C++ EN UN MOTOR DE VIDEOJUEGOS: GODOT

Pedro J. Estébanez

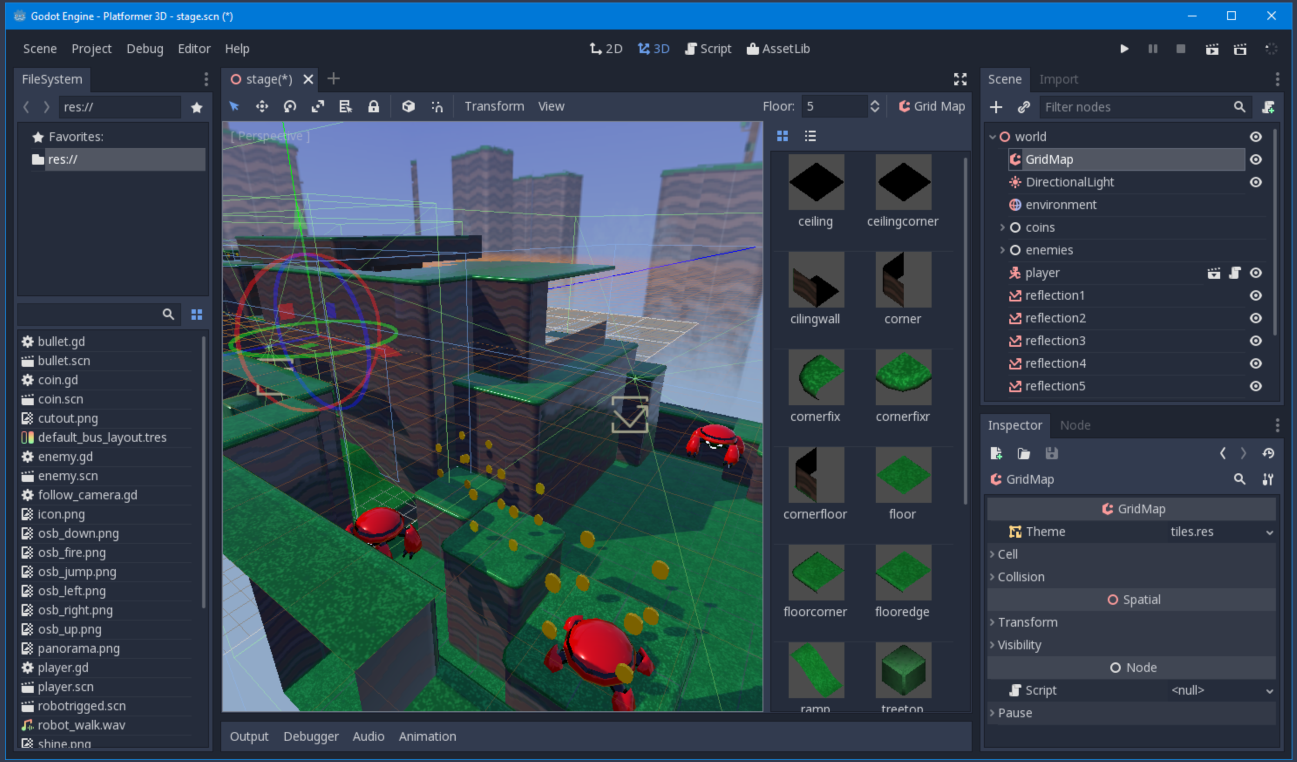
(a.k.a. RandomShaper)

@RandomPedroJ

Madrid C/C++ User Group

FUNDAMENTOS

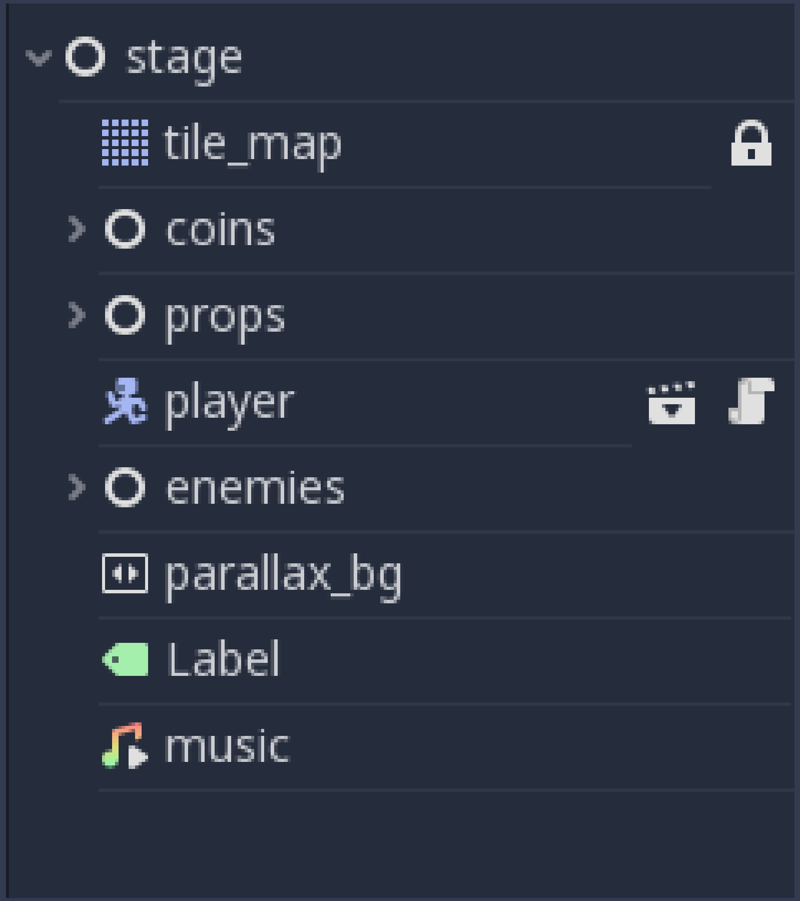
MOTOR DE JUEGOS + EDITOR



- 2D y 3D
- Muchas herramientas visuales
- Muy fácil de aprender

TODO ES UN NODO

- Una escena también es un nodo
- Muchas opciones de organización (instanciación, herencia, anidación)
- Trabajo paralelo en equipos
- Programadores y grafistas pueden colaborar sin interferencia



MULTIPLATAFORMA

- Escritorio · · · · ·
  (juugos + editor)
- Móvil · · · · · · · ·
- Web · · · · · · · · ·
  (WebAssembly)
- Consolas · · · · · ·
  (terceros)

LIBRE Y ABIERTO

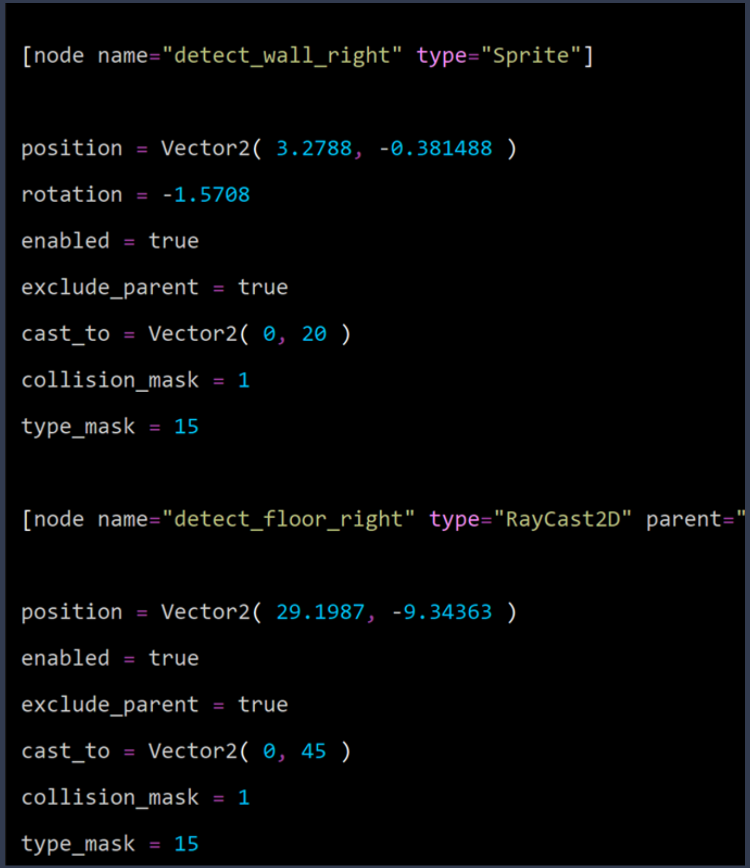
- Código abierto en 2014
- Alojado en GitHub
- Travis + AppVeyor CI
- Licencia MIT

PRESTACIONES

3D CONTEMPORÁNEO

- Physics Based Rendering
- Iluminación global en tiempo real
- Efectos típicos soportados (Tone-mapping, SSAO, SSR…)

AMIGABLE CON VCS (Y HUMANOS)

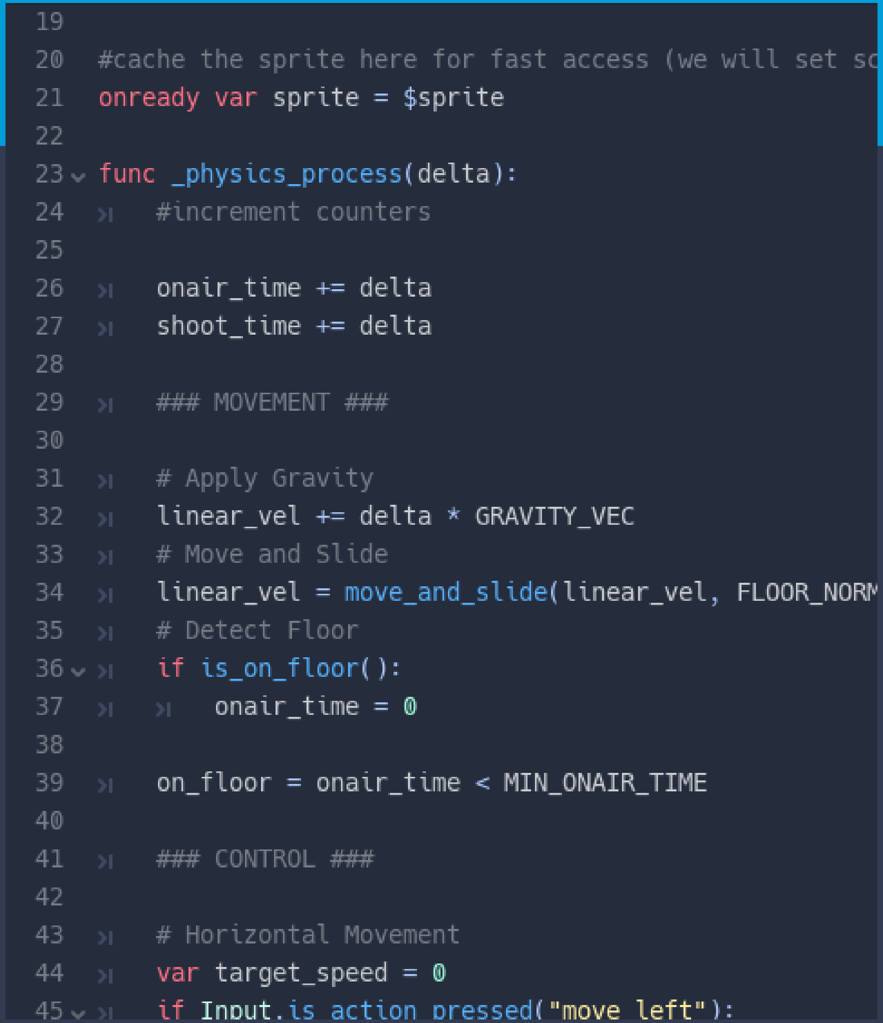


```
[node name="detect_wall_right" type="Sprite"]

position = Vector2( 3.2788, -0.381488 )
rotation = -1.5708
enabled = true
exclude_parent = true
cast_to = Vector2( 0, 20 )
collision_mask = 1
type_mask = 15

[node name="detect_floor_right" type="RayCast2D" parent="

position = Vector2( 29.1987, -9.34363 )
enabled = true
exclude_parent = true
cast_to = Vector2( 0, 45 )
collision_mask = 1
type_mask = 15
```

- Casi todo son archivos de texto
- Formato amigable con VCS (*diffs* elegantes)
- Fácil de leer y editar por humanos

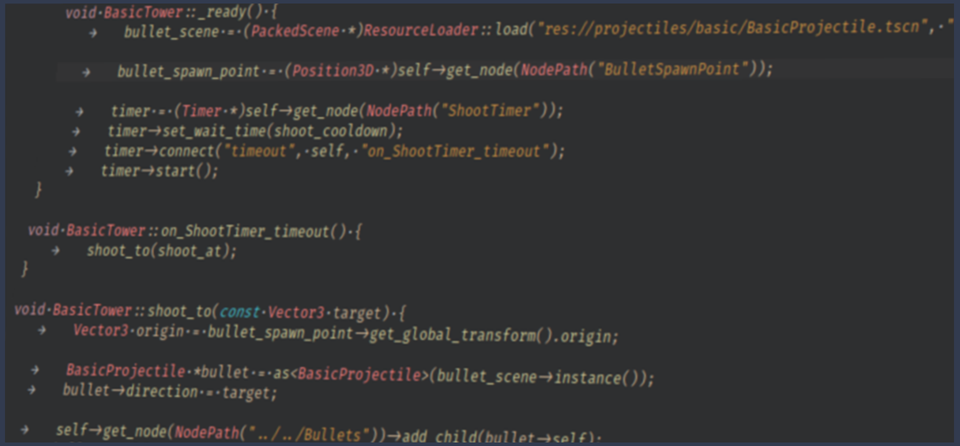
MÚLTIPLES LENGUAJES

- GDScript
- C# 7.0 (mediante Mono)
- Visual Scripting
- C++ (NativeScript)
- No oficiales: Nim, Rust, Python, D



```

#cache the sprite here for fast access (we will set sc
onready var sprite = $sprite

func _physics_process(delta):
	#increment counters

	onair_time += delta
	shoot_time += delta

	### MOVEMENT ###

	# Apply Gravity
	linear_vel += delta * GRAVITY_VEC
	# Move and Slide
	linear_vel = move_and_slide(linear_vel, FLOOR_NORM
	# Detect Floor
	if is_on_floor():
		onair_time = 0

	on_floor = onair_time < MIN_ONAIR_TIME

	### CONTROL ###

	# Horizontal Movement
	var target_speed = 0
	if Input.is_action_pressed("move_left"):
```

EXTENSIBLE

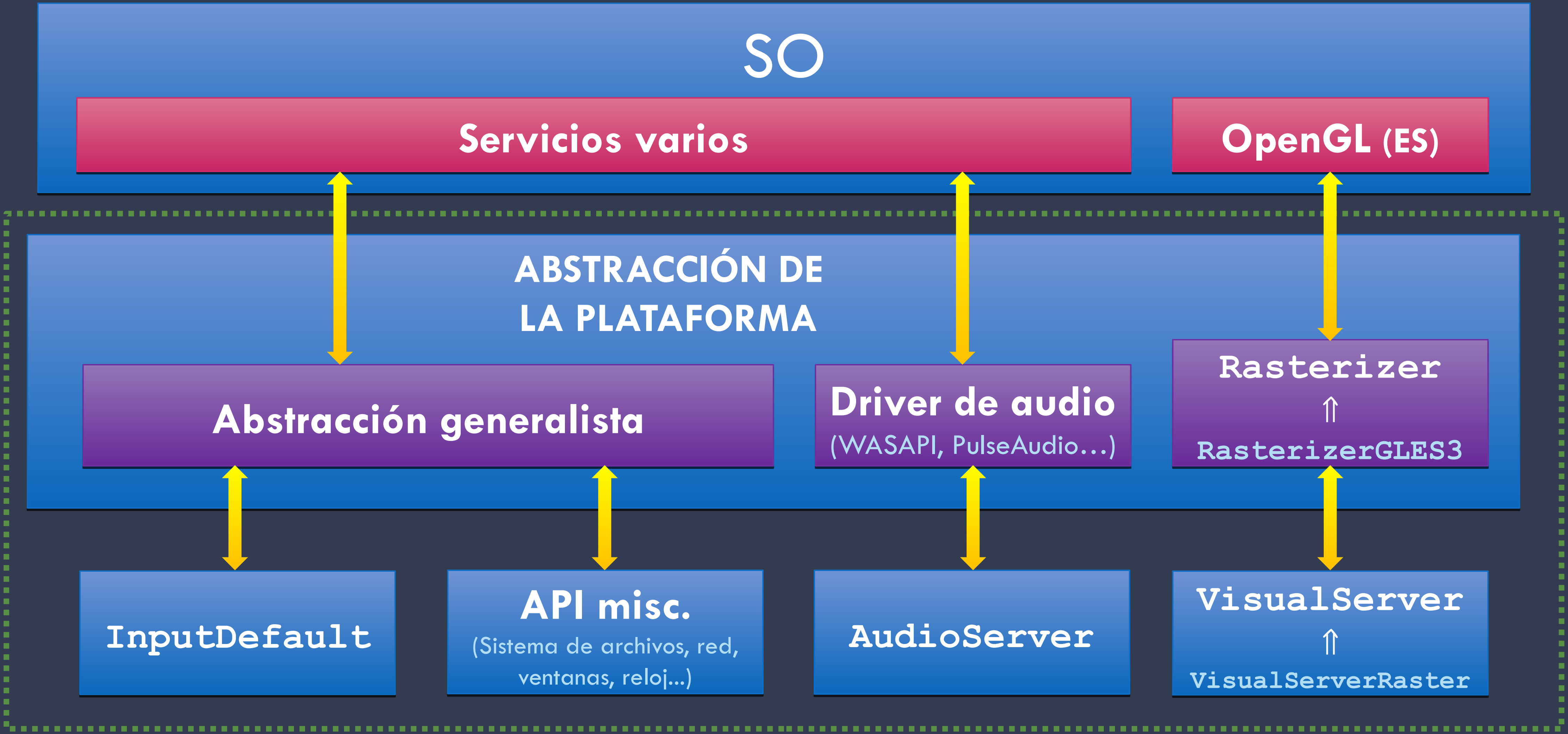


```
void BasicTower::_ready() {
    bullet_scene = (PackedScene *)ResourceLoader::load("res://projectiles/basic/BasicProjectile.tscn", "

    bullet_spawn_point = (Position3D *)self->get_node(NodePath("BulletSpawnPoint"));

    timer = (Timer *)self->get_node(NodePath("ShootTimer"));
    timer->set_wait_time(shoot_cooldown);
    timer->connect("timeout", self, "on_ShootTimer_timeout");
    timer->start();
}

void BasicTower::on_ShootTimer_timeout() {
    shoot_to(shoot_at);
}

void BasicTower::shoot_to(const Vector3 target) {
    Vector3 origin = bullet_spawn_point->get_global_transform().origin;

    BasicProjectile *bullet = as<BasicProjectile>(bullet_scene->instance());
    bullet->direction = target;

    self->get_node(NodePath("../../Bullets"))->add_child(bullet->self);
```

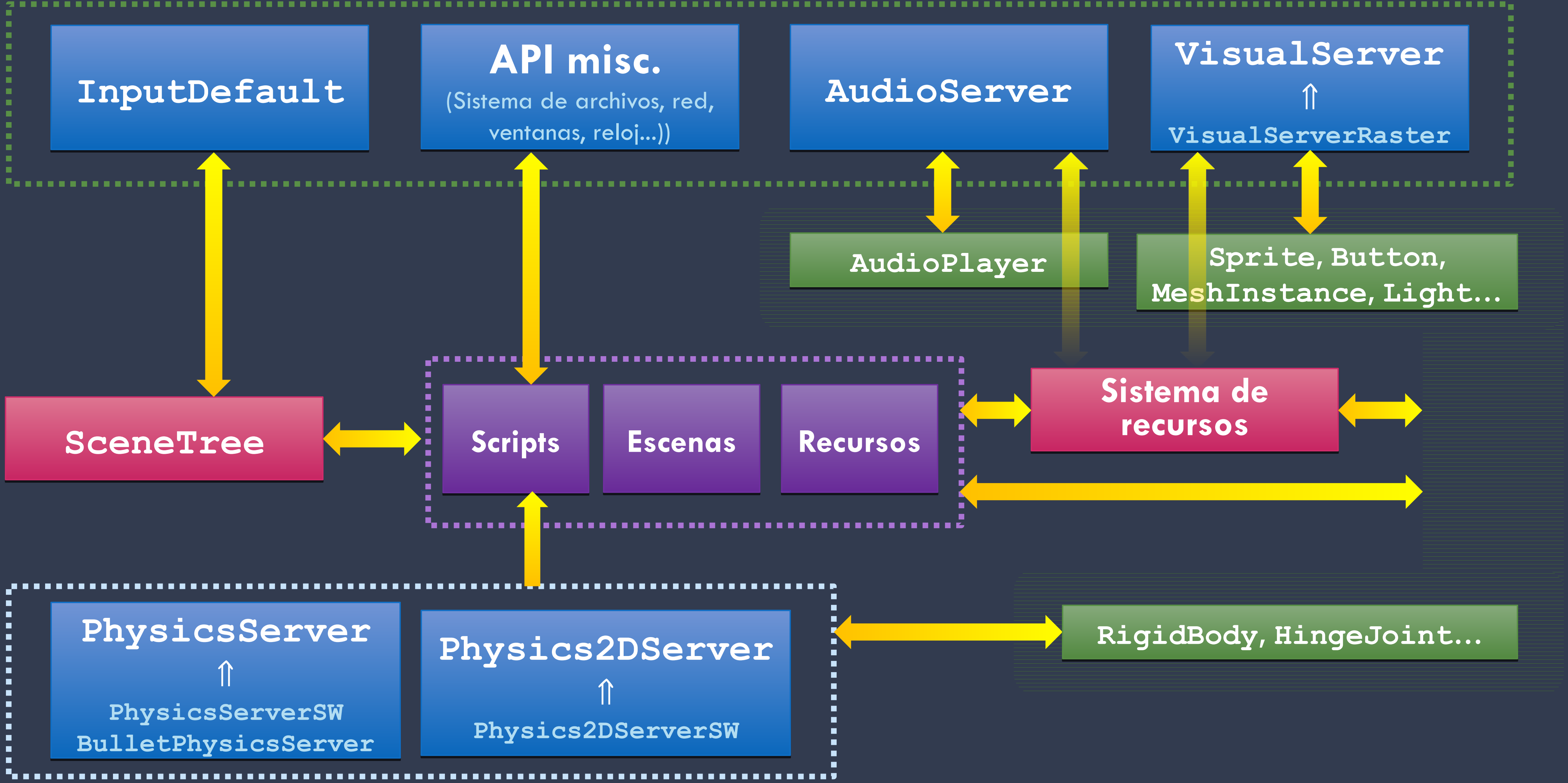
- Arquitectura de *plugins*
- Biblioteca de *assets*
- GDNative

ARQUITECTURA

MOTOR ⟷ MUNDO EXTERIOR



MOTOR ⟷ JUEGO


InputDefault
API misc.
(Sistema de archivos, red, ventanas, reloj...))
AudioServer
VisualServer
⇑
VisualServerRaster
AudioPlayer
Sprite, Button, MeshInstance, Light...
SceneTree
Scripts
Escenas
Recursos
Sistema de recursos
PhysicsServer
⇑
PhysicsServerSW
BulletPhysicsServer
Physics2DServer
⇑
Physics2DServerSW
RigidBody, HingeJoint...

CONSTRUCCIÓN

SCons

OPCIONES INTERESANTES

`target`

`debug | release_debug | release`

`-DNDEBUG`

OPCIONES INTERESANTES

`tools`

`(yes) | no`

`-DTOOLS_ENABLED`

~~`target=release tools=yes`~~

OPCIONES INTERESANTES

`platform`

`platform/*/detect.py`

`x11 | windows | iphone | android | ...`

`-D*_ENABLED` `-DANDROID_ENABLED`

OPCIONES INTERESANTES

```
arch

('') | arm | arm64 | x86 | x64 | ...

bits

(default) | 32 | 64
```

OPCIONES INTERESANTES

```
use_llvm
```

```
yes | (no)
```

OPCIONES INTERESANTES

```
use_lto

yes | (no)
```

OPCIONES INTERESANTES

```
builtin_freetype
builtin_libogg
builtin_zlib
...

(yes) | no
```

OPCIONES INTERESANTES

```
use_static_cpp

yes | (no)

LINKFLAGS +=
  -static-libgcc
  -static-libstdc++
```

OPCIONES INTERESANTES

`platform`

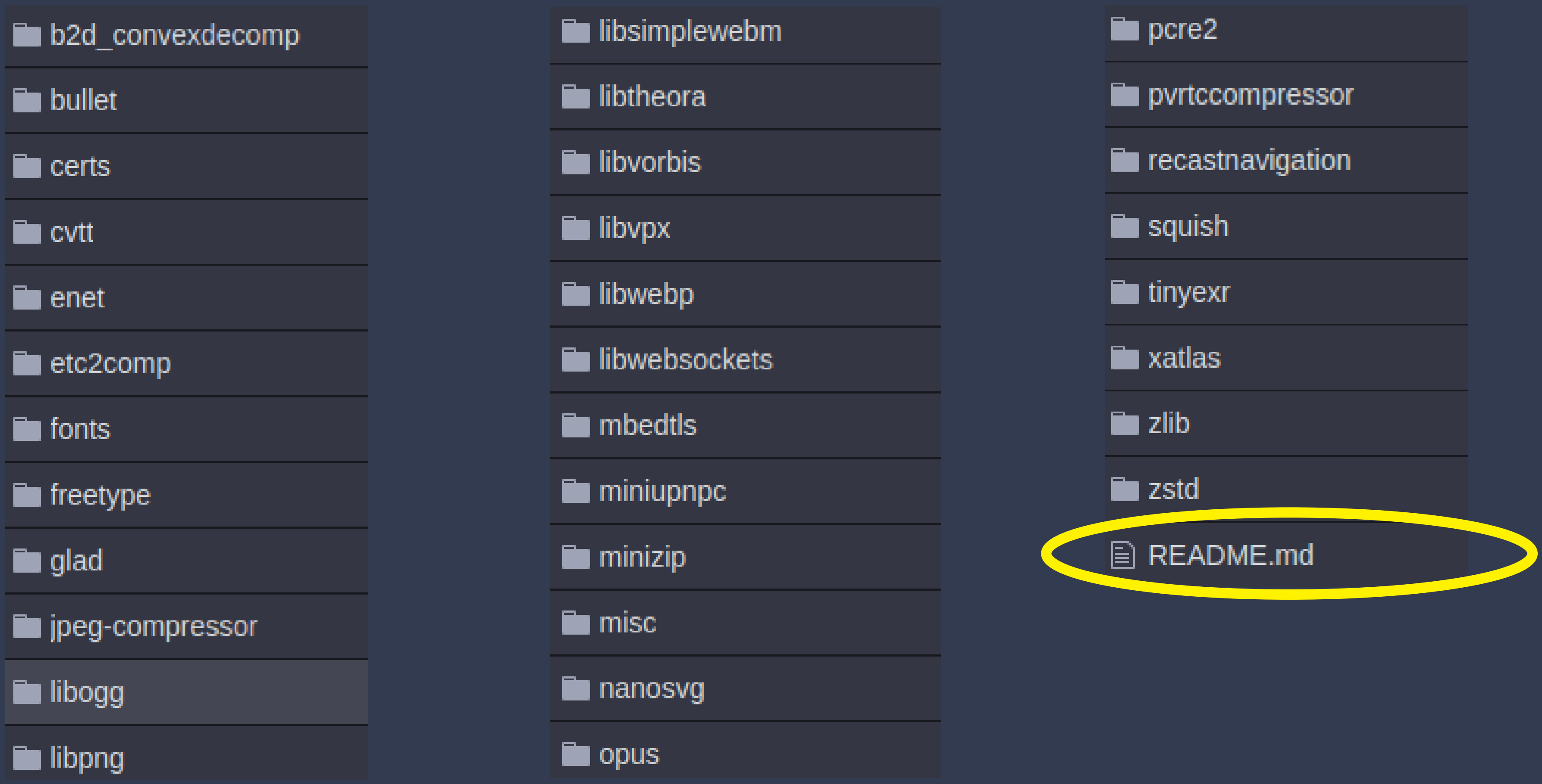
`platform/*/detect.py`

`x11 | windows | iphone | android | ...`

`-D*_ENABLED` `-DANDROID_ENABLED`

DEPENDENCIAS

`thirdparty/*/`

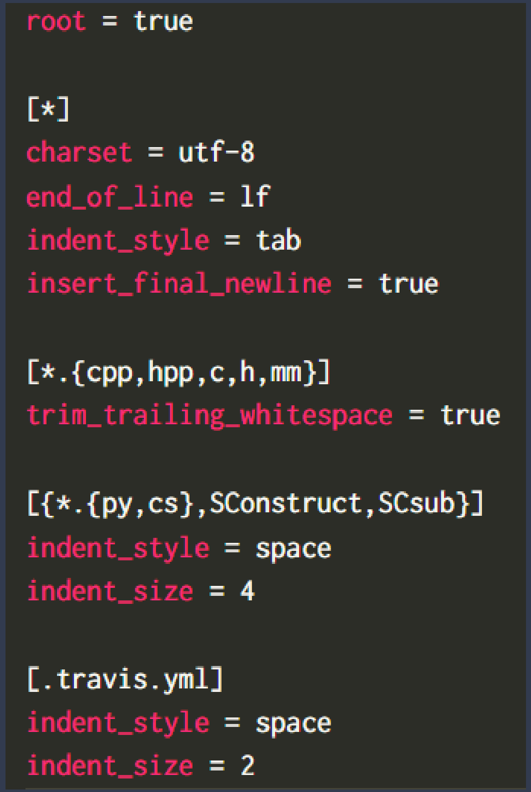


b2d_convexdecomp
bullet
certs
cvtt
enet
etc2comp
fonts
freetype
glad
jpeg-compressor
libogg
libpng
libsimplewebm
libtheora
libvorbis
libvpx
libwebp
libwebsockets
mbedtls
miniupnpc
minizip
misc
nanosvg
opus
pcre2
pvrtccompressor
recastnavigation
squish
tinyexr
xatlas
zlib
zstd
README.md

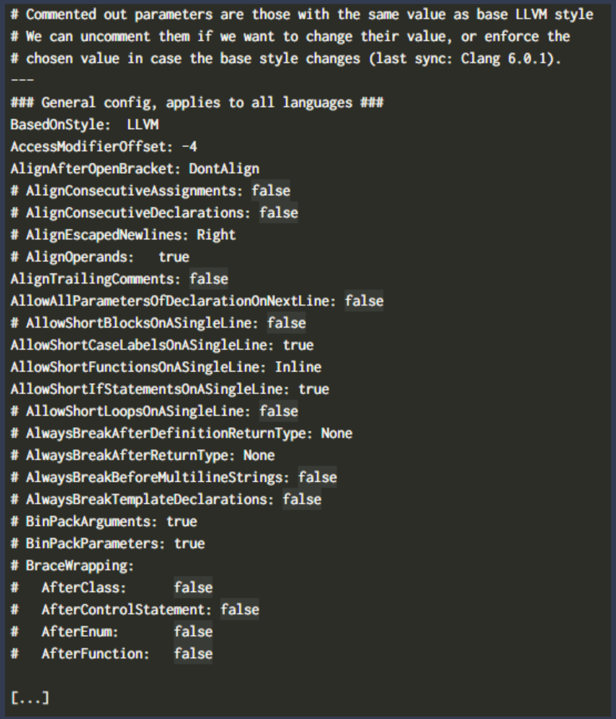
C++

ESTILO: FORMATO

.editorconfig



```
root = true

[*]
charset = utf-8
end_of_line = lf
indent_style = tab
insert_final_newline = true

[*.{cpp,hpp,c,h,mm}]
trim_trailing_whitespace = true

[{*.{py,cs},SConstruct,SCsub}]
indent_style = space
indent_size = 4

[.travis.yml]
indent_style = space
indent_size = 2
```



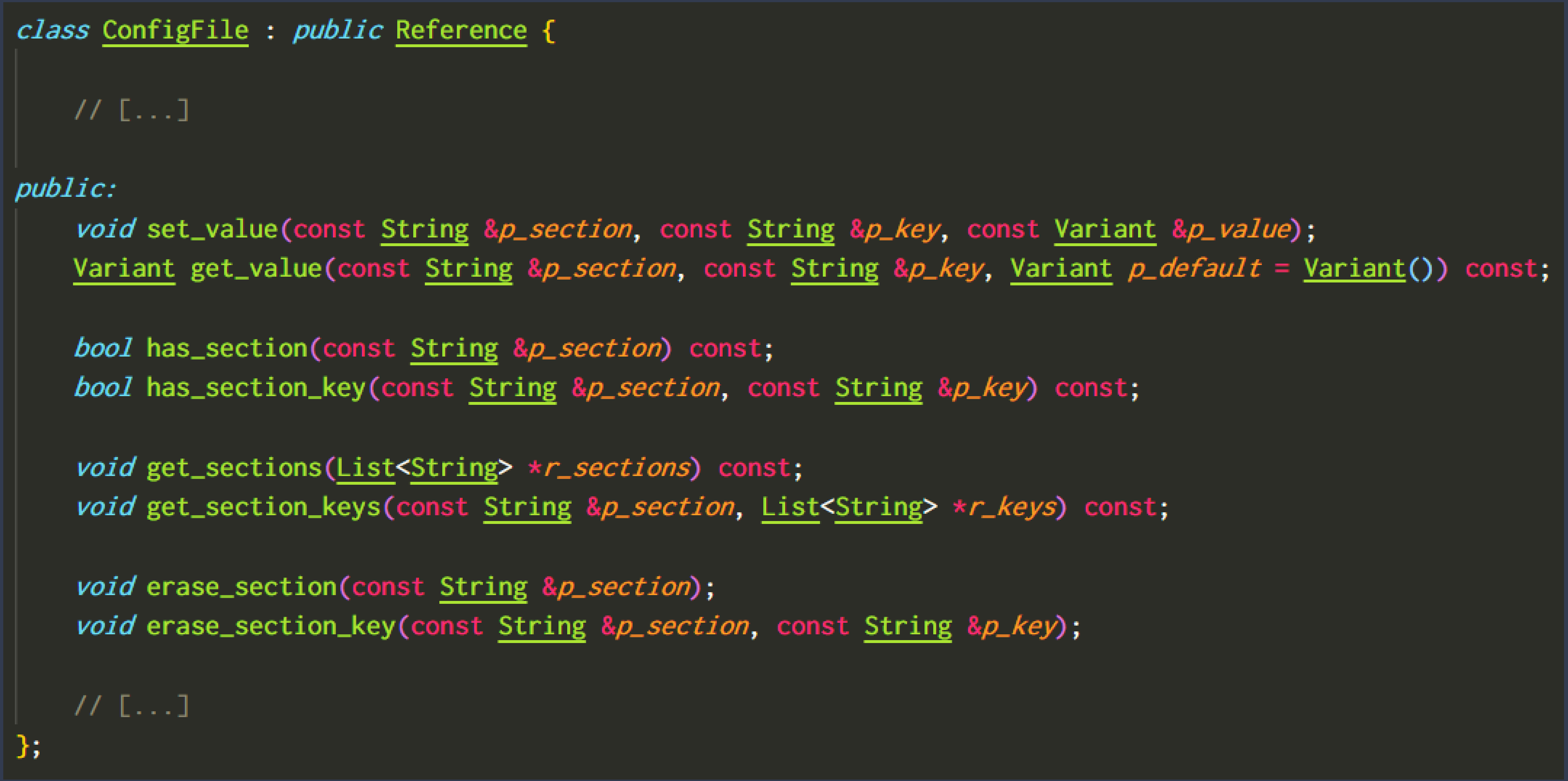
```
# Commented out parameters are those with the same value as base LLVM style
# We can uncomment them if we want to change their value, or enforce the
# chosen value in case the base style changes (last sync: Clang 6.0.1).
---
### General config, applies to all languages ###
BasedOnStyle:  LLVM
AccessModifierOffset: -4
AlignAfterOpenBracket: DontAlign
# AlignConsecutiveAssignments: false
# AlignConsecutiveDeclarations: false
# AlignEscapedNewlines: Right
# AlignOperands:   true
AlignTrailingComments: false
AllowAllParametersOfDeclarationOnNextLine: false
# AllowShortBlocksOnASingleLine: false
AllowShortCaseLabelsOnASingleLine: true
AllowShortFunctionsOnASingleLine: Inline
AllowShortIfStatementsOnASingleLine: true
# AllowShortLoopsOnASingleLine: false
# AlwaysBreakAfterDefinitionReturnType: None
# AlwaysBreakAfterReturnType: None
# AlwaysBreakBeforeMultilineStrings: false
# AlwaysBreakTemplateDeclarations: false
# BinPackArguments: true
# BinPackParameters: true
# BraceWrapping:
#   AfterClass:       false
#   AfterControlStatement: false
#   AfterEnum:        false
#   AfterFunction:    false

[...]
```

.clang-format

+ Pre-commit hook

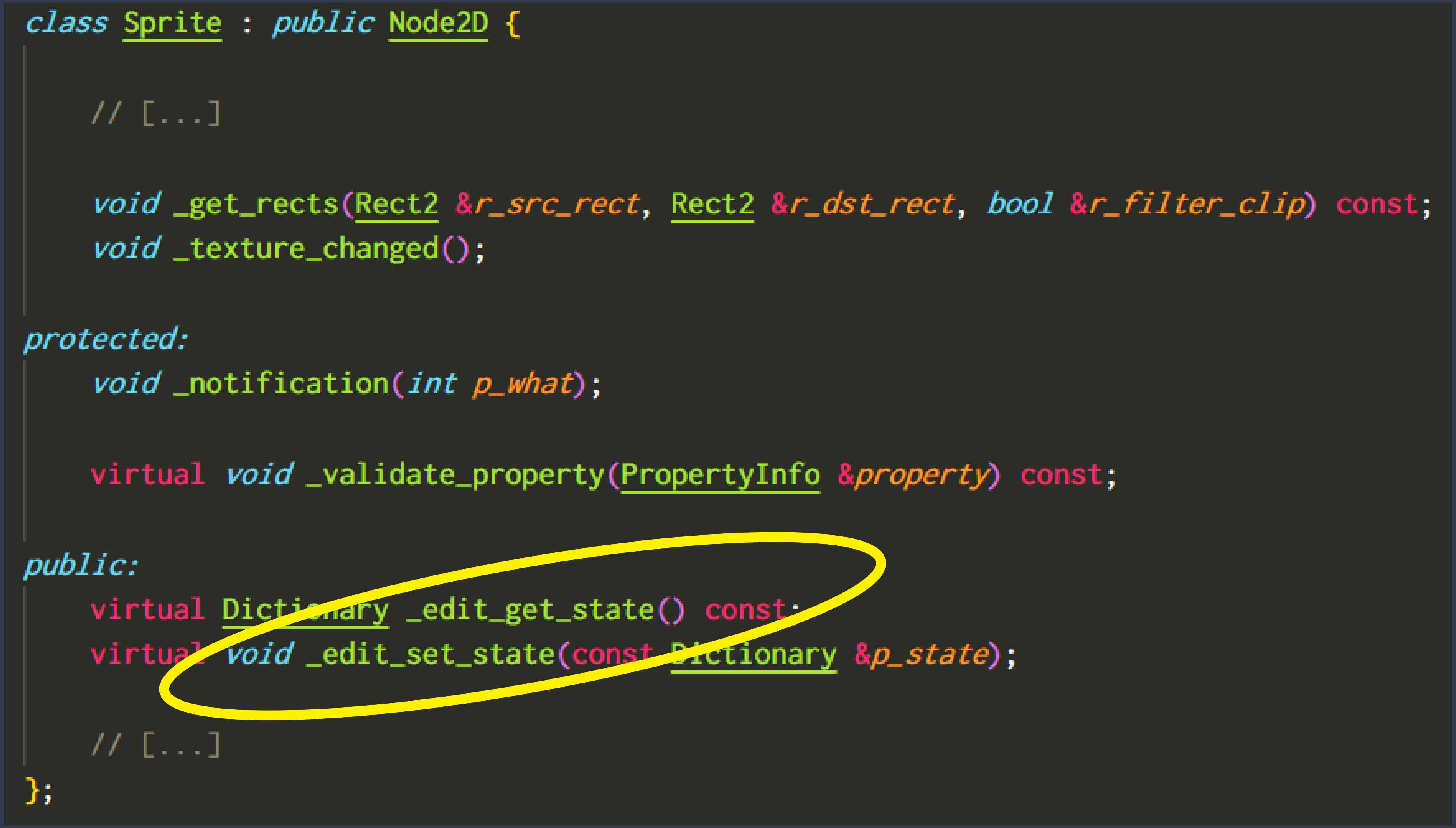
+ Validación en CI

ESTILO: PARÁMETROS



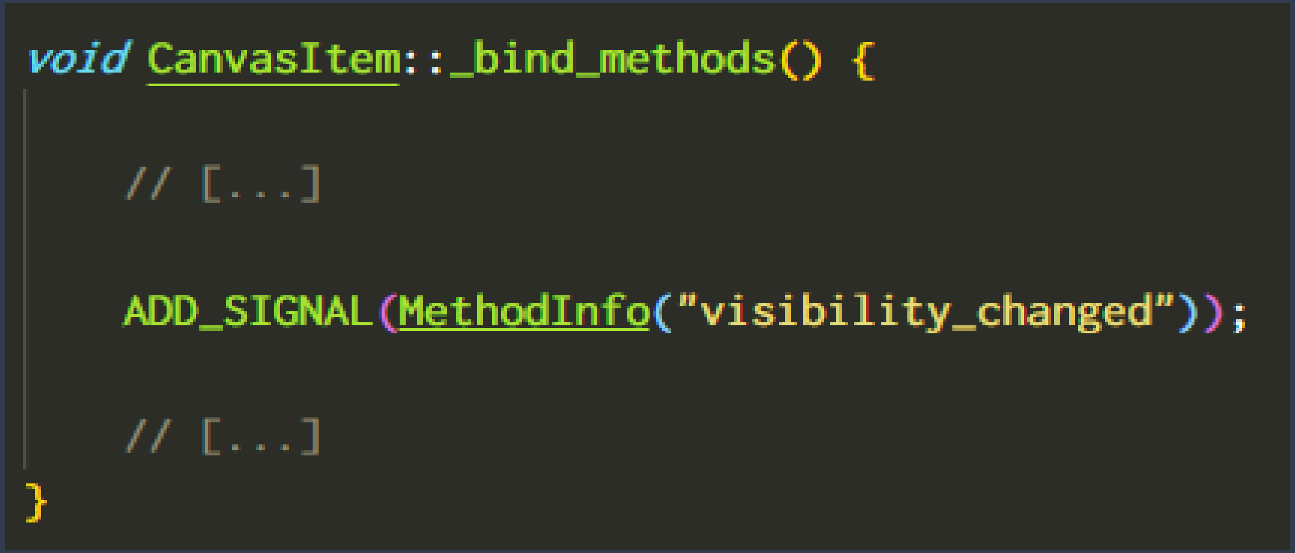
```
class ConfigFile : public Reference {

    // [...]

public:
    void set_value(const String &p_section, const String &p_key, const Variant &p_value);
    Variant get_value(const String &p_section, const String &p_key, Variant p_default = Variant()) const;

    bool has_section(const String &p_section) const;
    bool has_section_key(const String &p_section, const String &p_key) const;

    void get_sections(List<String> *r_sections) const;
    void get_section_keys(const String &p_section, List<String> *r_keys) const;

    void erase_section(const String &p_section);
    void erase_section_key(const String &p_section, const String &p_key);

    // [...]
};
```

ESTILO: FUNCIONES NO PÚBLICAS



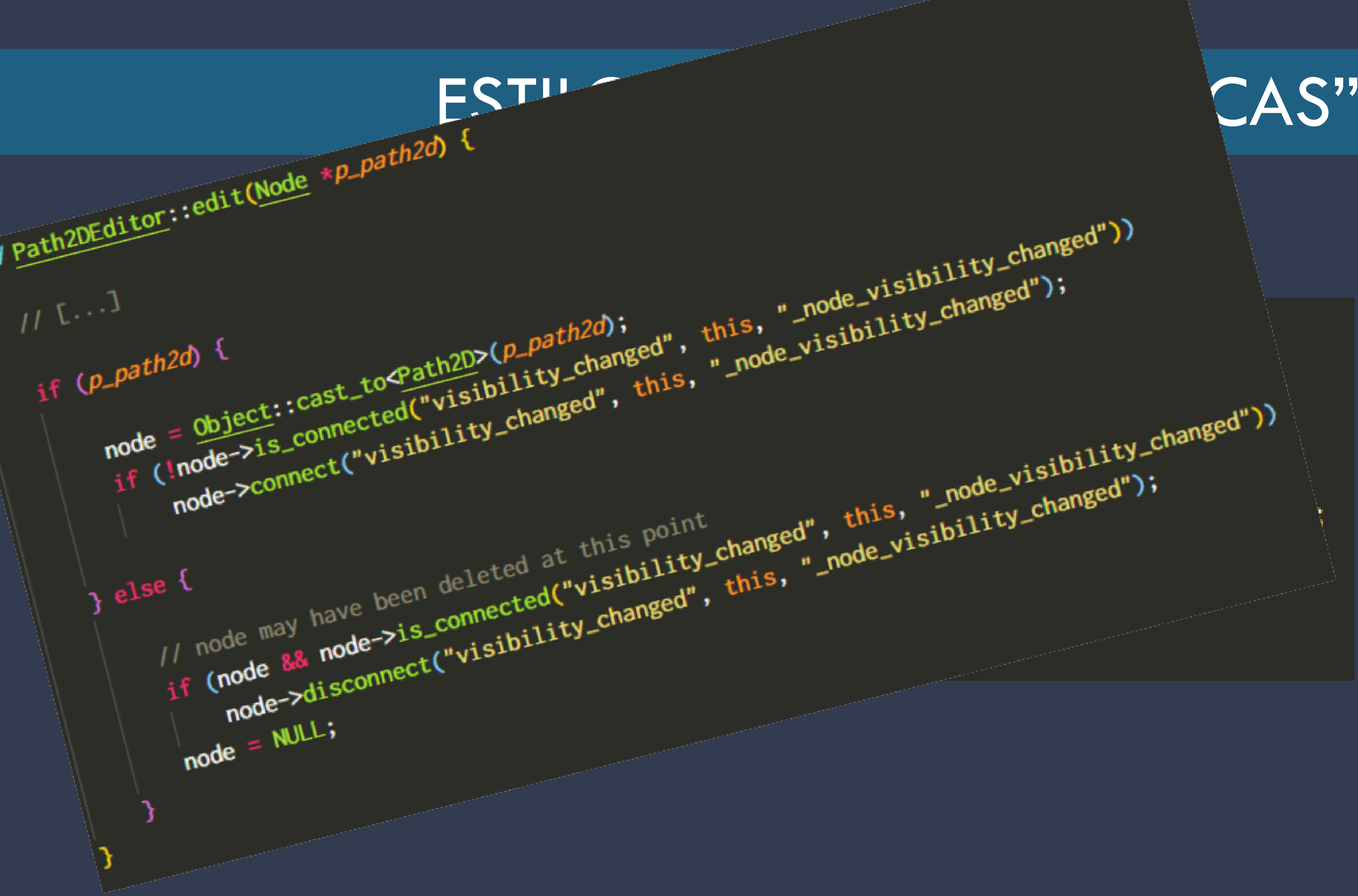
```cpp
class Sprite : public Node2D {

    // [...]

    void _get_rects(Rect2 &r_src_rect, Rect2 &r_dst_rect, bool &r_filter_clip) const;
    void _texture_changed();

protected:
    void _notification(int p_what);

    virtual void _validate_property(PropertyInfo &property) const;

public:
    virtual Dictionary _edit_get_state() const;
    virtual void _edit_set_state(const Dictionary &p_state);

    // [...]
};
```

ESTILO: CADENAS “MÁGICAS”



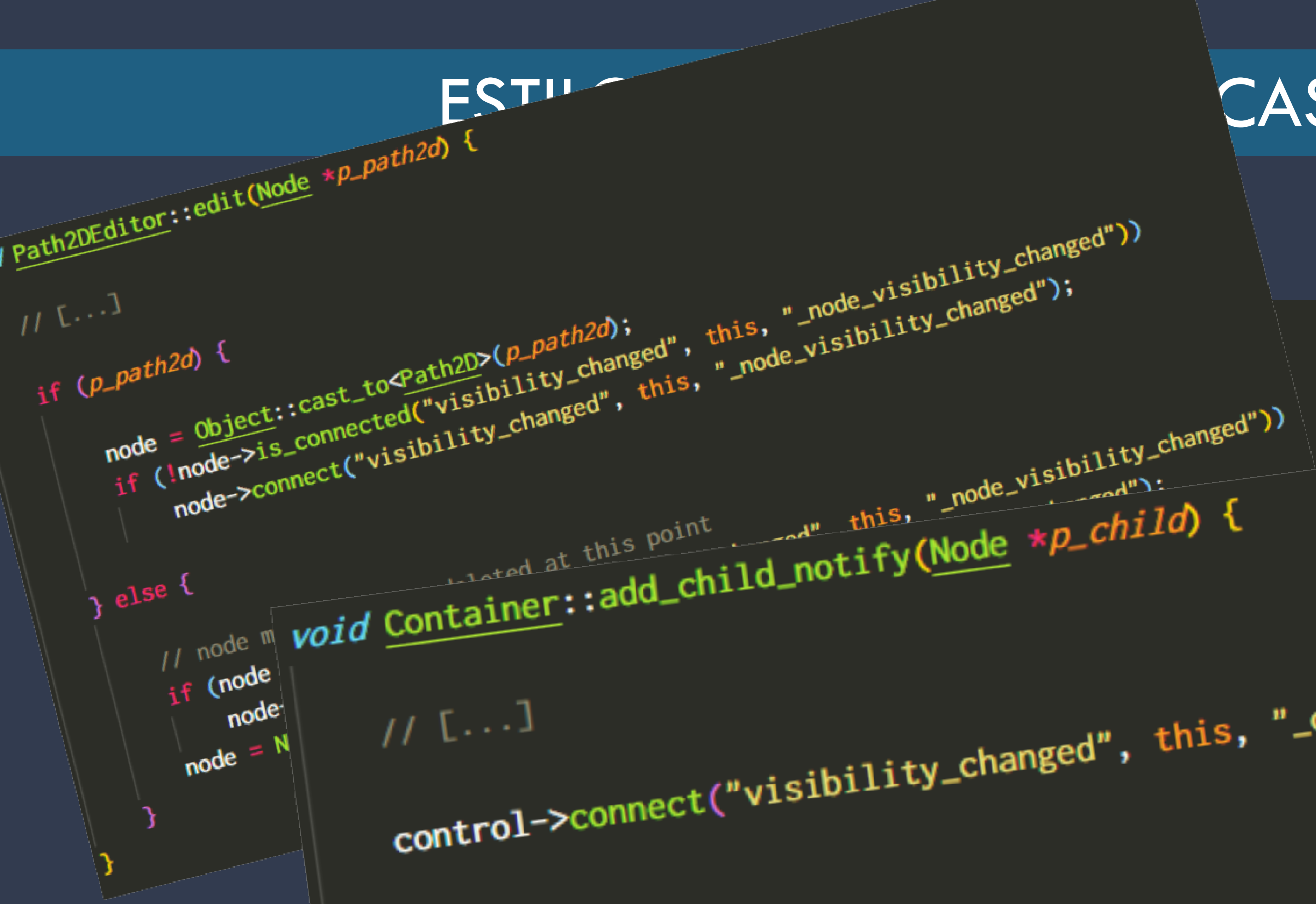
```
void CanvasItem::_bind_methods() {

    // [...]

    ADD_SIGNAL(MethodInfo("visibility_changed"));

    // [...]
}
```

ESTI CAS”

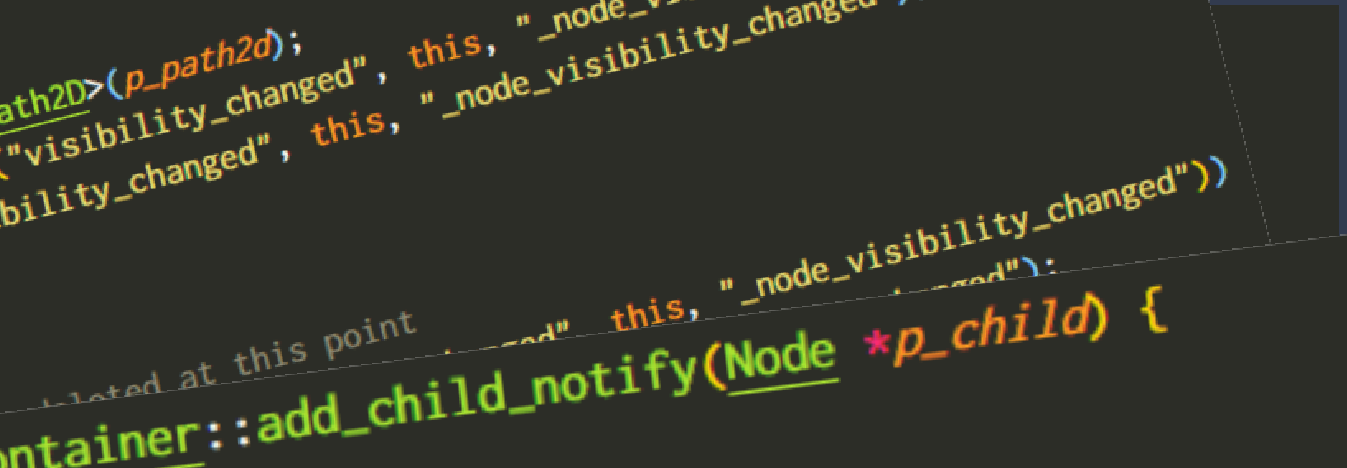


```
Path2DEditor::edit(Node *p_path2d) {

	// [...]

	if (p_path2d) {

		node = Object::cast_to<Path2D>(p_path2d);
		if (!node->is_connected("visibility_changed", this, "_node_visibility_changed"))
			node->connect("visibility_changed", this, "_node_visibility_changed");

	} else {

		// node may have been deleted at this point
		if (node && node->is_connected("visibility_changed", this, "_node_visibility_changed"))
			node->disconnect("visibility_changed", this, "_node_visibility_changed");
		node = NULL;
	}
}
```

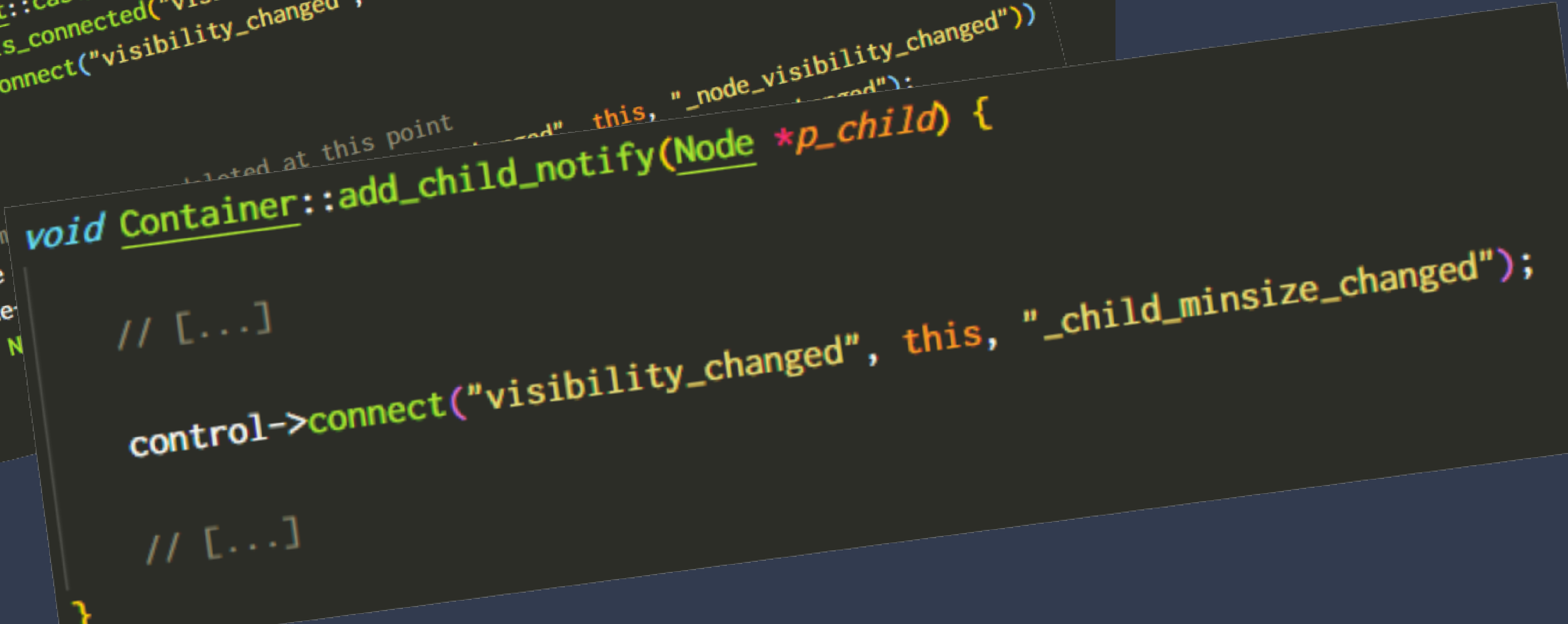
ESTI CAS”



```
Path2DEditor::edit(Node *p_path2d) {

    // [...]

    if (p_path2d) {

        node = Object::cast_to<Path2D>(p_path2d);
        if (!node->is_connected("visibility_changed", this, "_node_visibility_changed"))
            node->connect("visibility_changed", this, "_node_visibility_changed");

    } else {

        // node m
        if (node
            node-
        node = N
    }
}
```

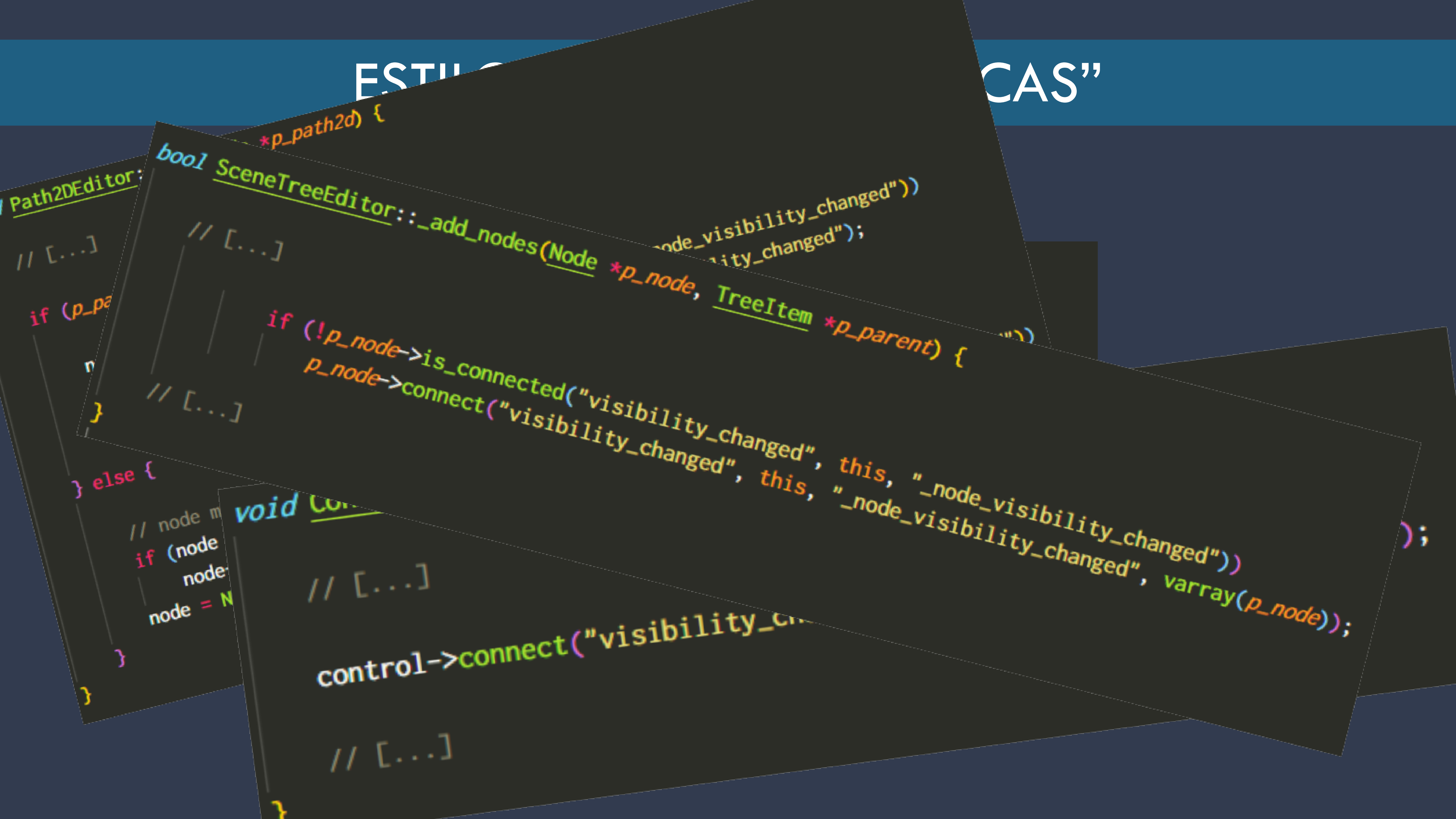


```
                                              this, "_node_visibility_changed"))
                                                                         ged");
             leted at this point
```



```
void Container::add_child_notify(Node *p_child) {

    // [...]

    control->connect("visibility_changed", this, "_child_minsize_changed");

    // [...]

}
```

ESTI CAS”



```
Path2DEditor:
    // [...]
    if (p_pa
        n
    } else {
        // node m
        if (node
            node-
        node = N
    }
}
```

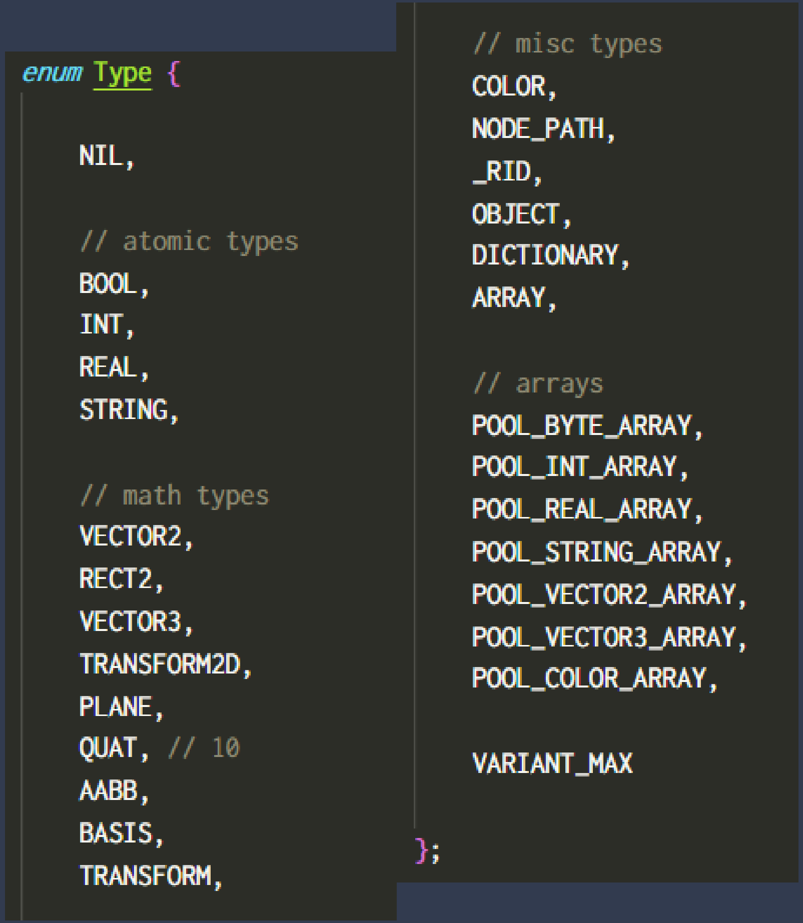
```
*p_path2d) {
ode_visibility_changed"))
ity_changed");
"))
);
```

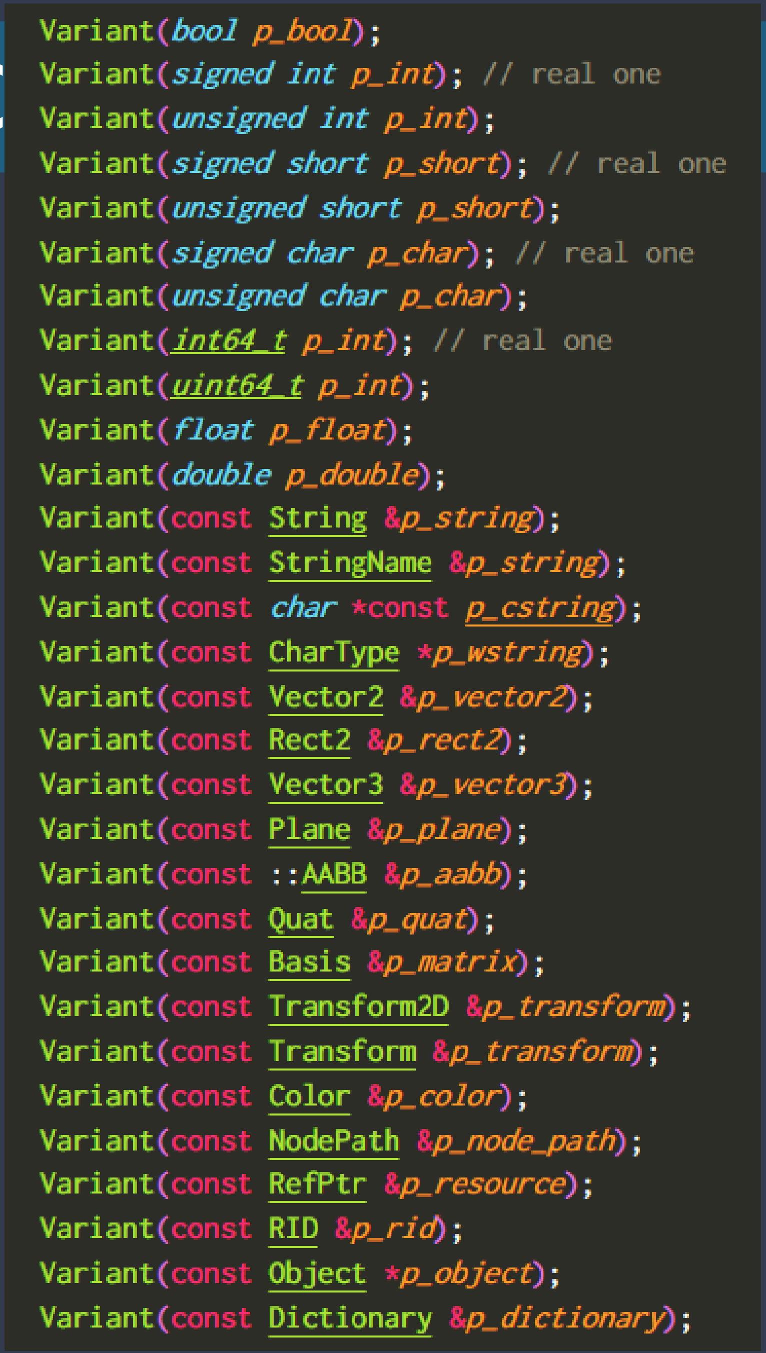
```
bool SceneTreeEditor::_add_nodes(Node *p_node, TreeItem *p_parent) {
    // [...]
            if (!p_node->is_connected("visibility_changed", this, "_node_visibility_changed"))
                p_node->connect("visibility_changed", this, "_node_visibility_changed", varray(p_node));
    // [...]
}
```

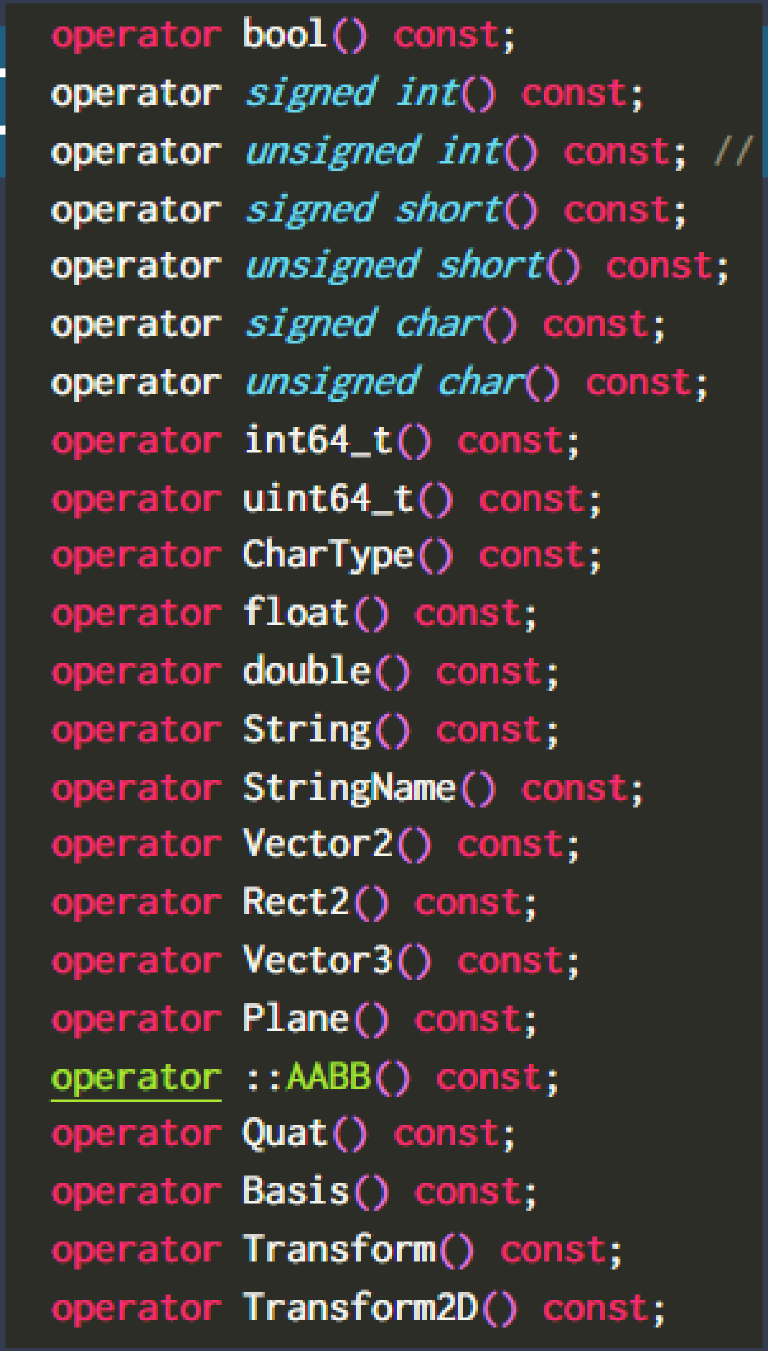
```
void Co
    // [...]
    control->connect("visibility_ch
    // [...]
}
```

INFRAESTRUC D

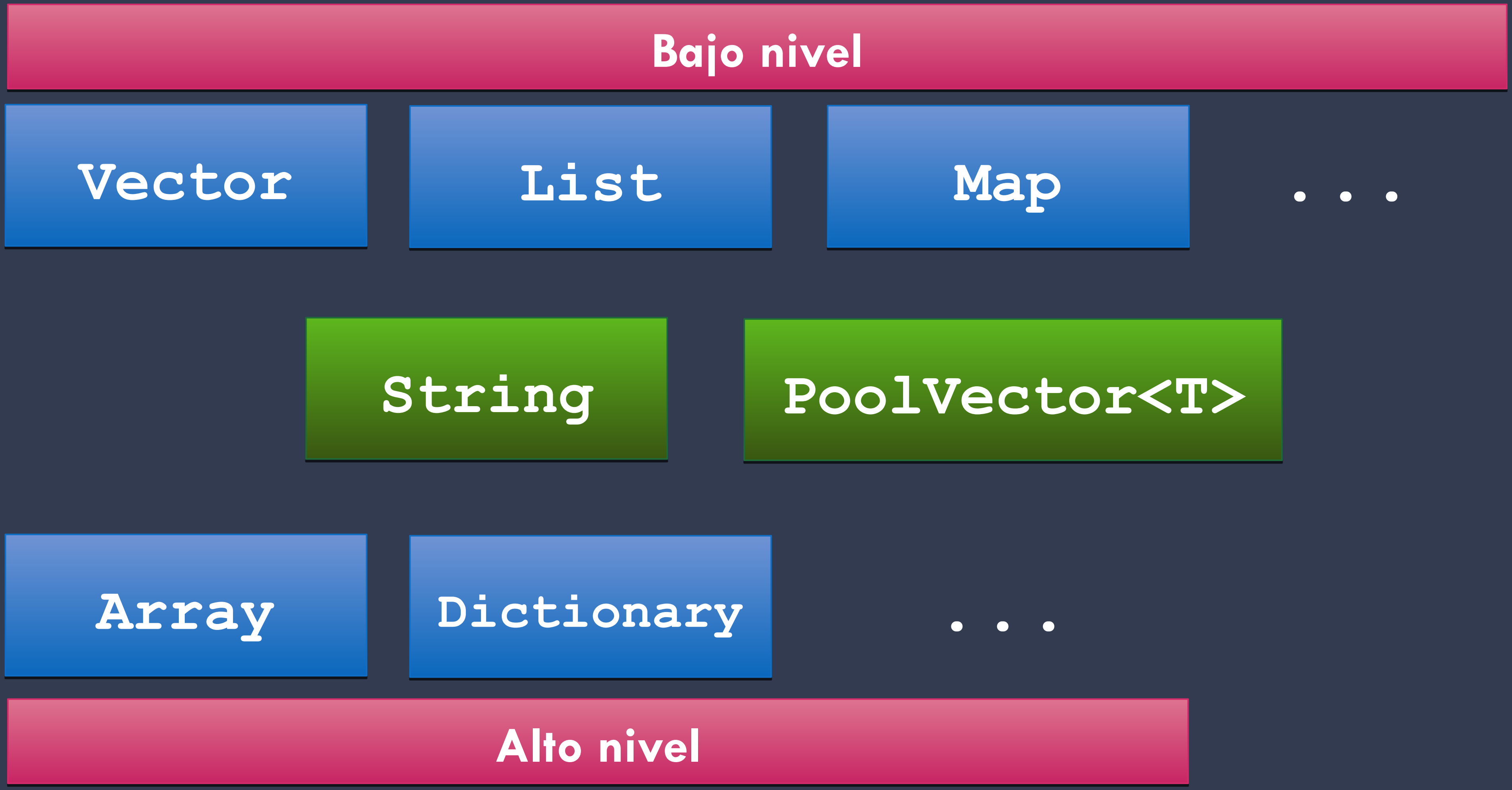
Variant


```
enum Type {

    NIL,

    // atomic types
    BOOL,
    INT,
    REAL,
    STRING,

    // math types
    VECTOR2,
    RECT2,
    VECTOR3,
    TRANSFORM2D,
    PLANE,
    QUAT, // 10
    AABB,
    BASIS,
    TRANSFORM,

    // misc types
    COLOR,
    NODE_PATH,
    _RID,
    OBJECT,
    DICTIONARY,
    ARRAY,

    // arrays
    POOL_BYTE_ARRAY,
    POOL_INT_ARRAY,
    POOL_REAL_ARRAY,
    POOL_STRING_ARRAY,
    POOL_VECTOR2_ARRAY,
    POOL_VECTOR3_ARRAY,
    POOL_COLOR_ARRAY,

    VARIANT_MAX

};
```


```
Variant(bool p_bool);
Variant(signed int p_int); // real one
Variant(unsigned int p_int);
Variant(signed short p_short); // real one
Variant(unsigned short p_short);
Variant(signed char p_char); // real one
Variant(unsigned char p_char);
Variant(int64_t p_int); // real one
Variant(uint64_t p_int);
Variant(float p_float);
Variant(double p_double);
Variant(const String &p_string);
Variant(const StringName &p_string);
Variant(const char *const p_cstring);
Variant(const CharType *p_wstring);
Variant(const Vector2 &p_vector2);
Variant(const Rect2 &p_rect2);
Variant(const Vector3 &p_vector3);
Variant(const Plane &p_plane);
Variant(const ::AABB &p_aabb);
Variant(const Quat &p_quat);
Variant(const Basis &p_matrix);
Variant(const Transform2D &p_transform);
Variant(const Transform &p_transform);
Variant(const Color &p_color);
Variant(const NodePath &p_node_path);
Variant(const RefPtr &p_resource);
Variant(const RID &p_rid);
Variant(const Object *p_object);
Variant(const Dictionary &p_dictionary);
```


```
operator bool() const;
operator signed int() const;
operator unsigned int() const; //
operator signed short() const;
operator unsigned short() const;
operator signed char() const;
operator unsigned char() const;
operator int64_t() const;
operator uint64_t() const;
operator CharType() const;
operator float() const;
operator double() const;
operator String() const;
operator StringName() const;
operator Vector2() const;
operator Rect2() const;
operator Vector3() const;
operator Plane() const;
operator ::AABB() const;
operator Quat() const;
operator Basis() const;
operator Transform() const;
operator Transform2D() const;
```

INFRAESTRUCTURA: CONTENEDORES



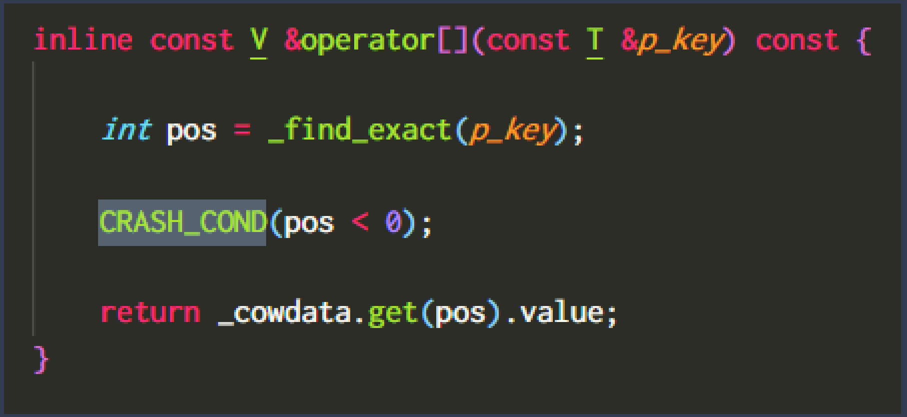
INFRAESTRUCTURA: CONTENEDORES

```
ADDFUNC0R(DICTIONARY, INT, Dictionary, size, varray());
ADDFUNC0R(DICTIONARY, BOOL, Dictionary, empty, varray());
ADDFUNC0NC(DICTIONARY, NIL, Dictionary, clear, varray());
ADDFUNC1R(DICTIONARY, BOOL, Dictionary, has, NIL, "key", varray());
ADDFUNC1R(DICTIONARY, BOOL, Dictionary, has_all, ARRAY, "keys", varray());
ADDFUNC1R(DICTIONARY, BOOL, Dictionary, erase, NIL, "key", varray());
ADDFUNC0R(DICTIONARY, INT, Dictionary, hash, varray());
ADDFUNC0R(DICTIONARY, ARRAY, Dictionary, keys, varray());
ADDFUNC0R(DICTIONARY, ARRAY, Dictionary, values, varray());
ADDFUNC1R(DICTIONARY, DICTIONARY, Dictionary, duplicate, BOOL, "deep", varray(false));
ADDFUNC2R(DICTIONARY, NIL, Dictionary, get, NIL, "key", NIL, "default", varray(Variant()));
```

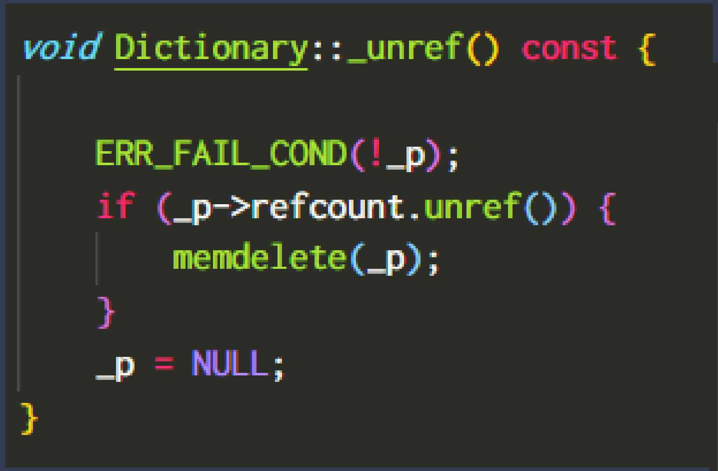
Alto nivel

INFRAESTRUCTURA: GESTIÓN DE ERRORES

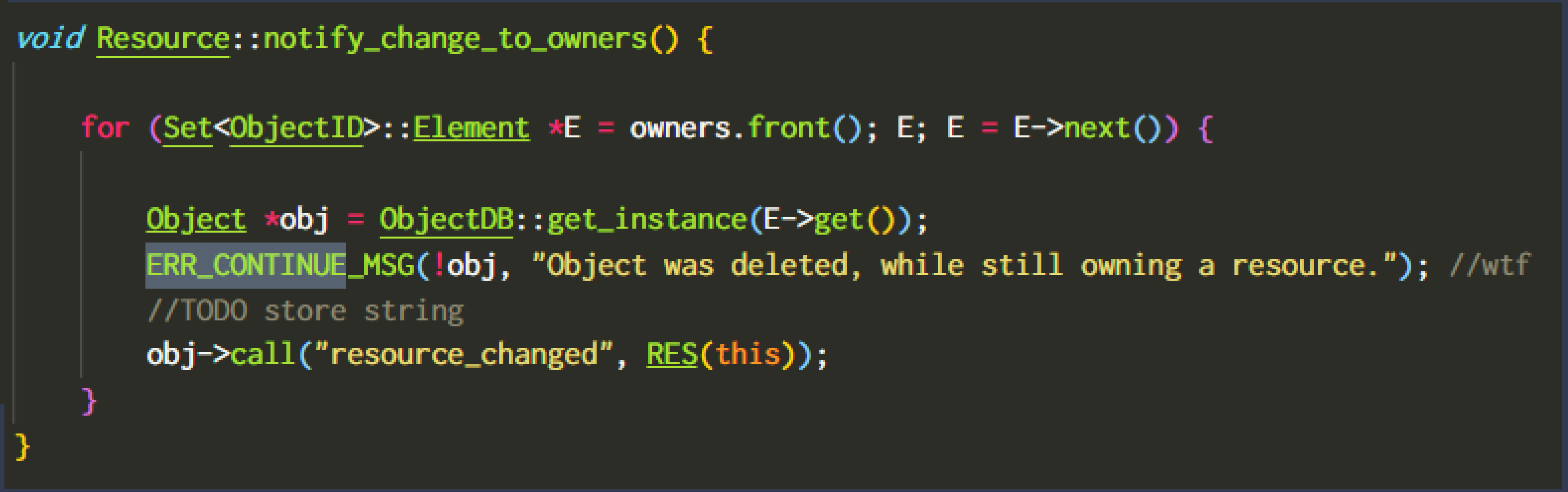
-fno-exceptions



```
inline const V &operator[](const T &p_key) const {

	int pos = _find_exact(p_key);

	CRASH_COND(pos < 0);

	return _cowdata.get(pos).value;
}
```

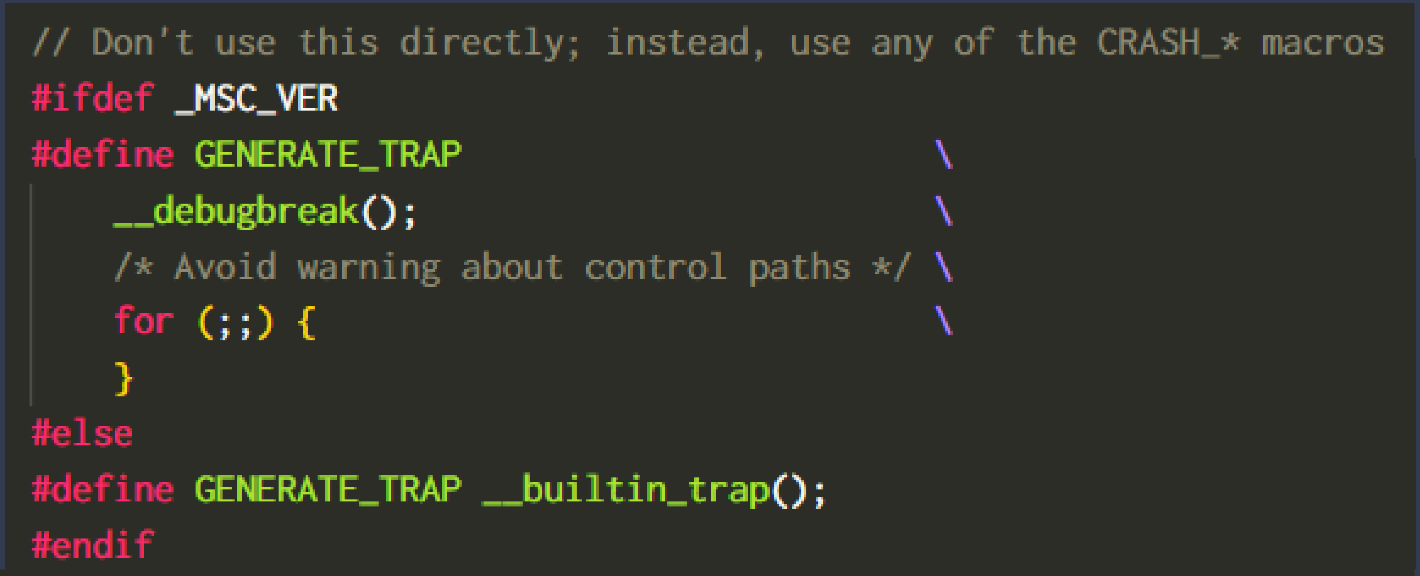


```
void Dictionary::_unref() const {

	ERR_FAIL_COND(!_p);
	if (_p->refcount.unref()) {
		memdelete(_p);
	}
	_p = NULL;
}
```

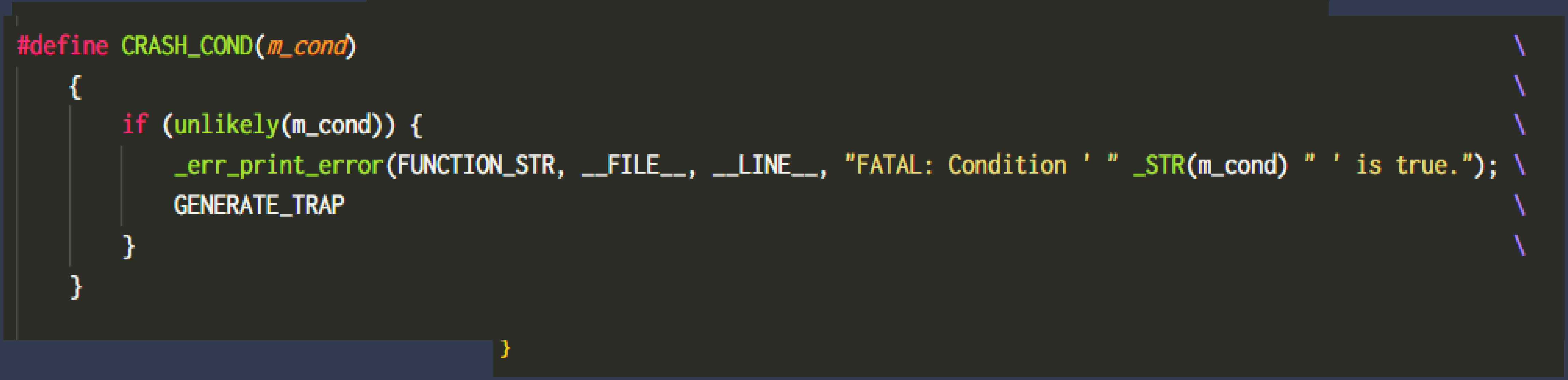


```
void Resource::notify_change_to_owners() {

	for (Set<ObjectID>::Element *E = owners.front(); E; E = E->next()) {

		Object *obj = ObjectDB::get_instance(E->get());
		ERR_CONTINUE_MSG(!obj, "Object was deleted, while still owning a resource."); //wtf
		//TODO store string
		obj->call("resource_changed", RES(this));
	}
}
```

INFRAESTRUCTURA: GESTIÓN DE ERRORES

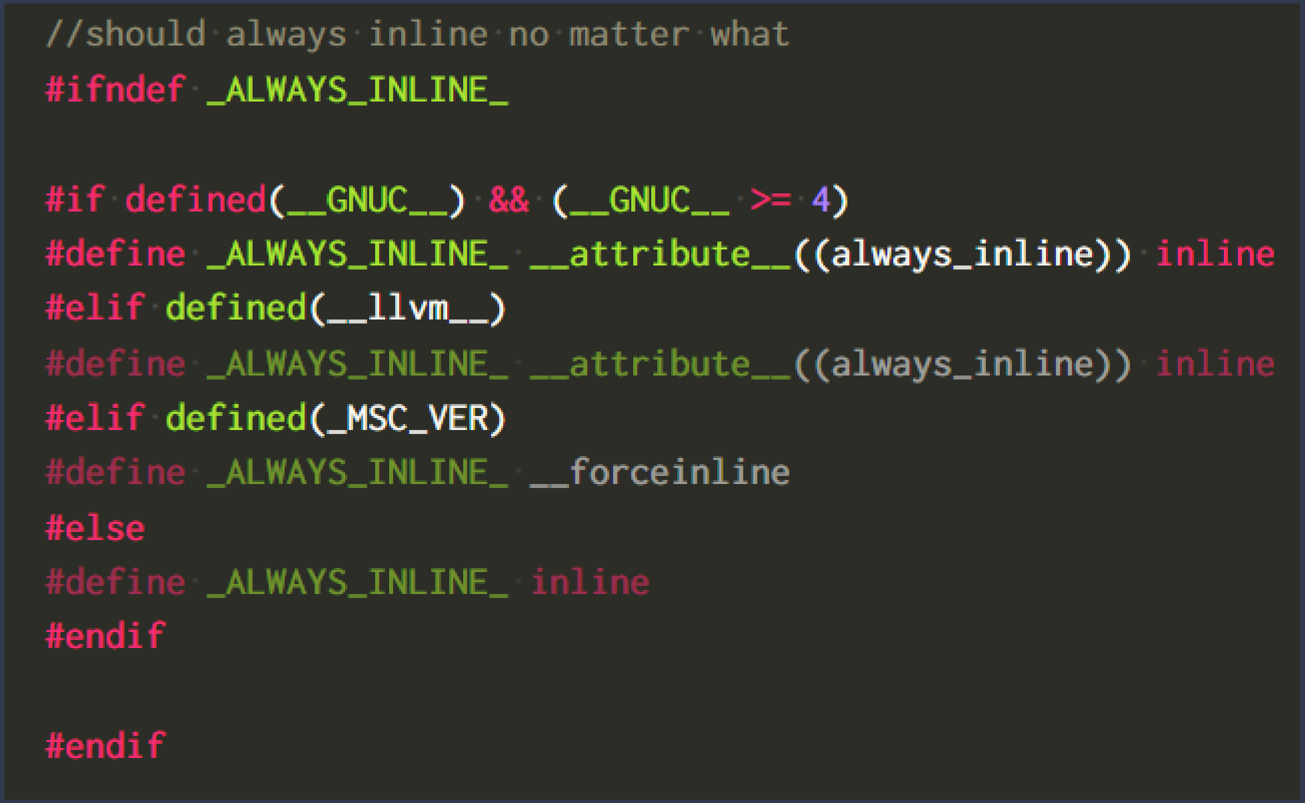


```
// Don't use this directly; instead, use any of the CRASH_* macros
#ifdef _MSC_VER
#define GENERATE_TRAP                            \
    __debugbreak();                              \
    /* Avoid warning about control paths */      \
    for (;;) {                                   \
    }
#else
#define GENERATE_TRAP __builtin_trap();
#endif
```



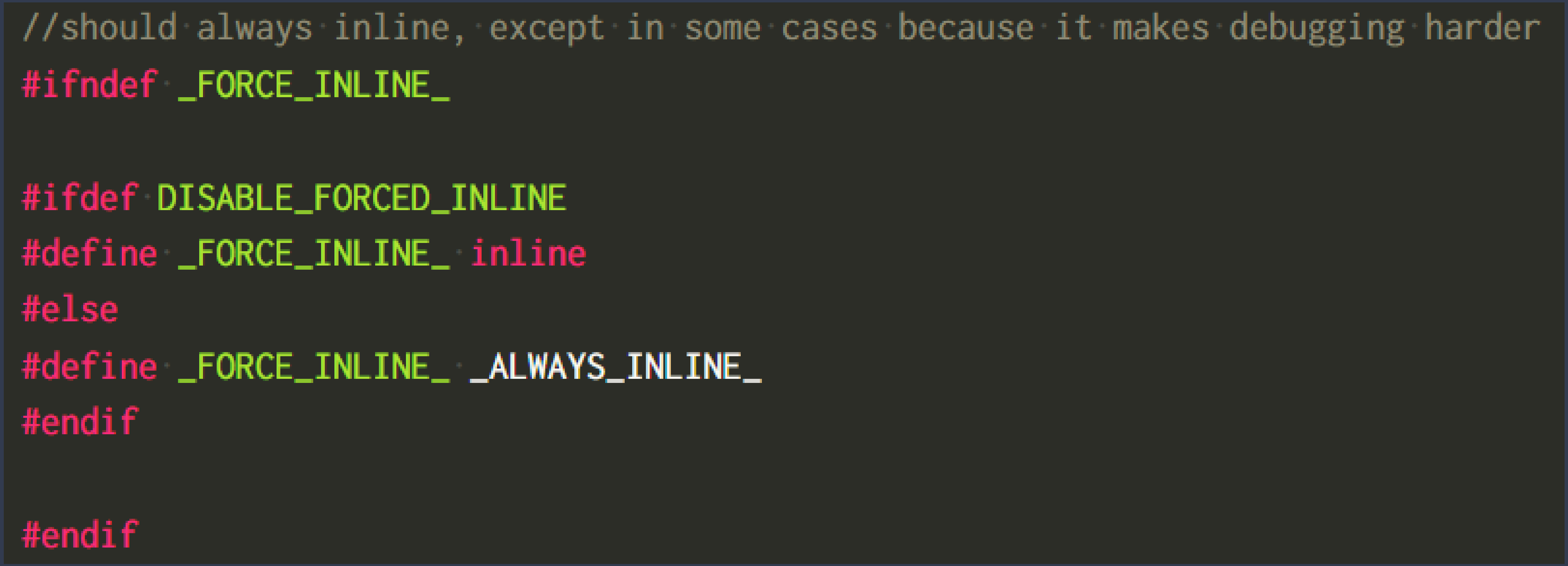
```
#define CRASH_COND(m_cond)                                                                                      \
    {                                                                                                           \
        if (unlikely(m_cond)) {                                                                                 \
            _err_print_error(FUNCTION_STR, __FILE__, __LINE__, "FATAL: Condition ' " _STR(m_cond) " ' is true."); \
            GENERATE_TRAP                                                                                       \
        }                                                                                                       \
    }

                    }
```

OPTIMIZACIÓN: *INLINING*



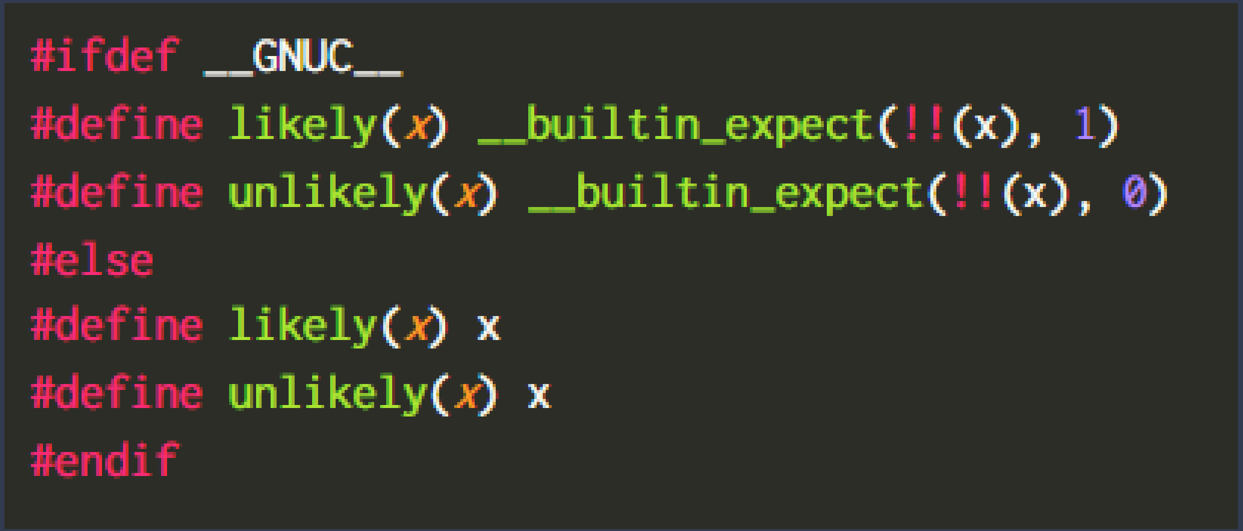
```
//should always inline no matter what
#ifndef _ALWAYS_INLINE_

#if defined(__GNUC__) && (__GNUC__ >= 4)
#define _ALWAYS_INLINE_ __attribute__((always_inline)) inline
#elif defined(__llvm__)
#define _ALWAYS_INLINE_ __attribute__((always_inline)) inline
#elif defined(_MSC_VER)
#define _ALWAYS_INLINE_ __forceinline
#else
#define _ALWAYS_INLINE_ inline
#endif

#endif
```

OPTIMIZACIÓN: *INLINING*



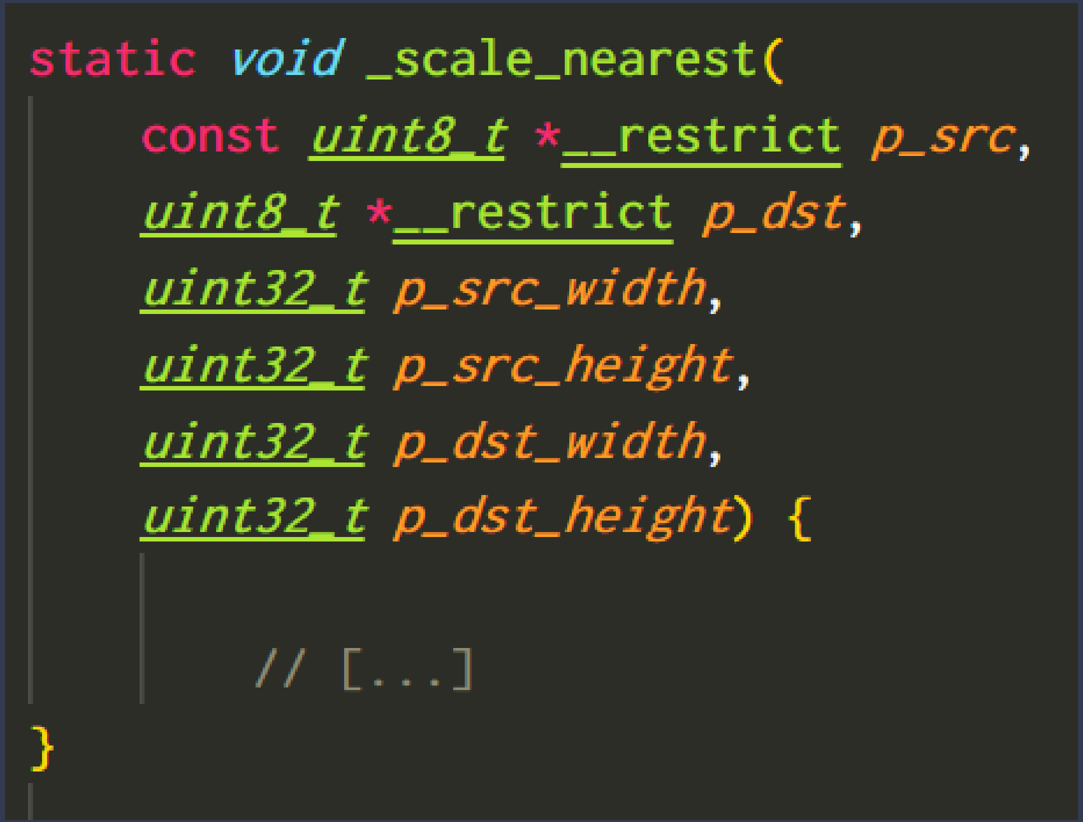
```
//should always inline, except in some cases because it makes debugging harder
#ifndef _FORCE_INLINE_

#ifdef DISABLE_FORCED_INLINE
#define _FORCE_INLINE_ inline
#else
#define _FORCE_INLINE_ _ALWAYS_INLINE_
#endif

#endif
```

OPTIMIZACIÓN: RAMAS



```
#ifdef __GNUC__
#define likely(x) __builtin_expect(!!(x), 1)
#define unlikely(x) __builtin_expect(!!(x), 0)
#else
#define likely(x) x
#define unlikely(x) x
#endif
```

OPTIMIZACIÓN: __RESTRICT



```
static void _scale_nearest(
		const uint8_t *__restrict p_src,
		uint8_t *__restrict p_dst,
		uint32_t p_src_width,
		uint32_t p_src_height,
		uint32_t p_dst_width,
		uint32_t p_dst_height) {

			// [...]
}
```

MODERNIZACIÓN

Contenedores STL

NO

Motivos:

Migrar a la STL a estas alturas manteniendo la semántica sería mucho esfuerzo para poco o nulo beneficio.

Punteros inteligentes estándar

NO

Motivos:

Gran esfuerzo para adaptarlos al código actual, que ya tiene su propia manera de hacerlo.

Inicialización uniforme {}

`SÍ, en la declaración`

Motivos:

Sobre todo, legibilidad: no tener que ir al constructor para ver el valor inicial.

`initializer_list`

SÍ

Motivos:

Mejora la interfaz de los contenedores propios, así como el rendimiento, en comparación con varias inserciones individuales.

Sin espacio entre >

SÍ

Motivos:

Mejora la legibilidad.

`override`

`SÍ`

Motivos:

Con un poco de esfuerzo por nuestra parte, el compilador detecta casos erróneos de sobrescritura de métodos.

`final`

SÍ

Motivos:

Por seguridad, especialmente en el núcleo del motor.

`explicit`

SÍ

Motivos:

Para evitar conversiones implícitas no desedas, que dan lugar a errores.

`nullptr`

SÍ

Motivos:

Porque es lo correcto.

async/promise

SÍ, donde tenga sentido

Motivos:

Porque facilita mucho programar ciertos comportamientos.

Multiproceso

`SÍ (thread, atomic)`

Motivos:

Para abandonar las implementaciones específicas para cada plataforma y eliminar usos erróneos de `volatile`.

¡GRACIAS!

¿PREGUNTAS?

¡GRACIAS!

¡Nos vemos en Twitter!

@RandomPedroJ

Apoya Godot en

PATREON