NEXT MEETING

NOTE: Add anything you would like to discuss! Include your (user)name so you can intro the issue/PR/topic to the group

- Topics for discussion at this meeting:
    - Bromeon: in 4.7 cycle - dependencies between GDExtensions?
    - Naros: MethodBind needs (follow-up from last meeting)
        - MethodInfo - Name, Flags, Argument Details, Argument Count, Return Details
        - Instance Class Name
        - Call, PtrCall
    - Naros: Status of GDExtension low-level API access, e.g. using node unique ids vs names.
    - van800: Update about godot-cpp.natvis (here).
- Review issues?
    - https://github.com/godotengine/godot-cpp/issues
    - https://github.com/godotengine/godot/issues?q=is%3Aissue+is%3Aopen+label%3Atopic%3Agdextension+-milestone%3A3.x
- The full PR list (if we have time, we can start going down it):
    - https://github.com/godotengine/godot-cpp/pulls
    - https://github.com/godotengine/godot/pulls?q=is%3Aopen+is%3Apr+label%3Atopic%3Agdextension

Long-term topics to maybe (re-)discuss:

- Shortage of documentation (discussion with ivorius here: https://chat.godotengine.org/channel/gdextension?msg=Q8ng7gvDMHPyTRSqW)
    - The following points are paraphrased and shortened from the perspective how I, Patrick understood them
        - Should the godot-C++ documentation be expanded in the main godot docs page or should it be split up to advance the production of documentation?
        - Godot SCons interaction documentation is not existent specifically for godot-cpp
        - If godot-cpp had its own section with sub-articles it would help because it is hard to find everything relevant for godot-cpp in the godot docs
            - External docs would be better to search / scan through
                - Example: Godot Rust docs

        - My (Patrick) answer:
            - SCons specific docs are missing
            - Current issue is not the searchability but more docs content itself and won't be solved by splitting the godot-cpp docs from the main docs page
                - Splitting the page will make it probably even harder since many will wonder why they are split and will be an extra step since most people have it in their muscle memory to go to the godot docs when looking for something
            - Docs of godot-rust are really great but are also written by several people
                - We need more people donating time on the docs (the following are new points and do not stem from the discussion mentioned above)
                    - Needed pages
                        - Using third-party libraries (possibly with vcpkg and/or conan) (working on that myself)
                        - Debugging gdextensions
                        - Editor extensions
                        - Runtime extension classes
                        - Handling custom project settings
                        - Extending servers and adding your own systems
                            - Rendering
                            - Physics
                            - …

2026-05-12

- Attendees:
    - dsnopek
    - van800 / Ivan
    - realkotob
    - Ivorious / Lukas
    - Chris (Naros)
    - Jan (Bromeon)
    -
- Topics for this meeting:
    - realkotob: Expose mesh surface RD buffer RIDs #118973
        - dsnopek: This is the purview of the rendering team; they have their team meeting tomorrow at this same time 🙂 - see #rendering channel on RocketChat

        - Lukas: marking these APIs is as experimental is an option
    - Ivorius: [godot-cpp] Remove api_version default?
        - https://github.com/godotengine/godot-cpp/pull/1972#pullrequestreview-4220083394
        - dsnopek: We have the default to maintain support for older workflows, but the v10 update is an OK time to break these
        - Lukas: we could have a warning if it's not specified (so folks have time to update their workflows)
        - realkotob: personally had confusion with this recently - it was unclear which version to use for Godot 4.6
        - Bromeon:
            - had similar dilemma with the Rust bindings, because it wasn't possible to require the version
            - Users had confusion when a new version of Rust bindings came out, why it didn't work with old Godot versions anymore
        - Naros: usually pin support for a particular version, it's unclear why `master` has a default - I would want to control this from my extension
        - dsnopek: remove now or do something in between?
        - Naros: fine with doing something in between, if the long-term plan is to remove it
        - Lukas: removing it now has the advantage to educate folks on how the version targeting works
        - **AI:** Lukas will make a proposal and link on RocketChat to see if anyone is opposed
    - van800: I can share what is upcoming in Rider regarding godot-cpp.natvis and addon template with GDExtension
        - Ivan:
            - Working for JetBrains and improving Godot support in Rider
            - Re natvis file: would be nice to merge what we have, and improve it more later (ie add more Godot types)
            - Re addon template for making a GDExtension:
                - dsnopek: Super cool!!!!
                - dsnopek: Only concern is the uncommon project layout with the C++ source files under addons/NAME/src
        - **AI:** Chris will give it a try the natvis file with CLion and/or Visual Studio to verify that it works
    - Bromeon: Object::call_const expose to GDExtension?
        - dsnopek: Expose a call_const to GDExtension or expose a "get info" to get a struct of info about a method bind object
            - Which one? Will we need the info for anything else
        - Chris: having the info about the method bind would be useful my extension that adds a script language
        - **AI:** Chris will make a list of info that they would like to access about a method bind

    - Bromeon: ClassDB::class_get_default_property_value calls memdelete -> UB for RefCounted
        - Relates to some work Lukas is doing on ensuring that RefCounted objects are always handled through Ref<T>
        - **AI:** Bromeon to make a Godot issue - Lukas will look at later
        - Done: https://github.com/godotengine/godot/issues/119425
    - Bromeon: Engine::register_singleton accepts RefCounted, but causes UB
        - see Singleton struct
        - dsnopek: Should just be an error to add a RefCounted singleton
        - **AI:** Bromeon to research and potentially open a PR
    -

2026-04-29

- Attendees:
    - dsnopek
    - Ivorius / Lukas
    - cerberus1746
    - aaronfranke
    - Quentin Quaadgras (Splizard)
    - enetheru
- Topics for this meeting:
    - Enetheru: GitHub Actions deprecation warning and deprecated actions/pre-commit
        - Is there a plan to migrate from the pre-commit action to the pre-commit.ci service or add a step, or find an alternative?
        - https://github.com/pre-commit/action
        - Discussion:
            - dsnopek: Let's do what Godot did and make our own action: https://github.com/godotengine/godot/blob/master/.github/actions/pre-commit/action.yml
    - dsnopek: Updating JSON files in godot-cpp for Godot 4.7-beta1
        - https://github.com/godotengine/godot-cpp/pull/1972
    - dsnopek: Plan to release godot-cpp v10-stable shortly after Godot 4.7-stable?
        - This gives us some time to try sync'ing some files from Godot
    - dsnopek: A couple more godot-cpp PRs that could use some reviews
        - https://github.com/godotengine/godot-cpp/pull/1973
        - https://github.com/godotengine/godot-cpp/pull/1974
        - https://github.com/godotengine/godot-cpp/pull/1975
    - Enetheru: More work on my re-work of the integration testing to python
        - Am I doing too much, or too little? I can solve some of this in multiple ways eg annotations vs additional jobs
        - https://github.com/godotengine/godot-cpp/pull/1922

    - Quentin Quaadgras: integrate profiling from GDExtension into the editor?
        - Needs exploration

2026-04-14

- Topics for this meeting:
    - dsnopek
    - Chris (Naros)
    - Ivorius / Lukas
    - Jan (Bromeon)
    - ValorZard
    - Yarwin
- Topics for this meeting:
    - https://github.com/godotengine/godot/pull/118214
        - Yarwin did some testing, so this should be good to go!
        - **AI:** David to review one more time before feature freeze
    - Naros: Ability to delegate Variant::callp to ScriptExtension delegates for script-level static functions https://github.com/godotengine/godot-proposals/issues/14614
        - GDScript overrides Script::callp()
        - To support this we need to add ScriptExtension::_callp() which overrides Script::callp()
    - Naros: Automated checks for common struct inconsistencies, e.g., Method/Property Infos https://github.com/godotengine/godot-cpp/pull/1965#issuecomment-4189639732
        - Lukas: Occasionally run AI to compare the two implementations and see if it reports anything for us to look into
        - **AI:** Perhaps make an issue explaining the idea and putting it up for grabs (having full details of the prompt used in the experiment would be essential)
    - Bromeon: candidates for required ptrs
        - SceneTree::get_root, SceneTree::get_current_scene (?)
        - SceneTree::get_multiplayer (?), SceneTree::set_multiplayer arg (?)
        - Engine::get_main_loop,
        - Engine::[un]register_singleton arg, Engine::[un]register_script_language arg
        - OS::add/remove_logger
        - **AI:** Bromeon to make a PR (or two, if we want to target Godot 4.7, and merge the easy ones quickly)
    - Bromeon: GDExtension casts need post-validation – by design?
        - gdextension_object_cast_to() takes “class tag”, checks is_class_ptr()
            - Native class (e.g. Node): unique tag per class

                - GDExtension class: uses nearest native ancestor's tag (not unique anymore)
            - Past issue: https://github.com/godotengine/godot-cpp/issues/995
            - Code: gdextension_object_cast_to, gdextension_classdb_get_class_tag, godot-cpp cast, godot-rust cast
            - **AI:** Lukas to make an `Object::derives_from(StringName)` method that could be used instead of the current `object_cast_to` GDExtension interface function
    - Bromeon: RPC IDs in GDExtension vs. GDScript
        - Script-level RPCs (@rpc in GDScript)
        - Node-level RPCS (Node::rpc_config, 15th bit `0x8000` set)
        - dsnopek: I think this is an unintentional bug
        - **AI:** Bromeon to make an issue for discussion (feedback from Fabio would be great)
    - ValorZard (~~Not~~ here)
        - Proposal from core: Migrate GDScript to GDExtension https://github.com/godotengine/godot-proposals/issues/14652
            - This would make a GDScript file no longer a resource, is there a way to make a GDExtension C++/Rust file a resource?
            - Also, is there a way to make unnamed classes work in GDExtension
                - Valor: This is really important, I do not want to have to give all my scripts class names
            - Ditto with built-in scripts
                - Valor: I did not know this was a feature in Godot until now.
            - Y: it would be great time to establish common interfaces which could be used through whole GDExtension system - like common classes for handling async and whatnot (so Cpp/C#/Rust can return coroutine or whatever which can be used flawlessly by gdscript user and vice versa)
    -

2026-03-31

- Attendees:
    - dsnopek
    - Patrick/FlameLizard
    - Chris (Naros)
    - Ivorius / Lukas
    - Yarwin
- Topics for this meeting:
    - Patrick/FlameLizard: Are we still going forward with this PR?: https://github.com/godotengine/godot-docs/pull/11652

        - Discussion: Still seems like a good idea; needs a rebase
    - Samuel/Enetheru: https://github.com/godotengine/godot-cpp/issues/1954 Just want to clarify that I think that deviating from the exact options that scons has is OK if I am omitting them, but no extra features is the goal?
        - Discussion: We think this is fine (assuming we understand correctly): in cases where cmake has a particular standard way of doing things, we should adopt that, rather than forcing our configuration to have an option just because scons has the same option
    - dsnopek: Alignment of builtin types not in extension_api.json
        - godot-cpp issue and PR where this came up:
            - https://github.com/godotengine/godot-cpp/issues/1956
            - https://github.com/godotengine/godot-cpp/pull/1958
        - This definitely affects any GDExtension binding, though
        - Discussion:
            - Lukas: this should definitely be in the extension_api.json, because there are some situations that can only be solved by having a higher alignment
            - dsnopek: is also in favor putting in extension_api.json; only downside is complexity
            - Patrick: is in favor of extension_api.json too
    - dsnopek: Add Variant::get_type_by_name to GDExtension Interface
        - https://github.com/godotengine/godot/pull/117160
        - Discussion:
            - Lukas: for a generic system, we could only address variant enums or object enums, we couldn't address both with the same functions at the moment (with a unification of variant and object reflection)
            - Naros: another option is deprecating the old function, and doing it entirely within language bindings; this would be more performant because we don't cross the boundary
            - Lukas: since these don't change, these can be cached
            - dsnopek: Given that these are not the exact enum types ("Vector2i" vs "TYPE_VECTOR2I") a general system doesn't exactly work; having the table in the bindings, means we can't handle unknown types (in the rare case where a new type is added, and the GDExtension itself doesn't need to understand the type, just pass it on, like the XML docs -> stub bindings discussion)
            - Lukas/Patrick: if this isn't too problematic, and we don't have the generic system, why not add it
    - Yarvin: Support immutable resources for thread-safety: https://github.com/godotengine/godot-proposals/discussions/14350
        - dsnopek: Usual approach in Godot is to promise to yourself that you won't change the resource and only do reading, and this should be thread safe.

Enforcing the immutability in Godot would be a core question, which might not be reasonable
        - Lukas: enforcing this in core would add a small performance cost to all setters
        - Yarwin: changing immutability is only possible if there is only one reference
        - Lukas: would be a matter for core and would require lots of discussion
    - Ivorius: Let's discuss Ref ownership transfer for GDExtension (pre-pr)
        - dsnopek: We need a new GDExtension interface function for creating RefCounted objects with new ownership semantics, while keeping the old one
        - Lukas: Keeping the old way could maybe block removing some bad stuff from RefCounted
        - Lukas or dsnopek will get to this eventually
        - We may also need a new version of the GDExtension interface function for looking up objects in the ObjectDB
    - Ivorius: Allow editing .gdextension files in the editor? (PR)
        - Lukas: this is a feature that can only break things
        - Naros: this can't be used by end users
        - Naros: this doesn't let you edit everything in the file to begin with
        - dsnopek: this only makes sense if you can create a GDExtension in the editor, which you can't
        - dsnopek: in GDNative we had it for free because it was a resource, in this case we need to maintain code to build the UI (which we need things add going forward)
        - Patrick: this would amount to issues from users when they break things
        -

2026-03-03

- Attendees:
    - dsnopek
    - Chris (Naros)
    - Ivorius / Lukas
    - Patrick (FlameLizard)
- Topics for this meeting:
    - Naros: Accessing ClassDB::add_compatibility_class to GDExtension
        - A great way for plugins to map old classes to new classes, particularly for resources
    - Naros: Exposing niche engine/editor state to GDExtension
        - Areas like: ScriptServer, DocData, overall editor widget integration
            - Do we use script bindings or gdextension interface?
    - dsnopek: When to release godot-cpp v10.0-stable?

        - Try to sync more stuff from Godot before stable release
        - Let's sync with `master` (as opposed to sticking with `4.6`) - they shouldn't change too much
        - (Try to make comments for important differences between godot and godot-cpp)
        - Let's remove math.compat.inc (and any similar compat code) for v10
        - We should sync defs.hpp because some things have moved into a namespace
        - Maybe address this for v10:
            - https://github.com/godotengine/godot-cpp/pull/1781
            - Needs some small changes; and we can the lower-case name rather than switching to upper-case

2026-02-17

- Attendees:
    - dsnopek
    - cerberus1746
    - Patrick/FlameLizard
    - Martin Delille
    - Valorzard
    - Jan (Bromeon)
    - sergi.perello
    - Chris (Naros)
    - Thaddeus Crews
- Topics for this meeting:
    - Start pinging for the meeting at the time (not just an hour in advance)
    - Ivorius: Should we adopt the Core triage project views? (for GDExtension Triage)
        - The core project's configuration looks much better than what we have currently, so let's do it! No objections
        - Thaddeus: Implemented during meeting; now we mirror core triage 1-1
            - Only exception is "status"; we use a string field instead of pre-defined labels
    - Enetheru: Should I continue on this path? Integration test bash to python, add docXML, run on windows and macos : #1922
        - Naros: Good for cross platform
        - Patrick: Is the performance the same on CI - there any penalty?
        - David: We should have only one script, replace the bash one, and use it always
        - **Overall:** This makes sense!
    - sergi.perello(DiplomaticRobot on gh): https://github.com/godotengine/godot/issues/105615

        - Wants to suppress the error about GDExtension library not being available on platforms the GDExtension doesn't support
        - Error happens every few seconds when using the editor
            - Needs to be investigated!
            - If the extension is not yet loaded and the EditorFileSystem performs a scan when a resource or other file is changed, a call is made to GDExtensionManager ensure_extensions_loaded that likely triggers this often when its not yet loaded.
        - Two ways to make iOS plugins: .gdip plugins, and SwitGodot (via GDExtension) which allows using Swift package manager
        - George's proposal was to mark platforms as not supported by setting a specific string as path (but not ideal because the list of not support can expand)
        - Chris: editor or project setting for disabling this error?
        - seri: Change from ERROR to WARNING, because their CI is set to fail if it sees an ERROR
        - Possible fix:
            - Mark GDExtensions as being "platform specific"
            - If it's set, then look at all the [libraries] entries, filter out the platforms, and then if the error is associated with one of those platforms, then we shown it. Otherwise, silently ignore the error
    - dsnopek: Allow viewing and editing GDExtensions from inside Project Settings
        - https://github.com/godotengine/godot/pull/115846
        - Naros: What is "autodetect_library_prefix"?
        - Patrick: More properties should be viewable in columns on the list of GDExtensions (rather than having to click through on each of them)
    - Bromeon: nullable/required object parameters in signals
        - How to start: start by figuring out what the implementation could look like, and then see how the core team feels about it

2026-02-03

- Attendees:
    - dsnopek
    - Ivorius / Lukas
    - aaronfranke
    - Chris (Naros)
    - Mr Samuel Nicholas
- Topics for this meeting:
    - General issues when a new GDExtension is loaded for the first time (or the first time the project is loaded):
        - https://github.com/godotengine/godot/pull/114131
        - https://github.com/godotengine/godot/pull/104868

        - Naros: Should we require users to reload the editor when a new GDExtension is discovered?
        - David: Would prefer to fix the bits in the editor that are assuming that things that adding a GDExtension won't change
        - Having the script creation dialog check when the dialog is first shown is fine
    - Enetheru: ”installable” godot-cpp see github issue
        - Enetheru: Lots of folks of want this - may be willing to take over PR for this
        - David: More OK with installable than previously
    -
    -

2026-01-20

- Attendees:
    - dsnopek
    - Patrick / FlameLizard
    - Chris (Naros)
    - Jan (Bromeon)
    - cerberus1746
    - Yarvin
    - Ivorius / Lukas
- Topics for this meeting:
    - Chris/Naros: Script vs ScriptExtension vs Object virtual behavior differences
        - Want to show which methods, properties, signals are on which classes
        - How to work around the fact there is no way to determine what methods are uniquely defined or overridden inside one script of a multi-tier script class hierarchy because `has_method` delegates to ClassDB, which always returns false.
        - **Make a ScriptExtension::has_script_method() to call the Script::has_method()**
    - patrick/flamelizard: Demo renaming for godot-cpp-template and gdextension docs
        - https://github.com/godotengine/godot-cpp-template/pull/109
            - Discussion of Lukas' comment
        - Should the project folder also be renamed for the gdextension docs since in that case it is actually a “demo”?
        - **Let's go with "project"**
    - dsnopek: Goals for Godot 4.7
        - [P0] Allowing one GDExtension to use another (including inheriting from classes)

                - https://godotengine.org/priorities/#allow-gdextensions-to-communicate-with-one-another
            - [P1] Compatibility methods for utility functions
                - https://github.com/godotengine/godot/pull/113736
            - [P2] Ability to enable/disable GDExtensions in project settings
                - https://godotengine.org/priorities/#add-the-ability-to-enable-disable-gdextensions-in-project-settings
            - [P2] Change the GDExtension interface handles used internally to opaque pointers and allow C++ compiler to type-check the conversions
                - https://github.com/godotengine/godot/pull/112125
            - [P0] Moar Docs!!!
            -
            - How to handle marking required arguments on signals
                - **Needs further research**
            - How to handle not having an initialization level that comes after platform classes
                - https://github.com/godotengine/godot/pull/114143
                - Currently, we have main loop callbacks workaround
                - **Yarvin to make a new issue or proposal - then we'll discuss perhaps with the core team or the TLC**
    -

2026-01-06

- Attendees:
    - dsnopek
    - Grant McClure
    - Jan (Bromeon)
    - Patrick (FlameLizard)
    - Ivorius / Lukas
    - Chris (CraterCrash Studios)
- Topics for this meeting:
    - Yarwin: Allowing to specify load order of GDExtensions (related github issue)
        - dsnopek:
            - At the moment we don't "officially" support extension depending on each other (even though some users have made it work)
            - Will need a proper dependency system
        - Grant: how to handle versioning (if two extensions depend on the same extension but at different versions)
    - Yarwin: Runtime/non-tool GDExtension classes are being run in the editor / Placeholder-related logic via GDExtension (how should we create placeholders?) - issue
        - dsnopek: unfortunately, this is how it works by design
        - Related docs PR: https://github.com/godotengine/godot-docs/pull/11578

- dsnopek: [godot-cpp] Review of godot-cpp v10 PRs would be appreciated!
  - Add back support for Godot 4.4
    - https://github.com/godotengine/godot-cpp/pull/1898
  - Add back support for Godot 4.3
    - https://github.com/godotengine/godot-cpp/pull/1899
- dsnopek: [godot-cpp] Should we support Godot versions older than 4.3?
  - Godot 4.3 is the first version that supports documentation, which is a baseline feature
  - Agreed 🙂
- Bromeon: ClassDB initialization: https://github.com/godotengine/godot/issues/114192
  - https://github.com/Bromeon/godot/commit/4813b1a6dd8083d18a035f3e229e313eef24ef93
  - Bromeon: may attempt to make a test that calls all exposed ClassDB methods at the earlier initialization levels
  - dsnopek: The exposed methods seem pretty safe
  - Lukas: It would be good to double check any methods that do anything specific for GDExtension
- paddy-exe: Discuss PR to godot-cpp-template for publishing to the asset library: https://github.com/godotengine/godot-cpp-template/pull/108#issuecomment-3710233047
  - Lukas: complexity doesn't seem to be worth the time savings (a minute or two to upload to the asset library)
  - Chris: if it's going to be automated it needs to be version centric (which may be possible with the asset store)
  - Lukas: at the moment, asset library is hand approved, so releasing too quickly creates work for the Godot Foundation contractors
  - dsnopek: who should our audience be? Only making reusable extensions, or also for making game logic in an extension?
    - Let's go for generic!
    - The only code change we need is renaming the 'demo/' directory to 'project/' (or something else generic)
    - But also some docs need to be updated:
      - https://docs.godotengine.org/en/latest/tutorials/scripting/cpp/gdextension_docs_system.html
      - https://docs.godotengine.org/en/latest/tutorials/scripting/cpp/gdextension_cpp_example.html
- Paddy-exe: Cross-compile naming issue in godot-cpp-template: https://github.com/godotengine/godot-cpp-template/issues/107
  - Looks like there is a bug with our SConstruct where we aren't adding the `lib` prefix building for Android (potentially only when building on Windows)
  - Needs more investigation!

2025-12-09

- Attendees:
    - dsnopek
    - Ivorius / Lukas
    - Quentin Quaadgras
- Topics for this meeting:
    - dsnopek: Minor improvements to gdextension_interface.json:
        - Structured deprecation info: https://github.com/godotengine/godot/pull/113697
        - Based on Bromeon's notes: https://github.com/godotengine/godot/pull/113754
    - dsnopek: [godot-cpp] Generate GDExtension interface header and loader from JSON
        - https://github.com/godotengine/godot-cpp/pull/1895
    -

2025-11-25

- Attendees:
    - dsnopek
    - Ivorius / Lukas
    - Jan (Bromeon)
    - Thaddeus Crews
    -
- Topics for this meeting:
    - Enetheru: ”installable” godot-cpp see github issue
        - Could make sense depending on how vcpkg works!
        - We need someone to explain vcpkg to us so we can understand how it really works
        - Patrick may know more about vcpkg and could help
    - dsnopek: Add RequiredParam<T> and RequiredResult<T> to mark Object * arguments and return values as required
        - PR: https://github.com/godotengine/godot/pull/86079
        - **Was just merged!!**
        - I'd like to do one more PR marking a "bunch" of stuff as required, starting with those things that are most important to language bindings - what are those things?
        - Action Item:
            - Jan will look at old issues and ask the Rust community
            - David will ping Miguel and ask if he has a list too

    - Bromeon: Discuss unifying methods (ClassDB, variant builtin, utility) into one system
        - Possibly related proposal:
            - https://github.com/godotengine/godot-proposals/issues/12622
            - I swear Ivorius had a proposal related to this, but I'm not sure (maybe it's that one, although, looking at it now it's not as related as I remember)
        - We think it makes sense to unify from the perspective of the GDExtension interface for use by extension bindings (even though it would likely duplicate systems within Godot, until we can remove the old system)
            - If it gets onerous, we could discuss removing compatibility for very old Godot versions
        - Proposed plan:
            - Could unify on MethodBind's
            - For Variant types these could be GDType's holding the list of MethodBinds
                - Depends on GDType advancing somewhat
            - Utility functions could be static methods on a utility class (similar to how godot-cpp exposes them)
                - Or, non-static methods on a singleton
                - We could do this now, if we want
        - Would benefit most new GDExtension bindings. Lower priority than other things that unlock new use cases
        - Should get a proposal, but need someone interested enough to write it :-)
    - Bromeon: raw char access for StringName?
        - Ivorius: It is now cheap to convert StringName to String, so you can do that and grab the pointer to the underlying buffer
            - PR that did it: https://github.com/godotengine/godot/pull/104985
            - Proposal: https://github.com/godotengine/godot-proposals/issues/11517
            - So long as the StringName is alive, then the buffer will be valid
        -

2025-11-11

- Attendees:
    - dsnopek
    - Quentin Quaadgras
    - Enetheru
    - Ivorius / Lukas
    - Logan Lang
- Topics for this meeting:
    - dsnopek: Use single godot-cpp branch for Godot 4.1+ and version it independently from Godot

        - https://github.com/godotengine/godot-proposals/issues/10243
    - dsnopek: GDExtension: Store source of gdextension_interface.h in JSON
        - https://github.com/godotengine/godot/pull/107845
        - The last open review note was to add a JSON schema, which I've done!
        - Draft follow-up PR:
            - https://github.com/godotengine/godot/pull/112125
            - Shows something I'd like to do with this functionality once it's available
    - Ivorius: Wild thought, can we make godot-cpp statically link with Godot?
    - dsnopek: Please review the RequiredParam<T> / RequiredResult<T> PR if you have time:
        - https://github.com/godotengine/godot/pull/86079

2025-10-28

- Attendees:
    - dsnopek
    - Patrick (FlameLizard)
    - dog_molecule
    - Jan (Bromeon)
    - Invorius / Lukas
    -
- Topics for this meeting:
    - dsnopek: Add RequiredParam<T> and RequiredValue<T> to mark Object * arguments and return values as required
        - https://github.com/godotengine/godot/pull/86079
        - Rename "RequiredValue" to "RequiredResult" - David to update PR
        - Everyone else to review and approve
    - Bromeon: macOS reloads extension if only `.gdextension` file changes. This causes issues because the dylib (unchanged) stays in memory, and is loaded a 2nd time.
        - https://github.com/godot-rust/gdext/pull/1367#issuecomment-3408190057
        - We should tell users (and binding implementors) that they need to fully clean up global static variables when the extension is deinitialized
            - For users, this is only needed if they want to support reload
        - David had an old idea about detecting if we were able to unload the extension (and show a warning)
            - Need to dig up the technical details about how to detect this
        - Worth considering removing the changed check on the .gdextension file
            - If no one (or very few people) depend on this, it's not worth the trouble
            - Doesn't change very often, so when it does, folks can restart

                - Could add a message in the editor when there is a change to the .gdextension file, telling folks that they'll need to restart the editor for the change to take effect
            - Patrick: could have an affect on the "creating extensions from the editor" PR, but depends on how that's implemented
        - Ivorius: godot-cpp helper functions proposal
            - https://github.com/godotengine/godot-proposals/issues/13475
            - Proposal comes from the games industry
            - How best to handle QOL features that are unlikely to be integrated into Godot soon?
            - David will write a response saying this is cool but should be an external project (and maybe if it becomes popular we can recommend it in the docs or consider it again)
        -
-

2025-10-14

- Attendees:
    - dsnopek
    - Zorbathut
    - Quentin Quaadgras
    - Ivorius / Lukas
    - Gergely (kisg / Migeran)
    - dementive
    - Samuel (enetheru)
- Topics for this meeting:
    - dsnopek/zorbathut: Convert LibGodot over to instance IDs, for safety and ease of use in GDExtension libraries
        - https://github.com/godotengine/godot/pull/111553
        - Would mean potentially fewer changes to GDExtension bindings to support LibGodot
        - In particular, would help with using LibGodot with GodotSharp
        - David doesn't like the idea of making decisions for GDExtension based on the state of GDExtension bindings as they were before LibGodot existed or based on integrating with GodotSharp (which would probably be temporary, since C# is going to GDExtension eventually)
            - But otherwise he's fine with the actual change - it shouldn't really matter if those APIs return object IDs or pointers
        - Gergeley and Zorbathut prefer the PR (switching to instance IDs)
        - Consensus wasn't reached
    - Dementive: Are the differences in MethodBind in Godot-cpp from Godot intended or an oversight? I also found that ClassDB does not expose engine classes, only classes registered with Godot-cpp is that an oversight or a limitation?

        - Dementive is going to take a stab at sync'ing `method_bind.h` from Godot to godot-cpp
    - Dementive: I'd like to discuss some of the issues I had when converting my module over to godot-cpp.
        - https://github.com/godotengine/godot-cpp/issues/1866
        - Great list of differences! However, each would need to be handled on a case-by-case basis - needs a champion or initiative to work down the list and eliminate the API differences between Godot and godot-cpp
        - Some changes could be made to Godot to sync with godot-cpp's API in some cases (rather than only the other way around)
-

2025-09-30

- Attendees:
    - dsnopek
    - Aaron Franke
    - Ivorius / Lukas
    - Yarvin
    - George (vnen)
- Topics for this meeting:
    - dsnopek (from Yarvin): Issues related to initializing new instance in GDExtension/Godot modules
        - https://github.com/godotengine/godot/issues/111075
        - Proposal:
            - Have a version of memnew() for RefCounted classes that keeps a reference to the object, and returns that, so that there's at least a strong reference for the postinitialize notification
            - Add some kind of message when trying to make a Ref<T> to an object that hasn't finished being constructed
            - Document how bindings need to handle constructed RefCounted object (potentially add new functions to gdextension_interface.h if this changes semantics: maybe incrementing the refcount in advance, and now the binding owns that reference)
            - Check if there are any scenarios in which the user needs to know what their instance is being created for (editor purposes such as docs/placeholders or actual use by the user) – one can (and should) do cleanup in NOTIFICATION_PREDELETE. (Yarvin: I'll do that)
    - dsnopek: We implemented that PR to prevent compatibility breakage for one extension, and... it broke compatibility for a different extension
        - https://github.com/godotengine/godot/pull/108614#issuecomment-3335505455
        - Should we attempt to address? If so, how?

                - David & Lukas leaning towards addressing
                - Aaron leaning against addressing
                - Could be handled by adding special flag to prevent `ClassDB` instantiating classes created with older versions of GDExtension to get their property values
                    - David will make draft PR to do this, so we have it in case we want to go this way
                - If we do it, we should aim to get it into Godot 4.5.1
            -

-

2025-09-15

- Attendees:
    - dsnopek
    - dementive
    - Jan (Bromeon)
    - Ivorius / Lukas
    - cerberus1746
    - George (vnen)
    - fkeyz
- Topics for this meeting:
    - Ivorius: Missing any godot-cpp contribution guidelines?
        - Setup might be good to cover
        - How to use the test project
        - (Some missing redirects after the documentation move)
        - What parts should be in ClassDB bindings, what should be in gdextension_interface.h, and what could be "GDExtension only" bindings
            - Related: https://github.com/godotengine/godot-proposals/issues/12622
            -
        -
    - dsnopek: Godot 4.5 is out! What will we focus on for the 4.6 dev cycle?
        - From dsnopek:
            - RequiredPtr:
                - https://github.com/godotengine/godot/pull/86079
                - TODO: Mark "required" arguments in the API docs to expose this to users too
                    - may conflict with "required" already used for required virtual functions - could use "not null"
            - Storing source of gdextension_interface.h in JSON:
                - https://github.com/godotengine/godot/pull/107845

        - From Ivorius:
            - https://github.com/godotengine/godot-proposals/issues/12622
            - godot-cpp version switch:
                - https://github.com/godotengine/godot-proposals/issues/10243
                - Questions:
                    - How do we version? What version do we start on?
                        - Restart on 1.0? Or 2.0 because it's the new version?
                        - Or a really high version? 1000, 2000, etc?
                        - Use the year? godot-cpp 25?
                        - Maybe do an "objection window"?
                - Next steps:
                    - Figure out the remaining unanswered questions on the proposal and get consensus on them
                    - Put proposed version scheme and open "objection window"
            -
            -
    - Bromeon: RequiredPtr plans? :)
        - https://github.com/godotengine/godot/pull/86079
-

2025-09-02

- Attendees:
    - dsnopek
    - Patrick (FlameLizard)
    - George (vnen)
    - Jan (Bromeon)
    -
- Topics discussed at this meeting:
    - dsnopek: godot-cpp cherry-picks (please review if you have some time!):
        - https://github.com/godotengine/godot-cpp/pull/1836
    - dsnopek: Fix WindowUtils::copy_and_rename_pdb regression
        - https://github.com/godotengine/godot/pull/110033
        - vnen will take a look if time
    - Patrick: Rider support for godot-cpp-template: https://github.com/godotengine/godot-cpp-template/pull/95
    - Bromeon: Array::get_typed_script questions
-

2025-08-18

- Attendees:
    - dsnopek
    - Quentin Quaadgras
    - Yarvin
    - Jan (Bromeon)
    - Ivorius / Lukas
- Topics discussed at this meeting:
    - Bromeon: access to base class during inititalization/destruction
        - GDScript predelete: https://github.com/godotengine/godot/issues/80834
        - GDScript postinit: https://github.com/godotengine/godot/issues/108395
        - GDExtension destructor: https://github.com/godotengine/godot/issues/109762
        - PR preventing RefCounted from being freed during notification PREDELETE: https://github.com/godotengine/godot/pull/101269 (Yarvin: I'll check if it solves the issues with destructor on GDExtension side as well)
        - Notes:
            - Stuff related to predelete is rather messy and might require bigger analysis/deeper dive (we need to know what is specific to GDExtension only and what is the problem in core as well). At least some problems might be related to core itself and might require input/work from the core team.
            - **Next:**
                - Make a tracker issue and review related issues
                - Investigate which issues also happen in GDScript or from a Godot module (maybe using the Godot tests)
    - Quentin Quaadgras
        - feasibility of GDExtension-implemented collection types (ie. PackedByteArray)
            - It is not possible/feasible according to Ivorius (would require a lot of tradeoffs because of memory allocation sheninganas).
            - **Workaround:** Creating a PackedByteArray, resizing it, and then writing the data directly to it
        - ownership semantics for non-RefCounted objects.
            - When passing an Object* into function, do I have to be careful not to free it?
            - When I receive an Object*, can I free it / do I need to free it?
            - Does the method callback into a GDExtension?
            - **Next steps:** Quentin to make an issue sharing a list of function with unclear ownership of Objects

2025-08-05

- Attendees:
    - dsnopek
    - Zylann
    - Jan (Bromeon)
    - George (vnen)
    - Ivorius / Lukas
    -
- Topics discussed at this meeting:
    - Bromeon: thread-safety of builtin types
        - Original Questions:
            - String, StringName, NodePath
            - Variant (if it holds a thread-safe type)
            - Callable (if the held function is thread-safe)
            - Not documented in Thread-safe APIs
        - String:
            - Each individual instance is thread-safe, so long as you don't try to use the same instance from multiple threads, it should be fine
            - Writing to it makes a copy (CoW)
        - StringName: Should be the same
        - NodePath: uses reference counting, but doesn't appear to be CoW - it is mutable, and so not thread safe
        - Variant:
            - assigning one Variant to another is not thread-safe
            - user code needs to do synchronization
            - reading from it on multiple threads is OK
        - Array and Dictionary:
            - mostly not thread-safe (except for reading)
            - really ugly issue with it not being safe to read from "read only" ones from multiple threads (two year-old bug)
        - Callable: should be thread-safe, they should be immutable after creation (baring some implementation issue)
        - TODO: Look at if arrays changed in a function will pass there changes up
    - dsnopek: Call startup callback only after reload is fully finished
        - https://github.com/godotengine/godot/pull/109309
    - Yarwin/Bromeon: Postpone adding new extension plugins to the editor
        - https://github.com/godotengine/godot/pull/109310

2025-07-22

- Attendees:
    - dsnopek

    - Zylann
    - Aaronfranke
    - Chris (CraterCrash Studios)
    - Ivorius / Lukas
    - Patrick (FlameLizard)
- Topics for this meeting:
    - dsnopek: No longer possible to create "unexposed classes" (aka "internal classes") via `ClassDB::instantiate()`
        - PR to preserve compatibility: https://github.com/godotengine/godot/pull/108614
        - Naros has a project that depends on this (this one, I think). In his case, it seems like it's fairly easy to work around, although, older binaries of his extension won't work right on Godot 4.5+
        - I suspect not many folks depend on this, but we could provide compatibility for this?
    - dsnopek: [godot-cpp] Using a release build of a GDExtension (godot-cpp) with a debug build of Godot will lead to heap corruption
        - https://github.com/godotengine/godot-cpp/issues/1820
        - Proposed addition to GDExtension interface to help fix:
            - https://github.com/godotengine/godot/pull/108725
        - **Note from the meeting:**
            - Aaron / Chris: It may be useful to give a warning when a debug build is loaded in a release build of Godot for performance reasons or differences in behavior with DEBUG_ENABLED
            - Patrick: Can we give a warning at export time?
                - We could check the .gdextension file and make sure that both "debug"/"editor" and "release" in there
                - We could also check if two entries point to the same library
            - We can work on this for Godot 4.6 🙂
    - Ivorius: Binding layer, is inserting subclasses into hierarchy possible?
        - Related to adding support for structs
        - Adding a new minimal super type (that is basically structs), and the question is if we can add the new class compatibly
        - Answer: yes, it is fine for compatibiltity
    - Zylann: User with two copies of the extension was having trouble, but didn't realize it because no errors were shown in the editor
        - Can we ensure that GDExtension errors during startup are shown in the editor?
        - When launching as editor, we could save any errors until later when we know that we can successfully show a popup?
        - Would be good for core, not just GDExtension
        -

2025-06-24

- Attendees:
    - dsnopek
    - aaronfranke
    - cerberus1746
    - Ivorius / Lukas
    - lodetrick
    - Logan Lang
    - Jan (Bromeon)
    - George (vnen)
    - cerberus1746
- Topics for this meeting:
    - dsnopek: [godot-cpp] Files generated by SCons should depend on the build_profile (if given)
        - https://github.com/godotengine/godot-cpp/pull/1795
        - There's a SCons question that I think is fine, but would be nice to double check if the more SCons-savvy are around
    - dsnopek: [godot-cpp] Double precision builds have been disabled on CI for now
        - https://github.com/godotengine/godot-cpp/pull/1799
    - dsnopek: [godot-cpp] Cherrypicks!
        - Possibly last round for Godot 4.3:
            - https://github.com/godotengine/godot-cpp/pull/1803
        - Very belated 2nd round for Godot 4.4:
            - https://github.com/godotengine/godot-cpp/pull/1805
    - Bromeon: breaking changes policy
        - https://github.com/godotengine/godot/pull/98566#issuecomment-2989930621
        - https://github.com/godotengine/godot/pull/84701
        - Policy idea: By default, all changes should have a compatibility method. But, on a case-by-case basis, if there is a strong justification, we may make an exception and allow not having a compatibility method.
    - dsnopek: Store source of gdextension_interface.h in JSON
        - https://github.com/godotengine/godot/pull/107845
-

2025-06-10

- Attendees:
    - dsnopek
    - Jan (Bromeon)
    - aaronfranke

    - George (vnen)
    - Theddeus Crews
- Topics for this meeting:
    - aaronfranke: Add header builders script for `env.GLSL_HEADER` and SVG icons
        - https://github.com/godotengine/godot-cpp/pull/1789
        - David will spend more time looking into it
    - Ivorius: variant_dir support (Add SCons variant_dir support)
        - Thaddeus will look at it
        - David will ping Fabio
    - Ivorius: What to do with limited platform extensions? Stubs? Compile error?
        - Where you have classes that only work on certain platforms, but you still want them to be available in the editor for code completion
        - There's lots of examples working around this in other extension and Godot itself
        - Let's talk about it again when Lukas is around
    - dsnopek: Add function to register main loop callbacks
        - https://github.com/godotengine/godot/pull/106030
        - Would be good to add an explanation for these callbacks into gdextenison_interface.h
            - David to make a PR
    - Bromeon: RequiredPtr – what's left for a 4.5 MVP?
        - https://github.com/godotengine/godot/pull/86079 (currently merge conflicts)
        - David will rebase
        - Thaddeus will talk to Remi and Clay about 4.5 inclusion
-

2025-05-27

- Attendees:
    - dsnopek
    - Patrick Exner (FlameLizard)
    - Ivorius / Lukas
    - George (vnen)
    - Jan (Bromeon)
    - Zylann
- Topics for this meeting:
    - Bromeon: `RequiredPtr` / `RequiredBind`
        - Latest PR by Repiteo: https://github.com/godotengine/godot/pull/106847
        - Larger PR by Repiteo: https://github.com/godotengine/godot/pull/106756
        - TODO:
            - Schedule a dedicated call that includes Jan, David and Thaddeus

                - David to review the C++20 PR: https://github.com/godotengine/godot/pull/100749
        - dsnopek: Alert the user if a GDExtension library file is missing
            - https://github.com/godotengine/godot-proposals/issues/7001
            - The folks at the meeting thought it was a good idea!
        - Bromeon: determine if a class is “final” (can be inherited from)
            - Multiple concepts in Godot like "abstract" and "virtual", and some can't be inherited from from GDExtension (the "abstract" ones)
            - Perhaps the concept of "abstract" is sufficient to mark classes as not extendable?
            - There's probably some classes (like "OS", "Time", etc) that should be marked as "abstract" that aren't!
        - Patrick: Moving blog post about using external libraries into Godot docs
            - Original blog: https://paddy-exe.github.io/posts/how-to-use-scons-with-cpp-package-managers/
        - Bromeon: Hot reloading
            - Addressing hot reload in game (not just in the editor)
                - Draft PR: https://github.com/godotengine/godot/pull/97991
            - Issues with hot reload not working in some (usually OS specific) cases:
                - https://github.com/godotengine/godot/issues/90108
                -
        -
-

2025-05-13

- Attendees:
    - dsnopek
    - Radiant
    - Ilikan
    - Hunt Sparra
    - Adam Scott
    - Valerie Bettaque
- Topics for this meeting:
    - dsnopek: GodotCon Boston
        - Lots of GDExtension presentations!
            - "Building a Godot Plugin with GDExtension" (workshop): https://talks.godotengine.org/godotcon-us-2025/talk/KMS9DA/
            - "Xogot": https://talks.godotengine.org/godotcon-us-2025/talk/KDZBDH/

                - "Embedding Godot: spicing up your app with SwiftGodotKit and more": https://talks.godotengine.org/godotcon-us-2025/talk/R9CSPE/
                - "LibGodot - Embed Godot Engine Everywhere": https://talks.godotengine.org/godotcon-us-2025/talk/XBJFYV/
                - "Adding new Script Languages to Godot": https://talks.godotengine.org/godotcon-us-2025/talk/FYE3UE/
                - "Introducing 3D Tiles For Godot": https://talks.godotengine.org/godotcon-us-2025/talk/JCDV3A/
                    - The Cesium plugin (for GIS data) is a GDExtension!
    - Closing old issues confirmation.
        - Saving for the next meeting when there are more people.
    - Documentation
        - Hunt: What are runtime classes?
        - David: The inverse of a tool script in GDExtension. A way to register it with ClassDB was added in either 4.3.
            - It's even in the new docs!
        - Hunt: Is the consensus to keep everything under GDExtension/godot-cpp in the existing docs?
        - David: Yes. We need to consider if new documentation should be under godot-cpp or be under the section for writing Engine code.
    - Hunt: https://github.com/godotengine/godot-cpp/issues/45
        - Hunt: This is a GDNative-era issue where some identifier names were *technically* illegal according to the C++ spec, but it is not a blocking issue in practice. In a prior meeting, we agreed that fixing this in GDNative is not worth the risk of breakage and that, since the main Godot repo has the same issue and we want to be as close as possible to that, if we were to fix this for godot-cpp, we would only do so for those methods/classes/etc. that are unique to godot-cpp.
        - Hunt: The only instance that I found was __swap_tmpl in defs.hpp and math.hpp. Do we want to change this for GDExtension?
        - David: It looks like Godot uses std:swap, so we can match that.
        - Hunt: I'll make a PR for it and then we can decide if we want to close the issue or not.
            - Actually already addressed by https://github.com/godotengine/godot-cpp/pull/1758

2025-04-29

- Attendees:
    - dsnopek
    - Patrick Exner (FlameLizard)
    - Aaronfranke
    - Ivorius / Lukas

    - Scott Doxey
    - Hunt Sparra
    - enetheru
    - Cerberus1746
- Topics for this meeting:
    - Ivorius: godot-cpp-template example class?
        - PR: https://github.com/godotengine/godot-cpp-template/pull/87
        - Related: Example class in aaronfranke's PR https://github.com/godotengine/godot/pull/90979
        - David: We had this discussion before when the template was first being created, although I don't remember who was for the example class.
        - Patrick: I think Remi was against it because it was too much boilerplate code.
        - David: Maybe we should ping him and see what his feelings are. Personally, I think I am fine with some example classes, and it would make my use cases easier.
        - Patrick: I guess, the question would be "how much code will we add"? Once we start, people will start asking why we showcase X but not Y. It could lead to a slippery slope.
        - aaronfranke: It is worth mentioning, I included an example class in my PR (https://github.com/godotengine/godot/pull/90979). I think, in particular, showcasing how to do notification ready or notification process are good to include because people will create _ready() and be confused why it does not work as they are expecting.
        - Ivorius: My thought process was to showcase the most functionality with the least amount of cose.
        - Ivorius: I can see the slippery slope argument, and I personally would not add anything more. That would be better covered by the docs examples. I think that most template creation tools give you a template so you can get started with going through the docs. It's frustrating to have a template and not be able to immediately test it.
        - Patrick: I get that, but the docs should still be the #1 place.
        - Ivorius: You have no idea how much time I spend referencing the example. I ended up there because there is nothing on Google if you look it up.
        - Ivorius: For the slippery slope, why don't we add a note to the PR that we will avoid it by not expanding on the example in the current PR.
        - aaronfranke: I still want to propose adding something about on-ready notification, but that is my only comment. In my PR, I think Calinou recommended adding that.
            - Calinou's comment on my PR: "The example extension code should probably print something in its constructor (or NOTIFICATION_READY), so you see it having an effect right away when it's added to the scene tree."

        - Ivorius: I think the difference between our PRs is that yours is for single-game GDExtensions and mine is for libraries.
        - aaronfranke: Actually, it's designed for all usecases.
        - David: I think it would be great to have a lot of examples people could look at. The reason I don't like people copying the test project is because it is very specialized for that.
        - Patrick: Let's post and see what Remi says. I don't think we are going to get anything just discussing it.
        - (Posted on PR that people are open to including a simple example and pinged Remi for feedback)
    - dsnopek: Add ScriptInstanceExtension class which mimicks the API of Godot's internal ScriptInstance class
        - PR: https://github.com/godotengine/godot-cpp/pull/1544
        - David: This would be for GDExtensions that add a scripting language.
        - David: The main hold up is testing.
        - Ivorius: I am kind of torn. I really like this idea because it makes it a lot easier for people to get into this. At the same time, I am wondering why you would want to use scripts and what the benefit is over regular ClassDB classes.
        - David: I think there is a benefit for pure scripting languages so you could, for example, link against the Lua interpreter and allow people to script that way.
        - David: I think this is very much needed since most GDExtensions that add a language implement this their own way. I originally copied from godot-luau-script and adapted it for the Gravity language.
        - Ivorius: The best I can suggest is to make a bunch of mocks.
        - David: That might be the best long term option since I won't need to make changes every time the script API changes.
        - Ivorius: I can see this encouraging more people to use godot-cpp. Basing things off of the C-binding is very difficult, from what I've heard.
        - David: Yes, the raw APIs have some difficult memory management where you have to persist data after it is passed to Godot.
    - dsnopek: Expose non-template versions of vararg methods
        - Issue: https://github.com/godotengine/godot-cpp/issues/1696
        - Looking for suggestions on how best to respond?
        - Ivorius: emit_signal copies the arguments since they may not be variants. Technically, we could be ignoring the copy for arguments that are already variants, but it would require changes. That should alleviate the overhead the author mentioned.
        - David: They could also write this code themself, and the only tricky part would be getting the hash from the extension_api.json.
        - Ivorius: My suggestion would be that most of the overhead comes from variant copying and the best way to address that is to make a PR to remove that or forking godot-cpp.

        - Hunt: If the APIs are in Godot, isn't it up to core?
        - David: These only exist in godot-cpp.
        - Ivorius: Then those would be useful to expose.
        - David: The problem is then modules would not have an equivalent API.
        - Ivorius: It makes sense to me for it to exist in both godot-cpp and Godot.
        - (It turns out godot-cpp does not an equivalent to callp but we have an emit_signalp)
        - David: Maybe we need these functions and just need to name them to be what they exist in core.
        - David: If all these exist in Godot, then we should make them exist in godot-cpp as well.
        - (David will look into the methods and reply)
    - aaronfranke: Document that memdelete() is the GDExtension C++ version of free()
        - PR: https://github.com/godotengine/godot/pull/103231
        - Adds a single sentence to Object's documentation mentioning that free() is "equivalent to the memdelete function in GDExtension C++ and in-engine C++"
        - Ivorius: This is maybe a weird suggestion, but what if we had a free() in object, but, instead of making a default, we disabled it an had a comment that was "use memdelete" instead.
        - David: There might be a way in docgen to document functions that do not actually exist (such as free()).
        - Ivorius: That would work for IDEs but not when the code is compiled.
        - Ivorius: I feel similar to ATS that it would be weird to include. The C++ has a lot of oddities that we don't want to start documenting. I guess it comes down to a question of why we don't have C++ docs.
        - aaronfranke: It is also applicable to GDExtension, which is why I brought it up in a meeting. I disagree with the goal that we don't want to document in-engine C++ works. I think that users modifying the engine is a perfectly valid use case. We should not make that more difficult than necessary.
        - David: It looks like it is approved and in the merge queue, with just a comment from Thaddeus.
    - aaronfranke: Make RST: Exclude godot-cpp test project files
        - PR: https://github.com/godotengine/godot/pull/103625
        - aaronfranke: The docbuilder fails because it finds the test project XMLs for both the GDExtension and engine module.
        - David: Seems good to me.
    - Enetheru: Closing old issues confirmation. I just wanted to wait for one more meeting.
        - https://chat.godotengine.org/channel/gdextension?msg=LtgzadSBRoZRj8drg

        - “We're cleaning up old issues, so the godot-cpp team is closing any issues that haven't been updated since 2022. If we closed an issue by mistake, just leave a comment, and we'll take another look at it.”
        - Enetheru: There was enough pushback that I stopped pushing it.
        - David: I’d like to do it.
        - Patrick: Maybe we should save this for the next meeting? That should work.
        - (It has been tabled)

2025-04-15

- Attendees:
    - dsnopek
    - Ivorius / Lukas
    - Thaddeus Crews
    - Hunt Sparra
    - George (vnen)
    - aaronfranke
    - Zylann
- Topics for this meeting:
    - Hunt: Identifiers with underscores (old issue)
        - https://github.com/godotengine/godot-cpp/issues/45
        - Both GDExtension and GDNative use identifiers that are technically reserved but have no functional impact. Do we want to fix this or take Godot’s stance of “we’ll fix it when it is an issue”?
        - Thaddeus: At the very least, I think we could gradually migrate by deprecating the old macros and introducing new ones that do not use reserved names. This would be similar to our warning wrappers.
        - Thaddeus: The API concerns are much less so because an underscore is valid so long as it is in a class namespace.
        - David: To maintain API compatibility, we will need to keep anything that is present in Godot.
        - David: For GDNative, not many people are working on it and I don’t think it is worth the risk of breakage to change it.
    - [3.x] Add generics to PackedScene instance
        - PR: https://github.com/godotengine/godot-cpp/pull/589
        - vnen: **I don’t think there is any harm in merging this** because it is already there, but, for future things, I think we can close them and not accept future PRs.
        - David: I know people are still using GDNative, and I don’t want to break it for them (with new PRs and few reviewers).
    - dsnopek: Register the classes used with the Godot editor
        - PR: https://github.com/godotengine/godot-cpp/pull/1743

        - This is adding the GDExtension parts to godot-cpp that were added in the Godot PR (https://github.com/godotengine/godot/pull/104129). How it works is by registering a callback that appends all the classes godot-cpp is aware of.
    - Ivorius: godot-cpp-template example class?
        - PR: https://github.com/godotengine/godot-cpp-template/pull/87
        - Ivorius: Wanted to reopen the discussion, but we should probably wait for Patrick.
        - David: Agreed. As an initial thought, I wouldn't mind having an example class in the template. It would make it quicker to create test projects, although that is not necessarily the most common use case.
    - Global Editor Plugins
        - David: I looked at some of the existing PRs and issues. We are really close, but the work stalled at some point.
        - Vnen: C# is moving the GDExtension. It seems that if both C# and GDScript are going to use this, then it makes sense for there to be a generic interface for any language to use.
        - Aaronfranke: The code in the PR is pluggable, but I did not expose it because I did not want to solidify the API.
        - David: That could be a way forward.
        - aaronfranke: I can rework the PR to do that, but I would prefer that we have a more solid plan before I do the additional work.
        - David: I think godot PR 87586 is the latest related PR. If this is added, then we could use it to walk an editor directory looking for GDExtensions (with some added safeguards).
        - Related GIP: https://github.com/godotengine/godot-proposals/issues/9806
        - vnen: GDExtension can do almost everything that a script can, which is why we are considering moving C# and GDScript to GDExtension.
        - Thaddeus: That would make things easier, but it would require more upkeep for GDExtension (for better or worse).
        - Zylann: For C#, one thing I am unsure of is how C# can use classes exposed by GDExtension. Would this go as far as creating C# classes from the C++?
        - David: I worked on that a year ago. I think we can have classes created in one language binding extend classes for another.
        - David: There will be overhead in that we have to through Godot to talk to the other extensions, but we're already sort of doing that. The only special cases that I can think of are virtual methods and the tooling for api.json.
        - David: Right now, you can have a project using the extensions and they'll be included when dumping the JSON. However, that doesn't work if you have extensions depending on extensions.
        - David: I have not tried it myself, but some other people have told me they have been able to use the JSON to integrate with other extensions.

        - Ivorius: Is there a roadmap? There is a lot of interest in the community and a willingness to work on it.
        - David: The origin discussion is https://github.com/godotengine/godot-proposals/issues/831, but there was no consensus. We discussed it at one of the Godot springs, people agreed the best way to do it was to have an editor file system, and Juan made a PR (https://github.com/godotengine/godot-proposals/issues/6307) for it.
        - David: The biggest unexplored area is how these editor plugins would be managed in the UI.
        - Ivorius: So the closest thing we have to what is actionable is the PR?
        - David: Right, that is the next step. And we are most of the way there with Adam's PR (https://github.com/godotengine/godot/pull/87586).
    - Nullability
        - Thaddeus: I wanted to confirm that this is still worth working on.
        - David: It is. The roadblock is finishing up a PR.
    - Warnings in GDExtension
        - Zylann: There is no built-in helper for that.
        - David: It depends on if we want to try and eliminate all warnings in godot-cpp itself. We would need to run the CI with warnings treated as errors to make sure we don't introduce new warnings.
        - Zylann: The only issue I ran into was with abs().
        - David: So we might be closer to this.

2025-04-01

- Attendees:
    - dsnopek
    - Samuel (Enetheru)
    - Henry (unvermuthet)
    - Ivorius / Lukas
    - Hunt Sparra
    - aaronfranke
- Topics for this meeting:
    - Henry: godot-cpp-template empty build profile (https://github.com/godotengine/godot-cpp-template/pull/82)
        - Adds an empty build profile and passes it to Scons by default. This is a hint that people can speed up their build times with a build profile (there is also a new comment in Scons to this effect).
        - Henry: This should also be in the docs, but this is an alternative way to inform people.
        - David: This is great. The only thought I had was that the build profile does not have comments… but you can't do that in JSON.

        - Patrick: A lot of people are going to look at the Scons file anyways. Some people may be suspicious of why the JSON file is there, but, even if they delete it and look at the Scons file later, they're going to find it and create/reimport it.
        - Henry: I think it may break if you delete the file.
        - Lukas: This is a dumb idea, but could we just add the comment as a key-value in profile?
        - David: We could, but we probably shouldn't.
        - David: I think the template should showcase enabled_classes instead of disabled_classed since most use cases will be using that.
        - Lukas: We could have several examples and just comment out the default in Scons.
        - Patrick: We could also just have it in a README. It'd be commented out in Scons and the README would explain how to write the build profile.
        - David: I like having it in the README, but I also like having the default build_profile as an example.
        - Hentry: If we don't load the file by default, then I prefer having it in the README. We can leave it in there, but, at that point, it can just be passed as a Scons argument.
        - David: So, how do we move forward? The goal is to make people aware of the feature and lower the amount of effort for them to do it.
        - Lukas: We want most people to use this, so we should comment it out and then people have to deal with it by uncommenting it or deleting it.
        - Henry: If you put the README information before Mac signing, I think people will read it.
        - Henry: I will comment out the scons and switch out disabled_classes for enabled_classes.
    - Henry: godot-cpp-template cross compile mingw CI (https://github.com/godotengine/godot-cpp-template/pull/81)
        - Henry: The initial problem was that the Windows job was taking a long time to install minGW but was not using it. This PR now uses it, but I am not sure it is worth it because of testing complications.
        - Henry: You can't build and run it in the same job since it is made on a separate OS (the Windows build is made on Linux).
        - David: I think it is fine for it to be another job.
        - Lukas: Was the reason for this just speed?
        - David: One reason was that we were installing it but using msvc instead. The other reason is speed.
        - Lukas: I'm pretty sure mingw does better with binaries without optimization than msvc (in general).
        - Henry: If we want to merge it, then I just need to check if the extra scons argument is necessary. Otherwise, from an implementation perspective, this is done.

- Henry: godot-cpp-template use caching actions from godot-cpp (https://github.com/godotengine/godot-cpp-template/pull/80)
  - This is about Github Action caches being immutable. With the old code, the cache would be created and would still be used, even if there were big changes to the code. With this change, we use the caching actions from godot-cpp that creates a unique cache per git commit and fuzzy-matches.
  - Lukas: I know that Repiteo(spelling) has been looking at it for the main repo. For the main repo, we have arrived at a pretty bespoke caching method. I don't know if that would be useful here. Although, that could be due to the complexity of all the branches the main repo and that may not be needed here.
  - David: I think it looks good.
  - Lukas: Can you explain the issue with a single cache step?
  - Henry: Instead of having a restore and a save after, you only have the composite action before and it automatically inserts a step. We could do it as one, but I did not do that here because godot-cpp does not do that either.
  - Lukas: Sounds good to me.
- dsnopek: TypedDictionary initializer support (https://github.com/godotengine/godot-cpp/pull/1750)
  - David: The PR includes a constructor that isn't in the Godot version: is this intentional or an oversight? PR author made a Godot PR in case this is an oversight: https://github.com/godotengine/godot/pull/104664
  - David: I wanted to ask Thaddeus if there was a reason for that, but I'll re-bring this up at the core meeting tomorrow.
- dsnopek: Stop referring to GDExtension as experimental
  - godot-cpp: https://github.com/godotengine/godot-cpp/pull/1755
  - godot-docs: https://github.com/godotengine/godot-docs/pull/10827
  - David: I have it marked as a draft because it conflicts with Lukas' docs changes. I can easily rebase mine once that is merged. Also, I will probably link to the new section on compatibility.
  - Patrick: I approved it.
  - Henry: Looks good to me.
  - Henry: Is there any update on the redirects issue that is blocking Lukas' changes.
  - Lukas: I asked again but did not receive any replies. I might ping one or two of the people that know more. It seems weird to be waiting for those redirects for months.
  - Patrick: You may want to request a review from bruvzg. Is he the main maintainer, David?
  - David: He has not been doing as much doc review lately, but he has the most knowledge.
- Enetheru: Closing old issues confirmation. I just wanted to wait for one more meeting.

        - https://chat.godotengine.org/channel/gdextension?msg=LtgzadSBRoZRj8drg
        - “We're cleaning up old issues, so the godot-cpp team is closing any issues that haven't been updated since 2022. If we closed an issue by mistake, just leave a comment, and we'll take another look at it.”
        - David: I don’t quite remember how it ended. I dimly remember Remi saying that we could but there could be trouble.
        - Patrick: He did mention that we were getting into 3.x issues that maybe should not be closed.
        - David: I know fire was very against us closing issues, but he didn’t say a ton on it.
        - Patricks: I think that Remi’s main argument is that it is comparable to the main repo, but I disagree. In my view, it is a very specific subsystem that does not have the same attention as the main repo.
        - David: The number of issues in godot-cpp is nice because you can look at the last 1-2 pages to see if anything was missed. Once it gets past a certain point, it becomes useless.
        - David: Remi reacted with a thumbs up and fire reacted with no good.
        - Hunt: Did fire give any specifics?
        - David: No, but he linked to some past discussions.
        - Patrick: There is a difference between closing issues and deleting issues. We’re not deleting them and the information is still there.
        - Lukas: I understand it. I’ve had several issues that I was still interested in but they got closed.
        - Patrick: Is it better to have no reply forever or have some final say?
        - Lukas: The bare minimum for closing issues could be we asked a question and they never replied (after several years).
        - Lukas: Why are we deleting these? So we can find more relevant issues?
        - David: That is my main reason.
        - David: We’ll ask in RocketChat for objections. If there are no objections, we can move forward.
        - David: Alternatively, we could add them to the triage project.
        - Henry: That would allow us to add comments.
        - David: That would be less controversial, but my worry is that we would be going through a lot of trouble and work for something that isn’t a problem.
        - Hunt: Why don’t we just send out the RocketChat message and table triaging all the old issues until we know if there are issues?
    - Aaronfranke: Allow creating GDExtension plugins from inside the Godot editor (https://github.com/godotengine/godot/pull/90979)
        - Henry: where is the scons file coming from?
        - Patrick: I think a very basic one is being generated.
        - Lukas: The details of this are interesting, but I see the point that maybe we should merge it and figure out the details in followup PRs.

- Lukas: My main question is “what is the next step?” Can we just approve it and merge it?
- Henry: David, have we tested the framework signing?
- David: I forgot. Feel free to ping me on RocketChat.
- Lukas: The last comment is from Thaddeus that this may need core approval.
- David: I think Editor
- Patrick: That would be KoBeWi. We can ask Thaddeus to clarify if core is needed.
- David: We can also bring it up at the core meeting and ask if it needs approval.
- (aaronfranke joins)
- aaronfranke: I want to point out that even if this needs core approval, that does not stop GDExtension people from putting their own review on it. I know in the past that Ivorforce has requested changes.
- Lukas/ Ivorius: I can dismiss my review and wait for a follow-up PR.
- David: I think this pretty low risk because it is a template.
- aaronfranke: You can edit the max-min versions, if it is reloadable, and the entry symbol. But, besides that, it is very hands off.
- aaronfranke: I also want to point out that we just released 4.4 and we want to merge this early in 4.5’s lifecycle so we can get feedback.
- Henry: Does this handle framework signing and macOS?
- David: This does not use the godot-cpp template.
- aaronfranke: I’ve tested that, as-is, this is work out of the box for Windows, iOS, macOS, and web. I have tested that the file runs but nothing more advanced than that.
- Patrick: I don’t think Github Actions should be part of this PR.
- Henry: Does this handle the macOS signing?
- David: You would handle that outside of this.
- Hentry: But macOS signing was why Lukas asked for changes.
- Lukas: That is an open question, but we can *probably* figure it out in a follow up.
- David: Given that it is a template, we have a lot of leeway. We can completely rewrite it without breaking the already generated versions.
- Lukas: The plan is to bring it up during the core meeting.
- David: It might be worth pinging Remi as well since, if I remember correctly, he was the main one who had opposition.
- aaronfranke: In a previous GDExtension meeting, I argued that the current scope and feature set is good. We agreed that we did not want to use the template and that the module feature was advertised early on in GDExtension.

2025-03-18

- Attendees:
    - dsnopek
    - Samuel (Enetheru)
    - Hunt Sparra
    - Henry (unvermuthet)
    - Ivorius / Lukas
    - Patrick Exner (FlameLizard)
    - Jan (Bromeon)
    - Adam Scott (from GDC)
- Topics for this meeting:
    - Enetheru: Issue tracker cleanup #442
        - GDNative Support
            - GDNative is technically still supported, but few people support it in practice.
            - David: Instead of asking if it is *really* still supported, it is better that we come up with a proposal.
            - Patrick: There are some on Godot 3 that support GDNative, so that should be a discussion with them.
            - David: If lawnjelly wants to take up GDNative, that would be great, so we should reach out to him.
            - Ivorius: Is lawnjelly a dedicated maintainer?
            - David/Patrick: I don't know if he is officially the 3.x maintainer, but most of his dev time is focused on 3.x.
            - David/Jan: There was discussion on if 3.x would be moved to its own repo and lawnjelly made the maintainer, but it is unclear what the status/consensus was.
            - Patrick: Is GDExtension out of experimental?
            - David: That's a good question. We probably could take it out of experimental break since we haven't had a compatibility break since 4.1 and I'm fairly confident we won't need one.
            - (See "Should we take GDExtension out of experimental topic below)
        - Addressing older issues
            - David: For issues over X years old, maybe we could close them with a comment saying we can reopen if it is still an issue.
            - Patrick: 2 years seems like a good timeframe.
            - Thaddeus: I am completely in favor of this because we can point to it when asking the main repo to do it.
            - David: What about anything before 2022 (4.0 alpha 1 was in January 2022)?
            - (General agreement)
            - (Enetheru will propose a general message for closed issues)

- David: Should we take GDExtension out of experimental
  - Ivorius: Why do we call it experimental? Because there can be breaking changes?
  - Patrick: Yes. Since we've had 3 versions since the stability work, we don't expect there to be any more major breakages (or they were circumvented by the compatibility system).
  - David: There are a lot of things we still need to do, but I feel that we won't have to break compatibility for them. We could say 4.4 is stable (since it just came out).
  - General consensus to find all places in the docs and update it (specifics TBD; David says he will take a look at it)
  - Ivorius: It may be worth it to do a blog post for the people who have avoided GDExtension since it was called "experimental."
  - David: I think it would be cool to do an announcement. We should figure out something in addition to "it's not experimental anymore" in the announcement.
  - Jan: Hot-reloading could be worth mentioning.
  - Patrick: We can show off the stability and doc changes.
  - (The last blog post on GDExtension was the "Successor to GDNative" post in 2021, old enough to have "...its contents might be outdated and no longer accurate…" at the top)
  - Henry: A recap sounds good.
  - Patrick: We could also advertise the template project. Some people still don't know it exists, and we've done some really great work on it.
  - Ivorius: The docs change is blocked by a redirect limit.
  - David: A recap sounds good. That will take awhile, so we should probably start it soon.
- Enetheru: Github CI workflow minimisation for PR1726
  - Enetheru: I got all platforms working for the CI.
  - David: I vaguely recall us not doing too much stuff on the CI because it would take minutes away from Godot, although I don't remember if that actually is a thing or I imagined it. Do the jobs you added run simultaneously?
  - Enetheru: They run simultaneously.
  - Patrick: I also remember something about not taking CI resources away from Godot. We might have to ask Thaddeus.
  - David: If someone who knows GitHub actions better than me could look at the PR, that would be helpful.
  - Enetheru: The main difference from Godot's actions is the runner.
- dsnopek: Exposing direct manipulation of API Type to extensions (https://github.com/godotengine/godot/pull/104287)
  - The author of Jenova Framework wants to expose set_current_api and get_current_api.

        - David: I'm not in favor in this (in particular set_current_api) because those should have nothing to do with GDExtension. We should be getting it from the initialization level from the GDExtension itself. So, I don't want to do exactly what he proposes, but I am open to allowing developers specifying the API type of a class during registration. Either way, it would change the existing API. Should we break that?
        - Jan: I also have the feeling that there is code out there in the wild that depends on the API level matching the initialization level. If it changes, there would be another check added.
        - Jan: Is there strict need for the change? It will add complexity.
        - David: I'm envisioning it as another field to the class info registration struct that will be optional. Godot-cpp would not use it at all, but other bindings could use it. Jenova's use case is having a GDExtension (at the editor level) that registers other extension levels. Admittedly, I don't entirely understand their use case.
        - Jan: One thing that holds right now is that you can rely that all classes for a specific initialization level would be registered when that initialization level ends. That no longer holds if that changes.
        - David: I was actually thinking of making it more strict so you could not register classes for a finished initialization level, but that is not implemented and Jenova is currently using that "feature." We may have accidentally made that "registration after initialization finished" part of the API.
        - Jan: Also, the API level is not exposed to the API. Only the API type is.
        - Jan: Do we know why it is useful to them?
        - David: That's a good question. We should figure that out.
        - Ivorius: At the top of the PR, they do mention that they need to initialize nested classes when running in debug mode.
        - David: I'm not sure why that would be the case.
        - Jan: Maybe we're solving an XY problem.
        - Ivorius: My best guess is that he is running the game in debug and it is being registered as editor, which is skipped in the build that does not load editor parts, but I'm just spitballing.
        - (David will reach out for more information from Jenova)
    - dsnopek: Optimizing memory layout of virtual function data (https://github.com/godotengine/godot/pull/104264)
        - Every time we have a GDExtension virtual function, it adds a pointer and bool to the class, and that bool is taking up a lot of extra space due to alignment. They are proposing just having the pointer and having an invalid value to cover the third state (and replace the pointer).
        - David: They've chosen maxint as the invalid value, which seems fine to me.
        - David: This seems good to me, but I want to run and verify their test project.

        - Jan: Is it defined behavior to compare pointers of different alignments?
        - David: I think in practice, no compiler makes a distinction between them.
        - Ivorius: I think it would be an issue if you were to dereference one, but I would be surprised if there is any issue when just comparing them.
        - David: I might bring this up at the next core meeting to double-check.
        - Jan: As I wrote, there is precedent for this in nmap(2) and in the Windows SDK. That doesn't mean it applies to all platforms though.
        - David: nmap(2) and the Windows SDK are data pointers and not function pointers though.
        - Ivorius: We could just make an empty function that fails if called and use that. That would also protect us from anyone calling it.
        - David: That reminds me of an issue of where people call the GDExtension function pointers before they are set, and someone suggested the same thing.
        - Jan: We had a similar thing in Rust, and using dummy functions were suggested. It looks promising, but we haven't implemented it yet. Although, there is overhead for that.
        - Ivorius: Would this break compatibility?
        - David: This shouldn't.
        - (David will bring up the PR in the core meeting to check. If there are any objections, then we can explore creating an invalid function.)
    - Henry: Implement use_static_cpp flag on Linux (https://github.com/godotengine/godot-cpp/pull/1747)
        - Henry: The use_llvm flag is defined twice (once on Linux and one on Windows), but that should be fine. Although, if someone knows a way to add it once and set the help text based on platform. It currently shows up twice when printing the help text.
        - David: The Godot build system does that, so it's probably fine.
        - Henry: I'll just update the help text to specify it is platform-specific.
        - Ivorius: It doesn't show for me (on Mac) in Godot, so it may be working in Godot.
        - Henry: When I run it on Linux, it shows up twice. I'll look again at Godot to see if there is something they do conditionally.
        - David: I think both will be loaded on Godot if it detects you are on Linux and have the cross-compiler stuff.
        - Henry: Someone else should review the CMake stuff.
        - Samuel: I can review that. Your stuff can go in global since it doesn't need to just be in Linux.
        - David: My preference for scons is to keep the files shaped like the Godot scons files because we occasionally just copy parts of the Godot files. If we think it can be organized better, we can also change it in Godot.
        - (Discussion to be continued after more research on how Godot does it)
    - Patrick: .uid files for godot-cpp-template: Yes or No? (https://github.com/godotengine/godot-cpp-template/pull/79)

        - David: My opinion is that we shouldn't have the .uid files since two GDExtensions could ship with the same UIDs.
        - Patrick: How are they generated?
        - David: I think they are generated randomly.
        - Henry: They should not be added since they will not be generated. However, if we add it to the .gitignore, then people would not be committing them in their projects, so we may just have to check commits to godot-cpp to make sure future PRs don't add it.
        - (The .uid file will not be added)

2025-03-04

- Attendees:
    - dsnopek
    - Thaddeus Crews
    - Henry (unvermuthet)
    - Ivorius / Lukas
    - Enetheru
    - Patrick Exner (FlameLizard)
    - Jan (Bromeon)
- Topics for this meeting:
    - [Enetheru] - Idea for documentation.
        - Issues with copy-pasting docs into main docs and indexing?
        - Enetheru: to do some more research on how the Godot docs work
        - Patrick:
            - There could be possible confusion from users about where to find documentation
            - All docs should be in the official documentation
            - If we divide docs it could become more chaotic
        - David: let's move the cmake.rst into the docs and remove from godot-cpp
        - Ivorius:
            - If PR https://github.com/godotengine/godot/pull/90979 is merged, then you can make a godot-cpp project in Godot, so the godot-cpp docs make sense in the main Godot docs
            - Doc splitting PR from a few meetings ago is progressing: https://github.com/godotengine/godot-docs/pull/10631
    - [Enetheru] - Backporting CMake what is needed? Should it be done?
        - David: Leave in 4.4+ for now, and backport after we shake out any issues, and see what folks you use it say
        - Thaddeus: Backporting the cmake config based on current scons, would cause problems in a 4.3 context
    - Ivorius: Branches on godot-cpp-template?

        - Lukas: Do we want to update to godot-cpp 4.4? We also want folks to use the oldest compatible godot-cpp version for compatibility with more Godot versions
        - Patrick: The project is very minimal, and you could switch back to a much older godot-cpp with minimal problems. We have a Godot 4.0 branch, but it's not maintained anymore
        - Lukas: People will just start from the current template, and not think about downgrading for better compatibility
        - Lukas: Maybe just put a paragraph explaining the issue and how to downgrade
    - Bromeon: resume RequiredPtr for nullability information in GDExtension API?
        - Bromeon: Marking arguments and return values as nullable or not would be very helpful for bindings (especially Rust, C#, Swift)
        - Bromeon: starting with RequiredPtr as opposed to OptionalPtr would allow for incremental migration of the API surface (at the moment, everything is optional) – would allow for proof-of-concept, with limited time investment and risk
        - David, Thaddeus: needs agreement in core team about RequiredPtr, error macros, code style. Good that we already have an immediate use case from the start (GDExtension bindings)
        - David: behind the scenes, RequiredPtr and null checks should generate same binary code
        - David: TODO: Finish adding support for Ref<T> to the RequiredPtr PR, and then bring it to the core team
    - dsnopek: How to continue on the "dev" suffix removal
        - David: Print warning when using .dev build instead of removing it
    - unvermutet: godot-cpp-template CI performance
        - unvermutet: Caches on CI do not seem to update themselves
            - Will make a PR to use proper cache actions from godot-cpp
        - unvermutet: CI sets up mingw on Windows and then doesn't use it (waste of time)
            - There was debate on whether to keep compiling with MSVC or switch to MinGW
                - As it is a template, the expectation is that is makes the correct decission for binary performance because people distribute the CI builds as releases
                - MinGW can produce faster LTO builds
                - Consider cross-compiling Windows builds?
            - Will make a PR adding `use_mingw=yes` to scons call
    - Ivorius: Quick update on frameworks
        - David to help test iOS build with Apple ID
    -

2025-02-18

- Attendees:
  - dsnopek
  - Patrick Exner (FlameLizard)
  - unvermuthet (Henry)
  - enetheru (Samuel)
  - green crow dev
  - Ivorius / Lukas
  - Hunt Sparra
  - Jan (Bromeon)
  - aaronfranke
- Topics for this meeting:
  - unvermuthet: godot-cpp-template MacOS / iOS binary and framework naming (https://github.com/godotengine/godot-cpp-template/issues/76)
    - Currently, the name only has the target and not env["suffix"], which results in two files with the same name that are then either merged or one is ignored.
    - David: Is this an issue with godot-cpp or the template?
    - Lukas: This is an issue with the godot-cpp template.
    - Lukas: We might not want to use frameworks at all and just use dylibs, but, for framework, just including the suffix is a no-brainer.
    - Henry: That would be my suggestion too. Another solution would be to disable double for macOS, but I think the oversight is the file name itself.
    - Henry: If you want multiple targets (e.g., single and double), you would want a different .framework folder for every target, which explodes the complexity.
    - David: This does seem like the bug.
    - Henry: To move forward with fixing it, I need to know if it is intended for there to be a one framework folder or one for each target.
    - Lukas: That is my blocker too. I have been talking bruvzg; he mentioned that Godot exports combine the dylibs into a framework and signs them, which would mean that we don't need to deal with frameworks at all. However, I have not been able to verify this.
    - Henry: If we could find the spot where the dylibs are bundled (export_plugin.cpp:1374), signing is handled, and the plist is generated, using dylibs seems like the right solution.
    - Patrick: That would simplify a lot.
    - Related PR: https://github.com/godotengine/godot-cpp/pull/1679
    - Henry: When the dylib file is inside the framework folder, is it possible to just set the library path to the framework folder. I always need to specify the library file itself.
    - Lukas: It does work with the framework folder, but there is a Godot issue with linking to the folder and/or using relative paths.

        - David: What is the next step on this?
        - Henry: For the specific issue I brought up, the fix would be to put the whole suffix in the file name.
        - Lukas: Yup. For the framework (to dylib) thing, that would take a little longer to verify everything works as expected and feedback from Fabio and bruvzg.
    - Ivorius: Remove `.dev` suffix? (https://github.com/godotengine/godot-cpp/pull/1705)
        - This PR removes the .dev suffix from at least the final build (objects TBD). In the .gdextension file, you can only specify the path with or without the .dev suffix (not both), which means Godot will fail to load the not-chosen format.
        - David: I'm for removing .dev. It doesn't bother me and users bring it up (via support requests) on a regular basis.
        - David: There was a concern about accidentally shipping dev builds, but adding a warning when a dev build is loaded would address this.
        - (Everyone who spoke is in favor of the change)
        - David: If we can get this in 4.4, then any breakage from this can come at the same time as any other 4.4 breakage (as opposed to developers having the 4.4 breakage and then breakage for this in godot-cpp a few weeks later).
    - unvermuthet: godot-cpp-template builds workflow matrix notation (https://github.com/godotengine/godot-cpp-template/pull/75)
        - Currently, we have a code block for every possible build combination. This PR improves readability and maintainability by using matrix notation instead.
        - Henry: I updated the Ubuntu version to get rid of a deprecation warning.
        - Patrick: Can we just put latest?
        - Henry: I'm not an expect, but I used a specific version in case the latest does not have long-term support.
        - David: There is a good reason to stay on an older version. If you build with a newer version and attempt to load it on an older verion of Ubuntu, it may fail because it requires a newer version of the C library.
        - Aaronfranke: We should use the latest version for maximum compatibility.
        - David: use_static_cpp=yes would give us the most support.
        - Lukas: I think it is fine to update in this PR because the runner will cease to exist soon.
        - (The godot build containers use Fedora 41, which is pretty new)
        - Henry: For this PR, why don't we change this to Ubuntu 22?
        - (People agreed to change it to 22 and punt on the long-term issue of Ubuntu versions)
    - dsnopek: Should we support loading from system directories on Linux/BSD? (https://github.com/godotengine/godot/pull/102153)
        - The PR would allow Godot projects to load the .so from system paths.

        - aaronfranke: What is the use case of this? Especially since compatibility isn't guaranteed for long periods of time.
        - David: I don't think we should support this at all, but we (unintentionally) partially support this now.
        - Lukas: This seems unsafe version-wise.
        - David: Do we want to remove the unintentional support for this? Or should we keep it for compatibility and put a comment in OS_Unit::open_dynamic_library(...) around when GDExtensions are loaded?
    - aaronfranke: Swap the names of the engine's internal "abstract" and "virtual" (https://github.com/godotengine/godot-proposals/issues/11791)
        - David: Your logic makes sense, but I don't think we can swap our terminology without it being Godot 5.
        - aaronfrake: Yeah, it would be Godot 5 (it has the milestone), but this is there for discussion.
        - Patrick: Shouldn't this have the label "compatability-breakage"?
        - Jan: This is the responsible PR here: https://github.com/godotengine/godot/pull/58972
        - aaronfranke: I disagree with the logic. It is an abstract class in C++, but users generally expect to be able to extend an abstract class.
        - Jan: That's my experience.

2025-02-04

- Attendees:
    - dsnopek
    - Radiant
    - Hunt Sparra
    - Patrick Exner (FlameLizard)
    - Radiant
    - Ivorius / Lukas
    - Enetheru
- Topics for this meeting:
    - Add interface functions for Object::set_script_instance() (https://github.com/godotengine/godot/pull/102373)
        - (This is for Godot 4.5 since we are in feature freeze for 4.4)
        - David: This used to be more complicated, but it was recently simplified
        - (Approved during discussion, funnily enough)
    - Relative paths for godot-cpp-template gdextension libraries? https://github.com/godotengine/godot-cpp-template/pull/73
        - The original proposal that spawned this was based on a misunderstanding, but this PR still adds good functionality.

        - Related issue #1681
        - Ovprois: The proposal is to use relative paths in .gdextension file in the godot-cpp template. This doesn't change anything, but this will make the template more future-proof by allow it to be used outside of res://bin/
        - David: This makes sense since there is some disagreement between GDExtension users since some like to put it in res://addons, etc
        - David: Is there any argument against relative paths?
        - Patrick: I originally thought absolute paths would make more sense, but I now agree with Ivorius' points.
    - Shortage of godot-cpp GDExtension docs - Chat Discussion continuation https://chat.godotengine.org/channel/gdextension?msg=Q8ng7gvDMHPyTRSqW
        - Ivorius: There are people who are trying to get started with godot-cpp, but aren't finding the resources they need for that. I want to help with that, but I don't know where to put it
        - David: We were discussing this about a half-year ago. Since so many of the APIs are the same, we got caught up in circles on if GDExtensions docs should be standard Godot docs or godot-cpp docs.
        - Ivorius: Now that I've been working with GDExtension more, I am understanding that more. We should document that.
        - Patrick: I definitely think there could be a page for useful information on GDExtension development.
        - Patrick: My concern with having godot-cpp/GDExtension docs be entirely separate would be unintuitive. You would have to find the link in the docs website and then click on the link and then another link for another binding. I'm not really sure if that should be separate.
        - Hunt: What do we want to have documented?
        - Ivorius: I view godot-cpp as one instance of a GDExtension. The GDExtension system is something that most people will not interact with. The vast majority of people will use one of the existing language bindings such as godot-cpp or godot-rust and write code in that. I would expect (as a godot-cpp user) to have a page that explains the binding to me.
        - David: Right now we have a mix of godot-cpp examples and some language-agnostic information. We may want to separate those.
        - Patrick: One problem with the documentation is that there is a lot of duplicate information (and there could easily be more) and you have to crosslink a lot.
        - Patrick: Could we add another subheader to the GDExtension docs for godot-cpp specifically.
        - Enetheru: I have an idea on how to structure the docs. If the documentation on the godot website was purely on how GDExtension links into the engine and then it links to the binding-specific information. The bindings would be responsible for their setup and usage details. This would also cleanup the main Godot docs.
        - Ivorius: That's how I see it.

        - Patrick: The main point against that is that the godot-cpp bindings are official, so people expect them to be there.
        - Enetheru: Why don't we have another godot-cpp subheading then? That could house all the setup and examples.
        - David: My vote would be for keeping it on the main doc sites so we can link overlapping examples (e.g., creating a Node).
        - David: **Proposed documentation layout:**
            - GDExtension
                - What is GDExtension
                - godot-cpp (or whatever we call the heading)
                    - C++ Example
                    - Documentation system
                    - ... ADD MORE STUFF HERE ...
                - .gdextension file
                - Internals / implementors / Making a new language binding (or something)
                    - C Example
        - Enethru, Ivorius, & Patrick: (Agreement)
        - (Discussion on how rearranging within the GDExtension section is easy; rearranging anything outside that is difficult)
        - (Ivorius will work on the docs PR)
        - Patrick: FYI, I'm working on the "Integration with Third Parties" documentation. I'm largely basing it on scons and leaving the CMake documentation to the CMake users/maintainers.
        - Enetheru: I can help with CMake.

2025-01-21

- Attendees:
    - dsnopek
    - fales
    - Thaddeus Crews
    - Hunt Sparra
- Recent news:
    - Godot 4.4-beta1 release, which means 4.4 is in feature freeze
        - We got the last big feature in in time, which was the compatibility system for virtual methods:
            - https://github.com/godotengine/godot/pull/100674
    -
- Topics for this meeting:
    - David will send out a message on RocketChat to poll about a new meeting time.

- Any important bugs to fix before Godot 4.4?
  - David: The only bug that stood out was one on compile time regressions, but a great PR already existed and it is merged now.
  - (No other major bugs were brought up by the group)
- (4.5 goal) GDExtensions depending on other GDExtensions
  - David: I was hoping to work on this for Godot 4.4, but didn't find time
  - David: We'll need this for C# to become GDExtension.
  - David: Raul was working on it, but I'm not sure of the current status.
- (4.5 goal) Thaddeus: I would like to have a way to handle nullable/non-nullable attributes.
  - Thaddeus: A lot of scripting languages (such as C#) have a nullable attribute.
  - David: The Godot Swift and Rust folks have wanted this for awhile. Bromeon and I have been working on it for awhile in PR#86079
    - https://github.com/godotengine/godot/pull/86079
  - Thaddeus: I'd additionally like this for value/scalar types. I don't expect this to be fully fixed by godot-cpp, but I would expect any PR adding nullability to apply to both objects and scalars.
  - Thaddeus: This could be helpful for when the typing system is eventually™ overhauled.
    - https://github.com/godotengine/godot-proposals/issues/162
  - fales: GDScript can implement it however it wants, but the internal binding system shouldn't change (since it would break compatibility everywhere). So, that's probably not happening until at least Godot 5.
  - (Discussion about languages such as Java moving away from nullability or nullable-by-default)
  - David: Backwards compatibility could be added if nullable scalars were different from regular scalars. That way the function hashes are different. Although I am not sure if this is a good idea overall.
  - fales: The only way I see this as doable is to stop using scalar and always pass by pointer.
  - (Thaddeus's PR: nullable https://github.com/godotengine/godot/pull/90767)
  - Thaddeus: How much has the RequiredPtr PR changed?
  - David: Without adopting a different naming scheme, there will be a big delta when switching from Node* to RequiredPtr in code.
  - David: The other change since we last discussed is that the API encourages godot-cpp developers to use RequiredPtr when needed.
  - Thaddeus: I'm not certain about the wrapper. Does the if-else-void method function like we expect it to, or will the error macros not work out?
  - David: I think we still need the error handling in the macro (instead of in RequiredPtr). When optimizing, I expect the compiler to pretend that RequiredPtr doesn't exist. …but we should probably test that.

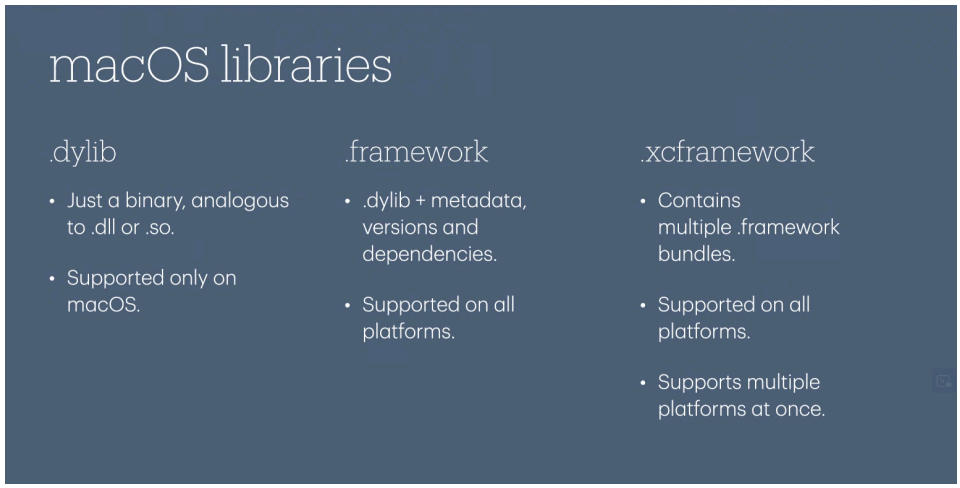
        - Thaddeus: I'm not too worried about it, but it's a thought that occurred to me. Also, I think we can make all the RequiredPtr methods const expr since it is dealing with a pointer type at all times.
        - David: If we can do that, let's do that.
        - fales: The PR looks nice, but how often are object parameters optional in their bound code. Does it make sense to have RequiredPtr the default?
        - David: When we were discussing this originally, the problem with OptionalPtr is that it would require more changes. We should check though.
        - Hunt: What about existing extensions? Isn't the current behavior that objects are optional?
        - David: This doesn't change the method hash, so it shouldn't affect binary compatibility. So, only source compatibility will be affected.
        - David: I think the next step is investigating if required or optional should be a default.
        - fales: I can only think of a few cases where a parameter being null would be okay. Off the top of my head, most of the APIs I can think of require the object to not be null.
        - Thaddeus: I like the new concept and direction, but I need to give it a test drive.

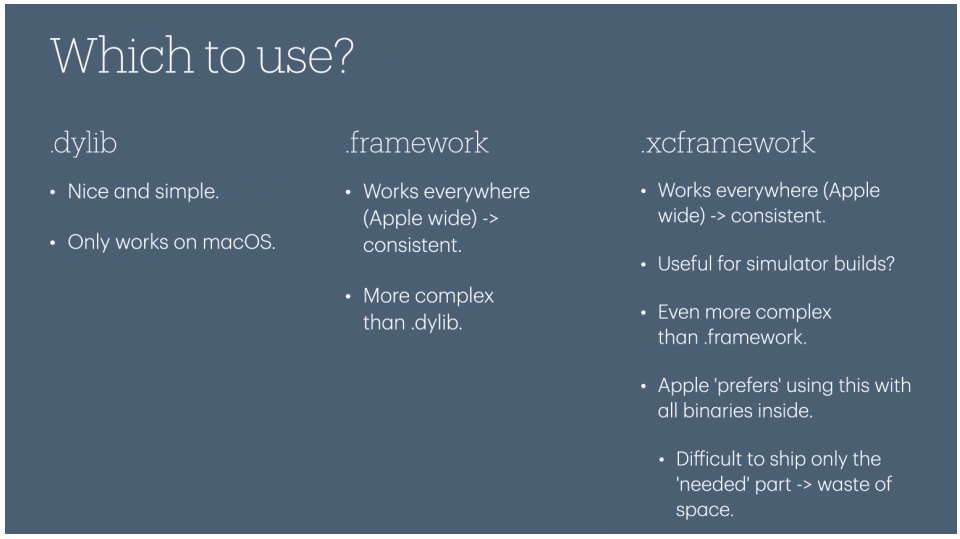
2025-01-07

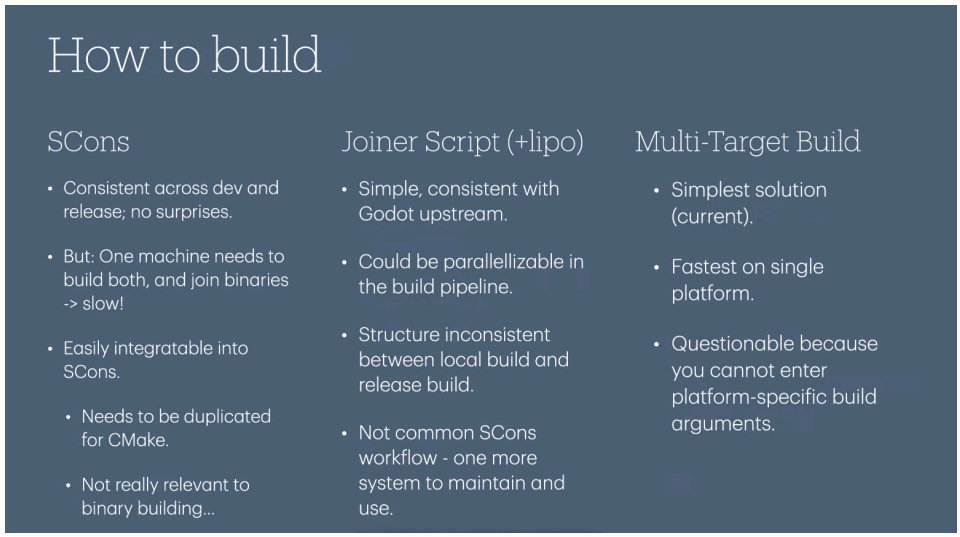
- Attendees:
    - dsnopek
    - Ivorius
    - Bromeon
    - Radiant
    - Julian
    - Logan
    - Hunt
    - Fabio
- Recent News
    - PR adding class icons (but not through .gdextension file)
- Topics for this meeting:
    - System for virtual method compatibility:
        - https://github.com/godotengine/godot/pull/100674
        - David: works with new macros that allow you to create compatibility methods with an alias. After doing this, you can try calling the latest version and fallback to the compatibility versions if it fails.
        - David: There is one point of ugliness still on the PR, but, aside from that, I would like to get this merged sooner rather than later.

        - Bromeon: It looks good; I can test again with Rust (referring to the “point of ugliness”).
        - David: If you could do one more smoke test, that would be awesome. Nothing on the interface has changed since it was last tested.
        - David: If there is a system that isn’t aware of the virtual compatibility setting (e.g., prior GDExtension versions), it will just return the virtual method it has. (https://github.com/godotengine/godot/pull/100674#issuecomment-2571654755)
        - Bromeon: Do you see any other potentially planned features that would be made harder by this such as required parameters?
        - David: That’s a good question. If a parameter is marked as required, should that change the hash used to store the compat method?
        - Bromeon: If an old method is passing null to an argument that is now required, that should cause a runtime error but not undefined behavior.
        - David: I don’t think anything about this makes required parameters inherently required, but we will still have to decide if that should change the hash or not. Once we start marking things as required, that could cause a lot of hash changes.
        - Bromeon: Worse case scenario, we would just need a mapping type like we had in older versions.
    - godot-cpp: Build profiles filtering out methods as well
        - https://github.com/godotengine/godot-cpp/pull/1680
        - Fabio: Some time ago, we added the option to specify which classes you wanted included in a build. It is a separate tool that, given a build profile and an api.json, generates a new api.json with only the classes and methods you want. This can be used to create more optimized builds of bindings.
        - Fabio: One catch with this is that there are a few classes that are forced to be included for godot-cpp.
        - Fabio: If someone else could see if this is useful for non-cpp bindings, that would be great.
        - Radiant: I could try generating the bindings.
        - Fabio: I think one of the advantages of this for external bindings could be faster compilations. Compiling and linking together allows for greater optimizations, but it can be very slow. By compiling with unused dependencies stripped and then linking, compile time can be greatly sped up.
        - Bromeon: Currently, for godot-rust, you need a binary to generate the extension_api for. In the future, I plan to introduce the ability to compile based on an extension_api.json.
        - Bromeon: To verify, this PR introduces a way to strip functionality you don’t need from Godot?
        - Fabio: It specifically strips the api.json.

        - David: The interface is to specify what classes to include and exclude.
        - Bromeon: Is this pulled from godot-cpp or the engine repo?
        - Fabio: It's a single Python file in godot-cpp. The file doesn't depend on anything in godot-cpp.
        - Bromeon: Do you intend for this to become a general mechanism that all bindings could use? Or is it mainly geared for godot-cpp?
        - Fabio: It was made for godot-cpp, but I can see it being useful for other bindings. If other bindings find it useful, I can see this being moved to the main godot repo or code it into the engine.
        - Bromeon: Since you mentioned build-profiles, is the plan to predefine specific groups of includes/exclusions (e.g., "no 3D").
        - Fabio: I'm not sure. We can add a separate collection. However, it is likely different per binding/project. The Godot engine has a tool for creating an engine compilation profile (Project -> Tools in editor), but we could create something similar for GDExtensions. The use case is different though; one case is recompiling the engine without the unused features while the other is just hiding the methods from api.json.
        - Bromeon: So the idea is to save time by just recompiling the api.json instead of the whole engine?
        - Fabio: Yes.
        - Bromeon: I think this can be used and useful for non-cpp bindings.
        - David: I think if we are going to make it a more general tool, we would need to move the current hard-coded list of files to always include some sort of argument.
        - David: But, for godot-cpp, I am approving it.
    - Ivorius: Generating .framework / .xcframework bundles for Apple targets? (https://github.com/godotengine/godot-cpp/pull/1679)
        - Ivorius: Frameworks are a way to distribute binaries on macOS. The reason we need that is because, on iOS, you cannot distribute the binary without a framework. Currently, there is nothing in-place to generate these frameworks for the GDExtension side, so any iOS GDExtension developer has to create the framework from scratch.
        - Ivorius: This PR adds a system to generate an Info.plist to assist with framework creation.

- 

- 

- 

- Fabio: If I remember correctly, the metadata file is an XML file for .framework but is a binary file for .xcframework, which is painful.

- Ivorius: I found a command where you input multiple framework binds and it creates the xcframework bundle.
- Fabio: Is that a Mac-only tool?
- Ivorius: Yes.
- Fabio: On old Intel-based Macs, if Godot loaded an unsigned GDExtension, the user would get a warning where they could allow the file to be loaded. However, on the new ARM-based Macs, you have to manually allow it on the file.
- Fabio: The Godot editor wraps the extensions in a .dylib framework when loading the extension. However, this doesn't help when you are exporting a project with the unsigned extension.
- Ivorius: Signing is difficult because Apples requires it so that either you or the GDExtension developer need to pay for an Apple developer license to sign their extension, which is a barrier.
- Fabio: If we can have a tool for the framework file, that should work. In the end, I think the joiner script is the easiest solution.
- Ivorius: I think we are on the same page. It also has the upside of being consistent with what Godot does, which reduces the complexity of godot-cpp.
- David: From the perspective of godot-cpp, does this mean removing the universal architecture and it is up to the user to create it?
- Fabio: We don't have to remove the universal option. We can deprecate it for those using it.
- David: Is there an advantage to the script rather than having scons run it after the first build. Godot does something like that for Android build (scons runs gradle which creates the arr file).
- Fabio: It would be a bit weird for scons to join the Intel and ARM builds because it becomes tricky when there is user code involved.
- Ivorius: I think a scons tool in the same vein as the PR could be possible. You would input both targets and scons would figure out how to create a framework based on the paths. The user would be free to call or note call it based on their needs.
- David: I don't think it needs to know how to do the two targets. It just needs to know where the two builds are.
- Fabio: That is easy to do in general, but I'm not sure how easy that would be to do in scons. We would probably need a special build target in scons for it.
- David: I don't know if scons is better or worse; it just seems that, fundamentally, building the universal is a build process (and thus scons).
- Ivorius: I think it would make more sense to the user for it to be in scons. That would be more consistent. However, having a separate script, it would be easier to call separately (e.g., you are building ARM and x86 builds on separate machines before combining).

        - David: I don't see how it is easier since it would be the same number of scons commands.
        - Ivorius: I guess that might work. My concern with files being built on different machines is that scons would try to rebuild the files when combining them.
        - Ivorius: With the current PR, in general, I think it makes too many assumptions. I am going to rework it and then we can look at it again.
    - Ivorius: GDExtension editor amnesia on macOS (https://github.com/godotengine/godot/issues/96403)
        - Ivorius: GDExtension reloads don't work on macOS. The PR details three cases where the reload fails. I've found out the cases and what causes it, but I don't know how to fix it. If someone is familiar with how GDExtensions are reloaded, I would appreciate it.
        - David: I am familiar with the code. If you share the underlying cause and the macOS-specific bits, we can work it out.
        - Ivorius: The worse case is if you link the .framework as the library target. If you click outside of Godot, it immediately forgets the target.
        - Ivorius: The next cause is if you link it inside the binary for the framework. It retains the information so long as you don't rebuild it. However, if you rebuild, it forgets it.
        - Ivorius: The final case is if you use a ./, which always works.
        - David: If I remember, reloading for GDExtension on macOS has several different fallbacks it uses. There is also an issue with reloading on macOS in general.
        - David: I don't know if it is just godot-cpp, but the issue has to do with thread locals (#1594). I think the PR removing the thread locals has been merged. It hasn't been cherry-picked to 4.3 yet.
        - David: We will talk more offline.
-

2024-11-26

- Attendees:
    - dsnopek
    - Patrick (FlameLizard)
    - Logan
    - aaronfranke
    - Thaddeus Crews
    - Lukas / ivorius
    - Fabio Alessandrelli
    - Zylann
- Recent news:
    - The big CMake modernization PR was finally merged!

        - See https://github.com/godotengine/godot-cpp/pulls?q=is%3Apr+is%3Amerged+cmake
        - There have been a few follow-up PRs (for additional bugs or regressions) that will be merged soon too
        - Samuel's absentee comments
    - PR doing earlier registration of some singletons was approved by core team!
        - See https://github.com/godotengine/godot/pull/98862
- Topics for this meeting:
    - Ivorius: github action: pull out to a separate repository?
        - Make into a proper action (in the marketplace?)
        - If we split it off, who's going to maintain it?
            - Patrick already too busy
        - Could be put into the godot-cpp repo?
            - Wouldn't allow tagging/versioning it
        - Or, we could have a "Godot actions" project with actions for Godot, godot-cpp, etc
        - If we refer to it in the godot-cpp submodule there are no surprises, because you need to explicitly update the submodule
            - Like the cache action, which is already used elsewhere: https://github.com/Faless/webrtc-native/blob/master/.github/workflows/build_release.yml#L129-L133
            - Then godot-cpp CI could help to test it as well
        - **TODO:** Lukas will make a PR to move it to godot-cpp
    - Ivorius: Structure of scons / SConscript? (https://github.com/godotengine/godot-cpp/issues/1646)
        - Unknown variables warning
            - Fabio:
                - It's only a warning, more of an aesthetic thing
                - Can we detect if we are included or used directly?
                - It's already become more and more of a tool
                - Moving to "site_scons/..." won't help when using as a sub-project
            - Lukas
                - Right now there are path issues if you try to use it as a tool
                - Either: (1) use a SConscript or (2) fully convert to a tool and use it that way in the template
            - Fabio:
                - Let's fix the path issues, but using a SConscript it's very easy for folks who may not know scons
                - At one time had a stub PR to generate all the MacOS stuff
            - Lukas:
                - If both were supported that would be the best of both worlds

        - Fabio:
            - Should be doable to fix the path issues, but needs some investigation
- dsnopek: [godot-cpp] Make godot-cpp installable with CMake/SCons
    - Proposal: https://github.com/godotengine/godot-proposals/issues/9364
    - PR: https://github.com/godotengine/godot-cpp/pull/1418
- Thaddeus: [godot] Add animation node extension
    - Specifically: Clarifying any hard/soft requirements for adding "extension" types to main repo
    - PR: https://github.com/godotengine/godot/pull/99181
    - Why use "*Extension" classes?
        - Fabio:
            - For historical reasons partly. It allows GDExtensions to extend abstract classes, while keeping them from instantiated (which would be useless and confusing)
            - Some were done this way, even though it may not have been necessary
        - Aaron Franke
            - Doesn't like using "abstract" classes this way - we could use "virtual" classes for this
            - GDREGISTER_VIRTUAL_CLASS() vs GDREGISTER_ABSTRACT_CLASS()
                - "ABSTRACT" can be used with C++ classes with pure virtual methods
                - To use "VIRTUAL" instead, you have to implement those methods, which could call GDExtension virtual functions
            - Old CanvasItem PR:
                - https://github.com/godotengine/godot/pull/67510
        - Fabio:
            - If you can't implement something in GDScript, then you shouldn't be able to instantiate
        - Aaron Franke:
            - More things could be implementable but calling GDExtension/GDScript virtual functions
        - Fabio:
            - Some classes use "*Extension" when they could not use it, is just for legacy reasons
            - If we could move away from that going forward, that would be good
        - David:
            - XR uses "*Extension" classes because you need to receive/return pointers which is only possible via GDExtension, and not scripting

        - Aaron Franke:
            - If it's possible to convert from abstract to virtual, I think it's worth doing
            - This shouldn't affect performance, vtable is already involved
    -

2024-11-12

- Attendees:
    - dsnopek
    - aaronfranke
    - Jan (Bromeon)
    - Ivorius / Lukas
    - Hunt Sparra
    - ughuu
- Topics for this meeting:
    - ughuuu: https://github.com/godotengine/godot-cpp/pull/1415#pullrequestreview-2399484398 module/addon import symmetry
        - **David:** Ughuu may have moved on to a different approach (module wrapper)
        - **David:** The overall approach looked good. I'll reach out and if he is not working on it anymore, I will take over. It is good to make godot-cpp's includes more compatible with Godot's.
        - **Ughuu:** I'm not working on it, but someone else can take it over if they want.
    - Bromeon: Classes without default constructor break reloading
        - Issue: https://github.com/godotengine/godot/issuesd/96823
        - PR: https://github.com/godotengine/godot/pull/99133
        - **Jan:** In languages where you can have multiple constructors for an object, this can break reloading. A common use case for this is classes that you primarily use in code instead of instantiating in a scene. The problem is that if a single class is not reloadable (due to not having a default constructor), it breaks the entire extension.
        - **David:** So these are not abstract classes.
        - **Jan:** One solution would be to require some sort of serialize function for classes that do not have a default constructor. Currently, there is a reload callback but not a save callback.
        - **David:** The only serialization we're currently doing is serializing the Godot object's property and restoring those. They are stored as variants and not serialized to disk. Currently, we do not have a system to serialize private data.

        - **David:** I think we have a bigger problem here with how to create classes without default constructors.
        - (Discussion about undefined behavior when recreate_instance_func fails; currently, it likely either crashes or takes on the behavior of its non-GDExtension parent class)
        - **David:** Properties should be restored from the GDExtension class and not the Godot base class (since the Godot object is not unloaded). If it doesn't do that, we should make it do that.
        - (The current PR doesn't exactly fix the issue; David will write some notes on it)
    - Enetheru(won't be present): cmake PR1598
        - This is the big change to make CMake much closer to Scons.
        - **David:** The code looks good, but I can't get it to work locally. It passes on CI but fails locally. This may just be a local issue. If anyone else has time to test this, that would be great.
        - **aaronfranke:** I'll do another pass once the PR author addresses your comments.
    - Ivorius: GH build action; setup-godot-cpp or pass down parameters? Extract into its own repo?
        - Some reviewers suggested moving the GitHub action for building godot-cpp to another repo.
        - **David:** I personally think we should move it to another repo. Once people copy it, then it is theirs to modify.
        - **Ughuuu:** Moving it somewhere would be good, if possible. I use the actions by reference.
        - **Lukas:** I also use the actions by reference instead of copying it, and it would be nice that I did not have to review it after every commit.
        - **David:** I think we need to bring Patrick in since he is likely the one stuck maintaining it.
    - Hunt: Register Engine, OS, ProjectSettings, and Time singletons in time for for INITIZATION_LEVEL_CORE
        - https://github.com/godotengine/godot/pull/98862
        - **David:** Some project settings are still not loaded until later, which would lead to unexpected results. However, it would be nice to have at least some accessible. For example OpenXR has some settings it would set in Core.
        - **Aaronfranke:** If there are any issues, I suspect they will be subtle. In the worse-case scenario, we could do a partial revert. At least the Time singleton should be safe since it depends on nothing.
    - dsnopek: Created documentation issues for anyone who would like to pick them up
        - https://github.com/godotengine/godot-docs/issues/10253
        - https://github.com/godotengine/godot-docs/issues/10254
    - Rapier

        - **David:** I think this is more of a question for the Physics folks. If they have changes for it, we can make it possible on our end.
    - Jenova Scripting: https://github.com/Jenova-Framework/J.E.N.O.V.A
        - **Aaronfranke:** I already have a PR for C++ scripting in Godot, but this looks much bigger.
        - **Ughuu:** I bring this up because it have more than godot-cpp and different APIs. Notably, it has autocomplete for *everything*. I think godot-cpp should go in this direction.
        - **David:** Do you know if they are using the Scripting API?
        - **Ughuu:** No.
        - **Aaronfranke:** This remind me of 3-4 years ago when Juan was proposing we go in the opposite direction and C# is treated like godot-cpp where C# objects are their own node type.
        - **David:** I think that is what Raul is doing.
        - **David:** w.r.t. Moving in this direction. Our main design goal is to match Godot's APIs, which limits us. If we wanted to go this direction, we would need to change things on Godot's side. Regardless, this is cool and it is good to have several different approaches. Still, I'd be interested to know if they were doing something similar to GDNative.
    - Current GDExtension/C++ Scripting PR: https://github.com/godotengine/godot/pull/90979
        - The current blockers are support for custom modules and hard-coding the scons setup (instead of reusing the template from godot-cpp).
        - **Aaronfranke:** In my opinion, custom modules are a generally useful feature. Also, this was one of the selling points of the new GDExtension framework.
        - **Aaronfranke:** Part of the reason we hard-code the scons setup is because we cannot know the developer's folder structure. It also allows compiling godot-cpp offline from a local version.
        - **Ughuu:** I like hard-coding the scons setup.
        - **David:** I don't think we should use the godot-cpp template because, if we do, then the Engine will depend on the template and we no longer can change it without downstream effects.
        - **Ughuu:** The only downside with the current approach is that compiling the addon is not done on threads, which freezes the editor.
        - **David:** It would be nice to have at least some feedback.
        - **Aaronfranke:** I'm not sure how to redirect OS output back into Godot.
        - **David:** Android is doing something like that for Gradle, so maybe there is something there.
    - Add `RequiredPtr<T>` to mark `Object *` arguments and return values as required
        - Draft PR for discussion: https://github.com/godotengine/godot/pull/86079
        - **Jan:** Do we still plan to pursue this?

        - **David:** Yes. I need to rework the PR for the type-state pattern discussed ~1,000,000 meetings ago.
        - **Jan:** I could take a look at it, but I'm not too familiar with the internals.
        - **David:** We can chat about it.

2024-10-29

- Attendees:
    - dsnopek
    - Patrick (FlameLizard)
    - Hunt Sparra
    - Ivorius / Lukas
    -
- Topics for this meeting:
    - Patrick (FlameLizard): **godot-cpp-template** Change the Github action to default fetch-depth (= 1) https://github.com/godotengine/godot-cpp-template/pull/59
    - Patrick (FlameLizard): **godot-cpp-template** "Clean up unnecessary compiledb parameters" https://github.com/godotengine/godot-cpp-template/pull/40
        - **AI:** Ivorius will test and make sure that the compiledb commands still work with this code removed
    - dsnopek: [godot-cpp] Update for new NOTIFICATION_POSTINITIALIZE handling
        - https://github.com/godotengine/godot-cpp/pull/1568
    - dsnopek: [godot-cpp] Allow unicode class names
        - https://github.com/godotengine/godot-cpp/pull/1574
    - dsnopek: [godot-cpp] Directly get object instance ID from Variant and implement Variant::get_validated_object()
        - https://github.com/godotengine/godot-cpp/pull/1591
    - dsnopek: [godot-cpp] Avoid thread_local on MacOS to prevent issues with hot reload
        - https://github.com/godotengine/godot-cpp/pull/1594
    - dsnopek: [godot-cpp] Fix crash in ClassDB::add_virtual_method() if arguments metadata is the wrong size
        - https://github.com/godotengine/godot-cpp/pull/1581
    - dsnopek: [godot-cpp] Don't print an error when decoding a null Ref<T>
        - https://github.com/godotengine/godot-cpp/pull/1616
    - dsnopek: Wording about GDExtension compatibility & stability in godot-cpp's README
        - https://github.com/godotengine/godot-cpp/pull/1588
    - dsnopek: Things that need documentation (have come up multiple times after the 4.3 release)
        - Godot & godot-cpp:
            - Runtime classes
        - godot-cpp:
            - `memnew()` and `Ref<T>`

                - Unable to use Godot types in global variables
                - Design goals around API compatibility with Godot
            - **AI:** dsnopek to put these into issue(s) on godot-cpp and godot-docs
        - Ivorius: macOS Universal builds vs single-arch
            - https://github.com/godotengine/godot-cpp/pull/1613
            - **AI:** Ivorius to write a new godot-cpp issue about switching to doing it like Godot, and we'll ping Fabio for his thoughts (Edit: Issue is here)

2024-10-01

- Attendees:
    - David Snopek
    - vnen
    - Adam Scott
    - Radiant
    - Zylann
- Topics for this meeting:
    - Radiant (RadiantUwU): Create RenderingServerExtension and RenderingServerManager to allow creating rendering methods from GDExtension API. (superseding or extending the idea of https://github.com/godotengine/godot-proposals/issues/4287)
        - Add new rendering methods, ex ray tracing, without having to edit the engine's code
        - Adam: Do we have an immediate use-case for this?
            - It will be easier to know how to design it if we have a concrete use-case
            - If it's just "interesting", maybe this isn't the time to work on this
        - Radiant: Wants to make a ray tracing renderer
        - David: Juan has mentioned that in the future he'd like to have a high-end ray tracing renderer in Godot, so this is something that could potentially be merged upstream
        - Next steps:
            - Chat with rendering team
            - Look at what physics is doing
            - Chat with physics folks to see if what they do for physics is working well
    - Radiant: Issue with creating Godot objects before GDExtension is initialized (in the context of Rust)
        - David: We discussed this with godot-cpp recently https://github.com/godotengine/godot-cpp/issues/1449
        - David: the process looks like this:
            1. Godot starts up
            2. GDExtension DLL/SO is loaded

            3. Global variables will be initialized
            4. Godot calls into the GDExtension so it can load the GDExtension interface
        - Let's start a new issue in the Godot repo to discuss with a wider group
    - Patrick (FlameLizard): Change folder structure of godot-cpp-template
        - https://github.com/godotengine/godot-cpp-template/pull/49#pullrequestreview-2328609733
        - Adam: template is supposed to be simple to get you started
            - For some people its for gameplay classes, not a re-usable addon
        - David: I agree - in my GDExtensions we don't use this layout
        - Adam: GDExtensions aren't required to be in addons, and aren't enabled/disabled like GDScript addons
        - Godot Jolt puts it in addons
        - Vnen: If you wanted it in addons, you could copy it there
            - The source structure doesn't affect the binary/distributed structure (it'd be a different repo or a ZIP file that the Godot Asset Library points to)
    - Enetheru: merge cmake PR's or tell me what you need to be confident
        - https://github.com/godotengine/godot-cpp/pull/1595
        - https://github.com/godotengine/godot-cpp/pull/1598
        - https://docs.google.com/spreadsheets/d/1aUjib11P9CfcMK7P5WhdRIs80jMO9mGPK4RbMlSccCM/edit?usp=sharing
    - Zylann: https://github.com/godotengine/godot-cpp/pull/1162
        - Hopefully Thaddeus will respond soon!
        - Otherwise, maybe the code is unused and we can remove it?
    - Adam:
        - Quick overview of the priorities, make sure that we explain well the subject

2024-09-17

- Attendees:
    - David Snopek
    - Adam Scott
    - Noah (Nlupugla)
    - Thaddeus Crews (Repiteo)
    - Zylann
- News
    - Last batch of cherry-picks exposed some bugs
        - Unsupported use-case after `is_instance_valid()` removal: https://github.com/godotengine/godot-cpp/issues/1587
        - Hot reload broken: https://github.com/godotengine/godot-cpp/issues/1589
            - Probably since June

                - PR from David to fix this
            - Probably means at least one more 4.1 cherry-pick
        - Work has picked up on cmake, led by enetheru
            - Adam
                - Could be interesting to see how's maintaining cmake vs scons
                    - It's an interesting use-case to see if cmake is somewhat viable for the godot repo
        - Allow unicode class names:
            - https://github.com/godotengine/godot/pull/96501
            - Trouble with CI on companion godot-cpp PR: https://github.com/godotengine/godot-cpp/pull/1574
                - Help would be appreciated!
                - Maybe it has to do with system's encoding or the versions of the compilers
                - Doesn't happen all the time, weird.
- Topics for this meeting:
    - Repiteo: Typed dictionaries
        - https://github.com/godotengine/godot-cpp/pull/1162
        - Discussion
            - Repiteo
                - godot-cpp is failing due to CI
                - Currently works one-to-one as it works on the Godot repo.
                - CI and static checks are fused in the Godot-cpp project, instead of being separated in the main repo.
            - David
                - If you're stuck, let us know!
            - Repiteo
                - That was actually the versioning of the API that failed the PR checks.
                    - A new PR is there to fix this:
                        - https://github.com/godotengine/godot-cpp/pull/1593
    - nlupugla: Struct integration in GDExtension
        - https://github.com/godotengine/godot/pull/82198
        - Discussion
            - nlupugla
                - Been working part-time on reduz's idea of structs
                - A relatively new user wanted to have some stuff going in C#
                    - The user is pretty keen to make it work in C#
                    - Someone suggested the person to work on godot-cpp before making it work on C#
                - There's both demand from C# and GDExtension to use structs

    - One thing I'm trying to figure out is how to expose the feature
    - I know roughly on what I want things to happen, but I don't know how.
    - Is the feature to export C++/C# structs to the inspector wanted in the first place?
        - It's just saving boilerplate on something that you could already do technically
- David
    - I'm a little unsure on what people would use this for
    - To be able to expose a C++ struct to the inspector, it's actually different features
- Adam
    - Actually, it's to expose native Godot structs, isn't it?
- nlupugla
    - What I was thinking about:
        - Doing some scripting in C++ and I want to define a Bullet stuct
            - Position, velocity
        - You cannot expose this to Godot directly
            - So, what you would do is to create a Godot resource
                - But it's super heavy
            - Or to have boilerplate to transfer from the heavyweight resource to a lightweight C++ struct
    - In core, structs are a fancy array
        - Like typedarrays, they have extra metadata
    - Notion in lightweight structs in multiple languages
        - Only time you would want to use Godot structs, it's for transferring data
- David
    - So you would want to copy data? But that's performance intensive.
- Nlupugla
    - This was inspired from this user coming from C#
        - This is especially about displaying data from the inspector
- Adam
    - There's two things
        - First,
        - Secondly, we need to make sure to differentiate C# and C++, the godot-cpp team will decide how to implement the Godot structs.

- Nlupugla
    - How does it work with GDExtension?
- David
    - It’s still arrays, right? So it would be very similar to the implementation of arrays and add the specific metadata and such.
- Zylann
    - Unless I miss something, Unity doesnt even support exposing structs in the inspector. In C# you actually have to use classes
    - Which are reference types and therefore quite different from what a "struct" means in C# and other languages in terms of semantic
    - So the equivalent in Godot is to make a custom Resource type which is a significant undertaking in languages that lack metaprogramming/reflection
    - Generally exposing basic "data" classes the way Unity supports out of the box is something even Godot itself is lacking with GDScript
    - (sure you can make them resources and use export and all that but that's more work and it doesnt even have the same UX)
- David
    - Integrating GDExtension would be pretty simple
- Nlupugla
    - ClassDB, usually, GDScript classes are not registered
    - Is there a way to mitigate this?
    -
- Zylann
    - They are technically not structs in the sense of C# or C++
        - They are “Godot” structs
            - You cannot use C# or C++ directly, you need to go through the process of inheritance and such
            - It doesn’t map memory wise
        - The way I see it is that these are dictionaries with predefined keys.
- Nlupugla
    - Yes, exactly, about the dictionaries.
- Zylann
    - What happens if GDScript defines a struct and sends it to the GDExtension
        - What does GDExtension see?
- Nlupugla

    - It would just see an array
- Zylann
    - You mean something like “StructArray” or something like that?
- Repiteo
    - Why we are using these kinds of workarounds is because Godot has a very primitive type system
- Nlupugla
    - property_info.h
        - Macros fill the boilerplates and such
        - Then, later on, there’s a bind_struct macro where it works similarly to other bind_ macros.
    - For godot-cpp, I could rewrite the macros in core
        - For a different language, I could find a way to do something
- Adam
    - You don’t have to do this right now.
- David
    - We would be better to not have any gdextension modifications that aren’t testable right now
- Nlupugla
    - How does TypedArrays work in GDExtension?
- David
    - They define metadata and such
- Zylann
    - Would structs be able to contain structs?
- Nlupugla
    - Not now, but it’s planned
- Zylann
    - I don’t know how it would look in C#
        - Probably gonna be better, but not a native struct
- David
    - We already can pass C structs as data from gdextension
    - Maybe we should do something like this
- Zylann
    - How does it would look like, that’s a question
    - Technically, we could define a memory layout and the extension would pass a piece of data that corresponds to the memory
    - We need though that the order doesn’t change
- Nlupugla
    - Yes, order matters in my structs
    - Structs are designed for performance, but specific type of performance

            - They're designed for classes with only two properties
            - They aren't design to fix the "raycast issue"
    - Conclusion:
        - Nlupugla
            - Summarizes the things to do
        - David
            - We need to think through how to expose stuff
            - If there's a way to do this before exposing stuff to gdextension, in two parts, it will be better
            - To prevent making it exposed, it's to remove methods that returns structs
        - Adam
            - Structs must not get in the json extension api
            - And quality of life upgrades should be done afterwards
        - Nlupugla
            - And what about the users that want to map C# structs to the inspector
        - Adam
            - I would concentrate on the Godot side instead, you shouldn't feel pressure to do additional stuff

2024-09-03

- Attendees:
    - David Snopek
    - FlameLizard
    - Jan (Bromeon)
    - Fales
- Recent news:
    - GDExtension "loaders" PR merged:
        - https://github.com/godotengine/godot/pull/91166
    - Allow ClassDB to create classes with NOTIFICATION_POSTINITIALIZE PR merged:
        - https://github.com/godotengine/godot/pull/91018
    - Fix editor needs restart after adding GDExtensions PR merged:
        - https://github.com/godotengine/godot/pull/93972
    - Jan: Added CI to compile gdextension_interface.h with C complier
    - Fales:
        - Resurrecting video decoder extension
            - Seems to be working, needs a little more work to update to modern Godot/godot-cpp
            - Should GDExtension team take on maintenance on this?
                - Move into the Godot organization

                - We still have Theora playback in Godot, but maybe this could be a way to move that out
                    - In Godot 4, we removed the webm format - this is what the extension supports
                    - Copied from Godot 3 - could use some more work, to support more platforms
                - Patrick: YouTuber (TheVoylin?) made video editing app with video encoding in GDExtension
                    - Maybe we could collaborate with him?
                    - Bundles ffmpeg - would support everything?
                    - One of the videos: https://www.youtube.com/watch?v=C9ptuhAB3GI&t=347s
                - Fales: Original purpose of the extension was as an example
                    - Maintain an AVImage extension
                    - Supporting more formats would show that Godot can support any desired format
                - Should we have a gdextension demo repository?
                    - One example for image, one for video, etc
                    - A way to validate that we don't break use cases
                - Patrick: We should try to minimize maintenance burden so it doesn't get abandoned
                    - An argument for using ffmpeg
            - Porting PSD module to GDExtension
                - Missing way to extend the ImageTexture class to allow that
                - Old PR: https://github.com/godotengine/godot/pull/73417
                -
            -
- Topics for this meeting:
    - Add `RequiredPtr<T>` to mark `Object *` arguments and return values as required
        - Draft PR for discussion: https://github.com/godotengine/godot/pull/86079
        - Jan suggested using "type state" pattern as a potential way to improve the `RequiredPtr<T>` proposal

2024-08-21

- Attendees:
    - David Snopek
    - Hunt Sparra
    - Aaron Franke
- Topics for this meeting:
    - Discussed doing a new survey for days/times to help address dwindling attendance
    - Discussed issues Aaron was having with porting a Godot module to GDExtension

    - Discussed early registration of singletons
    - Discussed possible ways to re-organize the GDExtension documentation
    - Briefly discussed the roadmap
    - **Given low attendance, it was a very informal meeting 🙂**

2024-08-09

- Attendees:
    - David Snopek
    - dragos
    - Gonzo
    - George (vnen)
    - Jan (Bromeon)
- Recent news:
    - dragos: Investigating Rust performance issue with custom physics engines:
        - https://github.com/godot-rust/gdext/issues/790
        - https://github.com/godotengine/godot-proposals/issues/10389
        - Might be the result of Callable performance with the Rust binding
        - dragos will profile Callables in C++ vs Rust
    -
- Topics for this meeting:
    - dsnopek: GDExtension: pointer parameters of type real are declared as float* in JSON
        - https://github.com/godotengine/godot/issues/93132
    - dsnopek: Is godot-cpp missing anything we need for the Godot 4.3-stable release next week?
        - Nothing that we could think of
    - ughuuu: General questions about using ScriptLanguage and ScriptExtension. Related to https://github.com/libriscv/godot-sandbox. Basically we want a way to dynamically have autocomplete at GDScript level for objects (and Script's seemed like the best way, but just wanted to check it during this meeting).
    - dsnopek: Add ScriptInstanceExtension class which mimicks the API of Godot's internal ScriptInstance
        - https://github.com/godotengine/godot-cpp/pull/1544
        - dsnopek will update the PR to convert all the C types to godot-cpp types
    - dsnopek: Fix property/method filtering in extension_api_dump.cpp
        - https://github.com/godotengine/godot/pull/64825
    - dsnopek: [godot-cpp] Use single godot-cpp branch for Godot 4.1+ and version it independently from Godot
        - https://github.com/godotengine/godot-proposals/issues/10243
    -
-

2024-07-24

- Attendees:
  - David Snopek
  - Zylann
  - Aaron Franke
  - Hunt Sparra
- Recent news:
  - [godot-cpp] Unexpose UtilityFunctions::is_instance_valid()
    - https://github.com/godotengine/godot-cpp/pull/1513
    - It is not safe to pass an object pointer to is_instance_valid() since, if it has been freed, it will crash. The only safe way to do this was to wrap the object in a variant. This has been a “trap” for many developers, so it is being removed.
  - [godot-cpp] Remind developers about memnew() in crash message when missing binding callbacks
    - https://github.com/godotengine/godot-cpp/pull/1510
- Topics for this meeting:
  - dsnopek: [godot-cpp] Use single godot-cpp branch for Godot 4.1+ and version it independently from Godot
    - https://github.com/godotengine/godot-proposals/issues/10243
    - David: Having multiple branches has caused “communication problems” due to some loose cherry-picking between master and previous version.
    - David: This proposal advocates for us switching to a single branch for 4.1+. Builds would look at extension_api.json to determine the version to build. This also allows us to have nice changelogs, patches, and differentiate between different types of versions.
    - Hunt: How would this handle breaking versions? I’m thinking about new functions in the GDExtension interface.
    - David: We would use ifdefs in that case.
    - Hunt: Would scons determine the version?
    - David: Scons would pick the latest in extension_api.json. We could add an option to scons to override it. Worse case scenario, that can be done directly in the scons file.
    - Aaron: I’m contributing to a Godot project that is using CMake. I picked around in Godot’s Cmake… and it needs a lot of fixing. I’m having a friend look at the CMake file, and he’s working on fixing it up. If this change goes through, then any new scons flags will have to be brought over to CMake.
    - Aaron: Aside from that, I still think it is helpful to have a submodule/separate branch for specific versions. That makes it easy to guarantee you are building for the right version. Reducing cherry-picking is great, but, if it is 90 minutes (as David said earlier), is it making this change? Are we expecting to have less cherry-picking over time?

        - David: You're right about the cherry-picking. If we've decided to maintain only the previous two versions, the cherry-picking amount remains constant. However, the real problem I am hoping to solve is the communication one. There have been a number of instances where there has been an important change bundled with a subtle behavioral change, which has resulted in unexpected behavior when people upgraded.
        - Aaron: I'm not sure if communication would be enough to solve the problem. I do see where you're coming from with a godot-cpp version vs. a godot version. That makes a lot of sense.
    - CMake
        - Zylann: I had the idea of having a build system for godot using Zigg, but I don't think there is enough interest in it.
        - David: Aaron, I agree the current CMake configuration in godot-cpp is a mess, and I don't know how to fix it. What we really want is for the CMake **to do exactly what is in Scons and no more**.
        - Aaron: On that topic, that's my entire goal here. Right now there is no support for multiple architectures nor Mac. Right now, the CMake file assumes everything that isn't msvc is gcc, which means it doesn't work with Clang. My friend and I are working on it and are hoping to submit a PR in a few weeks.
        - Aaron: Limitations in CMake are making this is a huge challenge. If you go to the test project without having godot-cpp built, Cmake won't call godot-cpp, won't build the godot-cpp files, and fail to compile. Right now we are working on having CMake automatically compile godot-cpp when needed. However, CMake is not designed to do this. It does not like to have the test project be a subfolder of the CMake directory. Still, we're going to do our best.
        - David: What if the project is at top level and godot-cpp is a subfolder?
        - Aaron: That would work for most projects with CMake. Still, we want to get the project layout to be identical to scons.
        - Aaron: Also, CMake will not automatically detect new cpp files. You have to clean and then rerun. The CMake developers suggest manually listing every file that exists and not glob-starring.
        - David: I think it is fine for the tests to be broken for CMake, so long as the common use case is supported. We have the godot-cpp-template now.
        - Hunt: What's the layout of the template?
        - David: godot-cpp is a subdirectory under the project. So, it should work with CMake.
        - David: Really, we need a dedicated CMake maintainter. There are a lot of CMake PRs now. Well, only 11, but I don't have any way to evaluate them.
        - Aaron: I'm not an expert at CMake, but I may still look at those PRs.
        - David: Excluding the test project, we will want our CI to make sure the CMake process works.

        - Aaron: What if the CI copied the test directory outside of godot-cpp for CMake testing?
        - David: That would be fine if we documented that the different folder structure was required for CMake.
        - Aaron: I don't think it is worth looking through the CMake PRs, since the CMake file has to be rewritten already.
        - David: If we could have someone look at https://github.com/godotengine/godot-cpp/pull/1355, that would be great
    - dsnopek: GDExtension: pointer parameters of type real are declared as float* in JSON
        - https://github.com/godotengine/godot/issues/93132
        - In extension_api.json, we are handling primitive types differently. The question is: do we document how the meta field works in JSON, or do we try to convert the types so, even when they are not following what is expected, it still works.
    - dsnopek: Web export with gdextension not working on 4.3 if threads/nothreads variants are mismatched
        - https://github.com/godotengine/godot/issues/94537
        - Problem: If you build with/without threads, you must use a template that was built with/without threads.
        - We are thinking about documenting this, but are there any other solutions?
        - Hunt: Is there anyway to determine if a loaded template was compiled with or without threads?
        - David: Scons may put stuff in the file name.
        - David: Godot-cpp maybe could print something from scons when doing a web-build with threads=yes.
    - C#
        - Zylann: Currently, using GDExtension with C# is a headache.
        - David: The best I've seen so far with C# is some community work with extensions. Really, we need code generation.
        - Zylann: The main issue is that the C# bindings is hard-coded into Godot.
        - Hunt: What do you mean?
        - Zylann: Right now the only way to classes to show up in C# is to compile them into the engine. That's why I am keeping Godot-Voxel as a module for now instead of porting it to GDExtension.
        - David: Code-generation sounds like the answer here.
        - Zylann: Godot should generate all the C# together (instead of it being hard-coded), and it should include the GDExtensions too.
        - David: That's not too different from the community solution I saw.
        - David: In 4.4, I think we'll be making some advances in interoperability. Raul is working on bringing C# to GDExtension.
        - David: It's a similar problem to having dependencies between GDExtensions.

        - Zylann: The one way I could think of doing it was to include the sources and compile the separate extensions as one extension. Another way to do it would be to have Godot generade the binding between them.
        - David: I think we can do this the same way we do native classes. I have some ideas and half-worked branches that make me think we can do this.
        - Zylann: Personally, I would not think of doing things like this (mixing languages).
        - David: I don't think calling between languages would be any less efficient than calling from Godot into GDExtension.
        - Zylann: It's probably better than variant calls.
        - Zylann: One big limiter with the Godot API is that you're limited to Godot's data types. For example, a byte array has to be a PackedByteArray in Godot.

2024-07-12

- Attendees:
    - David Snopek
    - Hunt Sparra
    - Naros
    - Thaddeus Crews
    - Riteo
    - Jan (Bromeon)
    - George Marques
    - spectius
- Recent news:
    - Merged: Fix setting base class properties on a runtime class
        - https://github.com/godotengine/godot/pull/94089
        - Fixed issue found with Rust: https://github.com/godotengine/godot/issues/93676
    - Bromeon: Godot-Rust community has been experimenting with WebAssembly export support
        - Got working on (some people's) Firefox, but some work needs to be done with threads
        - Reproducible setup is still a work in progress, but some people have gotten it working in Firefox
- Topics for this meeting:
    - Bromeon: Should bitfield types be unsigned integers instead of signed integers?
        - https://github.com/godotengine/godot/issues/88962
        - History: Negative values for bitfields cause issues when casting from a smaller negative integer size to a larger one (e.g., int32 to int64). This is complicated by the default type for integers varying between compilers and flags.

- Options:
    - Document that it is signed
    - Create an unsigned API (encouraging unsigned bindings)
- Thaddeus: Godot's Union type currently is always int64.
- Bromeon: I think a new (unsigned/bitfield) API could make this more clear for developers.
- David: If we added a new API for bitfield constants, would they still be stored the same way?
- Bromeon: I think we can store it the same way, but then, when you query for the constant through the Godot API, you would get back a signed value instead of an unsigned one.
- Thaddeus: If we were to change it, we would need to change the bitfield class from int64 to uint64.
- David: This change could probably be standalone.
- Thaddeus: Having bitfields look the same when retrieved as when registered would be good.
- Hunt: Do we want to force unsigned bitfields for all developers? There will be at least a few devs who think it should be signed.
- Bromeon: Even if the change was made, there would be issues with languages like GDScript and Java that don't have an unsigned type.
- Bromeon: Everything that is currently working has to keep working as-is. One option is to make the unsigned API a new methods for registration and retrieval.
- Hunt: What is the benefit of separating the bitfield retrieval APIs from the regular integer constant APIs.
- Bromeon: The benefit of it would be having a matching retrieval method for the registration method, although it doesn't necessarily need to be represented differently internally.
- Hunt: I'm concerned about separating them causing confusion when "legacy" bitfields were registered with `bind_integer_constant()` and "new" bitfields were registered with the new API.
- Bromeon: In David's proposal, `bind_integer_constant()` would return both new and old.
- David: In godot-cpp, you don't need to worry about the conversion from a uint64 type to the stored int64 type since it is implicitly converted.
- Bromeon: For APIs, do you think it is still reasonable to use signed values when registering an unsigned bitfelds?
- George: I don't think it matters what the APIs are doing so long as the end result is the same.
- David: The change would be BIND_BITFIELD_FLAG in godot-cpp calling a new bind_bitfield_constant() method instead of bind_integer_constant(). Storage, retrieval, and value marshaling should remain the same. The worst that could happen is users getting a compiler warning when calling

BIND_BITFIELD_FLAG since we would be passing an unsigned argument for a GDExtensionInt=uint64.
        - Bromeon: The new API could explicitly cast the argument and it would pass a warning if a signed value was passed.
        - David: We could add documentation for it to gdextension_interface.h. That would work for now, but I still think we need a separate place for GDExtension documentation.
        - Bromeon: I can take care of the PR for this change.
    - dsnopek: Fix editor needs restart after adding GDExtensions
        - https://github.com/godotengine/godot/pull/93972
        - David: I have tested it, but it did not work. But it is still exciting that this is in the works!
    - dsnopek: Improvement notification export for GDExtensions
        - https://github.com/godotengine/godot/issues/93366
        - David: The original issue was usererror, but we can still add a better error message. Do we want to add that error to ClassDB or leave it to godot-cpp?
        - Bromeon: Both work. I also think there are a lot of these cases with export annotations that are specific to types. This may deserve a better discussion at some point.
        - George: GDScript could use the same PropertyInfo test if the change was made global. However, this is not easy because there are many possible combinations.
        - David: It would be nice if we could validate in ClassDB but leave it to the language to validate, but I don't know how we would do that. We could end up with a fallback check in ClassDB but allow bindings and scripting languages to have their checks before that.
        - George: Some languages should be able to do that at compile time with features such as templates. That is not something that GDExtension can do though.
        - Bromeon: Can the extension actually check the export values in advance? The question is "are there some cases that ONLY godot can validate"?
        - George: The language should be able to validate it in every case.
        - Bromeon: Are the export annotations documented sufficiently so that it is clear what is or isn't valid usage?
        - George: That's a good point. We should have good documentation.
        - David: If we had an API to validate PropertyInfo, that could be used by other languages such as GDScript and GDExtension, and they could validate that way.
        - Naros: That would be really helpful for folks implementing scripting extensions. It could even be limited to debug builds.
        - David: Documenting is nice, but asking Godot if it is valid is even nicer, because Godot can change and evolve.

        - George: Everything looks like it is documented now, but I agree that having something in code is better.
        - Naros: Having it in code also covers human error.
        - Naros: I'm not sure how maintainable the interface would be over time.
        - David: That's a risk in many, many areas in Godot. I think in this particular case, if we can make the validation function generic enough that GDScript can rely on it, then that is good enough (since the massive GDScript community will catch it if it is wrong).
        - Bromeon: Would it return errors or a boolean?
        - David: I think for this to be useful, it should return rich errors. Also, it would be nice to be able to turn off the validation if it has already been verified.
        - Summary: Adding a validation API would be nice, but is a task for later. However, bindings could validate the property at compile time though. The short term solution is for GDExtension bindings to validate PropertyInfo themselves (based on the documentation).

2024-06-14

- Attendees:
    - Hunt Sparra
    - David Snopek
    - Adam Scott
    - Thaddeus Crews
    - Gergely Kis
- Topics for this meeting:
    - Adam: Compiling native modules as GDExtension and vice versa
        - David: The design of godot-cpp is to allow this (although it is imperfect due to different APIs). A good example of this is TextServerAdvanced in Godot.
        - Adam: I'm thinking about the web. One of the main challenges is having the smallest downloads (i.e., builds) as possible. As Godot adds more features, this will get more difficult. One way to solve this is to compile/configure templates at export time. Maybe godot-cpp could help with this.
        - Adam: HP was mentioning maybe shipping Godot with a linker to facilitate this, but I don't know if this is feasible or possible.
        - David: I think HP's idea could work, but it's a high level question (as a project) what we do or don't want to bundle.
        - Adam: I've been thinking of trying to port some modules to GDExtension to see if GDExtension has all the needed features.

        - Thaddeus: I'm right with you. I want to move as much to GDExtension as possible. When working with modules, there are extra precautions that need to be taken with APIs that are not exposed.
        - Hunt: How does performance compare between the two?
        - David: There is a cost to calling GDExtension, but it is not noticeable unless you are doing some serious looping. In that case, the solution is usually to get the pointer for the object you are working with and then loop using that.
        - Thaddeus: It could be helpful to document the performance differences between the two.
        - David: We should document this (along with several other things).
        - (Tabled for now)
    - dsnopek: [godot-cpp] Using `float` in API when Godot uses `float` (rather than always `double`)
        - See: https://github.com/godotengine/godot-cpp/pull/1433
        - To cherry-pick or not?
        - David: We want to match the Godot APIs, so we should be generating the float arguments instead of double. However, this may cause breakage.
        - Thaddeus: Cherry-picking it back makes sense, even with the breakage because developers on older versions likely are already expecting some changes as they update godot-cpp.
        - David: Only a few systems appear to be affected. Mainly audio.
        - Hunt: Would it break already compiled extensions?
        - David: Nope. We're already casting doubles to float before sending them over the API boundary.
        - Adam: It's a small change. Let's cherry-pick it.
    - dsnopek: [godot-cpp] Set instance binding in constructor
        - See: https://github.com/godotengine/godot-cpp/pull/1446
        - Merge for 4.3 or save for the beginning of the 4.4 cycle?
        - To cherry-pick or not?
        - Adam: Fixes it being unsafe to call some methods from the constructor. It does not depend on any changes in Godot, so we can cherry-pick it to any version of godot-cpp. I've tested it and it seems very good, but there is always a chance for regressions.
        - Gergely: I would merge it to 4.3. 4.3 will be active for the next month, so a lot of users will benefit from it.
        - David: I agree. If it is a problem, we can still fix it. Another option for cherry-picking is to merge now and backport once we are confident with it. I'm leaning towards this option. We had a recent issue in 4.1 that we fixed in a later version but did not cherry-pick until people encountered it.
        - (Will merge for 4.3 and wait to cherry-pick)
    - Bromeon: real ptrs exposed as float* - https://github.com/godotengine/godot/issues/93132

        - Adam: The question was if we should stick to what we do now or if we should change it so that pointers to floats are more consistent with how we handle pointers. After discussion, we are leaning towards keeping it as-is.
    - dsnopek: Triage (and hopefully close) some Godot issues:
        - https://github.com/godotengine/godot/issues?q=is%3Aissue+is%3Aopen+label%3Atopic%3Agdextension+-milestone%3A3.x
        - Trying to close out or triage existing GDExtension issues.
    - Add `RequiredPtr<T>` to mark `Object *` arguments and return values as required
        - Draft PR for discussion: https://github.com/godotengine/godot/pull/86079
        - Thaddeus: I have a PR for the opposite way (marking optional). It works for non-pointers too.
        - David: This would require more changes than RequiredPtr since everything is optional right now. (EDIT: See below comments for why this might not be the case)
        - Thaddeus: We could have all values implicitly required and all objects implicitly optional to reduce the number of changes. Although, ideally, the same default would be used for both.
        - David: I'm not sure how we will use optional for scalar types on the C++ side (since values cannot be null).
        - Thaddeus: Nullability should be handled by the language but not Godot.
        - David: I think modeling it as optional makes it easier. For required, it's not obvious what we should do when null is passed for a required value (my implementation just creates a dummy option, but it's not ideal and a little bit of a mess). For optional, it's obvious.
        - Gergely: Wrapping pointers could be a code style requirement going forward. What do you think about that?
        - David: Making it easier to do the right thing is good. The dev team may have objections to not using raw pointers anymore.
        - (Looking at PR now)
        - David: Actually, if we're passing and listing it as a variant. There are "0" changes to the pointer call encoding.

2024-05-29

- Attendees:
    - David Snopek
    - aaronfranke
    - raulsntos
    - Hunt Sparra
    - Zylann
- General discussion topics:
- Issues/PRs folks specifically want to review:

- dsnopek: Bind compatibility GDExtension methods removed in #88418
  - PR: https://github.com/godotengine/godot/pull/91502
  - Adds compatibility methods for removed methods - we decided this was unnecessary because they were totally broken
  - Should we merge this anyway? The PR is good
    - Believed that nobody was using methods, but the Rust bindings for Godot have an option to load ALL methods on launch, which was causing a panic for the missing methods (even if they were unused).
    - Aaron was in favor of merging to maintain compatibility, even if nobody was reasonably using the methods.
    - Hunt agreed.
    - David commented that it was good to merge for compatibility, but this is not a change in overall policy.
- Flamelizard: (godot-docs) .gdextension file options
  - PR: https://github.com/godotengine/godot-docs/pull/9383
- aaronfranke: Allow creating GDExtension plugins from inside the Godot editor #90979
  - PR: https://github.com/godotengine/godot/pull/90979
  - Can you create the extension outside of res://addons?
    - Yes. res://addons is just the (current) default. Do people prefer a different default?
    - David: Some workflows put the GDExtension one level above the Godot project.
    - Aaron: Is res://.. even supported? Does that even make sense to manage code *outside* the project when the tool looks like it is for *your* project?
    - David: I think we need a wider discussion on what project structure we want to promote.
    - Aaron: The path I am promoting is very similar to the Asset Library (import to addons), but it uses a bin folder instead of separate platform folders.
  - Future support for listing NOT-loaded GDExtensions
    - David: GDExtension only knows of loaded extensions currently. We're looking at this for the (unspecified) future
    - Aaron: With how the code is written, it should be doable to switch the listing logic in the future.
  - In addition to the scons per extension, the tool also creates a top-level scons that will calls the scons for all extensions
    - David: Will this work with multiple godot-cpp directories?
    - Aaron: The code current clones godot-cpp per extension, so it will work. There is probably a way to work around it, but scons complained and wouldn't build when godot-cpp was shared between projects.

                - David: I don't know if it is a good idea to share godot-cpp, even if compiling it multiple times is wasteful. There could be problems when upgrading godot-cpp but not maintaining the extension's source code.
- Godot 4.4 goals (feature freeze in ~4 months [maybe])
    - GDExtension "loaders"
        - PR: https://github.com/godotengine/godot/pull/91166
    - LibGodot
        - PR: https://github.com/godotengine/godot/pull/90510
        - Hopefully will be updated to use GDExtension loader soon
    - Fix NOTIFICATION_POSTINITIALIZE and godot-cpp object creation
        - PR: https://github.com/godotengine/godot/pull/91018
        - PR: https://github.com/godotengine/godot-cpp/pull/1446
    - Inter-GDExtension communication
        - One GDExtension generating wrappers for classes in another GDExtension
            - David: May or may not already work…? Still need to figure out the "official" way to do this.
        - One GDExtension extending classes defined in another GDExtension
            - Including overriding virtual methods defined in the parent class
        - Related to .NET support
    - Ability to enable/disable GDExtensions in project settings
        - David: Should be easy, but I haven't looked at it yet.
        - This is important for when changes unintentionally break GDExtension compatibility (e.g., Godot Jolt had to add code to conditionally load based on the Godot version)
    - Editor support for creating/updating source code for GDExtensions
        - https://github.com/godotengine/godot/pull/90979
        - https://github.com/godotengine/godot-proposals/issues/9806
        - Useful for both distributing extensions as assets and for implementing your gameplay logic in C++
    - Singleton registration order
        - Latest case: https://github.com/godotengine/godot-proposals/issues/9134
        - Hunt says he is looking into it for OS, Engine, Time, and ProjectSettings
        - Previous workarounds (https://github.com/godotengine/godot/pull/74758) were rejected because, philosophically, we should be registering singletons/methods/etc through ClassDB
- Shaders
    - Zylann: An example use case would be a terrain shader.
    - Aaron: 3% of the Godot binary is just visual shader. We could save this by compiling the shaders at Editor time instead of runtime.

2024-05-17

- Attendees:
    - David Snopek
    - vnen
- Issues/PRs folks specifically want to review:
    - dsnopek: Implement GDExtensionLoader concept
        - PR: https://github.com/godotengine/godot/pull/91166
        - Vnen will take a look at the PR when he has a chance
    - dsnopek: (godot-cpp) Fix NOTIFICATION_POSTINITIALIZE sent twice
        - PR: https://github.com/godotengine/godot-cpp/pull/1447
        - Vnen reviewed and approved it
    - dsnopek: Change how NOTIFICATION_POSTINITIALIZE is handled
        - Godot PRs:
            - https://github.com/godotengine/godot/pull/91018
            - https://github.com/godotengine/godot/pull/91019
        - Would need updates to bindings (so a Godot 4.4 thing)
        - Talked through it - makes sense
    - dsnopek: (godot-cpp) Setup instance binding in constructor
        - PR: https://github.com/godotengine/godot-cpp/pull/1446
        - Vnen also tried to solve this in the past - will take a look when he has a chance

2024-05-01

- Attendees:
    - David Snopek
    - Zylann
    - Raul Santos
    - Hunt Sparra
- General discussion topics:
    - Add `RequiredPtr<T>` to mark `Object *` arguments and return values as required
- Issues/PRs folks specifically want to review:
    - dsnopek: Implement GDExtensionLoader concept
        - PR: https://github.com/godotengine/godot/pull/91166
    - dsnopek: (godot-cpp) Fix NOTIFICATION_POSTINITIALIZE sent twice
        - PR: https://github.com/godotengine/godot-cpp/pull/1447
    - dsnopek: Change how NOTIFICATION_POSTINITIALIZE is handled
        - Godot PRs:
            - https://github.com/godotengine/godot/pull/91018
            - https://github.com/godotengine/godot/pull/91019
        - Would need updates to bindings

    - dsnopek: (godot-cpp) Setup instance binding in constructor
        - PR: https://github.com/godotengine/godot-cpp/pull/1446
        -
- Action Items
    - Hunt: Will make a draft PR for singletons to move discussion forward
-

2024-04-19

- Attendees:
    - Dsnopek
    - Adam Scott
    - Hunt Sparra
    - Gergely Kis (kisg)
    - George (vnen)
    - Thaddeus Crews (Repiteo)
    - Ughuuu(Dragos)(added after start)
- General discussion topics:
    - kisg: Migeran LibGodot PR: https://github.com/godotengine/godot/pull/90510
        - Easy to use test setup: https://github.com/migeran/libgodot_project
        - Short presentation by Gergely Kis (@kisg)
            - To be able to use Godot as a library
            - GDExtension initialization function
                - Initialization function will be called during the Godot core initialization
                    - Turn an object pointer into a binding
            - Godot instance object
                - Main setup of Godot
                - Moved the shutdown part outside of the Godot instance object, as it was creating memory leaks
            - Godot can run on the main thread of the process or in a background thread
                - Makes it so that the UI can be responsive even if Godot is slow
            - GDExtension code as much as possible in order to limit the number of languages needed to support
        - Discussion of any questions by attendees
            - Adam
                - How does the GDExtension is used?
            - kisg
                - If you link Godot as a library and load it either static/dynamic into a host process, you can initialize a

GDExtension that is not a separate library, but initialized in the host process
    - Makes the whole GDExtension much more powerful
        - Lots of advantages
        - We can do really good things such as embedding Godot inside Swift/Qt applications
        - We can also embed it into Blender
        - .NET can host Godot too, will be used with the Godot.NET GDExtension Raul is developing
- Dsnopek
    - The fact that GDExtension can does this is super nice
        - People were asking for embedding Godot all the time
        - Having GDExtension being the way to access functions of Godot as the API we maintain already is a super way to do this
- Kisg
    - Thank you David for reviewing the PR
- George
    - Can you embed Godot inside Godot?
- Kisg
    - It should be possible
    - We haven't tried it yet, but it should be possible
    - The only thing that you need to make sure is that the symbols don't clash during linking time
        - macOS, there could be a problem
        - On Linux, you can isolate shared libraries to make them not clash
            - Hiding the symbols from the main process
        - But on macOS, I need to check (Note after the meeting: checked, and the MacOS linker does not have this option)
- Dsnopek
    - What would be the usecase of running Godot inside Godot?
- George
    - There's been talks about running a Godot game inside the editor
- Kisg
    - We need to take a look at that
- Dsnopek
    - Did you want to run through some of the issues, elements of the PR?
- Kisg

    - That's all for me, I brought everything that I wanted to discuss
    - One thing was about keeping the initialization object as a GDExtension object or not?
- Dsnopek
    - As long that we have total control of that class, that people don't use the class other than we thought, it would be alright
    - George, do you have any comments about this?
- George
    - I'm not sure, TBH.
    - If it's a bound class, I'm just thinking about, it's normally accessible to scripting, and we don't want
- Dsnopek
    - This is a little bit different, here, you can do actions that you want
        - Only useful from GDExtension
    - I'm open for open bound classes
- Thaddeus
    - We could mark this as experimental
    - We don't have to be locked in as much
    - Not putting all our eggs in the same basket could be nice
- Dsnopek
    - Maybe we should have a tag for GDExtension that if you just doing scripting, you should never use this.
- Kisg
    - Maybe we could extend the whole registration API
        - Going that destination instead of doing a GDExtension element
    - We could create sandboxes and lockdowns
        - To prevent untrusted code to access API
    - Lua extension already exist to access Lua vetted stuff instead of the whole GDExtension API
- Dsnopek
    - Sandboxing is always useful
    - But that's beyond the whole GDExtension teams purview
- Adam
    - I think that we shouldn't think about sandboxing yet
        - To think more about the developers instead of modders
- Kisg
    - I fully agree, and I agree that we should mark this code as experimental
    - Then, other question, is `libgodot.h` fine?

- Dsnopek
    - For now, it's super fine. We gonna have this discussion in the PR.
    - GDExtension loader concept has been begun with Raul
        - We may be able to take advantage of Raul's work for this PR
- Kisg
    - I'll reach out to him for that

- dsnopek: Generate gdextension_interface.h from a machine-readable format?
    - Proposal discussion: https://github.com/godotengine/godot-proposals/discussions/9524
        - Dsnopek
            - Good transition from kisg PR
            - Idea would be to have a XML/JSON to define the extension interface and generate the header file from it
            - Other languages do that
            - Would it be a good idea?
        - Thaddeus
            - Great idea
            - We already output JSON files for API bindings
        - Adam
            - I suggest XML rather than JSON, JSON is much more difficult to work with
            - Either case, we need a schema for both files
        - Thaddeus
            - How about putting the bindings inside the actual extension interface?
        - Dsnopek
            - Combining the extension interface with the binding JSONs?
        - Thaddeus
            - Would it be possible? Is this worth it?
        - Kisg
            - A single file generated from a Godot instance, could be super cool
        - Thaddeus
            - I'm gonna say that a single file would be a benefit
            - We are spread-thin a little bit too much
            - Wouldn't we want to merge these files inside one file, a single command for a single file
        - Dsnopek
            - For bindings that need all, that could be nice
            - But C/C++ projects really need a header file

        - In a large number of cases, folks would want to grab the headers
    - Thaddeus
        - I would agree that C/C++ projects would need that header file
        - But other projects don't have the ease of access that C/C++ projects have
        - With that file, projects could generate their files, even C/C++
    - Kisg
        - If you cannot generate C/C++ header file from the XML/JSON file, it means that something is missing in the latter
    - Adam
        - As stated by timothyqiu, Vulkan uses XML. JSON would be a mess, reading wise and size wise.
    - kisg
        - I don't have any objection for XML
    - Adam
        - So, it's pretty much agreed that's a good idea
    - Hunt Sparra
        - Should we document this with a proper proposal or just a discussion is fine?
    - Dsnopek
        - Yes, if someone wants to fill out a proposal, could be nice
    - Adam
        - I don't think it's that urgent
    - Dsnopek
        - Somebody can just add a task to do this
- Issues/PRs folks specifically want to review:
    - Ughuuu: Draft PR https://github.com/godotengine/godot-cpp/pull/1415 related to godot-compat layer:
        - Idea is to be able to compile both using godot and godot-cpp. Added some python files, 1 you would run in godot, 1 you would run in godot-cpp using the output of the first one in godot.
        - Wanted to see the general idea of people of current use of it, eg here you can see an example of WIP of it: https://github.com/V-Sekai/godot-speech/pull/10/files
        - Discussion
            - Ughuuu
                - I want to have files in another folder called "godot-compat" that would have includes for targets for addons and modules
                - Initially it would run for Godot and generate a file

    - Only bad thing is that you need to generate this from Godot
    - So I created a workflow to clone Godot and generate bindings from it automatically
    - If you're targeting modules, you would have to link these files, otherwise, you would have to link to Godot
    - Idea to commit this output header mapping
- Adam
    - You want to be able to target a module or a GDExtension, right?
- Ughuuu
    - Yes, my work would take care of either of the builds
    - Godot-speech really wants to be a module, but with my work, I can build it for a GDExtension
- Dsnopek
    - There's demand for this
    - About keeping the generated compat file, could be something we do
    - People would know how to replace the file if needed
    - Same thing that I said, this approach makes you need to have godot-cpp to build a module, even if you don't compile a module
        - Seems a little bit backwards to me
- Ughuuu
    - But if you target godot-cpp, you need godot-cpp
- Dsnopek
    - Yes, but the reverse use-case is nice too, when you're building a gdextension that needs to build a module
    - You could use a `-Igen/include/godot_compat for a search path not to pollute the whole directory
        - That would be my preference
- Kisg
    - One question: would it be possible to include the header files with some clever pre-process with ClassDB?
        - That we can actually include with the XML generated?
        - That we could use this information to generate compatibility headers
- DSnopek
    - Would be certainly possible
    - GDClass macro could create some stuff
    - I don't know how open the engine core developers would be to this

        - Always be a resistance to make things into API that we need to maintain
- Ughuuu
    - Not only about GDClasses, it's about other things too
- DSnopek
    - This approach seems nice to me
    - If this were to become the preferred way to do
    - But as a way to become a way to do in C, this is a good way to go
- Ughuuu
    - One counter argument is that people shouldn't develop modules, but should develop addons
    - It depends if you see this as an intermediate step for addons or if you are seeing this as a way to port existing modules
    - Godot-cpp first, then Godot modules afterwards
    - For this, I target foremost people that develop add ons first
- Adam
    - What's the use-case?
        - To port modules?
        - If so, limited use-case, if any
- George
    - Could be broader than to just port modules to GDExtension
    - Speed advantages to have a module instead of a GDExtension
- Ughuuu
    - Could be size advantages
- DSnopek
    - Could be useful for platforms that don't support GDExtensions, such as consoles
- Kisg
    - We can statically compile gdextensions with libgodot
    - That you can also bundle any gdextensions in a single binary
- DSnopek
    - There is some demand for this
        - i.e. zylann
    - The demand exist
- Kisg
    - Could be nice for platforms that need more juice, i.e. physics engines

2024-04-03

- Attendees:
    - Dsnopek
    - Adam Scott
    - Hunt Sparra
- General discussion topics:
    - Singletons not registered early enough
        - Maybe add OS to it
- Issues/PRs folks specifically want to review:
    - dsnopek: Passing `null` for `Object *` argument to method crashes with GDExtensions using godot-cpp
        - https://github.com/godotengine/godot/pull/87613
        - Adam will ping Reduz next week for his feedback
        - David will post a comment saying that we'll continue collecting feedback until the next GDExtension meeting, and if no other new feedback comes, we'll go with the godot-cpp PR
    -
    -

2024-03-06

- Attendees:
    - Dsnopek
    - Riteo
    - Adam Scott
    - Rune
    - Zylann
- General discussion topics:
    - Singletons not registered early enough
        - Latest case: https://github.com/godotengine/godot-proposals/issues/9134
        - **(Re-)discuss if stakeholders or interested folks turn up**
        - Possible solutions:
            - Registering the `get_singleton()` functions rather than the objects, and allowing GDExtension to call those
            - Moving current registration code to happen earlier
            - Register each singleton at the point that it is ready to be used (will need to think through each singleton)
            - Define a set of core singletons that are safe to use early and register them earlier

            - For example: Engine, Time, maybe ProjectSettings if it doesn't have issues
        - Adam Scott: Add project setting to not create window and allow project to create it manually later
    -
- Issues/PRs folks specifically want to review:
    - dsnopek: Passing `null` for `Object *` argument to method crashes with GDExtensions using godot-cpp
        - Change `Object *` encoding to not pass `nullptr` (but pointer to `nullptr`) in Godot
            - https://github.com/godotengine/godot/pull/87613
        - Alternative fix in godot-cpp:
            - https://github.com/godotengine/godot-cpp/pull/1405
        - **Would be great to get Bromeon's input on this!**
        - **Would be great to talk with George and Juan and see if they remember the intention for the encoding of `Object *`'s**
    -
-

2024-02-23

- Attendees:
    - Dsnopek
    - Vnen
    - Gallilus
    - Ughuuu
    -
- General discussion topics:
    - ughuuu: Module vs Addon build. Can they be somehow improved for people who want to build both?
        - Maybe a new project could be created to try and mimic the Godot includes?
        - Maybe include everything in one big header?
            - Could have issues with size - needs testing
            - However, explicit includes is usually preferred
        - Maybe add a secondary set of headers that match Godot's headers in godot-cpp
            - Ughuuu: will work on a proposal/PR
    - ughuuu: Addon build size(eg. including godot-cpp in all builds).
        - Fabio's PR for build profiles could maybe help:
            - https://github.com/godotengine/godot-cpp/pull/1167
        - Vnen: could scons dependency checker help?

            - Binding generator could still generate everything, but scons could only build the stuff that's included
    - Singletons not registered early enough
        - Latest case: https://github.com/godotengine/godot-proposals/issues/9134
        - Vnen / dsnopek: Think we should register the singletons individually when each is ready to be used
    -
- Issues/PRs folks specifically want to review:
    - ughuuu: Add reusable action to build, also add mac sign action.
        - PR: https://github.com/godotengine/godot-cpp-template/pull/23
    - dsnopek: [godot-cpp] Fix _notification() with parent and child classes
        - https://github.com/godotengine/godot-cpp/pull/1381
    - dsnopek: Add `RequiredPtr<T>` to mark `Object *` arguments and return values as required
        - Draft PR for discussion: https://github.com/godotengine/godot/pull/86079
    - Add renaming of PDB files to avoid blocking them:
        - https://github.com/godotengine/godot/pull/87117
    - Fix loading GDExtension dependencies on Android:
        - https://github.com/godotengine/godot/pull/88381
    -

2024-01-26

- Attendees:
    - Dsnopek
    - Adam Scott
    - Vnen
    - Bromeon
    - Gallilus
    - Allenwp
    - Mihe
    - Helehex
- Issues/PRs folks specifically want to review:
    - dsnopek: Remove binding for GDExtension::initialize_library()?
        - From docs PR: https://github.com/godotengine/godot/pull/86968#discussion_r1445490574
        - **Let's "unbind" initialize_library() and open_library()**
            - **Maybe not even needing compatibility methods?**
            - **Dsnopek will make PR**
    - Bromeon: errors with double builds
        - Issue: https://github.com/godotengine/godot/issues/86346
        - **Let's check if there's a difference between 64 vs 32 bit builds too**
        - **Dsnopek will make a PR later to fix the gdextension_compat_hashes.cpp bug**

- dsnopek: The size of enums?
  - Original issue: https://github.com/godotengine/godot/issues/85444
  - Short-term godot-cpp fix: https://github.com/godotengine/godot-cpp/pull/1320
  - **Need a way to expand the GDREGISTER_NATIVE_STRUCT() system to store:**
    - **Storing per field:**
      - **Types (sizes): uint32_t**
        - **Could be stored like "meta" on arguments in the extension_api.json**
        - **Should check to make sure it's an "allowed" type**
      - **Types -> what they should be treated as: TextServer::Direction enum**
      - **Field name**
      - **Offset of field into struct**
        - **Should probably represent this in a similar way to the variant offsets in extension_api.json**
  - **Should check the size of structs doesn't change via CI**
  - **Maybe some overlap with GDScript structs - how can we make this system compatible?**
- dsnopek: Gameplay classes
  - PR: https://github.com/godotengine/godot/pull/82554
- dsnopek: Add typehints to `binding_generator.py`
  - PR: https://github.com/godotengine/godot-cpp/pull/1365
  - Do we want to do this?
-

2024-01-10

- Attendees:
  - Dsnopek
  - Bastiaan
  - Adam Scott
  - Ansraer
  - Aaron Franke
- General discussion topics:
  - Adam Scott: Template documentation
    - https://github.com/godotengine/godot-proposals/issues/8594
  - Adam Scott: godot-cpp template command line app?
    - Juan brought up doing a similar thing in the editor itself
  - Adam Scott: godot-jolt impact for GDExtension if it's integrated as a module

        - Bastiaan:
            - Going to a module would be a step back
            - Would mean that one of the biggest GDExtension implementations (poster child) would no longer exist. Currently, puts pressure on Godot to improve GDExtension and managing GDExtenions, which we lose. If using a GDExtension is too difficult, solve those difficulties.
                - Ex: we could use dependencies and versioning in GDExt
            - We should be promoting *more* things moving into plugins, rather than moving into core
            - GDExtensions can be agile and focused and don't depend on Godot release cycle
            - GDExtension can provide multiple solutions to a problem for different needs
                - For example, marrs making a physics engine just for driving simulation
            - Perhaps we should have no physics engine, get a message when using a physics node, and given offer to install one of many physics engines
        - Aaron Franke:
            - Agree, but need to solve some problems:
            - 1. Make sure we have a system to recommend and curate extensions
                - https://github.com/godotengine/godot-proposals/issues/7231
                - https://github.com/godotengine/godot-proposals/issues/8114
            - 2. If we find problems with GDExtension and are tempted to make a module to avoid them, no, we should fix the problems
                - https://github.com/Zylann/godot_voxel/issues/442

2023-12-01

- Attendees:
    - Dsnopek
    - Adam Scott
    - Bromeon
    - Vnen
    - Ansraer
- Issues/PRs folks specifically want to review:
    - dsnopek: Tracking which arguments and return values are nullable in ClassDB (and put that into extension_api.json)
        - https://github.com/godotengine/godot-proposals/issues/2241
        - Option<T> may be tricky because it would require adding it everywhere

        - Something progressive would be better
            - Maybe starting with metadata only?
        - GDScript team is working on core proposal for unified type system
            - Planning to work on “nullable”
            - From Danil Alexeev: https://gist.github.com/dalexeev/bd4e952a601b18ae3a94919f89b98928
        - **Next steps:**
            - Wait to see unified type system proposal from GDScript team
            - [dsnopek] Make experimental Required<T> to see how that looks
    - Yuri: Crashes with GDExtensions in newer versions
        - Assetlib upgrade mechanism?
            - Maybe put info about incompatible versions in assetlib (so Godot can call home and check it)?
                - Opt in setting because Godot presently doesn’t call home (and people like this)
            - Maybe a URL in plugin.cfg that can be called to see updates?
                - Maybe the URL is a fixed end-point on assetlib or the new asset store
        - **(1) Safe mode?**
            - **Vnen: We think this is a good idea! Can store crash from crash handler and on next launch detect it, and offer safe mode.**
            - Dsnopek: maybe detect last version project was saved with (in project.godot file) and if new version offer to open in safe mode?
        - **(3) compatibility_maximum key in .gdextension?**
            - Dsnopek: might not help too much in practice, but it’s easy to implement, so we can add it to have another possible tool
            - Adam: Or record target version and show message when loaded in a newer version
            - Vnen: UX concerns about how it shows this warning, and hiding it if they are good
        - **(2) Some way to disable/enable GDExtensions**
            - Show warning when loading with newer Godot, and allow disabling some
        - hard-code hack into the editor for godot-jolt?
            - We aren’t crazy about this idea, but perhaps it’s the best quick solution for 4.2.1?
        - **Next steps in priority order highlighted in bold**
    - dsnopek: GDScript singleton cache with GDExtension hot reload:
        - https://github.com/godotengine/godot/pull/85373
        - George thinks it looks good

2023-11-15

- Attendees:
    - Dsnopek
    - Bastiaan Olij
    - Mai
    - Ryanabx
- Discussion topics:
    - Goals and priorities for Godot 4.3?
        - Gameplay classes
        - Virtual methods
        - Initialization order and singletons
            - At different initialization levels, some classes are registered or not, and/or the singleton is registered or not
            - A previous attempts/discussions:
                - https://github.com/godotengine/godot/pull/74758
                - https://github.com/godotengine/godot/pull/79584
            - Bastiaan's proposals:
                - Provide access to real ::get_singleton() rather than the current singleton registration
                - Rename "initialization" levels to "registration" and then provide another callback to actually initialize
            - Start discussing and making proposals/PRs during 4.3 cycle
- Issues/PRs folks specifically want to review:
    - dsnopek: Custom callable issues
        - https://github.com/godotengine/godot-cpp/pull/1280
        - https://github.com/godotengine/godot-cpp/pull/1294
    - dsnopek: Objects does not dispatch NOTIFICATION_POSTINITIALIZE
        - https://github.com/godotengine/godot-cpp/issues/1269
-

2023-10-18

- Attendees:
    - Dsnopek
    - Fabio
    - Riteo
    - Aaron Franke
    - Zylann
    - Mai
    - Adam Scott
- Discussion topics:

    - Riteo: (perhaps) see what's left for the documentation stuff?
        - Need to figure out clean up
    - Deficiencies that Mai found
        - non-code dependencies
            - some can go in pck
            - some cannot
            - same may be in project dir and committed
            - some may not
        - extension paths
            - Project
            - Editor
            - Export
            - .godot
            - cache
        - excludes / includes
        - dependency platform versions
        - singleton initialization order
        -
        - original argv lost
        - executable path
        - project path
        -
        - icon and doc install
            - Runtime?
            - static configs are a little problematic
        - initialization order is a mess
            - core, scene... seem to have no real meaning
- Issues/PRs folks specifically want to review:
    - aaronfranke: Allow extending CanvasItem in GDExtension #67510
        - Stuck on clarification from Juan
        - Could be a potential PR for the next core meeting that Adam is hoping to schedule
    -

2023-10-06

- Attendees:
    - Dsnopek
    - Gallilus
    - Bromeon
    - Ryanabx
    - Rahul
    - Vnen
    -

- General discussion topics:
    - Left over things on the Godot 4.2 roadmap:
        - https://docs.google.com/document/d/1Bx3mPxQsrwWGS2r6Q9mIt6r2QCtriwXJNia5Lw517X0/edit#heading=h.893krabv6qpc
    - Planning for Godot 4.3: goals and aspirations!
        - “Gameplay classes” aka non-@tool classes
        - Defining virtual methods from GDExtension classes that can be implemented in GDScript
        - Add Engine method to allow registering class reference data
            - Remi’s old PR: https://github.com/godotengine/godot/pull/75415
        - Improving documentation
            - Starter project and tutorial (using the template)
            - How to support many platforms
            - How to expose things to GDExtension (for contributors)
            - For C++ folks: how to get started with GDExtension (guide)
            - For GDScript folks (C++ newbies): how to get started with GDExtension (tutorial)
            - Backwards compatibility (for contributors)
        - Exposing more editor classes to allow GDExtensions to do more in editor plugins
        - Improve web export (document versions and tools)
- PRs/Issues discussed:
    - Documentation (signatures) not appearing in Godot help:
        - https://github.com/godotengine/godot/issues/82817
        - https://github.com/godotengine/godot/issues/76796
    - dsnopek: Fixes to allow object-less callables throughout Godot
        - https://github.com/godotengine/godot/pull/82695
    - dsnopek: GDExtension: Convert validated_call() to ptrcall() (rather than call())
        - https://github.com/godotengine/godot/pull/82794
    - dsnopek: [DRAFT] Registering “gameplay classes” (aka like non-@tool scripts):
        - Godot: https://github.com/godotengine/godot/pull/82554
        - godot-cpp: https://github.com/godotengine/godot-cpp/pull/1256
    - Web: Catch using GDExtensions in a non-dlink build
        - https://github.com/godotengine/godot/pull/82790
    - NativeStructs and Pointers in C#
        - https://github.com/godotengine/godot/pull/81004
-

2023-09-20

- Attendees:
    - Dsnopek
    - Bastiaan
    - Riteo

    - Adam Scott
    - Zylann
    - Mai
    - Ryan Brue
    - Max Hilbrunner
    - Lyuma
    -
- General discussion topics:
    -
- PRs discussed:
    - dsnopek: MethodBind::get_hash() fails to hash default arguments
        - https://github.com/godotengine/godot/pull/81521
    - dsnopek: Implement reloading of GDExtensions
        - Godot: https://github.com/godotengine/godot/pull/80284
        - Godot-cpp: https://github.com/godotengine/godot-cpp/pull/1200
    - adamscott: Add support to import custom variables from parent SConstruct (redux)
        - https://github.com/godotengine/godot-cpp/pull/1220
    - adamscott: Refactor compiledb implementation
        - https://github.com/godotengine/godot-cpp/pull/1230
    - dsnopek: [godot-cpp] Fix variant call compiler error
        - https://github.com/godotengine/godot-cpp/pull/1238
    - dsnopek: [godot-cpp] Handle missing instance binding callbacks by finding the closest parent
        - https://github.com/godotengine/godot-cpp/pull/1165
    - dsnopek: [godot-cpp] Implement `callable_mp()` and `callable_mp_static()`
        - https://github.com/godotengine/godot-cpp/pull/1155

2023-09-08

- Attendees:
    - Dsnopek
    - Bromeon
    - Paddie.exe
    - Pcting
    - Marcel
- PRs discussed:
    - dsnopek: [godot-cpp] Check that GDExtension is opened by compatible Godot version
        - https://github.com/godotengine/godot-cpp/pull/1208
    - dsnopek: [godot-cpp] Ensure that PtrToArg specializations for native structs are used
        - https://github.com/godotengine/godot-cpp/pull/1214
    - dsnopek: Add functions for non-ptr style virtual calls in GDExtension

        - https://github.com/godotengine/godot/pull/80671
    - dsnopek: MethodBind::get_hash() fails to hash default arguments
        - https://github.com/godotengine/godot/issues/81386
-

2023-08-23

- Attendees:
    - Dsnopek
    - Mai
    - Adam Scott
    - Fredia
    - Zylann
    - Gallilus
    - Riteo
    - Vnen
- General discussion topics:
    - Make a list of issues with GDExtension that we could fix in Godot 5 when we can wipe everything clean
        - Let's keep it out of GitHub to not send the wrong message publicly
    - Further Godot 4.2 roadmap discussion?
        - September is the last month of feature development!
    - dsnopek: For merging PRs into godot-cpp, how much approval is "consensus"?
        - At the last meeting, we decided on this rule: "Two team members approval (unless controversial and then we'll discuss at a meeting)"
        - It may be a little too restrictive?
            - Perhaps:
                - If the PR is authored by a team member (and it's non-controversial) only one team member approval is enough?
                - Or, maybe the merger (ie probably dsnopek) can have more discretion when it comes to "simple" or "small" changes?
            - **Meeting: one trusted (including team member) review is enough for non-controversial things**
- PRs:
    - dsnopek: Set varrarg methods' ptrcall of builtin classes, and let them can be called without arguments
        - https://github.com/godotengine/godot/pull/76047
        - Required for the vararg methods to be added to godot-cpp (approved at a previous meeting)
            - https://github.com/godotengine/godot-cpp/pull/1091
        - **Meeting: vnen will review it**
    - Remi Fasolasido : Status on Virtual functions (godot-cpp)

        - Example. Godot_cpp class with virtual functions. GDScript override those functions
        - Must be able to call the overridden functions from both C++ and Gdscript.
        - If already possible, at least a documented example with the correct binding.
        - See:
            - https://github.com/godotengine/godot-cpp/issues/1199
            - https://github.com/godotengine/godot-cpp/issues/1072
            - https://github.com/godotengine/godot-cpp/issues/910
        - **Meeting: the way forward is adding the GDVIRTUAL*() and GDVIRTUAL_CALL() macros to godot-cpp**
    - dsnopek: Fix Object::notification() order
        - https://github.com/godotengine/godot/pull/78634
    - Fredia: Godot Android plugin re-architecture
        - https://github.com/godotengine/godot/pull/80740
    -

2023-08-11

- Attendees:
    - Dsnopek
    - Bromeon
    - Vnen
    - Gallilus
    - Jeff Ward
    - Adam Scott
- General discussion topics:
    - Managing 4.1 branch via cherry picking
        - First cherry-pick PR: https://github.com/godotengine/godot-cpp/pull/1205
        - **Meeting: Let's do it!**
    - Doing our own merges on godot-cpp
        - Or hand back to the production team?
            - **Meeting: Let's keep doing it**
        - How much approval constitutes a consensus?
            - **Meeting: Two team members approval (unless controversial and then we'll discuss at a meeting)**
    - Maintaining GDNative / godot-cpp for 3.x: should we do it? How do we do it?
        - Recent PR adding some tests: https://github.com/godotengine/godot-cpp/pull/1169
        - **Meeting: Pick a "line" and say "these kind of things we merge, and these others we don't do for 3.x anymore"**
            - Provisionally: "Due to limited resources currently the GDExtension team is only able to maintain 3.x bug fixes"
- PRs:

    - Bromeon
        - StringName construction from char* https://github.com/godotengine/godot/pull/78580
        - Avoid unneeded ptrcall copies https://github.com/godotengine/godot/pull/80075
    - dsnopek: Zylann's GODOT_VERSION_* macros PR
        - https://github.com/godotengine/godot-cpp/pull/1193
    - vnen:
        - Copy DLL to a temp file before opening https://github.com/godotengine/godot/pull/80188
    - dsnopek: Calling virtual method implementations in GDExtension is geared to C++
        - https://github.com/godotengine/godot/issues/63275
    -

2023-07-26

- Attendees:
    - Dsnopek
    - Bastiaan
    - Zylann
    - Mai
    - Adam Scott
- dsnopek: godot-cpp binaries are ~1-5mb bigger than the previously were
    - https://github.com/godotengine/godot-cpp/issues/1160
    - Mai has an interesting idea to do automatic registration of the classes in the hash map (as an alternate way to how PR #1050 does it), example:
        - // some register header
        -
        - template<Class>
        - class Register {
        - Register(class_name) {
        - hash_map.insert({class_name, type_info(Class)});
        - }
        - };
        -
        - // node.hpp
        -
        - static inline Register<Node>("Node");
- Zylann: It's worth investigating if the changes in PR #1050 are thread-safe
- dsnopek: Expose methods from ClassDB singleton
    - Original PR: https://github.com/godotengine/godot-cpp/pull/936
    - Alternative PR: https://github.com/godotengine/godot-cpp/pull/1164
    - Discuss the general principle:

        - In godot-cpp only (not other bindings), should we generate code for some singletons (like ClassDB, ResourceLoader, etc) to use static methods for source compatibility with Godot modules?
        - Dsnopek: I think godot-cpp is a special case, where we want to be able to easily move code in/out of Godot itself.
        - **Yes, trying to make godot-cpp's API match Godot's "module API" is a design goal of godot-cpp**
- dsnopek: Handle missing instance binding callbacks by finding the closest parent
    - https://github.com/godotengine/godot-cpp/pull/1165
    - It depends on exposing methods from the ClassDB singleton per the previous point
- dsnopek: Allow resizing String's from GDExtension
    - https://github.com/godotengine/godot/pull/79156
- Further Godot 4.2 roadmap discussion?
    - Hashing out how to expose documentation from a GDExtension
- Bastiaan: Singletons in GDExtension - can we get them earlier?
    - Dsnopek's draft re-ordering PR: https://github.com/godotengine/godot/pull/79584
    - **Next steps:** schedule a meeting with folks from the core team to discuss!

2023-07-14

- Attendees:
    - dsnopek
    - Adam Scott
    - Repiteo
    - Galilus
    - Paddie.exe
    - Vnen
- Godot 4.2 roadmap discussion
    - Fixing bugs
        - Want to fix remaining major bugs to see if we'll need to break compatibility again
    - Documentation
        - How to interact with the different systems?
        - What is extendable? How to make something extendable? (the different ways to do it)
    - Improving editor integration from GDExtension
        - Add the ability to draw custom gizmos from GDExtension? (mihe brought up on physics team meeting)
- Issues/PRs discussed:
    - dsnopek: TileMap issue - registering a compatibility method for `const` change (Yuri asked for advice to unblock Groud)
        - https://github.com/godotengine/godot/pull/78328
        - What I told Yuri:

                - “I think you're right that it needs a compatibility method because const-ness is included the hash. I could definitely see a valid argument for not including const-ness in the hash, but making that changes would break compatibility for a huge number of methods, and at this point I don't know that it's worth doing. If we end up in a situation like in 4.1 where we had to break compatibility anyway to fix a critical bug, it could make sense to throw in the hash change too. But until that happens, I'd personally prefer to maintain compatibility.”
            - Do you agree/disagree?
                - Agree
                - Vnen: const makes sense on the hash, sets expectations
    - dsnopek: Allow CallableCustom objects to be created from GDExtensions
        - Godot: https://github.com/godotengine/godot/pull/79005
        - godot-cpp: https://github.com/godotengine/godot-cpp/pull/1155
        - Need some input from folks who maintain other bindings
    - dsnopek: Implement vararg methods of builtin classes
        - https://github.com/godotengine/godot-cpp/pull/1091
    - dsnopek: Full CharString implementation
        - https://github.com/godotengine/godot-cpp/pull/1150
    - Gallilus: add GDExtensionScriptInstanceGetClassCategory
        - https://github.com/godotengine/godot/pull/78995
    - adamscott: Quick PRs about the build
        - https://github.com/godotengine/godot-cpp/pull/1170

2023-06-28

- Attendees:
    - dsnopek
    - Mai
    - Bastiaan
    - Fabio
    - Raul Santos
    - Adam Scott
    - Lyuma
    - Zylann
-
- Issues/PRs discussed:
    - dsnopek: Improvements to GDExtension type safety (PR by Mai)
        - https://github.com/godotengine/godot/pull/78781
    - Fabio: Build system/toolchain improvements galore: https://github.com/godotengine/godot-cpp/pull/1147 (tiny compared to what I’d like to do 😓)
        - Attempting to address two problems:

                - Being able to customize the build toolchain and override what godot-cpp autodetects, because we can't possibly detect all possible toolchains
                - Being able to build the library multiple times (like when building a universal binary for MacOS)
            - Fabio will probably close this PR, and make 1-2 PRs to cover his new ideas
-

2023-06-16

- Attendees:
    - dsnopek
    - gallilus
    - vnen
    - rburing
    - Paddie.exe
-
- Issues/PRs discussed:
    - rburing: GDExtension missing functionality to assign native method to a Callable?
        - https://github.com/godotengine/godot-cpp/issues/1089
        - Original questions/notes:
            - Would be nice to have (especially `callable_mp_static`) for physics server GDExtensions.
            - Is there any obstruction to implementing this?
        - Discussion at meeting:
            - Should be possible to implement by making a child class of CustomCallable in Godot, and then have a way for GDExtensions to make one which can callback into GDExtension
            - Dsnopek has an idea about using MethodBind* that might work - or not 🙂
    - dsnopek: Unregister custom classes in reverse registration order
        - https://github.com/godotengine/godot-cpp/pull/1047
    - dsnopek: Fix GDExtension Variant type conversion
        - https://github.com/godotengine/godot/pull/75758
    - dsnopek: Fix: Include method_ptrcall.hpp on simple structs
        - https://github.com/godotengine/godot-cpp/pull/1086
    - dsnopek: Automatically remove editor plugins when deinitializing GDExtension
        - https://github.com/godotengine/godot-cpp/pull/1138
    - Paddy-exe: 4.x Add ARCore support
        - https://github.com/godotengine/godot/pull/77559
        - Specific question: How to expose the necessary class to GDExtension (or should it be done in the first place)?

https://github.com/godotengine/godot/pull/77559#issuecomment-1585148521

2023-05-24

- Attendees:
  - Dsnopek
  - Mux213
  - Ifire
  - Bromeon
-
- Issues/PRs discussed:
  - dsnopek: Allow GDExtensions to add editor plugins
    - https://github.com/godotengine/godot/pull/77010
    - Older PR:
      - https://github.com/godotengine/godot/pull/65592
  - aaronfranke: Allow extending CanvasItem in GDExtension
    - https://github.com/godotengine/godot/pull/67510
  - dsnopek: Add automated tests that run a GDExtension (rather than just building it)
    - https://github.com/godotengine/godot-cpp/pull/1101
    - Approved at last meeting, but changes were requested, and didn't get marked as approved on GitHub
  - dsnopek: Identifiers containing double underscore are reserved according to the C++ standard
    - https://github.com/godotengine/godot-cpp/pull/1048
  - Bromeon: versioning of new GDExtension typedefs?
    - @since or @after clause with a date
      - **Discussion:**
        - Seems like a good idea! The more information we can record about the API the better
        - The current format can be machine processed, and this information can be used by the bindings
        - Let's go with @since MINOR_VERSION
    - Machine-parseable format (current one is good if it's stable)
      - **Discussion:**
        - Bromeon is processing the header for the Rust bindings
        - As long as we stick with the info in the gdextension_interface.h, let's try to keep this format consistent and machine readable

            - We should only ever change this to be *more* machine-readable, not the reverse
    - Bromeon: some more functions could benefit from Uninitialized* pointer types
        - **Next steps:**
            - Open a new issue to investigate all the GDExtension methods and see which can have their type swapped
    - dsnopek: Regressions from godot-cpp PR #1044 and PR #1045
        - https://github.com/godotengine/godot-cpp/issues/1119
        - Draft “option nr 2” implementation:
            - https://github.com/godotengine/godot/pull/77410
        - saki7: I personally vote this one to be the top priority issue which should be resolved before 4.1 is released.
    -

2023-05-19

- Attendees:
    - Dsnopek
    - Paddie.exe
    - Vnen
    - Bromeon
    - touilleMan
    - Rburing
    - Mhilbrunner
    - Adam Scott
-
- Issues/PRs discussed:
    - Bromeon: status on library-reloading?
        - https://github.com/godotengine/godot/issues/66231
        - https://github.com/godotengine/godot-cpp/issues/955 (focuses on DLL locking issue)
    - dsnopek: Ensure GDExtension class is the correct type for the Godot engine class
        - Godot: https://github.com/godotengine/godot/pull/73511
        - godot-cpp: https://github.com/godotengine/godot-cpp/pull/1050
    - dsnopek: GDExtension: Add Engine method to allow registering class reference data
        - https://github.com/godotengine/godot/pull/75415

2023-05-13

- Attendees:
    - mux213

    - vnen
    - gallilus
    - paddie.exe
    - dsnopek
- Should we do two meetings in the next two weeks (as opposed to one) since the 4.1 feature freeze is coming up?
    - **Yes!**
- Issues/PRs discussed:
    - dsnopek: Add a backwards-compatibility system for GDExtension method
        - https://github.com/godotengine/godot/pull/76446
            - **Approved!**
        - And adding some compatibility methods where they are missing:
            - https://github.com/godotengine/godot/pull/76577
                - **Approved!**
        - And adding CI to see if the compatibility has been broken:
            - https://github.com/godotengine/godot/pull/76647
                - **Changes requested, and review needed from Akien**
    - dsnopek: Add automated tests that run a GDExtension (rather than just building it)
        - https://github.com/godotengine/godot-cpp/pull/1101
            - **Approved, with minor change requested**
    - touilleMan: *_operation_index logging an error if called with an out-of-bound index
        - https://github.com/godotengine/godot/pull/66185
            - **Approved!**
    - dsnopek: Rework GDExtension interface from a struct to loading function pointers
        - https://github.com/godotengine/godot/pull/76406
        - Companion godot-cpp PR:
            - https://github.com/godotengine/godot-cpp/pull/1095
        - **Both approved!**
    - dsnopek: Change Ref<T> to allow non const access to ptr
        - https://github.com/godotengine/godot/pull/64789
        - **Discussion:**
            - Need a good use case for why this is needed
            - May be an issue in the binding_generator.py in godot-cpp rather than something that needs to be fixed on the Godot side:
                - https://github.com/godotengine/godot-cpp/issues/693

2023-04-24

- Introductions?
-

- What is the compatibility promise we are striving for?
    - **Proposed:** Old GDExtensions should work in newer Godot versions (ie. an extension made for Godot 4.1 should work in 4.2), but newer GDExtensions won't work with older Godot versions (ie. an extension made for Godot 4.2 won't work in Godot 4.1). Incompatible extensions should fail to load with a message (as opposed to crashing).
    - **Folks at the meeting seemed to be in agreement!**
- Add CI step to godot-cpp to build and run a test GDExtension so we avoid reverting/regressing old fixes?
    - See for a recent example: https://github.com/godotengine/godot-cpp/pull/1045#issuecomment-1516145062
    -
    - Regarding adding the CI improvements:
        - Existing PR from touilleman:
            - https://github.com/godotengine/godot-cpp/pull/922
        - Faless: Maybe split CI to test (a) an example, and (b) a real test?
            - (a) is more for learning
        - **Next step: dsnopek will work on a PR (possibly continuing touilleMan's work)**
    -
    - Regarding the issue in the specific example:
        - saki7: For this specific issue, I propose 2x3x2x4 test matrix: https://github.com/godotengine/godot-cpp/pull/1045#issuecomment-1520202838
          I'm not sure if this test can be done solely inside godot-cpp, or do we need a meta CI step for ensuring that godot core is properly encoding an object before passing it to GDExtension (of any language)?
        - Possibly related issues:
            - https://github.com/godotengine/godot/issues/61967
            - [GDScript] https://github.com/godotengine/godot/pull/72654
        - Could require breaking change on the Godot side
        - Faless: Maybe have the binding generator in godot-cpp generate the methods differently depending on the types?
        - **Next steps: Get input from Bastiaan on the best way forward?**
-
- Issues/PRs to discuss:
    - Bromeon: prevent compatibility breaks in JSON API – maybe a strategy going forward?
        - https://github.com/godotengine/godot/issues/75779
        - Options:
            - Revert fix and save for 4.1.0
            - Or, in GDExtension, when we encounter a float, use the older broken hash as a hack?

                - Or, decide that we broke compat and just stick with what's in 4.0.2 now?
                - Or, have a dictionary of hashes to map from old to new?
            - For the future:
                - We need the CI step to detect and prevent doing this again
                - Creating compatibility functions for when functions change
            - **Next steps: Faless will take a look at this one**
    - touilleMan: GDExtension user-friendliness by being explicit on when return value should be passed initialized
        - https://github.com/godotengine/godot/pull/35813
        - **Next steps: touilleMan will rebase and fix CI, and Bromeon test with the Rust bindings**
    -

2023-04-14

- Introductions?
- Should we start regular meetings?
    - Does this day/time work for the future?
    - Or, maybe an alternating day/time to cover more timezones?
    - Or, schedule each individually for a little while before settling into a rhythm?
        - **Next meeting we'll just do another poll!**
- What are the priorities for getting GDExtension from "beta" to "stable"?
    - Setup CI to check for compatibility breakage (checking JSON to make sure only safe things are added)
        - Perhaps set it up as a warning first so community can get used to it
    -
-
- Compatibility (ClassDB):
    - Vnen:
        - Setup CI to check for compatibility breakage (checking JSON to make sure only safe things are added)
        - Use hashes to keep deprecated functions callable when new ones are available
    - Bromeon: changes to parameter defaults?
    - Need to be able to request function via class name, function, hash
    - New functions (with changed signature) need to get a new name
        - Maybe automatically route calls to old functions given the hash
        - Requires addition to ClassDB to store "original function name" to get the right routing
    - Need to find right balance between compatibility and ability to update the engine
    - Godot 5 could just be removing the old deprecated functions
    - Properties:

        - Can’t change type or remove them
        - Figure out setters and getters later
- How to safely extend/version GDExtension ABI? Is there a mechanism already in place?
    - dsnopek: This is a selfish question because PR #73511 needs to add a function pointer to `GDExtensionInterface` :-)
    - Maybe add a void* next to do safe additions?
    - Maybe add a size field?
    - **THIS:** Maybe a “void *LoadProcAddr(const char *p_name)” function to get low-level functions instead of a big struct?
        - And Godot 5, we could get rid of the big struct 🙂
        - Add function definitions to gdextension_interface.h (and then implementations like godot-cpp would be responsible for loading the pointers for them)
        - **dsnopek will work on a draft PR for review**
-