

Bridging the gap

Streamlining Pharo FFI Bindings

Esteban Lorenzano, ESUG 2024



No man is an island...



*No **smalltalk** is an island...*



*No **pharo** is an island...*



Foreign Function Interface (FFI) on Pharo

- [2009] **FFI Plugin**: no callbacks, not portable.
- [2011] **Alien**: callbacks, function as objects.
- [2013] **NativeBoost**: Very nice design, ASMJIT, callbacks, executable memory manager. Not portable.
- [2016] **uFFI**: libffi (callbacks, portable, tested)
- [2019] **uFFI**: libffi/ThreadedFFI (callbacks, portable, tested, multi-threading support)

We now have a very mature and reliable infrastructure :)

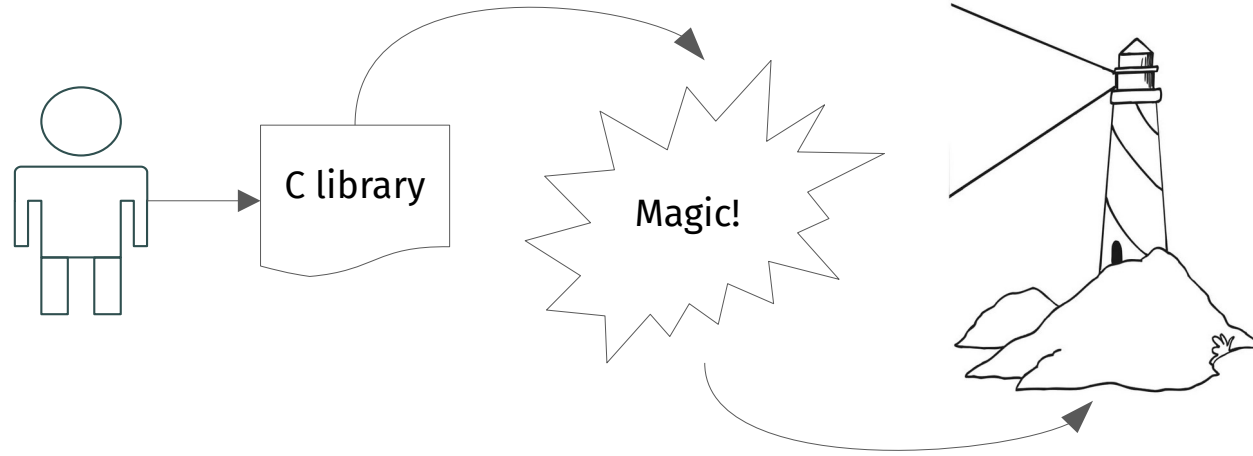


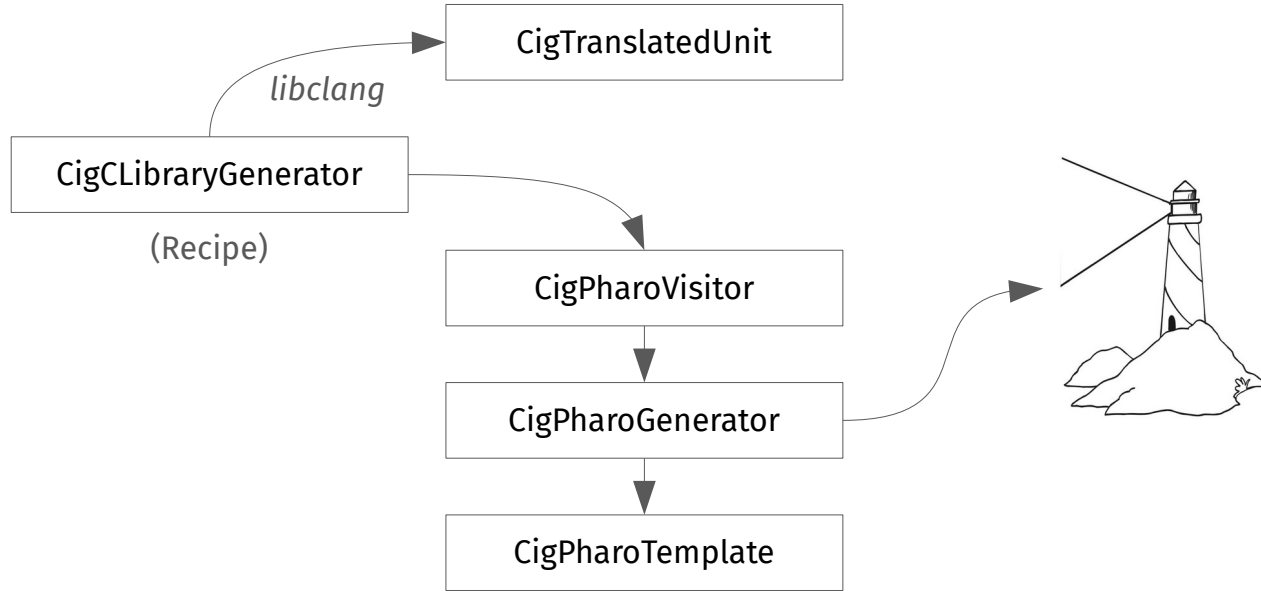
But you still need to manually create your bindings !

And this is often a painful process of copy & paste from a C header into your project, as classes or methods.



Pharo-CIG: C Interface Generator





Raylib demo

<https://www.raylib.com>

« raylib is a simple and easy-to-use library to enjoy videogames programming »



How a recipe looks

```
1 "a recipe"  
2 (lib := CigCLibraryGenerator new)  
3   prefix: 'Ray';  
4   libraryName: 'raylib';  
5   import: '/home/esteban/dev/vm/raylib/src/raylib.h';  
6   cIncludePath: '/home/esteban/dev/vm/raylib/src';  
7   useMainThread.  
8  
9 lib generate.
```



How the code looks

The screenshot shows an IDE window titled "Method: LibRaylib>>drawText:posX:posY:fontSize:color:". The interface is divided into several panes:

- Left Pane:** A tree view showing the "LibRaylib" package structure, including "Base" and "Library" sub-packages.
- Middle Pane:** A class hierarchy for "LibRaylib", listing classes like "LibRaylib", "RayBaseObject", "RayAudioBuffer", "RayAudioProcessor", "RayEnumeration", "RayBlendMode", "RayCameraMode", "RayCameraProjection", "RayConfigFlags", "RayCubemapLayout", "RayFontType", "RayGamepadAxis", and "RayGamepadButton".
- Right Pane:** A list of methods, with "drawText:posX:posY:fontSize:color" selected and highlighted in blue.
- Bottom Pane:** The source code for the selected method, showing a public method signature and an ffiCall implementation.

The code in the bottom pane is as follows:

```
drawText: text posX: posX posY: posY fontSize: fontSize color: color

self ffiCall: #(void DrawText(const char* text, int posX, int posY, int fontSize, Color color))
```

At the bottom of the IDE window, there is a status bar with the text "1/3 [1]" on the left and "public extension F +L W" on the right.



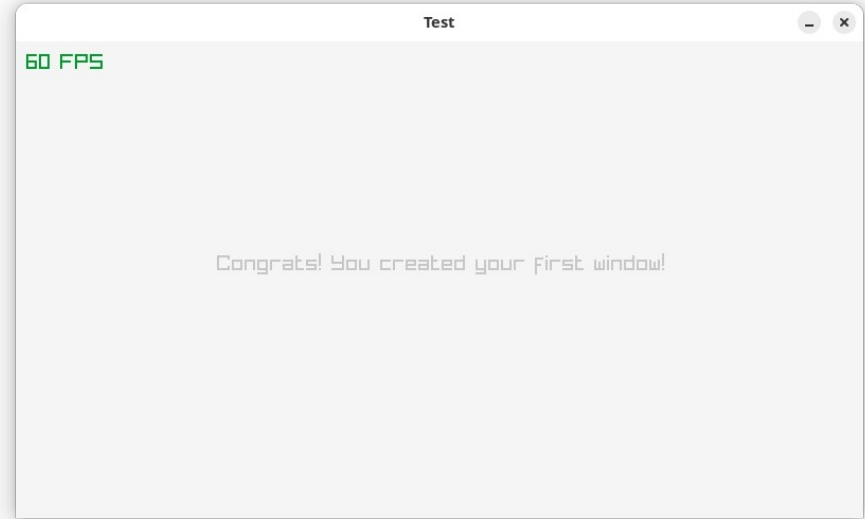
Challenges

- FFIFunctionParser → CigFunctionParser
- size_t → int
- Libc and other standard libraries usage, but I do not want to import everything ! → from:import:
- ...



How coding looks

```
1 "example of raylib"
2 ray := LibRaylib uniqueInstance.
3
4 ray initWindowWidth: 800 height: 450 title: 'Test'.
5 ray setTargetFPS: 60.
6 [ ray windowShouldClose = 0 ] whileTrue: [
7
8     ray beginDrawing.
9     ray clearBackground: RayColor white.
10    ray
11        drawText: 'Congrats! You created your first window!'
12        posX: 190
13        posY: 200
14        fontSize: 20
15        color: RayColor lightGray.
16    ray drawFPSPosX: 10 posY: 10.
17    ray endDrawing ].
18
19 ray closeWindow.
--
```



(Translation from <https://www.raylib.com/examples.html>)



libxml2 demo

<https://gitlab.gnome.org/GNOME/libxml2>



openssl demo
<https://www.openssl.org>

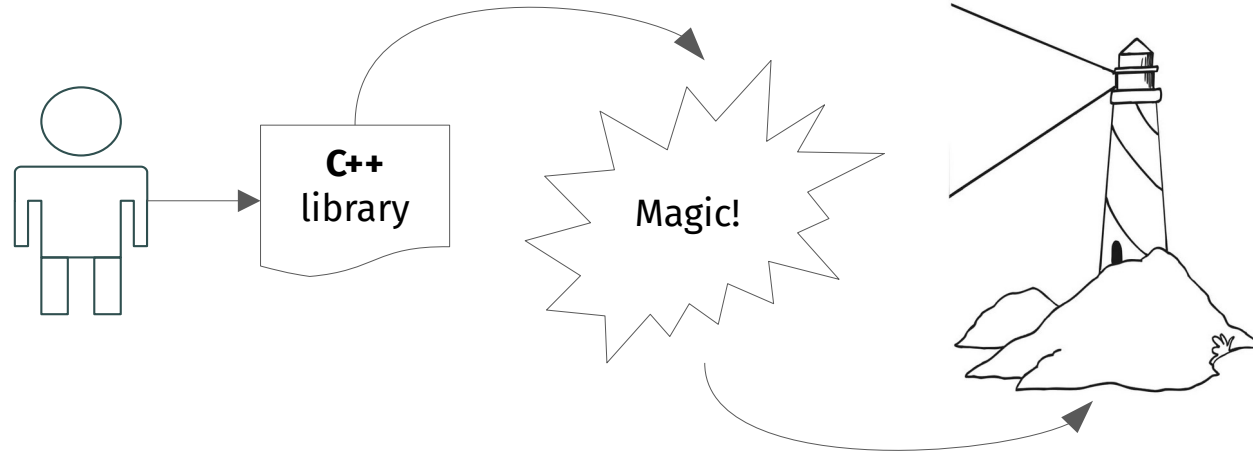


But many useful libraries are made in C++ !

And there is not standard ABI to be able to use them
is hard and painful to create C bindings for them.

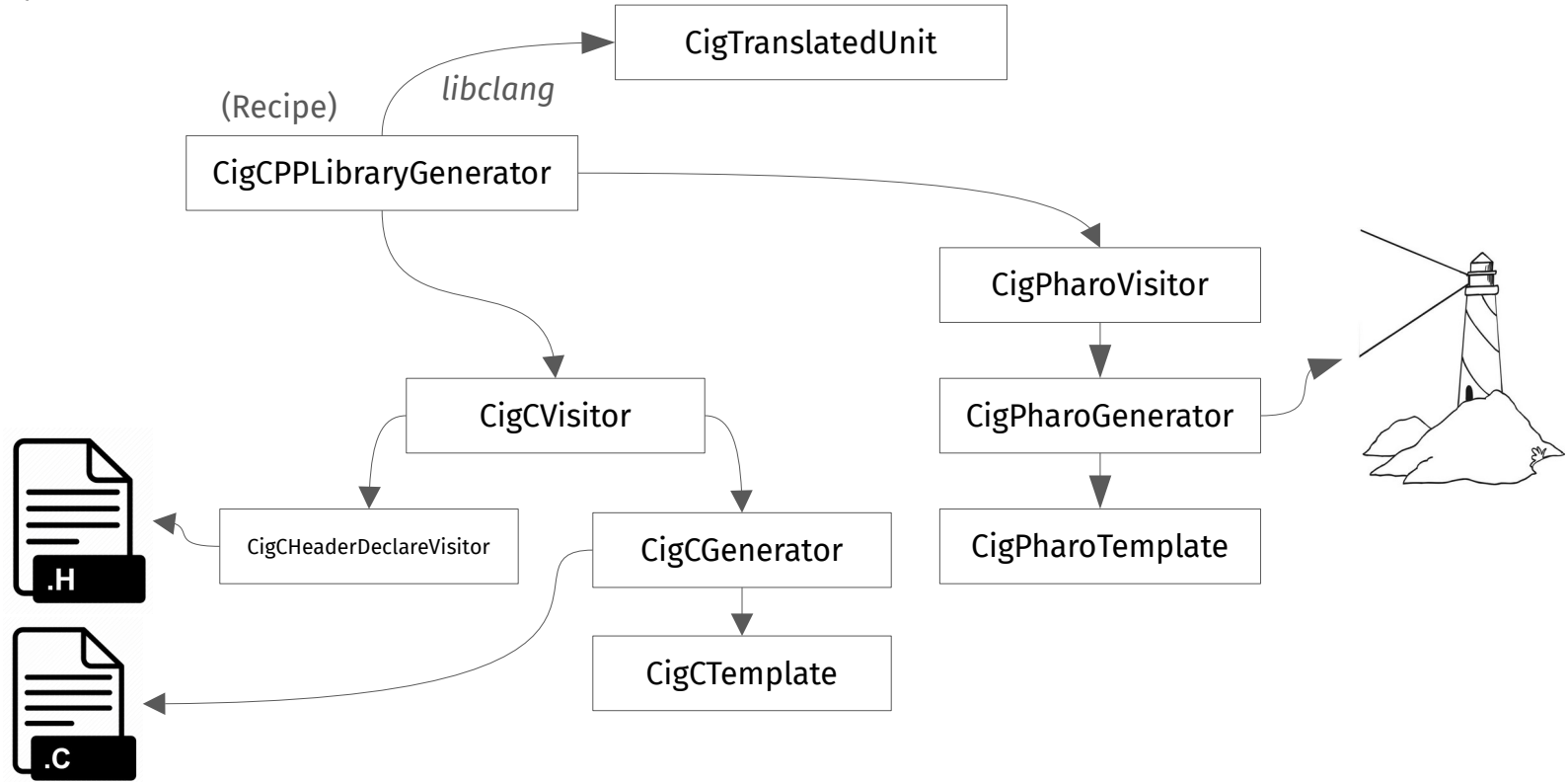


Pharo-CIG: C++ Interface Generator



This seems easy...





C++ “challenges”... sigh...

- Structs are classes
- Classes are structs
- Structs (and unions) can be anonymous
- && is not just “and”
- `std::lib`
 - Iterators, arrays, string... anything you want...
 - `shared_ptr`, memory handling, “pseudo” declarations...
- Namespaces
- Aliases
- Templates
 - Template classes
 - Template functions
 - Template everything
- Why this thing exists at all?



libnoise demo

<https://libnoise.sourceforge.net>

« libnoise is a portable **C++** library that is used to generate coherent noise, a type of smoothly-changing noise. libnoise can generate Perlin noise, ridged multifractal noise, and other types of coherent-noise. »



C++ recipe overview (1/2)

```
1 "example c++ basic"
2 (noise := CigCppLibraryGenerator new)
3   prefix: 'No';
4   libraryName: 'noise';
5   import: '/home/esteban/dev/vm/libnoise/include/noise/noise.h';
6   cIncludePath: '/home/esteban/dev/vm/libnoise/include';
7   cIncludePath: '/home/esteban/dev/vm/libnoise/include/noise';
8   cLib: 'noise'.
9
10 noise generate.
```



C++ recipe overview (2/2)

```
12 "example c++ with namespaces"
13 (noiseutils := CigCppLibraryGenerator new)
14   prefix: 'Nu';
15   libraryName: 'noiseutils';
16   import: '/home/esteban/dev/vm/noiseutils/noiseutils.h';
17   cIncludePath: '/home/esteban/dev/vm/noiseutils';
18   cIncludePath: '/home/esteban/dev/vm/libnoise/include';
19   cIncludePath: '/home/esteban/dev/vm/libnoise/include/noise';
20   cIncludePath: '../noise';
21   cLib: 'noise';
22   cLib: 'noiseutils';
23   namespace: NoNoiseNamespace;
24   namespace: NoModelNamespace;
25   namespace: NoModuleNamespace.
26
27 noiseutils generate.
```



How the code looks

The screenshot shows an IDE window titled "Method: NoCache>>getValueX:y:z". The interface is divided into several panes:

- Left Pane:** A tree view showing the project structure. "LibNoise" is expanded, and "Module" is selected.
- Middle Pane:** A class hierarchy view. "FFIOpaqueObject" is expanded to "NoBaseObject", which is expanded to "NoModule". "NoCache" is selected.
- Right Pane:** A list of methods for "NoCache". "finalizing" is expanded to show "public" and "overrides".
- Bottom Pane:** The source code for the "getValueX:y:z" method. The code is as follows:

```
getValueX: x y: y z: z
    self ffiCall: #(double no_noise_module_Cache_GetValue(no_noise_module_Cache* self, double x, double y, double z))
```

At the bottom of the IDE, there are tabs for "NoCache", "Comment", "getValueX:y:z", and "Inst. side me". The status bar at the bottom left shows "1/4 [1]" and the bottom right shows "public extension F +L W".



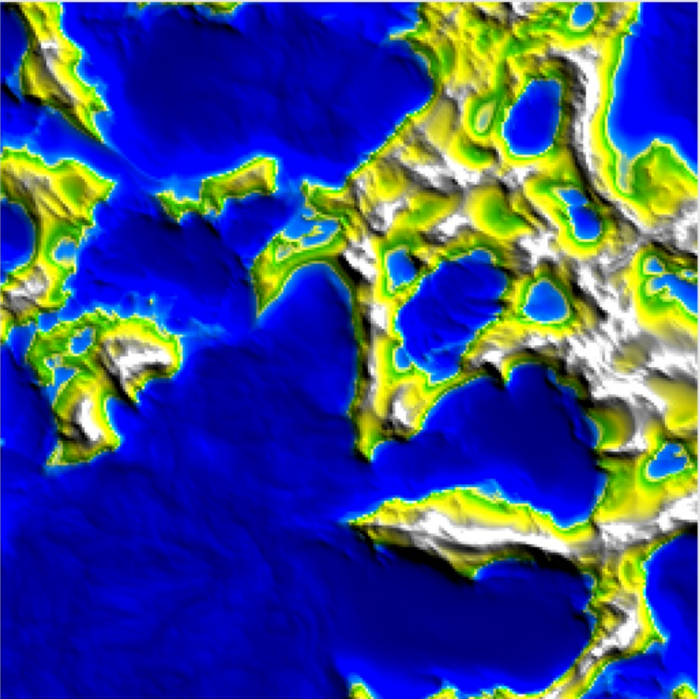
How coding looks

```
1 | "this is taken from libnoise tutorial"
2 | seed mountainTerrain baseFlatTerrain flatTerrain terrainType terrainSelector finalTerrain heightMap heightMapBuilder renderer image writer |
3
4 seed := 42.
5
6 mountainTerrain := NoRidgedMulti new autoRelease.
7 baseFlatTerrain := NoBilow new autoRelease.
8 baseFlatTerrain setFrequency: 2.0.
9
10 flatTerrain := NoScaleBias new autoRelease.
11 flatTerrain setSourceModuleIndex: 0 sourceModule: baseFlatTerrain.
12 flatTerrain setScale: 0.125.
13 flatTerrain setBias: -0.75.
14
15 terrainType := NoPerlin new autoRelease.
16 terrainType setFrequency: 0.5.
17 terrainType setPersistence: 0.25.
18 terrainType setSeed: seed.
19
20 terrainSelector := NoSelect new autoRelease.
21 terrainSelector setSourceModuleIndex: 0 sourceModule: flatTerrain.
22 terrainSelector setSourceModuleIndex: 1 sourceModule: mountainTerrain.
23 terrainSelector setControlModule: terrainType.
24 terrainSelector setBoundsLowerBound: 0.0 upperBound: 1000.0.
25 terrainSelector setEdgeFalloff: 0.125.
26
27 finalTerrain := NoTurbulence new autoRelease.
28 finalTerrain setSourceModuleIndex: 0 sourceModule: terrainSelector.
29 finalTerrain setFrequency: 1.0.
30 finalTerrain setPower: 0.125.
31
32 heightMap := NuNoiseMap new autoRelease.
33 heightMapBuilder := NuNoiseMapBuilderPlane new autoRelease.
34
35 heightMapBuilder setSourceModule: finalTerrain.
36 heightMapBuilder setDestNoiseMap: heightMap.
37 heightMapBuilder setDestSizeDestWidth: 256 destHeight: 256.
38 heightMapBuilder
39   setBoundsLowerXBound: 6.0
40   upperXBound: 10.0
41   lowerZBound: 1.0
42   upperZBound: 5.0.
43 heightMapBuilder build.
44
45 renderer := NuRendererImage new autoRelease.
46 image := NuImage new autoRelease.
47
48 renderer setSourceNoiseMap: heightMap.
49 renderer setDestImage: image.
50
51 renderer clearGradient.
52 renderer addGradientPointGradientPos: -1.0000 gradientColors: (NuColor newR: 0 g: 0 b: 128 a: 255) autoRelease.
53 renderer addGradientPointGradientPos: -0.2500 gradientColors: (NuColor newR: 0 g: 0 b: 255 a: 255) autoRelease.
54 renderer addGradientPointGradientPos: 0.0000 gradientColors: (NuColor newR: 0 g: 128 b: 255 a: 255) autoRelease.
55 renderer addGradientPointGradientPos: 0.0625 gradientColors: (NuColor newR: 240 g: 240 b: 64 a: 255) autoRelease.
56 renderer addGradientPointGradientPos: 0.1250 gradientColors: (NuColor newR: 32 g: 160 b: 0 a: 255) autoRelease.
57 renderer addGradientPointGradientPos: 0.3750 gradientColors: (NuColor newR: 224 g: 224 b: 0 a: 255) autoRelease.
58 renderer addGradientPointGradientPos: 0.7500 gradientColors: (NuColor newR: 128 g: 128 b: 128 a: 255) autoRelease.
59 renderer addGradientPointGradientPos: 1.0000 gradientColors: (NuColor newR: 255 g: 255 b: 255 a: 255) autoRelease.
60
61 renderer enableLight: true.
62 renderer setLightContrast: 3.0.
63 renderer setLightBrightness: 2.0.
64
65 renderer render.
66
67 writer := NuWriterBMP new.
68 writer setSourceImage: image.
69 writer setDestFilename: 'tutorial.bmp'.
70 writer writeDestFile.
71
72 (BMPReadWriteWriter formFromFileName: 'tutorial.bmp') scaledToSize: 512@512
```

Inspector on Form(512x512x32)

a Form (Form(512x512x32))

Form Raw Breakpoints Meta



1 self

(lol, go to LibNoise class>>example:)



Remarks

- Not all C++ libraries are designed to be used outside C++
- Good C libraries have a lot of information we can use
 - Function and argument naming
 - class comments
 - functions comments
- ... but this is not often the case, in consequence names are still ugly :)
- **IMPORTANT:** is also possible to build “on top” of the generated bindings, to get correct abstractions



Status

- Test coverage **~60%**
- **C** is “mostly done”
 - Most C libraries should parse, but there are missing things that should emerge by using it
 - Variadic support missing
 - #define support missing
- **C++** is not :)
 - std::lib
 - Complex declarations like moving references
 - ?
- **Documentation...**



*... send not to know
For whom the bell tolls,
It tolls for thee.*

- John Donne



Help wanted

- For its nature, this project will always be a “beta”
- You can contribute
 - By parsing libraries and publish them
 - By helping to improve the generators
 - ... and you can also ask for your library bindings to be generated, we will do our best



pharo-cig

<https://github.com/estebanlm/pharo-cig>

