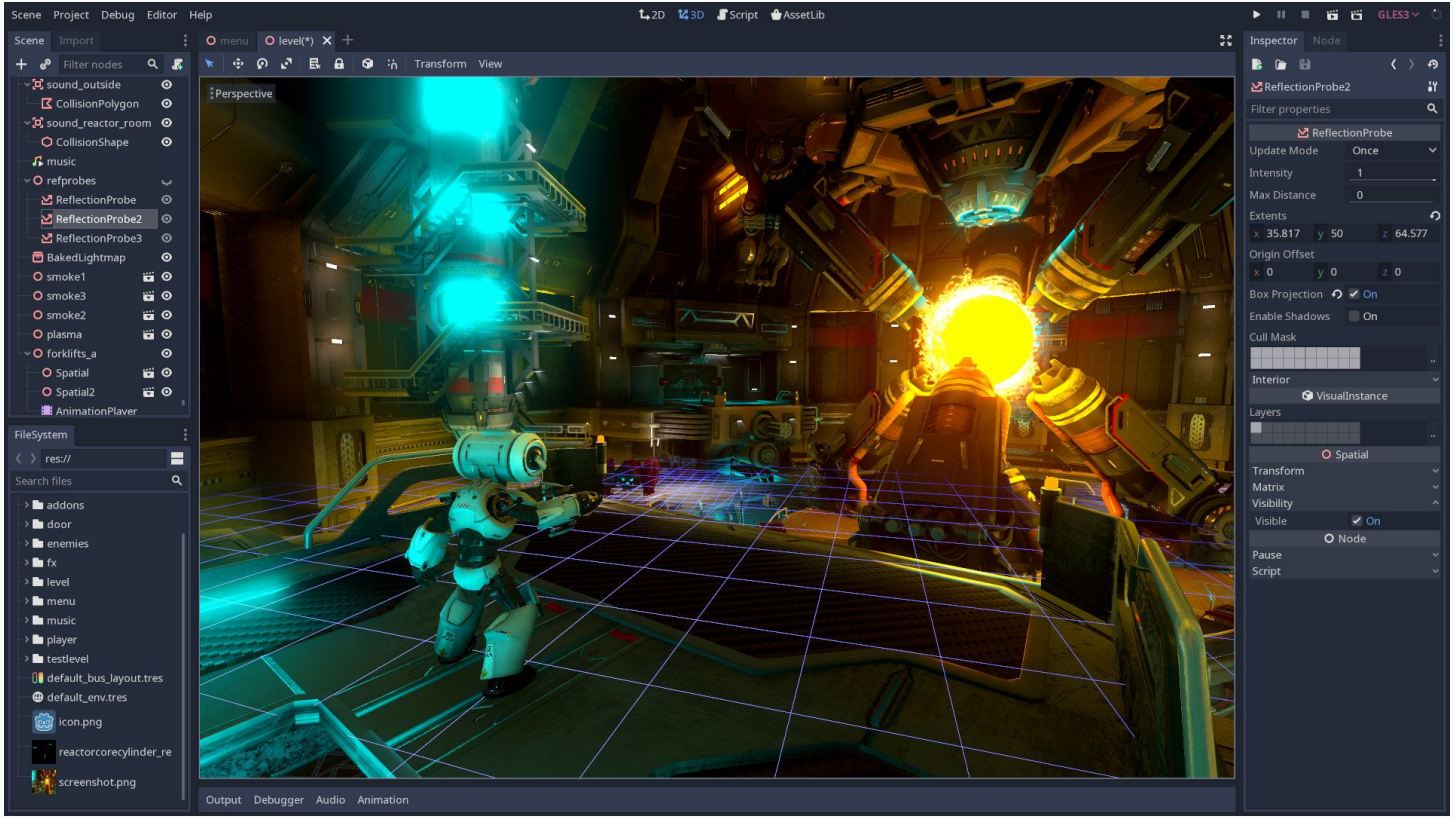
Bringing Python to Godot

01/02/2020

Emmanuel Leblond - https://github.com/touilleMan/godot-python

Godot ?



GODOT
Game engine

Open source (MIT)

Full featured

Linux support ❤️❤️❤️

Demo time !

Godot: dynamic from the ground

godot/node2d.hpp

```
class Node2D : public Node {

        void set_rot(float p_angle);
        float get_rot() const;
        ...

}
```

Godot: dynamic from the ground

godot/node2d.hpp

```
class Node2D : public Node {
    GDCLASS(Node2D, Node);
    void set_rot(float p_angle);
    float get_rot() const;
    ...
    void _bind_methods() {
        ClassDB::bind_method(D_METHOD("get_rot"), &Node2D::get_rot);
        ClassDB::bind_method(D_METHOD("set_rot", "degrees"), &Node2D::set_rot);
        ...
    }
}
```

Godot: dynamic from the ground

godot/node2d.hpp

```
class Node2D : public Node {
        GDCLASS(Node2D, Node);
        void set_rot(float p_angle);
        float get_rot() const;
        ...
        void _bind_methods() {
                ClassDB::bind_method(D_METHOD("get_rot"), &Node2D::get_rot);
                ClassDB::bind_method(D_METHOD("set_rot", "degrees"), &Node2D::set_rot);
                ...
        }
}
```

```
# C++ static traditional way
Node2D *obj = new Node2D();
obj->set_ret(4.2);
```

Godot: dynamic from the ground

godot/node2d.hpp

```
class Node2D : public Node {
        GDCLASS(Node2D, Node);
        void set_rot(float p_angle);
        float get_rot() const;
        ...
        void _bind_methods() {
                ClassDB::bind_method(D_METHOD("get_rot"), &Node2D::get_rot);
                ClassDB::bind_method(D_METHOD("set_rot", "degrees"), &Node2D::set_rot);
                ...
        }
}
```

```
# C++ static traditional way
Node2D *obj = new Node2D();
obj->set_ret(4.2);
```

```
# C++ Dynamic way
Variant obj = ClassDB::instance("Node2D");
MethodBind *mb = ClassDB::get_method("Node2D", "set_rot");
Variant args[1] = {Variant(4.2)};
Variant ret = mb->call(obj, args, ...)
```

Godot: dynamic from the ground

godot/node2d.hpp

```
class Node2D : public Node {
        GDCLASS(Node2D, Node);
        void set_rot(float p_angle);
        float get_rot() const;
        ...
        void _bind_methods() {
                ClassDB::bind_method(D_METHOD("get_rot"), &Node2D::get_rot);
                ClassDB::bind_method(D_METHOD("set_rot", "degrees"), &Node2D::set_rot);
                ...
        }
}
```

```
# C++ static traditional way
Node2D *obj = new Node2D();
obj->set_ret(4.2);
```

```
# C++ Dynamic way
Variant obj = ClassDB::instance("Node2D");
MethodBind *mb = ClassDB::get_method("Node2D", "set_rot");
Variant args[1] = {Variant(4.2)};
Variant ret = mb->call(obj, args, ...)
```

```
# GDscript
obj = Node2D()
obj.set_ret(4.2)
```

Godot: dynamic from the ground

godot/node2d.hpp

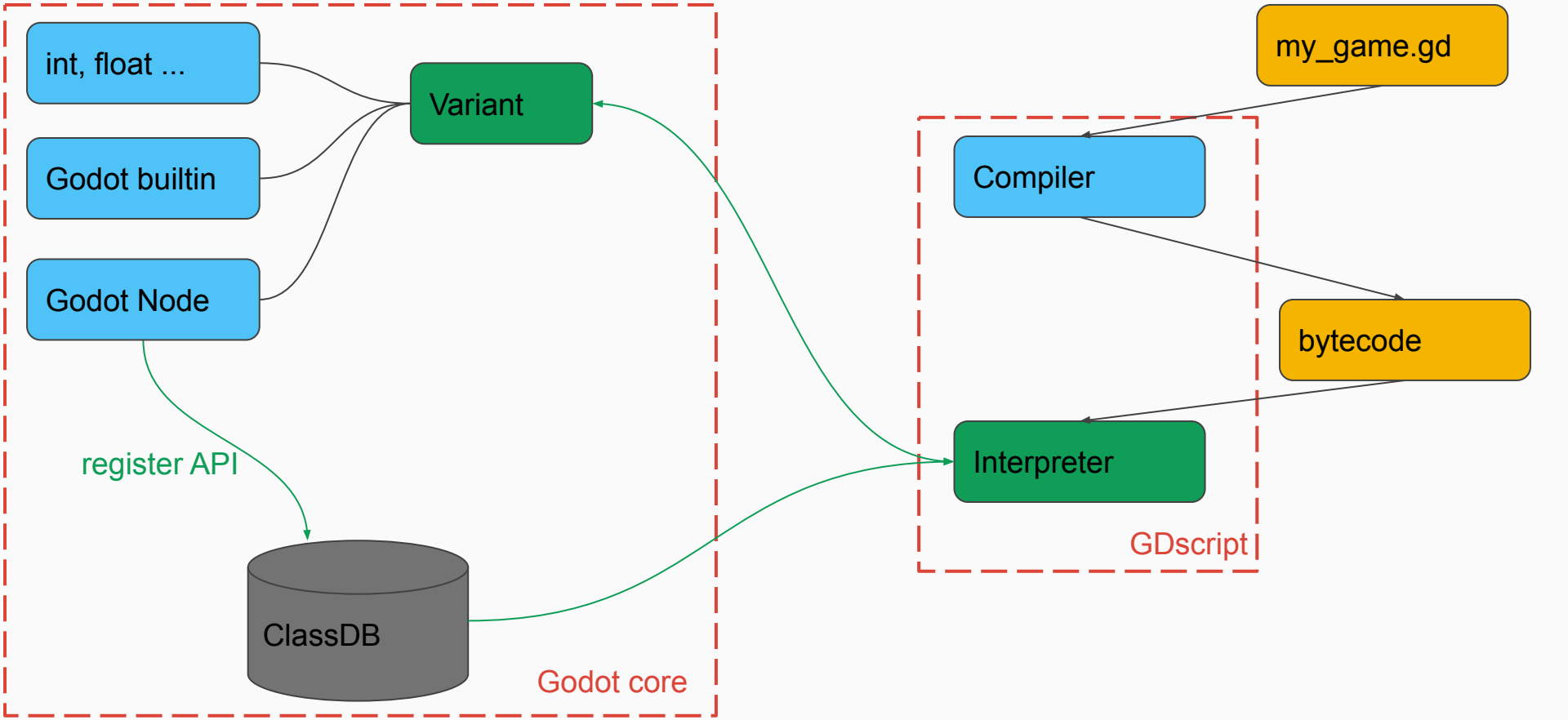
```
class Node2D : public Node {
        GDCLASS(Node2D, Node);
        void set_rot(float p_angle);
        float get_rot() const;
        ...
        void _bind_methods() {
                ClassDB::bind_method(D_METHOD("get_rot"), &Node2D::get_rot);
                ClassDB::bind_method(D_METHOD("set_rot", "degrees"), &Node2D::set_rot);
                ...
        }
}
```

```
# C++ static traditional way
Node2D *obj = new Node2D();
obj->set_ret(4.2);
```

```
# C++ Dynamic way
Variant obj = ClassDB::instance("Node2D");
MethodBind *mb = ClassDB::get_method("Node2D", "set_rot");
Variant args[1] = {Variant(4.2)};
Variant ret = mb->call(obj, args, ...)
```

```
# GDscript
obj = Node2D()
obj.set_ret(4.2)
```

GDscript ? Why ?

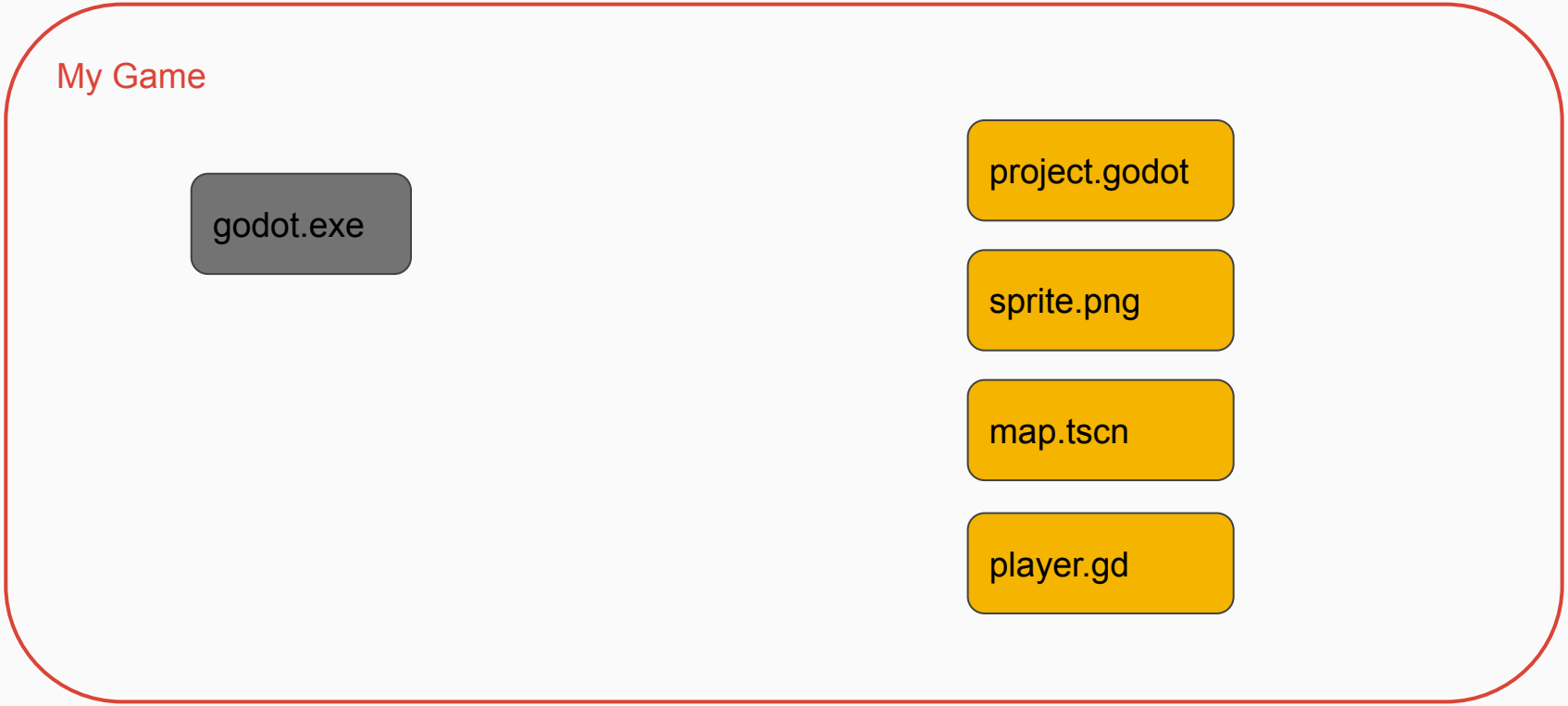


BUT...

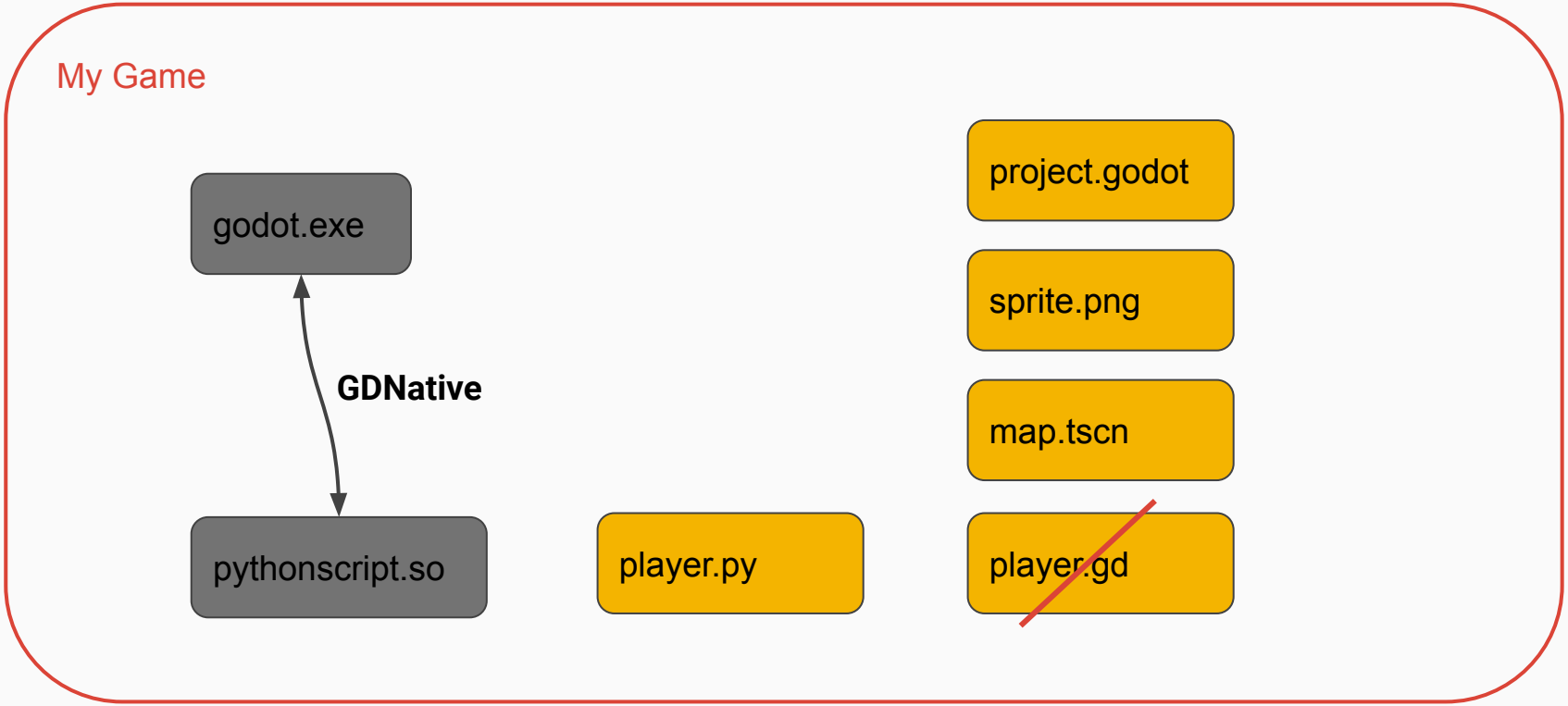
Python ? What for ?

- Game scripting
- Testing
- Asset pipeline with Blender
- Bring Python ecosystem to Godot ;-)

Bind to Python to Godot

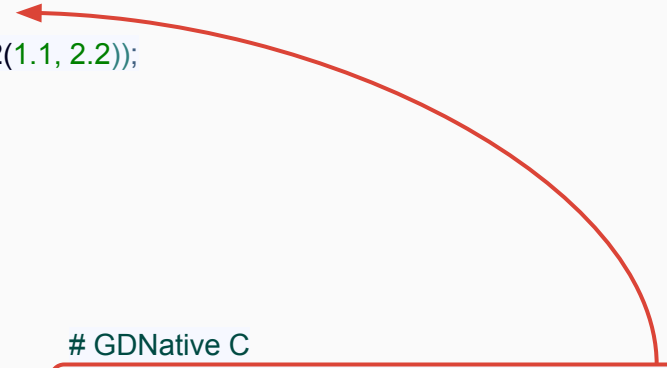


Bind to Python to Godot



Control Godot from C with GDNative

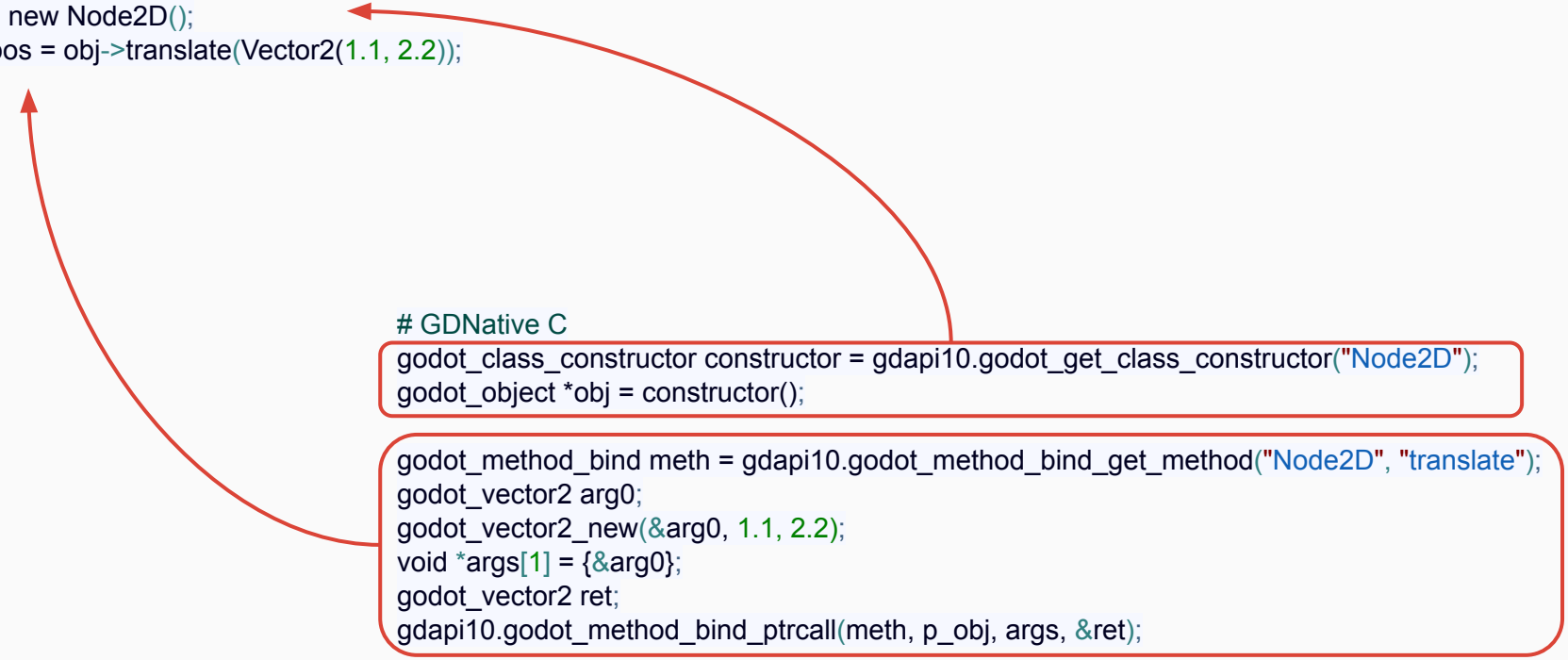
```
# C++ static traditional way
Node2D *obj = new Node2D();
Vector2 new_pos = obj->translate(Vector2(1.1, 2.2));
```



```
# GDNative C
godot_class_constructor constructor = gdapi10.godot_get_class_constructor("Node2D");
godot_object *obj = constructor();
```

Control Godot from C with GDNative

```
# C++ static traditional way
Node2D *obj = new Node2D();
Vector2 new_pos = obj->translate(Vector2(1.1, 2.2));
```



```
# GDNative C
godot_class_constructor constructor = gdapi10.godot_get_class_constructor("Node2D");
godot_object *obj = constructor();
```

```
godot_method_bind meth = gdapi10.godot_method_bind_get_method("Node2D", "translate");
godot_vector2 arg0;
godot_vector2_new(&arg0, 1.1, 2.2);
void *args[1] = {&arg0};
godot_vector2 ret;
gdapi10.godot_method_bind_ptrcall(meth, p_obj, args, &ret);
```

Bind to Python to Godot

WHAT ?

HOW ?

- Python as script language
- Godot Callbacks
- Builtins bindings
- ClassDB bindings

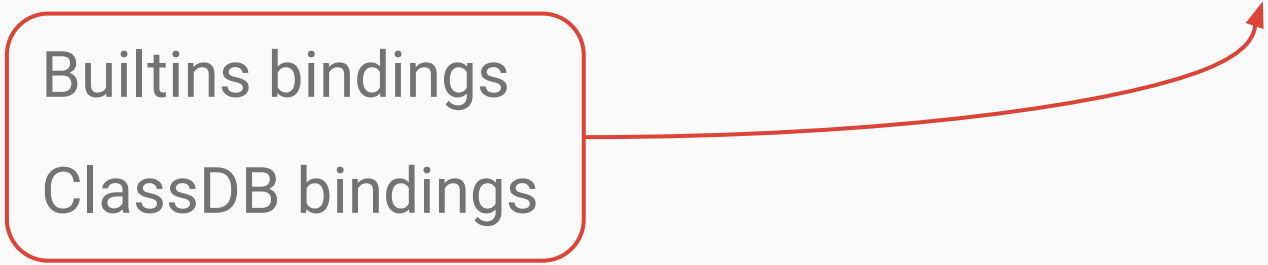
Bind to Python to Godot

WHAT ?

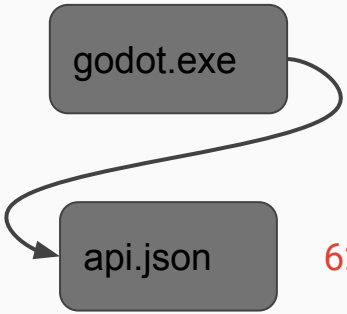
- Python as script language
- Godot Callbacks
- Builtins bindings
- ClassDB bindings

HOW ?

- Cython !
- A looot of templates ;-)

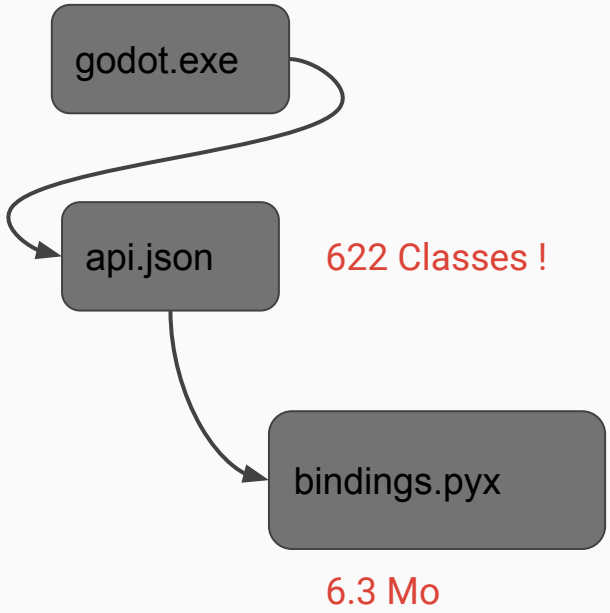


Generate ClassDB binding

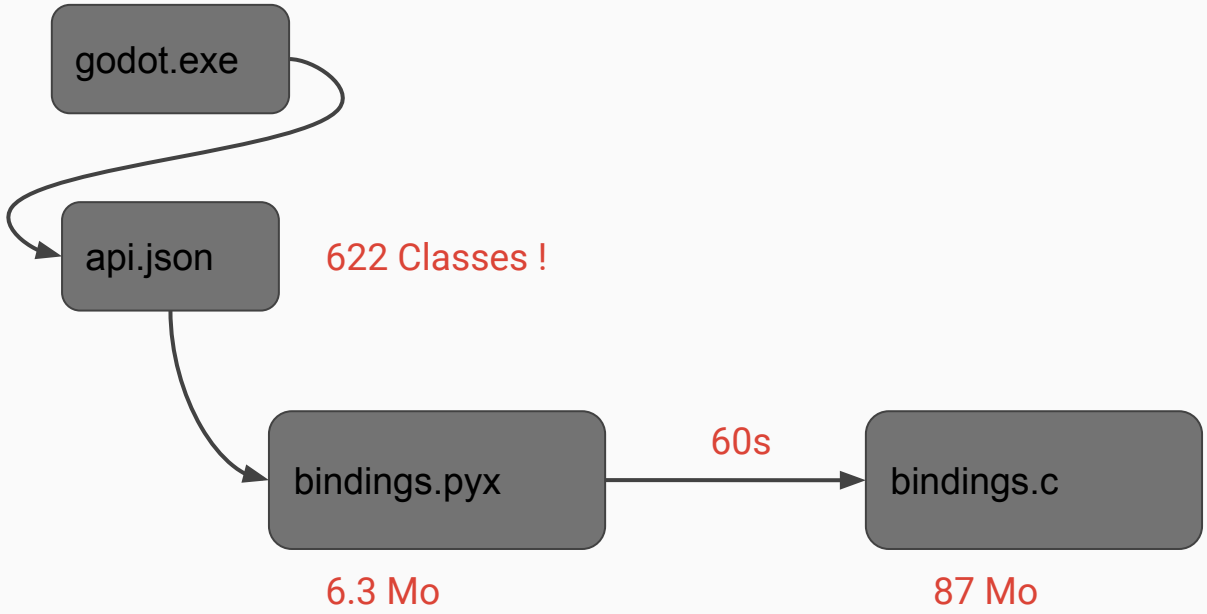


622 Classes !

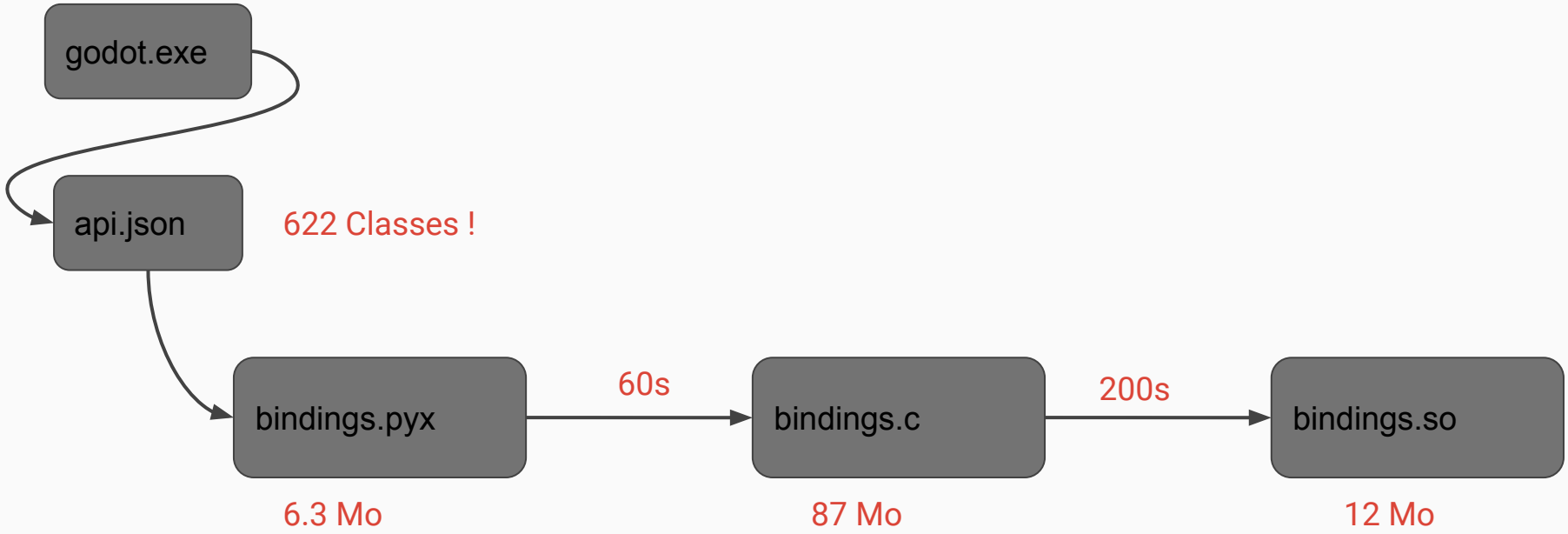
Generate ClassDB binding



Generate ClassDB binding



Generate ClassDB binding



Roadmap

- Beta is here (we need you !)
- MacOS build :'-(
- Editor integration
- Cython in games \o/
- Godot 4.0

Roadmap

- Beta is here (we need you !)
- MacOS build :'-(
- Editor integration
- Cython in games \o/
- Godot 4.0

Write a game !!!

Thank !
Q&A ?


touilleMan


@touilleMan

MicroPython

:’-(

- ~90% Python
- No pdb
- Static heap
- Documentation…

*^__^*/

- Smaller
- Embedding friendly
- Customization !
- GIL is optional

CPython + Pybind11

:’-(

- C++...
- ...with magic templates
- Compilation time
- Output size

*^__^*/

- C++ binding feels like Python
- Autogenerate bindings with api.json

CPython + CFFI

:’-(

- Memory management
- Overhead
- Dynamic binding not for Godot release

*^__^*/

- You bind in Python !!!!
  - Closure
  - Call resolution done by Python
  - Just in time binding
- Supports also Pypy

CPython + Cython

:’-(

- Compilation time
- Output size

*^__^*/

- C speed
- Python clarity
- Allow us to write games in Cython !!!!