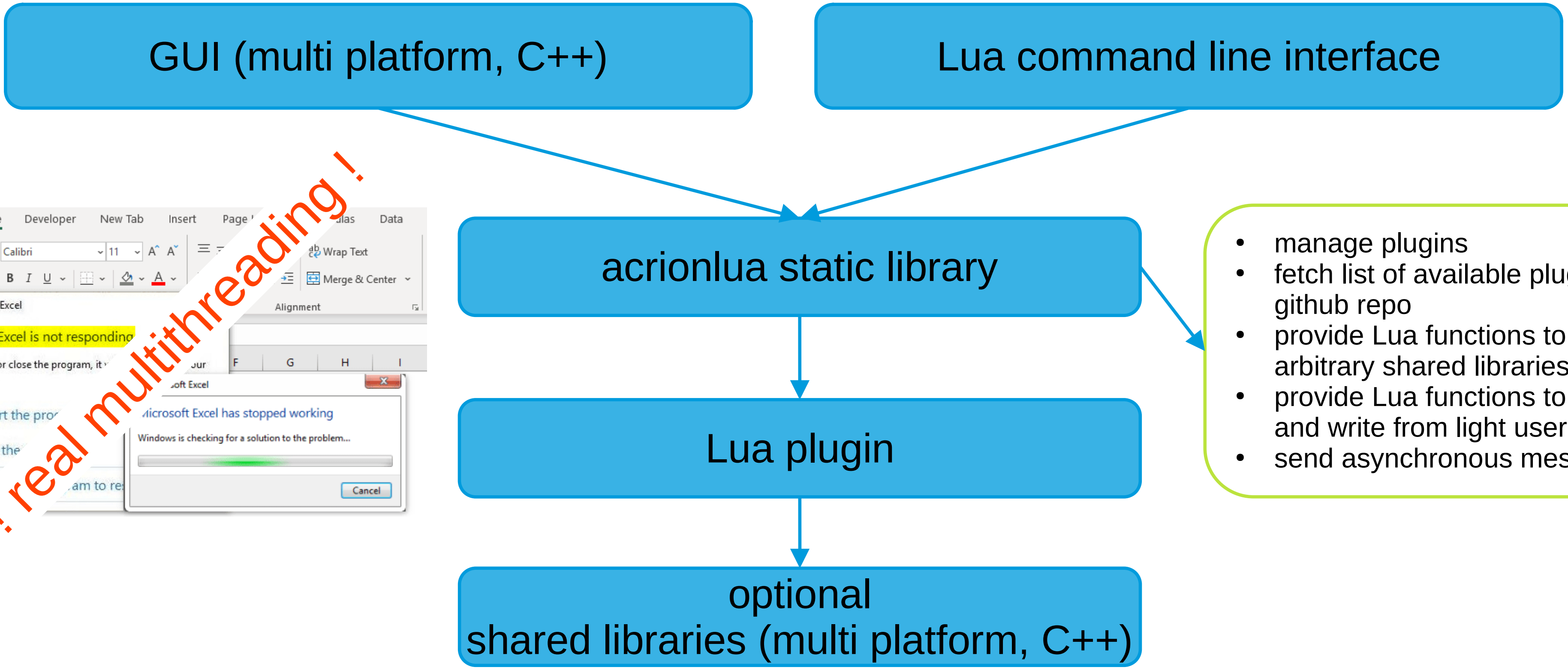
Microsoft Excel

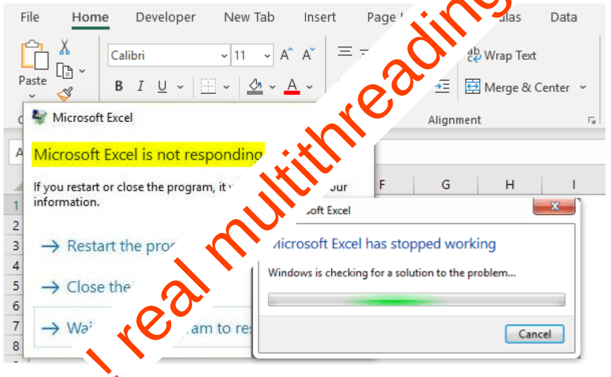
Microsoft Excel has stopped working

Windows is checking for a solution to the problem...

do Lua co-routines help?

take care of a common usability problem


GUI (multi platform, C++)
Lua command line interface
acrionlua static library
Lua plugin
optional
shared libraries (multi platform, C++)



Microsoft Excel

Microsoft Excel is not responding

If you restart or close the program, it … our information.

→ Restart the pro…

→ Close the …

→ Wai… …am to re…

…oft Excel

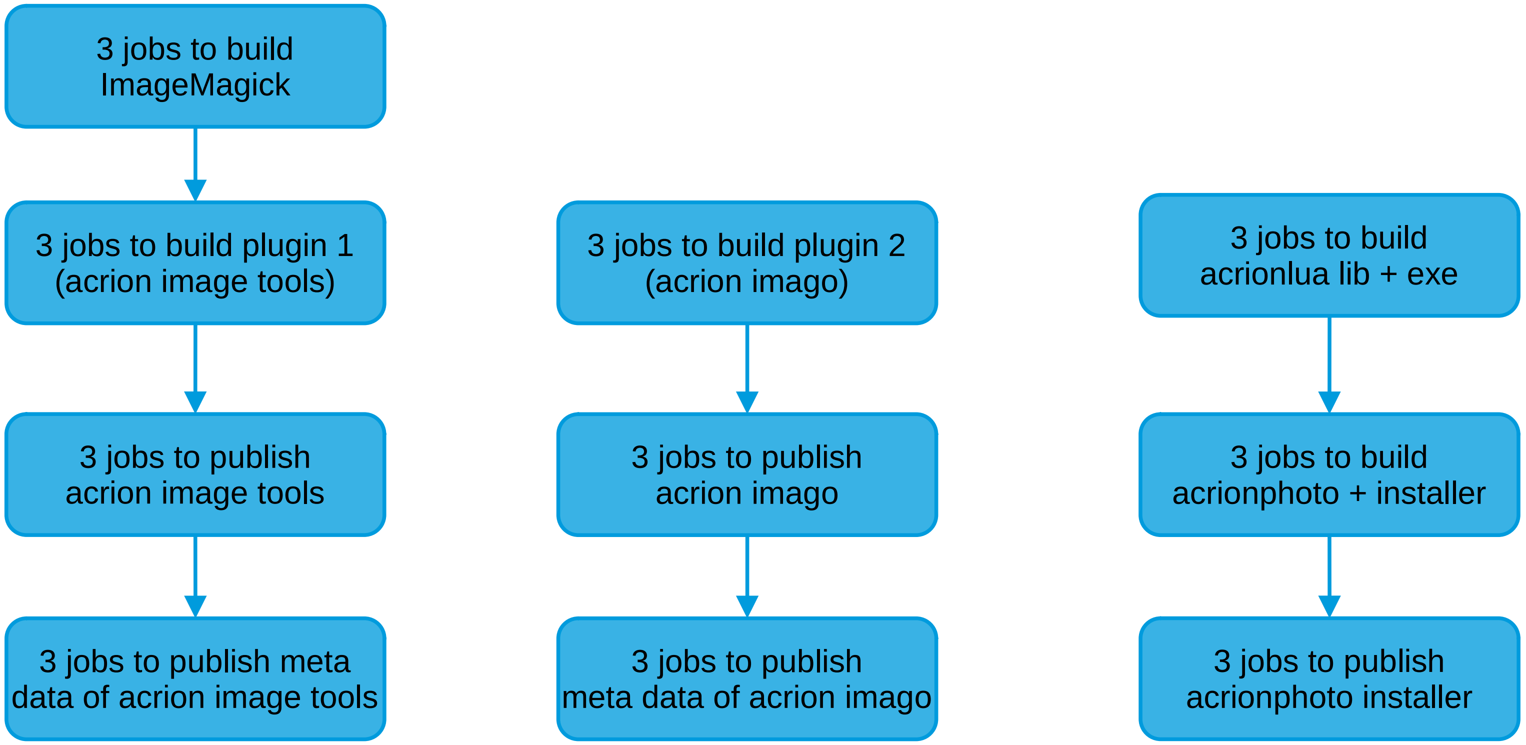
…icrosoft Excel has stopped working

Windows is checking for a solution to the problem...

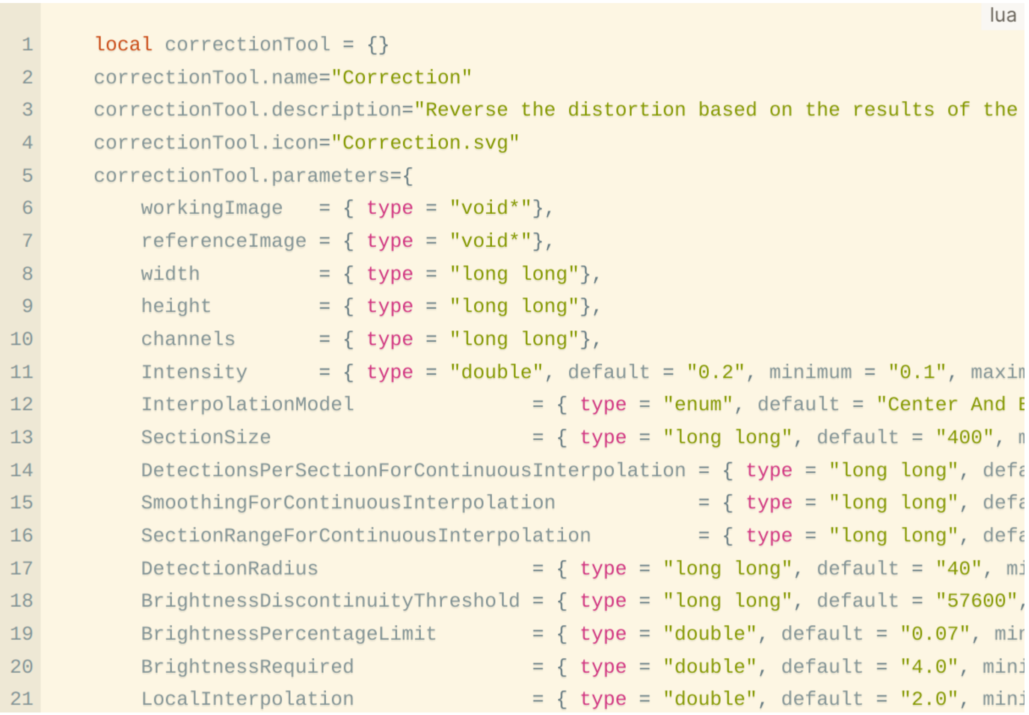
! real multithreading !

- manage plugins
- fetch list of available plugins from github repo
- provide Lua functions to load arbitrary shared libraries
- provide Lua functions to read and write from light user data
- send asynchronous messages

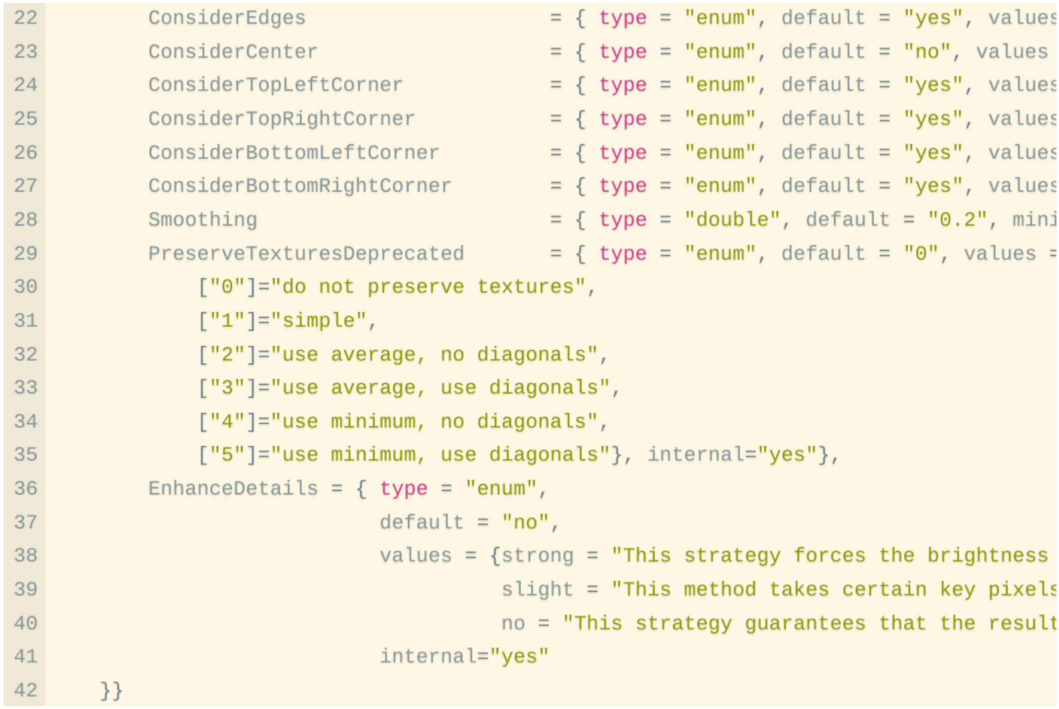
Jenkins


3 jobs to build ImageMagick
3 jobs to build plugin 1 (acrion image tools)
3 jobs to publish acrion image tools
3 jobs to publish meta data of acrion image tools
3 jobs to build plugin 2 (acrion imago)
3 jobs to publish acrion imago
3 jobs to publish meta data of acrion imago
3 jobs to build acrionlua lib + exe
3 jobs to build acrionphoto + installer
3 jobs to publish acrionphoto installer

Lua plugin interface



```lua
local correctionTool = {}
correctionTool.name="Correction"
correctionTool.description="Reverse the distortion based on the results of the
correctionTool.icon="Correction.svg"
correctionTool.parameters={
    workingImage   = { type = "void*"},
    referenceImage = { type = "void*"},
    width          = { type = "long long"},
    height         = { type = "long long"},
    channels       = { type = "long long"},
    Intensity      = { type = "double", default = "0.2", minimum = "0.1", maxim
    InterpolationModel                 = { type = "enum", default = "Center And E
    SectionSize                        = { type = "long long", default = "400", m
    DetectionsPerSectionForContinuousInterpolation = { type = "long long", defa
    SmoothingForContinuousInterpolation            = { type = "long long", defa
    SectionRangeForContinuousInterpolation         = { type = "long long", defa
    DetectionRadius                    = { type = "long long", default = "40", mi
    BrightnessDiscontinuityThreshold = { type = "long long", default = "57600",
    BrightnessPercentageLimit          = { type = "double", default = "0.07", mir
    BrightnessRequired                 = { type = "double", default = "4.0", mini
    LocalInterpolation                 = { type = "double", default = "2.0", mini
```

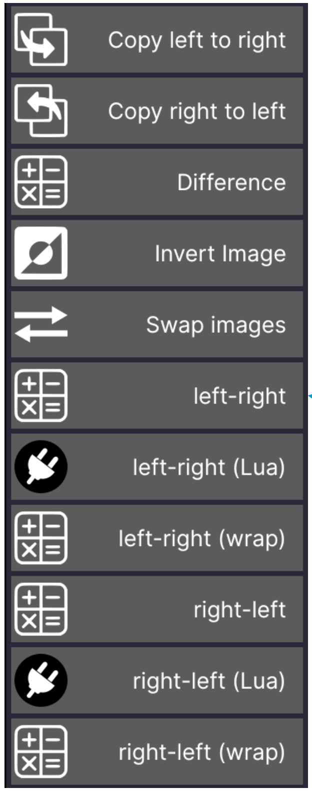


```lua
        ConsiderEdges                    = { type = "enum", default = "yes", values
        ConsiderCenter                   = { type = "enum", default = "no", values
        ConsiderTopLeftCorner            = { type = "enum", default = "yes", values
        ConsiderTopRightCorner           = { type = "enum", default = "yes", values
        ConsiderBottomLeftCorner         = { type = "enum", default = "yes", values
        ConsiderBottomRightCorner        = { type = "enum", default = "yes", values
        Smoothing                        = { type = "double", default = "0.2", mini
        PreserveTexturesDeprecated       = { type = "enum", default = "0", values =
            ["0"]="do not preserve textures",
            ["1"]="simple",
            ["2"]="use average, no diagonals",
            ["3"]="use average, use diagonals",
            ["4"]="use minimum, no diagonals",
            ["5"]="use minimum, use diagonals"}, internal="yes"},
        EnhanceDetails = { type = "enum",
                           default = "no",
                           values = {strong = "This strategy forces the brightness
                                     slight = "This method takes certain key pixels
                                     no = "This strategy guarantees that the result
                           internal="yes"
    }}
```

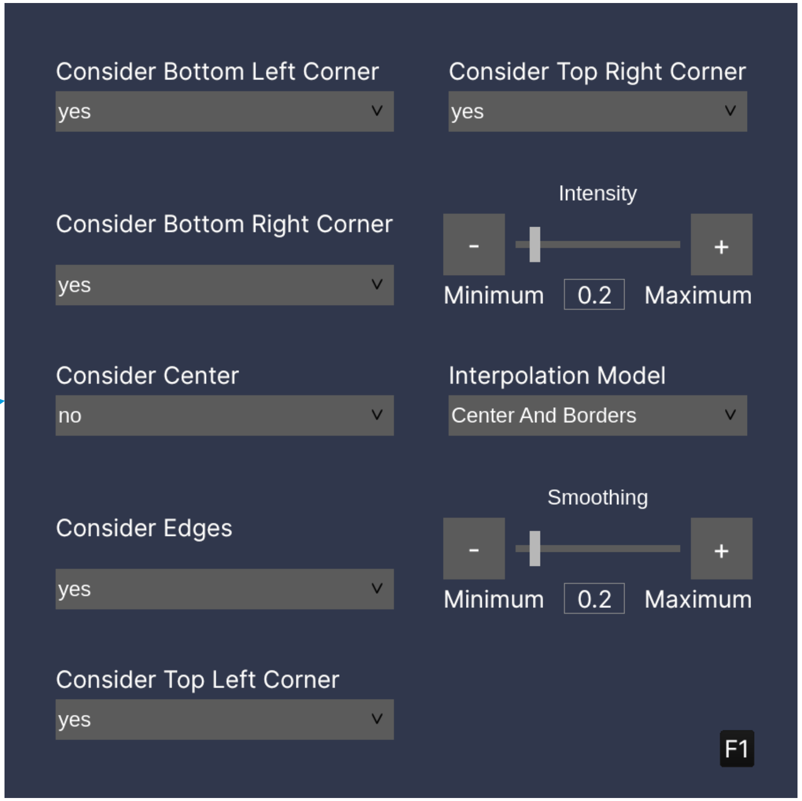
acrion.ch
github.com/acrion

Stefan Zipproth

showing plugin meta data in a GUI



generic widgets based on Lua code (plugin description)



Consider Bottom Left Corner: yes

Consider Top Right Corner: yes

Consider Bottom Right Corner: yes

Intensity: Minimum 0.2 Maximum

Consider Center: no

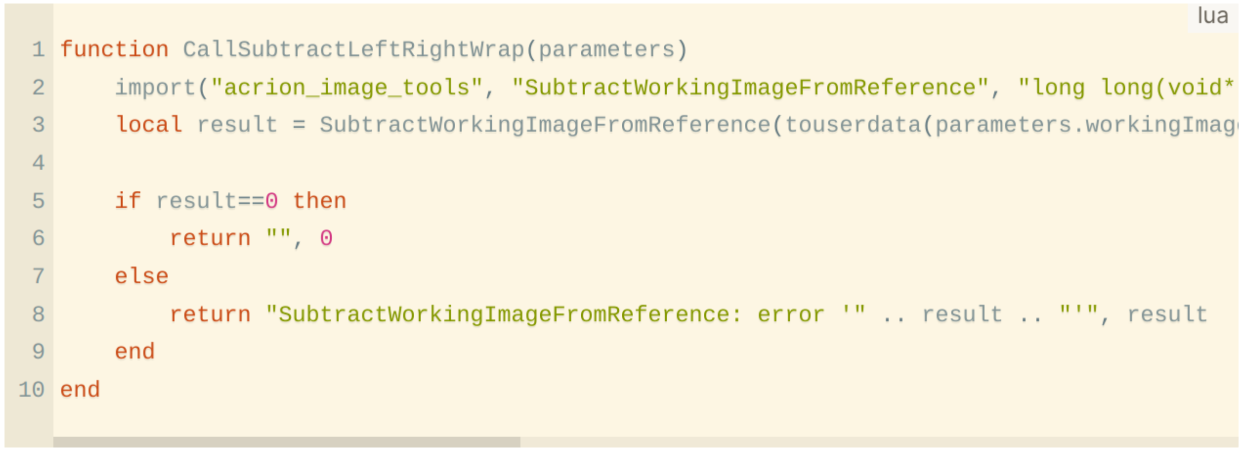
Interpolation Model: Center And Borders

Smoothing: Minimum 0.2 Maximum

Consider Edges: yes

Consider Top Left Corner: yes

new Lua function `import`

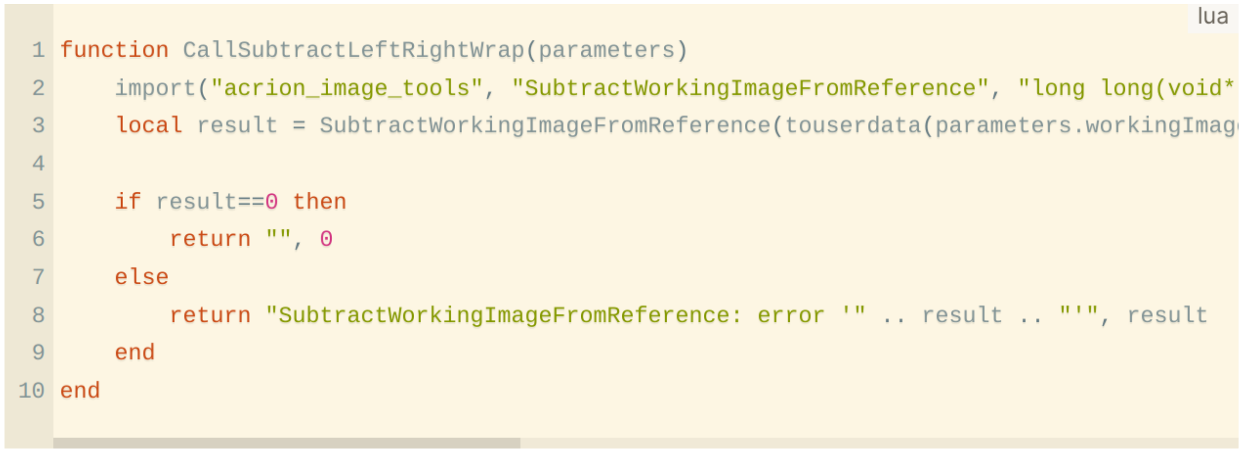


```lua
function CallSubtractLeftRightWrap(parameters)
    import("acrion_image_tools", "SubtractWorkingImageFromReference", "long long(void*
    local result = SubtractWorkingImageFromReference(touserdata(parameters.workingImag

    if result==0 then
        return "", 0
    else
        return "SubtractWorkingImageFromReference: error '" .. result .. "'", result
    end
end
```

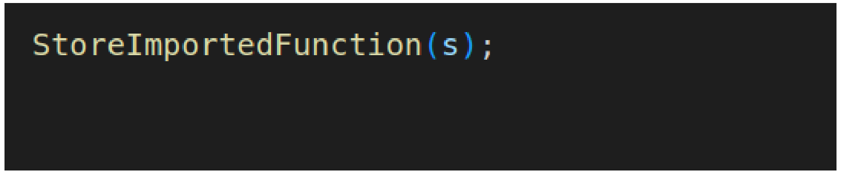
Lua plugin calls `import`

C++ function `Import`
- loads the shared library via `boost::dll_shared_library`

new Lua function `import`



```lua
function CallSubtractLeftRightWrap(parameters)
    import("acrion_image_tools", "SubtractWorkingImageFromReference", "long long(void*
    local result = SubtractWorkingImageFromReference(touserdata(parameters.workingImag

    if result==0 then
        return "", 0
    else
        return "SubtractWorkingImageFromReference: error '" .. result .. "'", result
    end
end
```



```
StoreImportedFunction(s);
```

boost C++ LIBRARIES

Lua plugin calls `import`

C++ function `Import`

- loads the shared library via `boost::dll_shared_library`
- stores the signature of the new lua function + reference to the shared library in a C++ map using the name of the new Lua function as key

new Lua function `import`

lua
```lua
function CallSubtractLeftRightWrap(parameters)
    import("acrion_image_tools", "SubtractWorkingImageFromReference", "long long(void*
    local result = SubtractWorkingImageFromReference(touserdata(parameters.workingImag

    if result==0 then
        return "", 0
    else
        return "SubtractWorkingImageFromReference: error '" .. result .. "'", result
    end
end
```

Lua plugin calls `import`

C++ function `Import`
- loads the shared library via `boost::dll_shared_library`
- stores the signature of the new lua function + reference to the shared library in a C++ map using the name of the new Lua function as key
- registers a new lua function that calls the C++ function `CallDllFunction`

all imported functions call the same C++ function

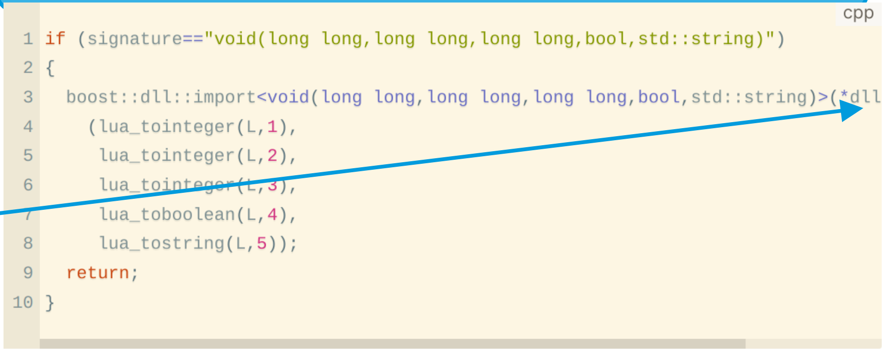
```cpp
StoreImportedFunction(s);
lua_pushcfunction(L, CallDllFunction);
lua_setglobal(L, s.functionName.c_str());
```

Calling the imported function

- `CallDllFunction` gets the name of the calling Lua function via `lua_getinfo`.
- In the map it finds the required data
- It searches the signature...

faster method
(default)

… through a code-generated chain of if clauses, e.g.



```cpp
if (signature=="void(long long,long long,long long,bool,std::string)")
{
  boost::dll::import<void(long long,long long,long long,bool,std::string)>(*dll
    (lua_tointeger(L,1),
     lua_tointeger(L,2),
     lua_tointeger(L,3),
     lua_toboolean(L,4),
     lua_tostring(L,5));
  return;
}
```

lua_find_signature_if_chain.cpp (3497 lines, 2 MB)

Calling the imported function

- `CallDllFunction` gets the name of the calling Lua function via `lua_getinfo`.
- In the map it finds the required data
- It searches the signature...

… in a C++ `std::map` of `std::function` instances that are pre-initialized like e.g.

lua_find_signature_map.cpp
(608 lines, 212 KB)

```cpp
callDllFunction["void*(void*,void*,long long,double)"] =
  [](lua_State* L,
     std::shared_ptr<boost::dll::shared_library> dll,
     std::string functionName)
  {
    lua_pushlightuserdata(L,
                           boost::dll::import<void*(void*,void*,long long,double
                             (lua_touserdata(L,1),
                              lua_touserdata(L,2),
                              lua_tointeger(L,3),
                              lua_tonumber(L,4)));
  };
```

… and called as follows …

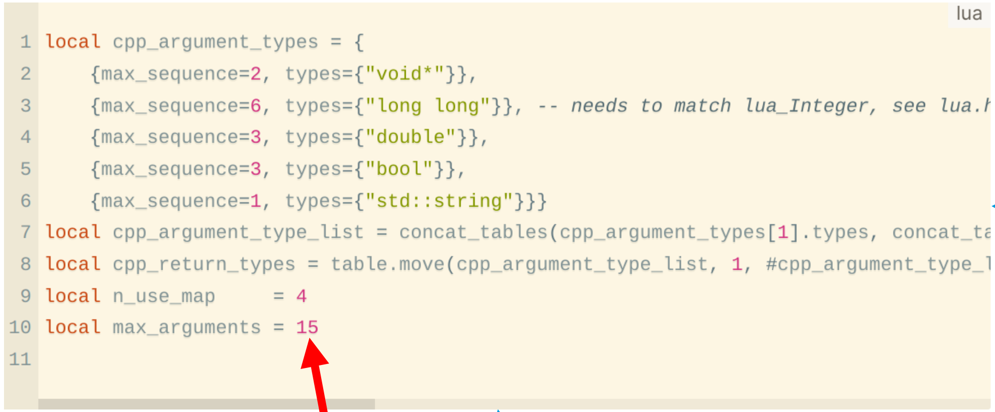
```cpp
callDllFunction[s.signature](L, s.dll, s.functionName);
```

… through a code-generated chain of if clauses, e.g.

```cpp
if (signature=="void(long long,long long,long long,bool,std::string)")
{
  boost::dll::import<void(long long,long long,long long,bool,std::string)>(*dll
    (lua_tointeger(L,1),
     lua_tointeger(L,2),
     lua_tointeger(L,3),
     lua_toboolean(L,4),
     lua_tostring(L,5));
  return;
}
```

lua_find_signature_if_chain.cpp (3497 lines, 2 MB)

what’s possible?



```lua
local cpp_argument_types = {
    {max_sequence=2, types={"void*"}},
    {max_sequence=6, types={"long long"}}, -- needs to match lua_Integer, see lua.
    {max_sequence=3, types={"double"}},
    {max_sequence=3, types={"bool"}},
    {max_sequence=1, types={"std::string"}}}
local cpp_argument_type_list = concat_tables(cpp_argument_types[1].types, concat_ta
local cpp_return_types = table.move(cpp_argument_type_list, 1, #cpp_argument_type_l
local n_use_map     = 4
local max_arguments = 15
```

plugin developers may use wither

(1) only Lua
(2) Lua with own C++ libraries

accessing 3rd party C++ libraries

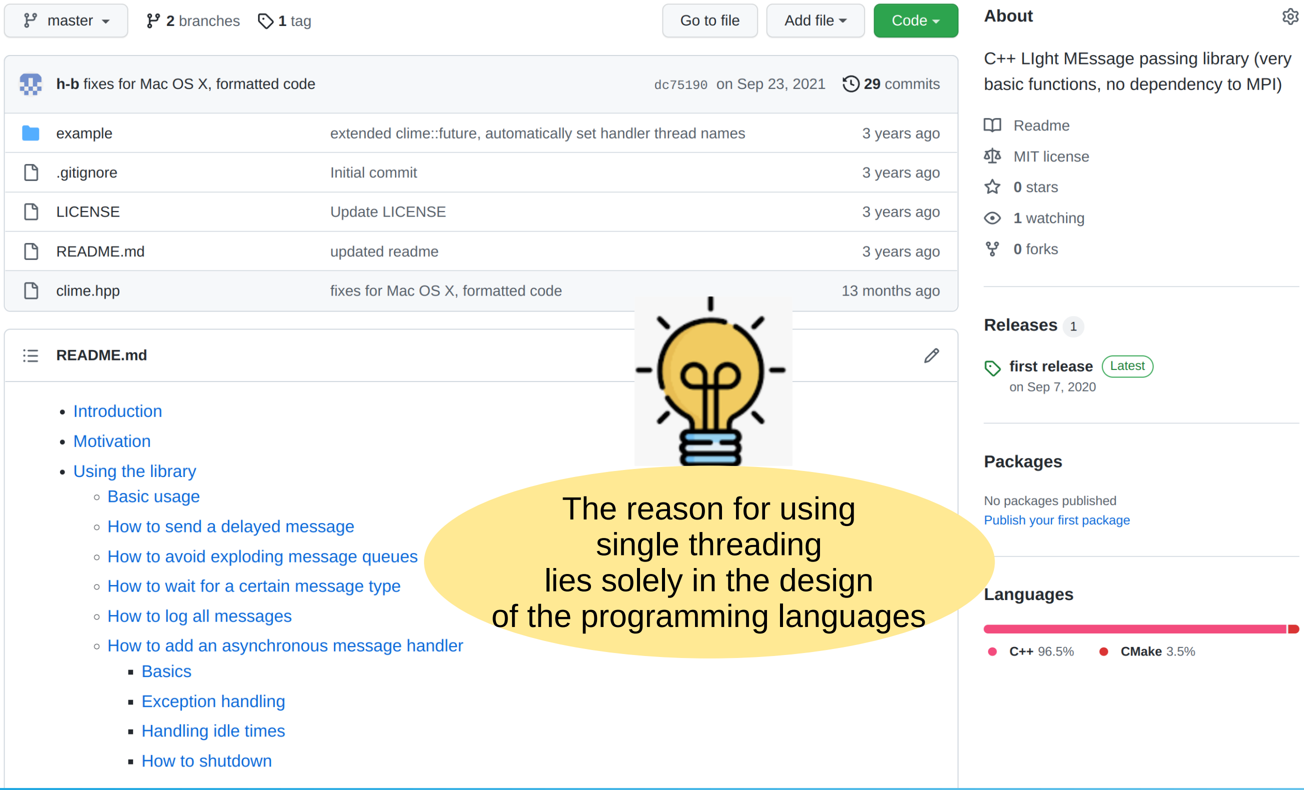
(1) is possible in many cases
(2) can be made possible easily

multithreading

clime
C++ **LI**ght **M**essage
passing library

- used in commercial product
- prevents common multithreading issues

github.com/h-b/clime



master · 2 branches · 1 tag · Go to file · Add file · Code

h-b fixes for Mac OS X, formatted code	dc75190 on Sep 23, 2021	29 commits
example	extended clime::future, automatically set handler thread names	3 years ago
.gitignore	Initial commit	3 years ago
LICENSE	Update LICENSE	3 years ago
README.md	updated readme	3 years ago
clime.hpp	fixes for Mac OS X, formatted code	13 months ago

README.md

- Introduction
- Motivation
- Using the library
  - Basic usage
  - How to send a delayed message
  - How to avoid exploding message queues
  - How to wait for a certain message type
  - How to log all messages
  - How to add an asynchronous message handler
    - Basics
    - Exception handling
    - Handling idle times
    - How to shutdown

About

C++ LIght MEssage passing library (very basic functions, no dependency to MPI)

Readme
MIT license
0 stars
1 watching
0 forks

Releases 1

first release Latest
on Sep 7, 2020

Packages

No packages published
Publish your first package

Languages

● C++ 96.5% ● CMake 3.5%

The reason for using single threading lies solely in the design of the programming languages

acrion.ch
github.com/acrion

Stefan Zipproth

what’s required to make clime available in Lua?

care about Lua state

github.com/h-b/clime

cpp

```cpp
struct lua_State;

namespace ACRIONLUA
{
    namespace fs = std::filesystem;

    class Lua
    {
    public:
        Lua(fs::path luaFilePath);
        fs::path    GetPath() const { return _luaFilePath; }
        Table       GetTable(const char* functionname) const;
        std::string GetLicensee() const;
        std::string RunPlugin(const char* functionname, Table parameters) const;

    private:
        static Table LoadTable(lua_State* L);
        static void  PushStringTable(lua_State* L, const StringTable& parameters);
        static void  PushTable(lua_State* L, const Table& parameters);
        static void  laction(int i);

        fs::path          _luaFilePath;
        lua_State*        _L{nullptr};
        static lua_State* _luaStaticState;
        static std::mutex _luaStaticStateMutex;
    };
}

```

what’s required to make clime available in Lua?

register function to send a message

github.com/h-b/clime

cpp

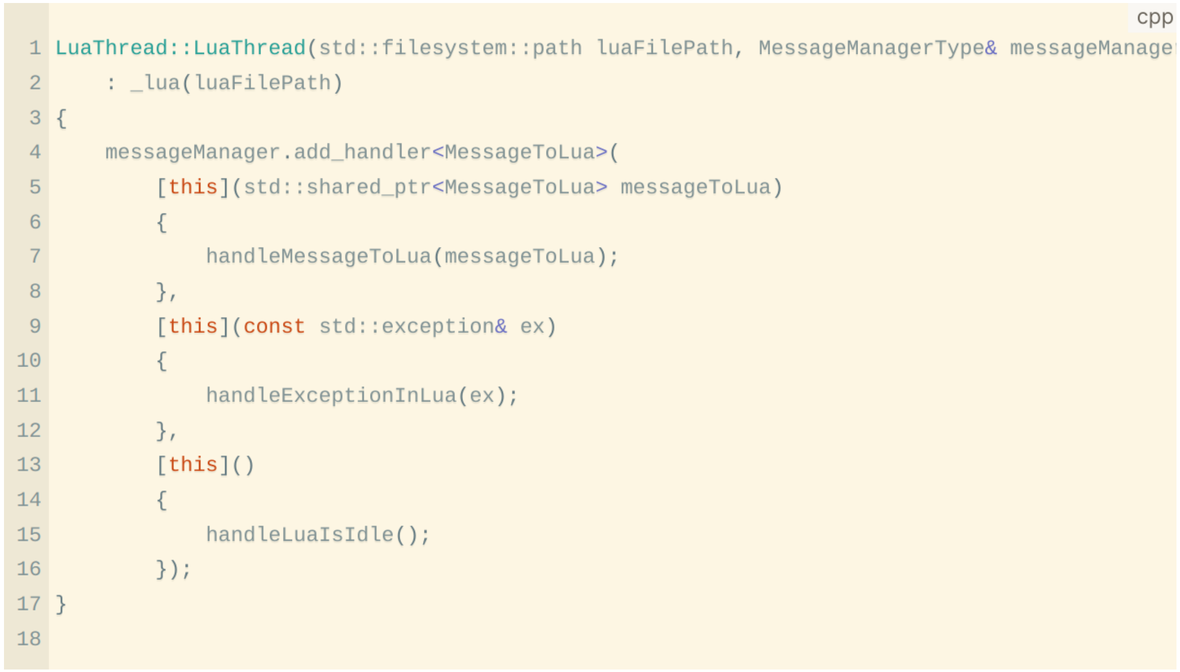
```cpp
namespace ACRIONLUA
{
    struct Table;
    typedef std::map<std::string, std::string> StringTable;
    typedef std::map<std::string, Table>       SubTables;

    struct ACRIONLUA_LIBRARY_EXPORT Table // We call a "Table" the composition of a table
    {
        StringTable data;
        SubTables   subTables;
    };

    class ACRIONLUA_LIBRARY_EXPORT MessageToLua
    {
    public:
        std::string name;
        Table       parameters;
    };
```

```cpp

    class ACRIONLUA_LIBRARY_EXPORT MessageFromLua
    {
    public:
        std::string name;
        Table       parameters;
    };

    using MessageManagerType = clime::message_manager<MessageToLua, MessageFromLua>;

    extern ACRIONLUA_LIBRARY_EXPORT std::string shortenClassName(const std::string& classN
    extern ACRIONLUA_LIBRARY_EXPORT int         demangling_status_;
}

#define CLASS_NAME_ ::ACRIONLUA::shortenClassName(__DEMANGLED_CLASS_NAME(::ACRIONLUA::dema

// convenience macros to make sending messages more readable
#define SEND_TO_LUA(name, parameters) ::ACRIONLUA::LuaThreadPool::Get().SendMessage(std::m
```

what's required to make clime available in Lua?

register function to handle a message

github.com/h-b/clime



cpp

```
LuaThread::LuaThread(std::filesystem::path luaFilePath, MessageManagerType& messageManage
    : _lua(luaFilePath)
{
    messageManager.add_handler<MessageToLua>(
        [this](std::shared_ptr<MessageToLua> messageToLua)
        {
            handleMessageToLua(messageToLua);
        },
        [this](const std::exception& ex)
        {
            handleExceptionInLua(ex);
        },
        [this]()
        {
            handleLuaIsIdle();
        });
}

```

multithreading

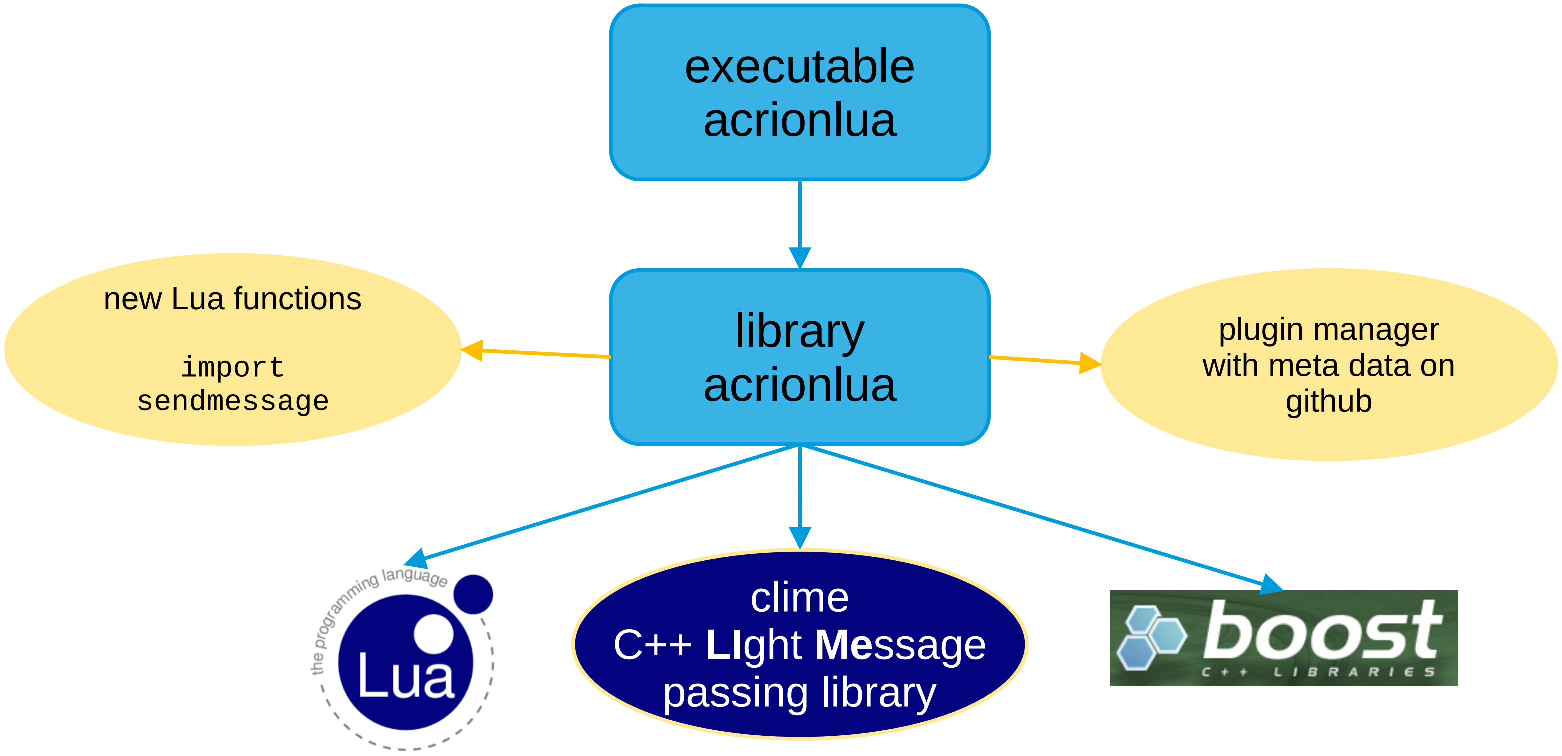
github.com/h-b/clime

suggestions on how

to design the new messaging functions

in acrionlua?

Summary


executable acrionlua
library acrionlua
new Lua functions
import sendmessage
plugin manager with meta data on github
the programming language
Lua
clime
C++ LIght Message passing library
boost
C++ LIBRARIES