

omigroup / omigroup

Public

[Code](#) [Issues 38](#) [Pull requests](#) [Discussions](#) [Projects 2](#) [...](#)

12/07/22 Weekly Meeting 85 Notes #323

mrmetaverse started this conversation in **General**



mrmetaverse on Dec 7, 2022

Maintainer

Weekly Meeting 85 is happening on Wednesday December 7, 2022 from 9:00 AM PST(17:00 UTC) - 10:00 AM PST (18:00 UTC)

You can add it to your calendar [here](#).

We'll be meeting again in the omi-weekly-meeting channel of the [AngellXR/OMI Discord](#).

We regularly use the # omi-meeting-chat channel for text chat.

Who we are and why we're here:

The Open Metaverse Interoperability Group (OMI) is focused on bridging virtual worlds by designing and promoting protocols for identity, social graphs, inventory, and more. Our members include businesses and individuals working towards this common goal.

If you're an observer or here for the first time, this is a meeting where people who work on the OMI group initiatives discuss updates and contribution opportunities for current projects and goals. This is your chance to jump in and help!

We welcome you to introduce yourself in #omi-meeting-chat as we dive into the agenda items. There will be an open floor in the last 15 minutes of the meeting (agenda permitting) where we can discuss any items you feel are relevant to the current or near future OMI goals.

Agenda

- Volunteer to take notes? Volunteer to record?
- Welcome newcomers

Working Group Updates:

- [MSF delegates](#) (setting up the repo)
- [UX Research](#)
- [Scripting group](#)
 - OMI developer hour*
- [glTF Extensions Group](#)
 - Progress on OMI_ref and authoring extensions

- [Media Group](#)
- [M3 avatar interop](#)
- [OMI Website update](#)
- any others?

Discussions:

- DEMO! :)
- OMC updates next week
- This is a time to build. What can we build?
 - Adding OMI to the Metaverse Street Jin demoed
 - M3 recently started a shop for wearables. This is a great example of things we could be doing and participating in.

Next week:

Demoing the OMI collider and physics body (in Godot!)

Future demo themes:

- [OMI-audio-emitter](#) exploration and demo
- [OMI-collider](#) exploration and demo (December 7!)
- [IPSME](#) and Me
- [Generative and AI tooling](#)
- Any other suggestions?

↑ 1

1 comment · 3 replies

Oldest

Newest

Top



mrmetaverse on Dec 7, 2022

Maintainer

Author

Demo time!

↑ 1

3 replies



mrmetaverse on Dec 7, 2022

Maintainer

Author

edited ▾

Two extensions here being designed by OMI meant to work together.

omi-collider

OMI_collider #118

robertlong wants to merge 12 commits into `main` from `OMI_collider`

Conversation 18 · Commits 12 · Checks 0 · Files changed 5 · +278 -0

robertlong commented 27 days ago

This is a continuation of the PR started by [@RangerMauve #63](#)

The goal for this PR is to get something together that is up to date with our current implementations of the spec and the latest feedback.

RangerMauve and others added 12 commits 10 months ago

- Added initial READMEs for collider and physics_body extensions
- Update extensions/2.0/OMI_collider/README.md
- Update extensions/2.0/OMI_collider/README.md
- Change wording from unity limits, use specific axes for box extents
- Merge branch 'main' of [github.com:HyperGodot/gltf-extensions](#)
- radius should only apply to sphere and capsule
- Remove defaults, use required fields based on type, add isTrigger to ...
- Merge branch 'main' of [github.com:omigroup/gltf-extensions](#)

Labels: None yet

Projects: None yet

Milestone: #393474

No milestone: #400166b

Development: 8cd5be9

Successfully merging this pull request may close these issues.

and omi physics body

OMI_physics_body #125

aaronfranke wants to merge 1 commit into `omigroup:main` from `aaronfranke:OMI_physics_body`

Conversation 0 · Commits 1 · Checks 0 · Files changed 10 · +2,185 -0

aaronfranke commented 20 hours ago

This PR adds OMI_physics_body. It supersedes #75 from earlier this year. This PR takes #75, overhauls the text to be more readable, updates the table of equivalent types in major engines, adds some discussed features, fixes and completes* the schema, and added a folder of example GLTF files.

I have the document listed as a Stage 1 proposal. This PR meets all of the requirements of Stage 1 and also meets nearly all of the requirements of Stage 2, except for that all practical purposes, since this depends on OMI_collider, the spec can't be truly complete without OMI_collider merged in first.

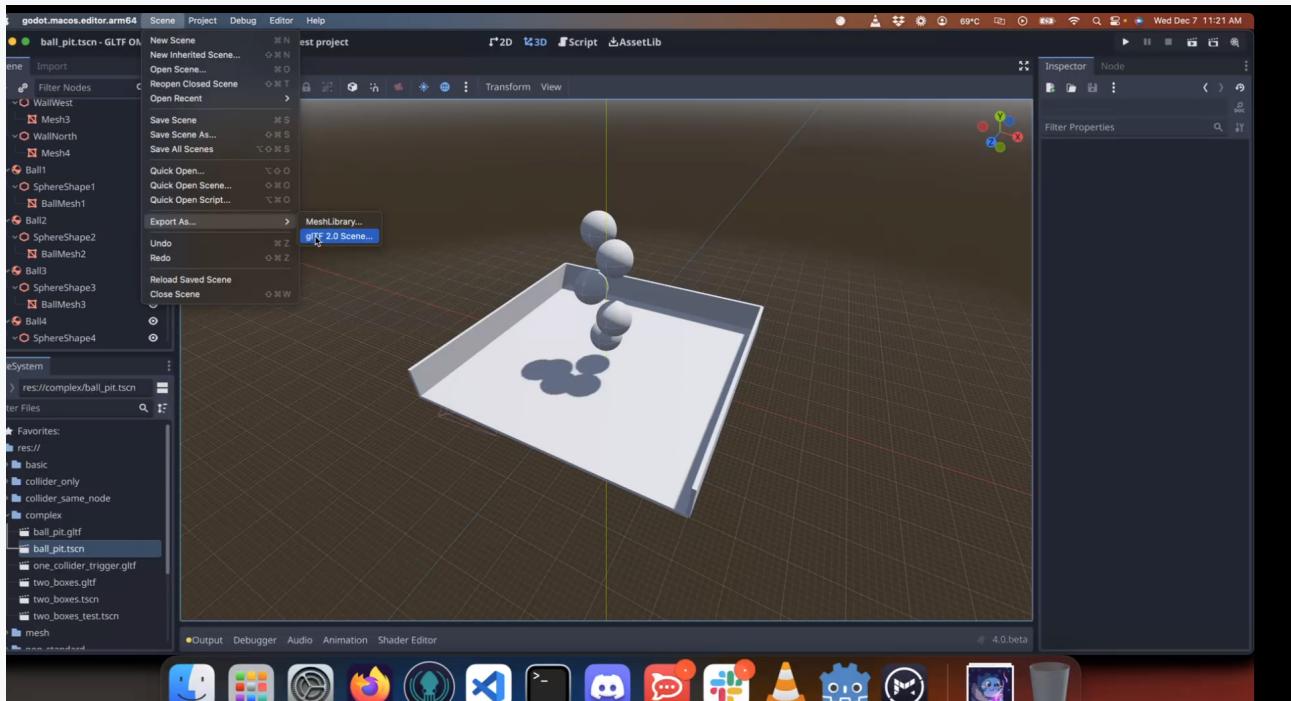
Stage 1

- High level outline published as a GitHub PR in this repository. (this PR)
- Signed W3C CLA. (yes)

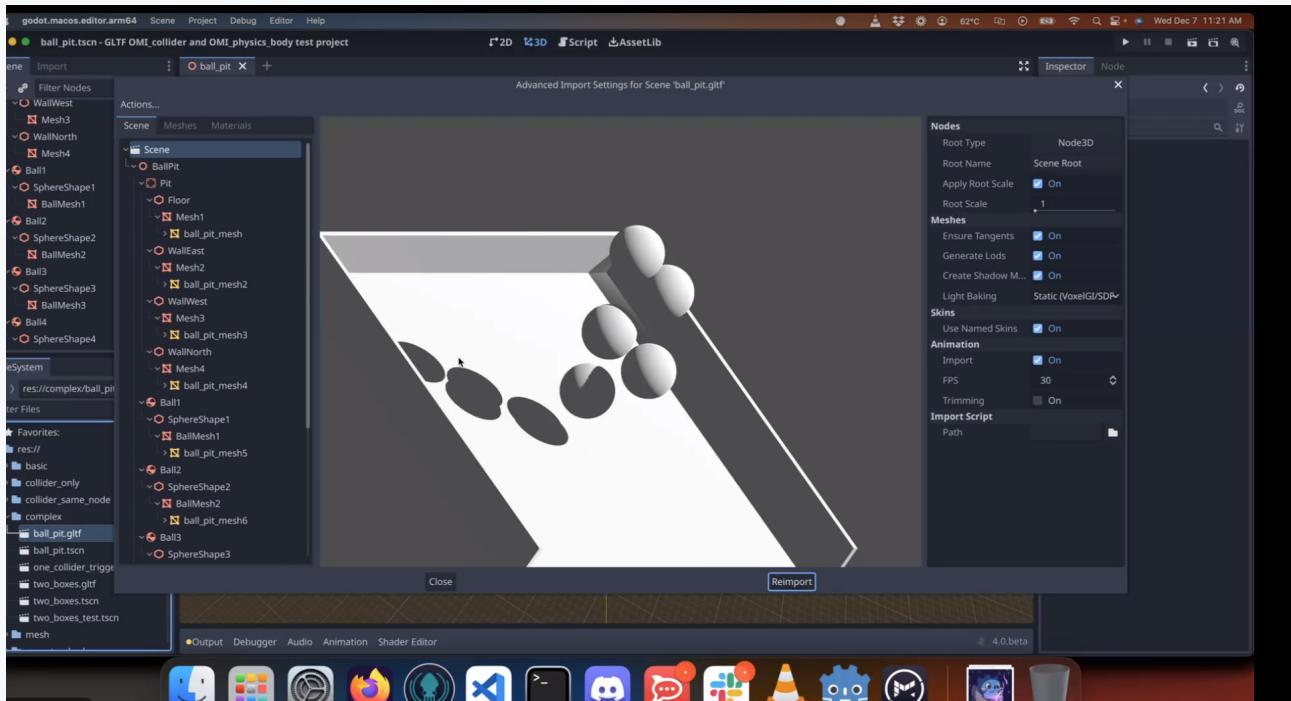
Stage 2

- Previous requirements from Stage 1. (yes)
- One metaverse implementation of the extension in development. (yes, in Godot Engine)
- Valid and publicly accessible sample assets. (yes, see `examples/` subfolder, but note these also use OMI_collider)
- Valid JSON schema describing the extension in its entirety. (this PR has a schema for OMI_physics_body, but a full description in its entirety depends on OMI_collider)
- Markdown document explaining the purpose, features, and properties of the extension. (yes, see the

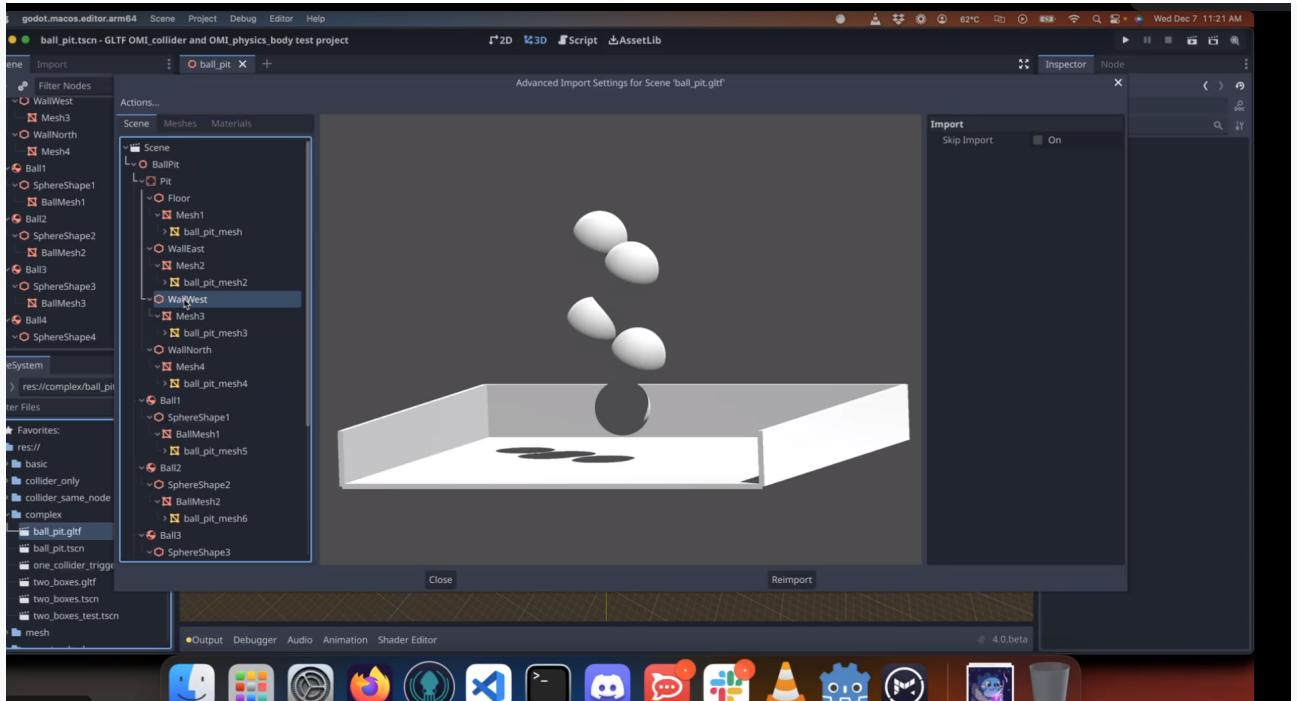
In godot we can export to gltf



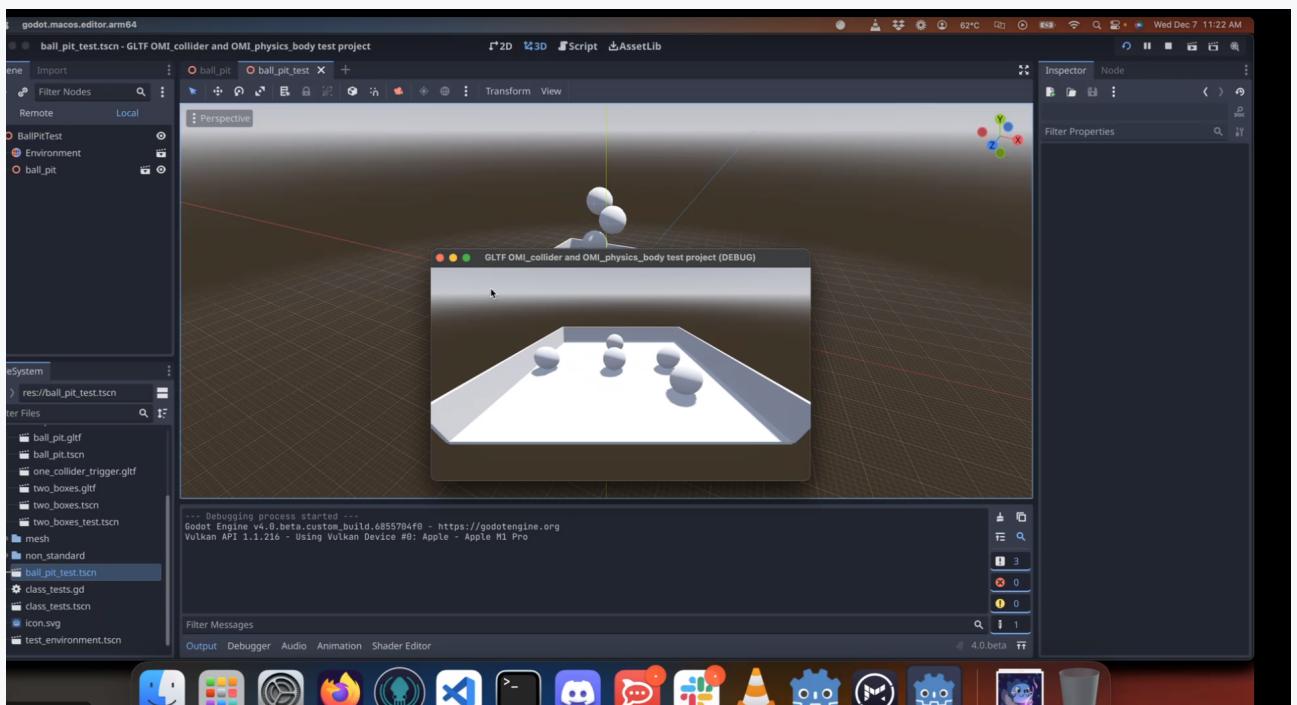
This creates this new gLTF file



Includes physics bodies and shapes imported as Godot nodes

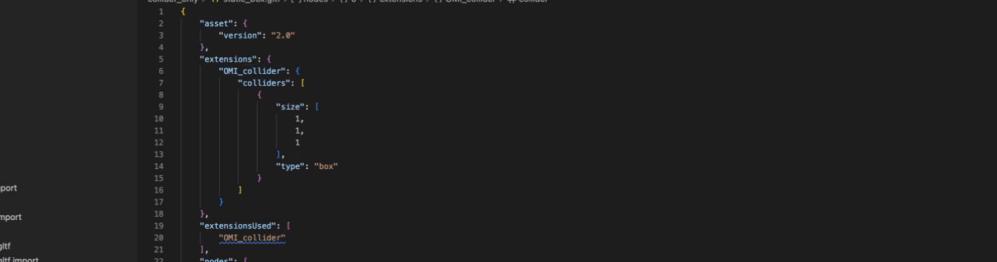


We can run this and the balls collide exactly as they should and we can bring into any game engine we want!



Other examples:

We want to handle the case where there is only an OMI_Collider. That is a simpler system that will work individually. This also aligns with one of our core values for modularity and severability.

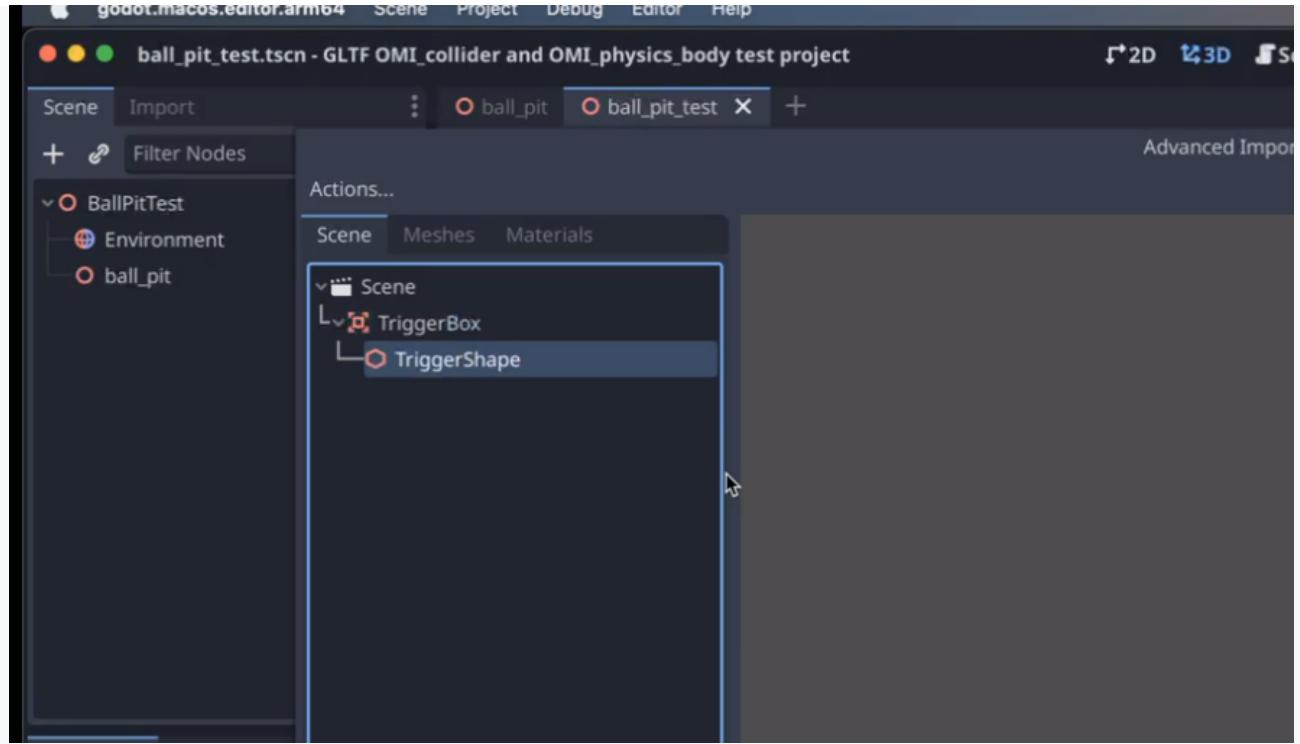


A screenshot of a macOS desktop environment. At the top is a dock with icons for Finder, Home, Mail, Safari, and others. The main window is a terminal application showing a code editor and a file browser. The file browser on the left shows a tree structure of files and folders related to GLTF_OML_physics, including godot, basic, empty.gltf, empty.gltf.import, empty.tsrn, static_box.gltf, static_box.gltf.import, static_box.tsrn, trigger_box.gltf, trigger_box.gltf.import, trigger_only_body.gltf, trigger_only_body.gltf.import, trigger_only_shape.gltf, trigger_only_shape.gltf.import, collider_only, static_box_with_parent.gltf, static_box_with_parent.gltf.import, static_box.gltf, static_box.gltf.import, static_box.tsrn, static_box0.bin, trigger_box.gltf, trigger_box.gltf.import, collider_same_node, static_box.gltf, static_box.gltf.import, trigger_box.gltf.import, trigger_only_body.gltf, trigger_only_body.gltf.import, trigger_only_shape.gltf, trigger_only_shape.gltf.import, complex, ball_pit.gltf, ball_pit.gltf.import, and OUTLINE, TIMELINE. The code editor on the right displays a GLTF file structure with various properties like asset, version, extensions, and nodes. The terminal at the bottom shows command-line output related to GLTF_OML_physics, including errors and a stack trace. The status bar at the bottom right shows the date and time (Wed Dec 7 11:23 AM).

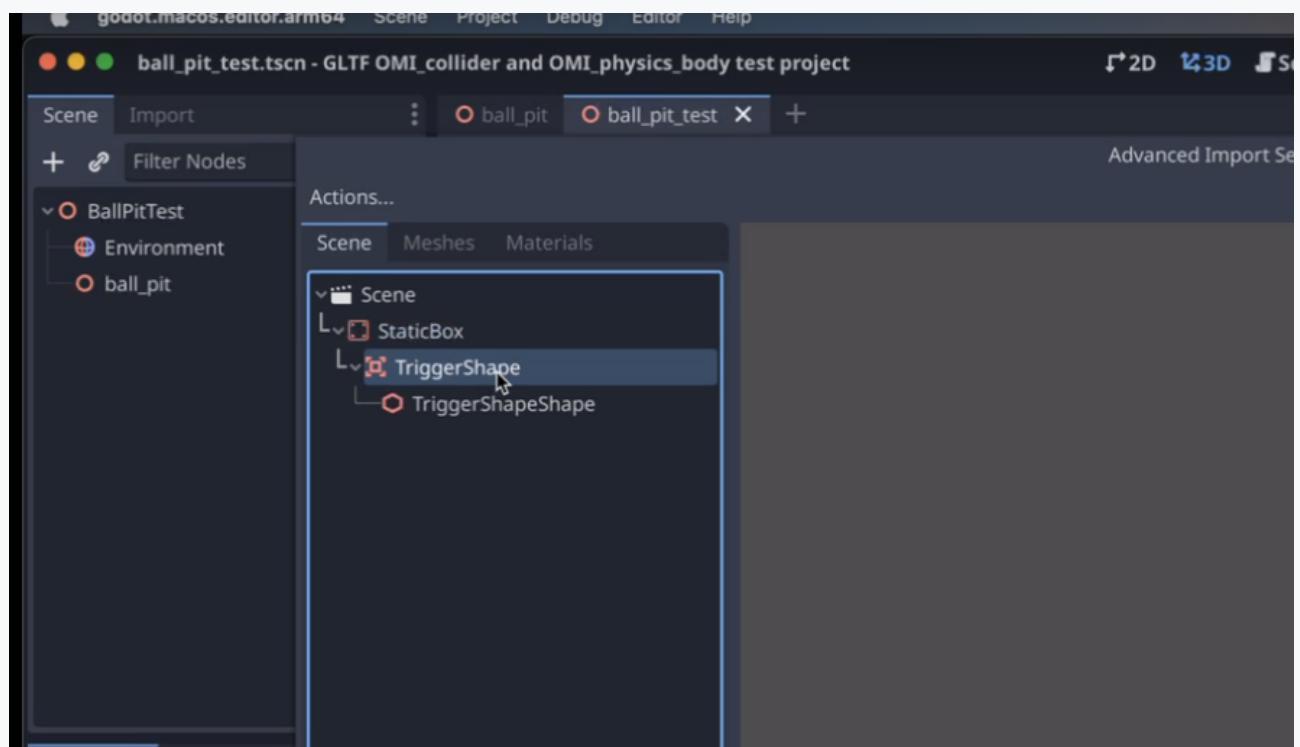
We strive to make this as simple as it can be, and then it can be used in any other engine.

Another example has both the collider and physics body

In unity you only need the triggershape. But we want to make it more portable between Unity and other engines as well. Whereas, in godot we require a body for all shapes.



Setup so that godot creates a child body so the shape behaves as expected.



We have a PR submitted for OMI_Collider that calls out various examples and use-cases:

OMI.collider #118

File filter Conversations

146 extensions/2.0/OMI.collider/README.md

```

70 +
71 + Specifying "isTrigger": true in the collider hints to engines that this collider should not be used for physics interactions and should exist as a "trigger volume" which emits entity intersects with it.
72 + By default an engine may assume that the collider is a static (or rigid) physics body.
73 +
74 + Collider Types
75 +
76 + We've looked at the different types of collision shapes between major game engines (Unity/Unreal/Godot) and looked at what properties are common between them.
77 +
78 + Colliders inherit whatever transform is given to them by the 'node' they are attached to. This includes rotation and translation, however it is discouraged to scale these nodes as problems in some physics engines.
79 +
80 + TODO: There's limits on the sorts of transforms, Unreal can't rotate colliders, has to be axis aligned (local axis to the game component) Maybe only more restrictive only when on directly.
81 +
82 + Here is a table of what the shapes can be represented with in different game engines:
83 +
84 + | Shape | Unity | Godot | Unreal | Ammo |
85 + |-----|-----|-----|-----|-----|
86 + | Box | Box | BoxShape | Box | Box |
87 + | Sphere | Sphere | SphereShape | Sphere | Sphere |
88 + | Capsule | Capsule | CapsuleShape | Capsule | Capsule |
89 + | Hull | MeshConvex | ConvexPolygonShape | Mesh? | ConvexHull |
90 + | Mesh | Mesh | ConcavePolygonShape | Mesh | Mesh |
91 + | Compound | Compound | Add multiple colliders | ??? | ??? |

```

Comment on lines +84 to +91

aaronfranke 27 days ago

Cylinder was discussed, but is missing from this list.

robertlong 21 days ago

It's sorta possible in PhysX 5 now: <https://github.com/NVIDIA-Omniverse/PhysX/blob/85fd98157dcbb7df3fcf26fa9c07c6052c320b92f/physx/snippets/snippetcustomconvex/CylinderConvex.cpp#L211>

Physics Body has a new PR as well. Re-worked

OMI.physics_body #125

1 Open aaronfranke wants to merge 1 commit into omigroup:main from aaronfranke:OMI.physics_body

Conversation 0 Commits 1 Checks 0 Files changed 10

aaronfranke commented 20 hours ago

This PR adds OMI.physics_body. It supersedes #75 from earlier this year. This PR takes #75, overhauls the text to be more readable, updates the table of equivalent types in major engines, adds some discussed features, fixes and completes* the schema, and added a folder of example GLTF files.

I have the document listed as a Stage 1 proposal. This PR meets all of the requirements of Stage 1 and also meets nearly all of the requirements of Stage 2, except that for all practical purposes, since this depends on OMI.collider, the spec can't be truly complete without OMI.collider merged in first.

Stage 1

- High level outline published as a GitHub PR in this repository. (this PR)
- Signed W3C CLA. (yes)

Stage 2

- Previous requirements from Stage 1. (yes)
- One metaverse implementation of the extension in development. (yes, in Godot Engine)
- Valid and publicly accessible sample assets. (yes, see `examples/` subfolder, but note these also use OMI.collider)
- Valid JSON schema describing the extension in its entirety. (this PR has a schema for OMI.physics_body, but a full description in its entirety depends on OMI.collider)

Reviewers
No reviews
Still in progress? Convert to draft

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Development
Successfully merging this pull request may close

Each of these types has meaning in multiple game engines

The extension must also be added to the glTF's `extensionsUsed` array and because it is optional, it does not need to be added to the `extensionsRequired` array.

The extension is intended to be used together with `OMI.collider`. Physics bodies without collision shapes on them will not have any function.

Physics Types

The `*type` property defines what type of physics body this is. Different types of physics bodies have different interactions with physics systems and other bodies within a scene.

Here is a table listing the mapping between the `OMI.physics_body` type and the equivalent types in major game engines.

Shape	Unity	Godot 3	Godot 4	Unreal
Static	Collider	StaticBody	StaticBody3D	WorldStatic, Simulate Physics = false
Kinematic	Rigidbody.isKinematic	KinematicBody	AnimatableBody3D	WorldDynamic, Simulate Physics = false
Character	Rigidbody.isKinematic	KinematicBody	CharacterBody3D	Pawn, Simulate Physics = false
Rigid	Rigidbody	RigidBody	RigidBody3D	PhysicsBody, Simulate Physics = true
Vehicle	Rigidbody	VehicleBody	VehicleBody3D	Vehicle, Simulate Physics = true
Trigger	Collider.isTrigger	Area	Area3D	Generate Overlap Events = true

Static
Static bodies can be collided with, but do not move. They are usually used for level geometry.

Kinematic
Kinematic bodies collide with other bodies, and can be moved using scripts or animations. They can be used for moving platforms.

Character
Character bodies are like kinematic bodies, except are designed for characters. If an engine does not have a dedicated character

We have this PR to Godot engine discussing implementation

Implement physics support in the GLTF module #69266

Implement and closes godotengine/godot-proposals#5268

This PR implements physics support in the GLTF module. This allows importing physics objects from a GLTF file using the `OMI.collider` and `OMI.physics_body` GLTF extensions. omigroup/gltf-extensions#118

The code is designed in a modular and carefully abstracted way. There are helper classes `GLTFCollider` and `GLTFPhysicsBody` that serve as an intermediary between the imported `OMI.*` physics data and Godot's nodes. The `*_dictionary` methods handle converting to/from the JSON Dictionary data that gets stored in the GLTF files, and the `*_node` methods handle converting to/from Godot nodes. `GLTFDocumentExtensionPhysics` is the stateless master class that manages the other two helper classes during the import and export processes.

`OMI.collider` is designed to be able to work independently of `OMI.physics_body` to allow for simpler files. This PR handles this case. If a collider is by itself, we create two nodes, a body (StaticBody3D or Area3D, depending on the collider's `isTrigger` flag), and a CollisionShape3D child.

Both importing and exporting GLTF files with physics is fully supported. GLTF -> Godot -> GLTF round-trip conversions should keep everything functional (even if the data is slightly different, such as Godot adding a physics body parent to GLTF files with standalone colliders). Godot -> GLTF -> Godot round-trip conversions should also keep everything functional for the data present in the standard (GLTF OMI physics won't keep track of inertia or angular velocity etc).

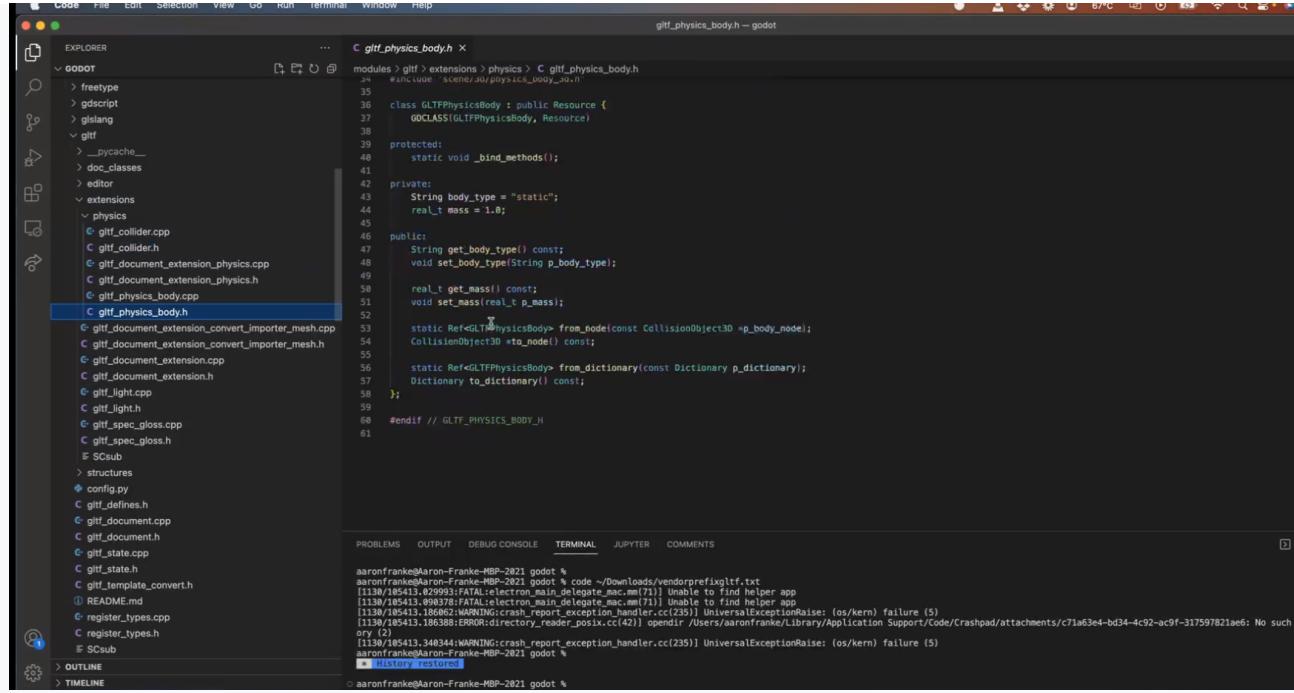
Some implementation notes:

- While I was in the process of making this PR, `MSFT.physics` appeared as a competing standard to OMI's physics extensions. Due to the abstractions I mentioned earlier, if `MSFT.physics` ends up taking off as a popular standard, it should be fairly simple to modify this code to allow reading (and maybe writing) Microsoft physics by modifying the `*_dictionary` methods. For now, this PR only implements reading and writing OMI physics data.
- The OMI PR for `OMI.collider` does not include cylinders as one of the available shapes, but this PR does. The standard has not been finalized yet, and my opinion is that `OMI.collider` should add cylinders to the standard. However, even if it's not part of the standard, when exporting a Godot scene with cylinders there's no good reason to throw away those shapes, and if we encounter a GLTF file with cylinders there's no reason to throw that away either.

Here is a test project that tests most things listed above: [Godot_GLTFOMI_physics.zip](https://godot-gltf-omiphysics.godot)

Question: Godot 4.0, or Godot 4.1? On one hand, Godot 4.0 is supposed to be in feature freeze, so delaying to 4.1 makes sense. However, this is a low-risk PR, it should have zero impact on anyone not using GLTF physics. I am also interested in

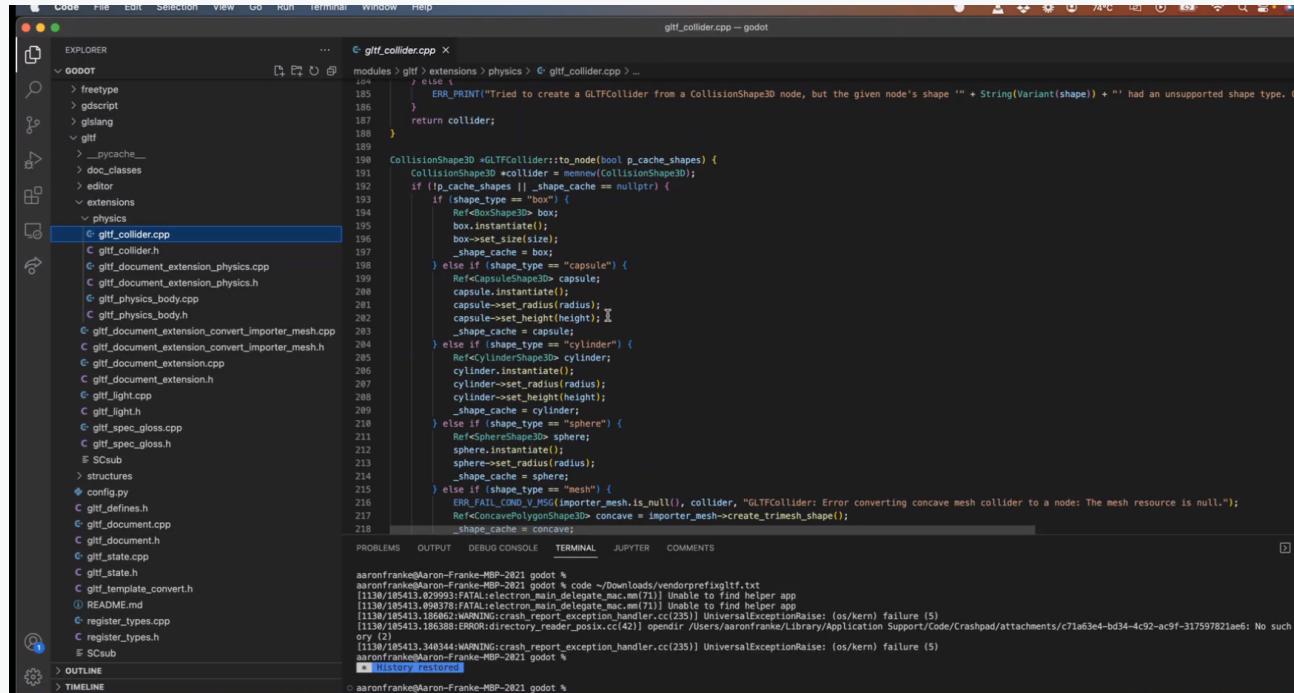
Inside of godots code we have these little methods for converting as well:



```

modules > gltf > extensions > physics > C gltf_physics_body.h
34  #include "scene/sd/physics_body_sd.h"
35
36  class GLTFPhysicsBody : public Resource {
37  public:
38      GOCLASS(GLTFPhysicsBody, Resource)
39
40  protected:
41      static void _bind_methods();
42
43  private:
44      String body_type = "static";
45      real_t mass = 1.0;
46
47  public:
48      String get_body_type() const;
49      void set_body_type(String p_body_type);
50
51      real_t get_mass() const;
52      void set_mass(real_t p_mass);
53
54      static Ref<GLTFPhysicsBody> from_node(const CollisionObject3D *p_body_node);
55      CollisionObject3D *to_node() const;
56
57      static Ref<GLTFPhysicsBody> from_dictionary(const Dictionary p_dictionary);
58  };
59
60 #endif // GLTF_PHYSICS_BODY_H

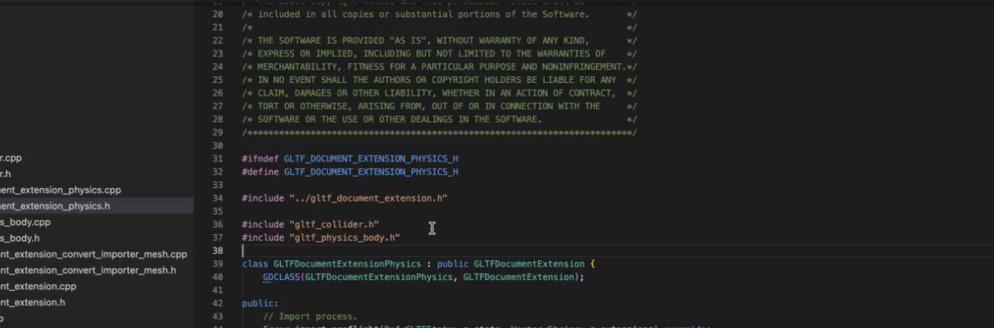
```



```

modules > gltf > extensions > physics > C gltf_collider.cpp > ...
185     ERR_PRINT("Tried to create a GLTFCollider from a CollisionShape3D node, but the given node's shape '" + String(Variant(shape)) + "' had an unsupported shape type.");
186
187     return collider;
188 }
189
190 CollisionShape3D *GLTFCollider::to_node(bool p_cache_shapes) {
191     CollisionShape3D *collider = memnew(CollisionShape3D);
192     if (!p_cache_shapes || !_shape_cache == nullptr) {
193         if (shape_type == "box") {
194             Ref<BoxShape3D> box;
195             box.instantiate();
196             box->set_size(size);
197             _shape_cache = box;
198         } else if (shape_type == "capsule") {
199             Ref<CapsuleShape3D> capsule;
200             capsule.instantiate();
201             capsule->set_radius(radius);
202             capsule->set_height(height);
203             _shape_cache = capsule;
204         } else if (shape_type == "cylinder") {
205             Ref<CylinderShape3D> cylinder;
206             cylinder.instantiate();
207             cylinder->set_radius(radius);
208             cylinder->set_height(height);
209             _shape_cache = cylinder;
210         } else if (shape_type == "sphere") {
211             Ref<SphereShape3D> sphere;
212             sphere.instantiate();
213             sphere->set_radius(radius);
214             _shape_cache = sphere;
215         } else if (shape_type == "mesh") {
216             ERR_FATAL("COND_V_NSGImporter_mesh.is_null()", collider, "GLTFCollider: Error converting concave mesh collider to a node: The mesh resource is null.");
217             Ref<ConcavePolygonShape3D> concave = importer_mesh->create_trimesh_shape();
218             _shape_cache = concave;

```



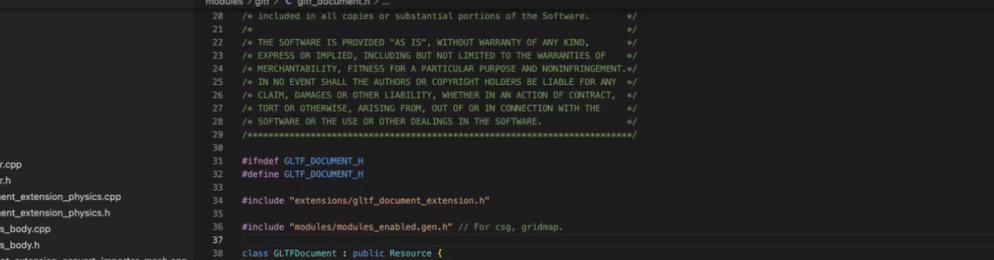
The screenshot shows a macOS desktop with two windows open. The terminal window in the foreground displays the Godot source code for `gltf_document_extension_physics.h`. The file browser window in the background shows the Godot source code structure under the `GODOT` folder, including files like `gltf.collider.cpp`, `gltf_physics_body.cpp`, and `gltf_document_extension_physics.cpp`.

```
Code File Selection View Go Run Terminal Window Help
gltf_document_extension_physics.h - godot
C gltf_document_extension_physics.h
modules > gltf > extensions > physics > C gltf_document_extension_physics.h ...
20  /* included in all copies or substantial portions of the Software. */
21  /*
22  // THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
23  // EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
24  // MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
25  // IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
26  // CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
27  // TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
28  // SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
29  ****
30
31 #ifndef GLTF_DOCUMENT_EXTENSION_PHYSICS_H
32 #define GLTF_DOCUMENT_EXTENSION_PHYSICS_H
33
34 #include "../gltf_document_extension.h"
35
36 #include "gltf.collider.h"
37 #include "gltf_physics_body.h"
38
39 class GLTFFDocumentExtensionPhysics : public GLTFFDocumentExtension {
40     GDCLASS(GLTFFDocumentExtensionPhysics, GLTFFDocumentExtension);
41
42 public:
43     // Import process.
44     Error import_preflight(Ref<GLTFState> p_state, Vector<String> p_extensions) override;
45     Vector<String> get_supported_extensions() override;
46     Error parse_node_extensions(Ref<GLTFState> p_state, Ref<GLTFNode> p_gltf_node, Dictionary &p_extensions) override;
47     Node3D *generate_scene_node(Ref<GLTFState> p_state, Ref<GLTFNode> p_gltf_node, Node *p_scene_parent) override;
48     // Export process.
49     void convert_scene_node(Ref<GLTFState> p_state, Ref<GLTFNode> p_gltf_node, Node *p_scene_node) override;
50     Error export_node(Ref<GLTFState> p_state, Ref<GLTFNode> p_gltf_node, Dictionary &p_node_json, Node *p_scene_node) override;
51 };
52
53 #endif // GLTF_DOCUMENT_EXTENSION_PHYSICS_H
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER COMMENTS
aaronfranke@Aaron-Franke-MBP-2021 godot %
aaronfranke@Aaron-Franke-MBP-2021 godot % code ~/Downloads/vendor/prefiglif.txt
[1138/185413.896373]FATAL:electron:main_delegate.main_delegate.macos(711) Unable to find helper app
[1138/185413.186062]WARNING:crash_report_exception_handler.cc(235) UniversalExceptionRaise: (os/kern) failure (5)
[1138/185413.186388]WARNING:crash_report_exception_handler.cc(42) opendir /Users/aaronfranke/Library/Application Support/Code/Crashpad/attachments/c7a63e4-bd34-4c92-acf3-317597821ae6: No such file or directory
[1138/185413.348344]WARNING:crash_report_exception_handler.cc(235) UniversalExceptionRaise: (os/kern) failure (5)
aaronfranke@Aaron-Franke-MBP-2021 godot %
* History restored

```

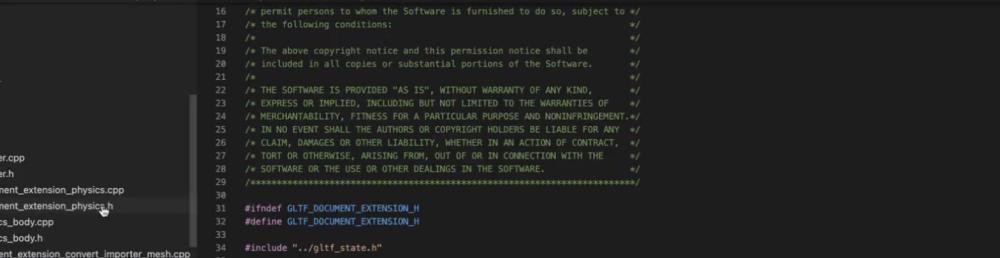
How is a glTF document extension different from an importer?

When godot imports it goes through this thing called glTF document



The screenshot shows a macOS desktop environment. In the top-left corner, the Dock contains icons for Finder, Homebrew, and the terminal application. The top menu bar includes 'Code', 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', 'Window', and 'Help'. The main window is a code editor for 'gltf_document.h' under the 'gltf' project. The code is a C++ header file for a GLTF document, defining a class 'GLTFDocument' that inherits from 'Resource'. It includes various GLTF-related functions and constants. Below the code editor is a 'PROBLEMS' tab, which is currently empty. The bottom of the code editor window shows the terminal output, which is also empty. The status bar at the bottom of the screen displays the current date and time as '63°C 10:59 AM'.

to add stuff you use this GLTF_DOCUMENT EXTENSION method



```
gltf_document_extension.h - godot
```

```
modules > gltf > extensions > gltf_document_extension.h > ...
16  /* permit persons to whom the Software is furnished to do so, subject to */
17  /* the following conditions: */
18  /*
19  * The above copyright notice and this permission notice shall be
20  * included in all copies or substantial portions of the Software.
21  */
22  /*
23  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
24  * EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
25  * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
26  * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
27  * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
28  * TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
29  * SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
30  */
31 #ifndef GLTF_DOCUMENT_EXTENSION_H
32 #define GLTF_DOCUMENT_EXTENSION_H
33
34 #include "../gltf_state.h"
35
36 class GLTFDocumentExtension : public Resource {
37     GDCLASS(GLTFDocumentExtension, Resource);
38
39 protected:
40     static void _bind_methods();
41
42 public:
43     // Import process.
44     virtual Error import_preflight(Ref<GLTFState> p_state, Vector<String> p_extensions);
45     virtual Vector<String> get_supported_extensions();
46     virtual Error parse_node(Ref<GLTFState> p_state, Ref<GLTFNode> p_gltf_node, Dictionary sp_extensions);
47     virtual Node3D *generate_scene_node(Ref<GLTFState> p_state, Ref<GLTFNode> p_gltf_node, Node *p_scene_parent);
48     virtual Error import_post_parse(Ref<GLTFState> p_state);
49     virtual Error import_node(Ref<GLTFState> p_state, Ref<GLTFNode> p_gltf_node, Dictionary &r_json, Node *p_node);
50 }
```

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	JUPYTER	COMMENTS

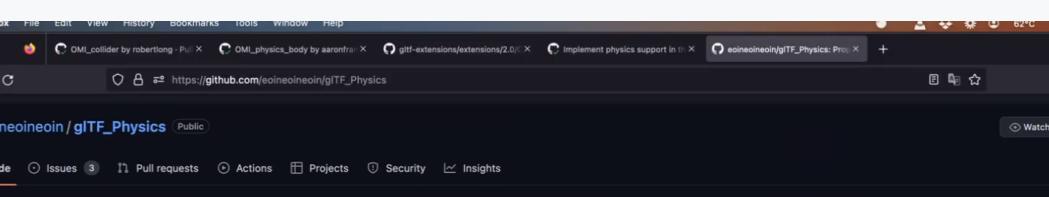
```
aaronfranke@Aaron-Franke-MBP-2021 godot %
aaronfranke@Aaron-Franke-MBP-2021 godot % code ~/Downloads/vendorpreflight/tsc
[1130/105413.908373]FATAL:electron_main_delegate.macos[771] Unable to find helper app
[1130/105413.186062]WARNING:crash_report_exception_handler.cc[235] UniversalExceptionRaise: (os/kern) failure (5)
[1130/105413.186388]ERROR:directory_reader_posix.cc[42] opendir: /Users/aaronfranke/Library/Application Support/Code/Crashpad/attachments/c71a63e4-bd34-4c92-acf9-317597821ae6: No such
file or directory
[1130/105413.340344]WARNING:crash_report_exception_handler.cc[235] UniversalExceptionRaise: (os/kern) failure (5)
aaronfranke@Aaron-Franke-MBP-2021 godot %
* [History restored]
```

If you want to implement it into godot, you can literally just write it and drop it into godot without recompiling the engine every time.

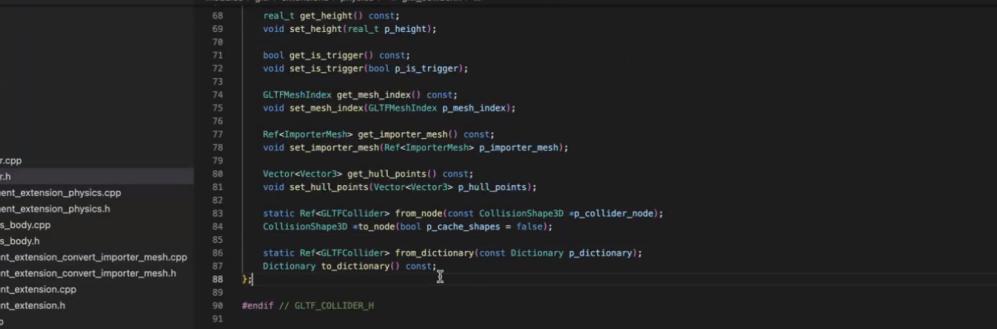


mrmetaverse on Dec 7, 2022 Maintainer Author edited

Also want to point out that this was designed in such a way to be clear about what is OMI, what is godot, and what is glTF



A "competing" standard recently emerged from inside of Microsoft. We could change a few things and use mostly the same codebase for QM1 physics *and* Microsoft physics.



gitf.collider.h - godot

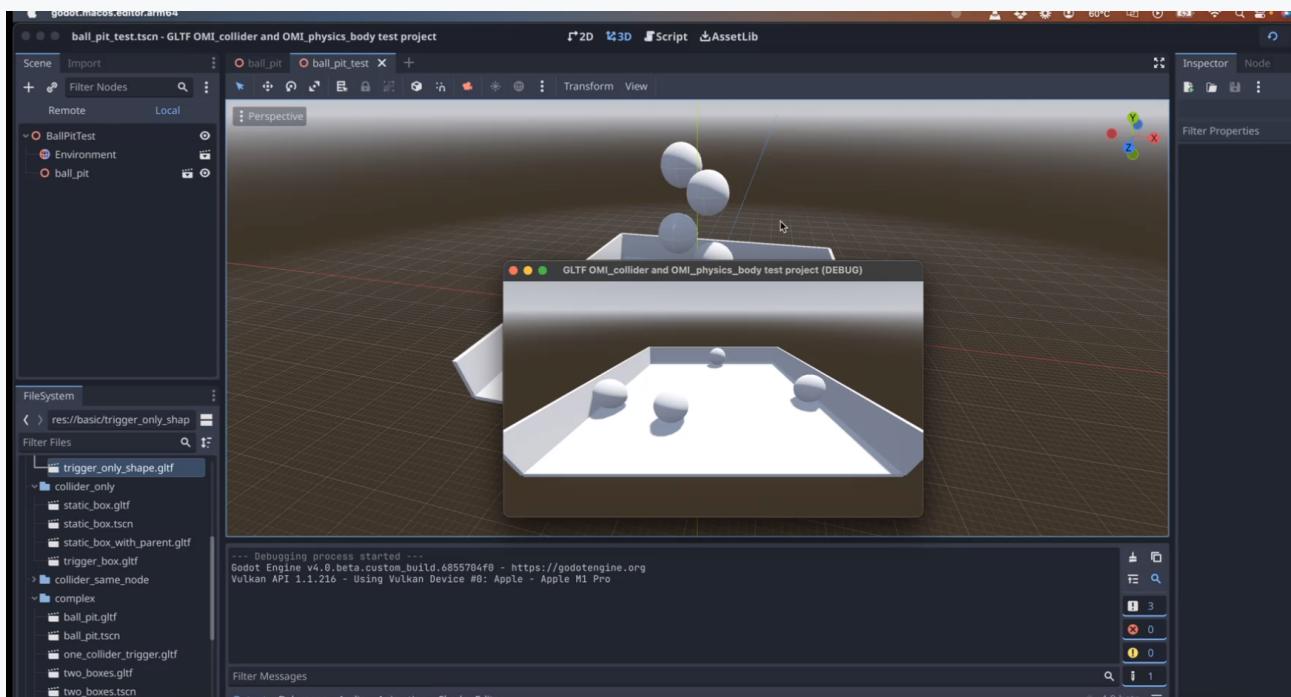
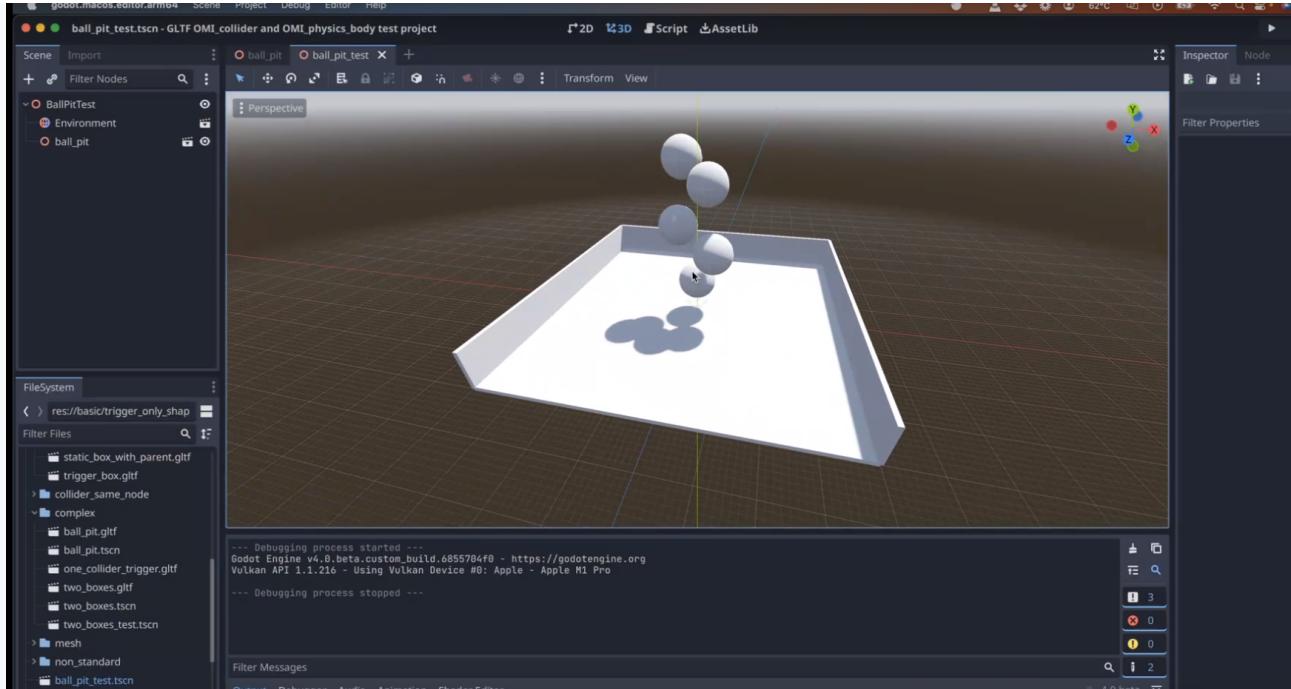
```
EXPLORER
... C gitf.collider.h x
modules > gitf > extensions > physics > C gitf.collider.h ...
68     real_t get_height() const;
69     void set_height(real_t p_height);
70
71     bool get_is_trigger() const;
72     void set_is_trigger(bool p_is_trigger);
73
74     GLTFMeshIndex get_mesh_index() const;
75     void set_mesh_index(GLTFMeshIndex p_mesh_index);
76
77     Ref<ImporterMesh> get_importer_mesh() const;
78     void set_importer_mesh(Ref<ImporterMesh> p_importer_mesh);
79
80     Vector<Vector3> get_hull_points() const;
81     void set_hull_points(Vector<Vector3> p_hull_points);
82
83     static Ref<GLTFCollider> from_node(const CollisionShape3D *p.collider_node);
84     CollisionsShape3D *to_node(bool p_cache_shapes = false);
85
86     static Ref<GLTFCollider> from_dictionary(const Dictionary p_dictionary);
87     Dictionary to_dictionary() const;
88 }
89
90 #endif // GLTF_COLLIDER_H
91
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER COMMENTS

```
aaronfranke@Aaron-Franke-MBP-2021 godot %
aaronfranke@Aaron-Franke-MBP-2021 godot % code ~/Downloads/vendorpref/sdlfix.txt
[1138/185413.186378]FATAL:electron_main_delegate.macos.mm(711) Unable to find helper app
[1138/185413.186862]WARNING:crash_report_exception_handler.cc(235)] UniversalExceptionRaise: (os/kern) failure (5)
[1138/185413.186388]ERROR:directory_reader_posix.cc(42)] opendir '/Users/aaronfranke/Library/Application Support/Code/ocrashpad/attachments/c71a63e4-bd34-4c9f-317597821ae6: No such
path'
[1138/185413.186344]WARNING:crash_report_exception_handler.cc(235)] UniversalExceptionRaise: (os/kern) failure (5)
aaronfranke@Aaron-Franke-MBP-2021 godot %
* History restored
```

We have a root node, and extensions, etc. (I tried to keep up 😊)

All 5 of these spheres only need the 1 shape resource:



mrmetaverse on Dec 7, 2022

Maintainer

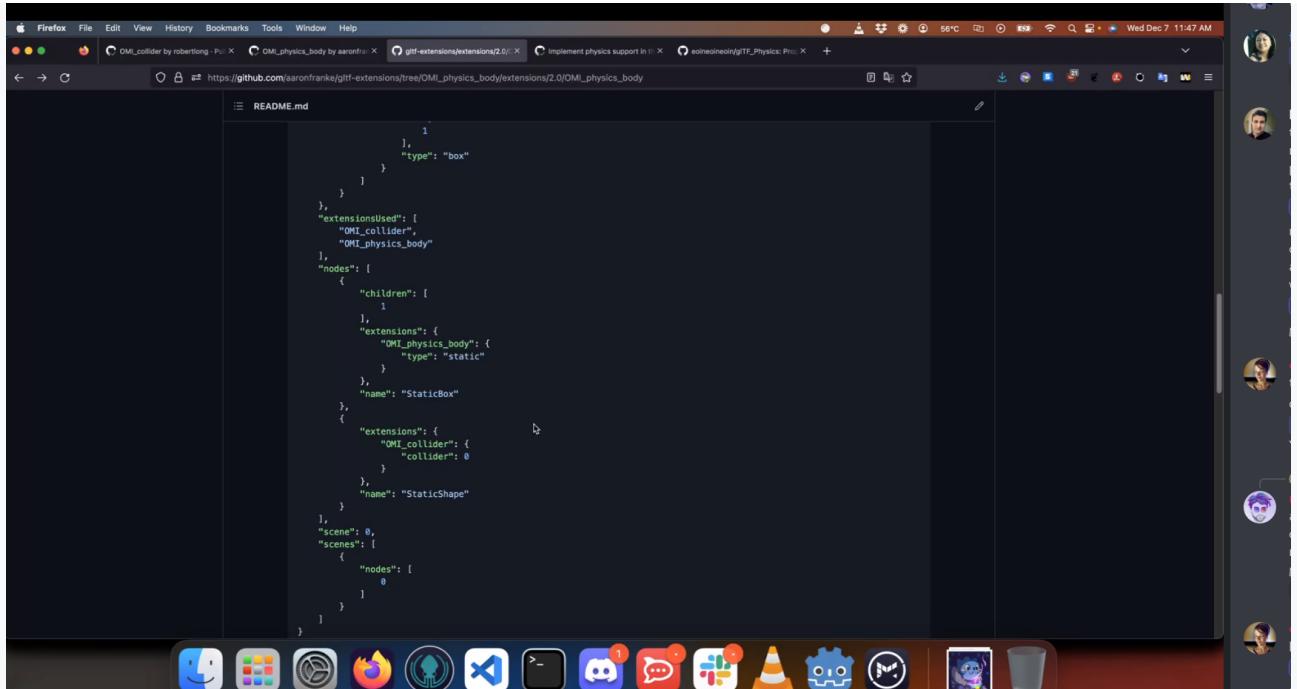
Author

edited ▾

CTA - HOW YOU CAN HELP

We are always looking for groups to test, experiment with, and implement our proposed extensions. (It's also part of our standard process for considering an extension or protocol as "ready").

We would love to work together with other experts, like an Unreal expert, and Unity expert to see what we could change or edit.

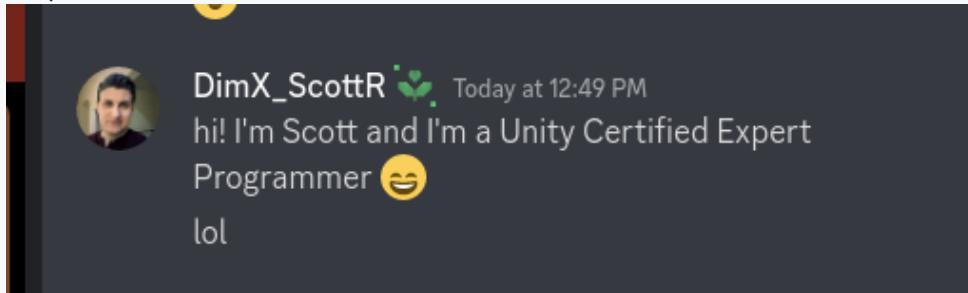


```

```json
 "children": [
 {
 "type": "box"
 }
],
 "extensionsUsed": [
 "OML.collider",
 "OML.physics_body"
],
 "nodes": [
 {
 "children": [
 {
 "extensions": {
 "OML.physics_body": {
 "type": "static"
 }
 },
 "name": "StaticBox"
 },
 {
 "extensions": {
 "OML.collider": {
 "collider": 0
 }
 },
 "name": "StaticShape"
 }
],
 "scene": 0,
 "scenes": [
 {
 "nodes": [
 0
]
 }
]
 }
]
}
```

```

Ha perfect, we're 2/3 there then! :



Next week... invite a friend! Who do we know that is an Unreal engine expert who could test/implement these extensions?

Category

 General

Labels

None yet

1 participant

