# OMI_physics_body

## Contributors


- Aaron Franke, The Mirror Megaverse Inc.
- Mauve Signweaver, Mauve Software Inc.


## Status

Open Metaverse Interoperability Group Stage 1 Proposal

## Dependencies

Written against the glTF 2.0 spec.

Depends on the `OMI_physics_shape` spec to be useful.

## Overview

This extension allows for specifying the type of physics body in glTF scenes.

Physics bodies are defined with a string enum for the type. Nodes with physics shapes defined using the `OMI_physics_shape` spec should be added as direct children of physics bodies. In order to be associated with a `OMI_physics_body` glTF node, `OMI_physics_shape` glTF nodes must be direct children, not indirect children.

Each glTF node with `OMI_physics_shape` may be associated with zero or one `OMI_physics_body` glTF node as its direct parent. Each glTF node with `OMI_physics_body` should have one or many `OMI_physics_shape` glTF node direct children (zero is valid but not recommended, since physics bodies will not collide with anything if they have zero physics shape children).

### Example:

This example defines a static body node which has a single box collider as a child:

```
{
    "asset": {
        "version": "2.0"
    },
    "extensions": {
        "OMI_physics_shape": {
            "shapes": [
                {
                    "size": [
                        1,
                        1,
                        1
                    ],
                    "type": "box"
                }
            ]
        }
    },
    "extensionsUsed": [
        "OMI_physics_body",
        "OMI_physics_shape"
```

```
    ],
    "nodes": [
        {
            "children": [
                1
            ],
            "extensions": {
                "OMI_physics_body": {
                    "type": "static"
                }
            },
            "name": "StaticBox"
        },
        {
            "extensions": {
                "OMI_physics_shape": {
                    "shape": 0
                }
            },
            "name": "StaticShape"
        }
    ],
    "scene": 0,
    "scenes": [
        {
            "nodes": [
                0
            ]
        }
    ]
}
```

More example assets can be found in the examples/ folder. All of these examples use both `OMI_physics_shape` and `OMI_physics_body`.

# glTF Schema Updates

This extension consists of a new `OMI_physics_body` data structure which can be added to a glTF node.

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_physics_shape`. Physics bodies without collision shapes on them are valid but will not collide with anything.

## Property Summary

| | Type | Description | Default value |
|---|---|---|---|
| **type** | `string` | The type of the physics body as a string. | Required, no default |
| **mass** | `number` | The mass of the physics body in kilograms. | 1.0 |
| **linearVelocity** | `number[3]` | The initial linear velocity of the body in meters per second. | [0.0, 0.0, 0.0] |
| **angularVelocity** | `number[3]` | The initial angular velocity of the body in radians per second. | [0.0, 0.0, 0.0] |
| **centerOfMass** | `number[3]` | The center of mass offset from the origin in meters. | [0.0, 0.0, 0.0] |

| | Type | Description | Default value |
|---|---|---|---|
| **inertiaTensor** | `number[9]` | The inertia tensor 3x3 matrix in kilogram meter squared (kg·m²). | [0.0, …, 0.0] |

## Physics Body Types

The `"type"` property is a lowercase string that 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.

| Body Type | 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 type, treat this as kinematic instead.

### Rigid

Rigid bodies collide with other bodies, and move around on their own in the physics simulation. They are affected by gravity. They can be used for props that move around in the world.

### Vehicle

Vehicle bodies are like rigid bodies, except are designed for vehicles. If an engine does not have a dedicated vehicle type, treat this as rigid instead.

### Trigger

Trigger bodies do not collide with other objects, but can generate events when another physics body "enters" them. For example, a "goal" area which triggers whenever a ball gets thrown into it. Trigger bodies can be added as children of other bodies to attach a trigger volume to another body.

## Mass

The `"mass"` property is a number that defines how much mass this physics body has in kilograms. Not all body types can make use of mass, such as triggers or non-moving bodies, in which case the mass can be ignored. If not specified, the default value is 1 kilogram.

## Linear Velocity

The `"linearVelocity"` property is an array of three numbers that defines how much linear velocity this physics body starts with in meters per second. Not all body types can make use of linear velocity, such as non-moving bodies, in which case the linear velocity can be ignored. If not specified, the default value is zero.

## Angular Velocity

The `"angularVelocity"` property is an array of three numbers that defines how much angular velocity this physics body starts with in radians per second. Not all body types can make use of angular velocity, such as non-moving bodies, in which case the angular velocity can be ignored. If not specified, the default value is zero.

## Center of Mass

The `"centerOfMass"` property is an array of three numbers that defines the position offset in meters of the center of mass in the body's local space.

This property is useful when converting assets with a center of mass, but when creating new assets it is recommended to leave the center of mass at the body's origin. Some physics engines support the center of mass being offset from the origin, but not all of them do. Implementations without support for a center of mass offset would have to adjust the node positions to make this work, which may be undesired.

## Inertia Tensor

The `"inertiaTensor"` property is an array of 9 numbers that defines the inertia tensor 3x3 matrix of the body in kilogram meter squared (kg·m²). We specify "tensor" in the name because this defines inertia in multiple directions and is different from linear momentum inertia. Only "rigid" and "vehicle" body types can make use of inertia. If zero or not specified, the inertia should be automatically calculated by the physics engine.

The inertia tensor matrix is a symmetric matrix. The inertia matrix represents the mass distribution of the body and determines how hard the body is to rotate. The values on the diagonal represent the inertia around the 3 principle axes (X, Y, Z), while the values not on the diagonal represent the 3 coupling values between the axes (XY, XZ, YZ). For more information, refer to the Wikipedia article.

Some engines only support specifying the inertia around the principle axes as a Vector3. In those engines, when importing a glTF file and reading the inertia matrix, only the diagonal principle axis values should be used, and the non-diagonal coupling values should be discarded. Similarly, when exporting a glTF file while writing the inertia matrix, write the Vector3 values to the matrix diagonal principle axis values, and set the non-diagonal coupling values to zero.

## JSON Schema

See schema/node.OMI_physics_body.schema.json.

## Known Implementations

- Godot Engine: https://github.com/godotengine/godot/pull/69266

## Resources:

- Unity colliders: https://docs.unity3d.com/Manual/CollidersOverview.html
- Unreal Engine Physics: https://docs.unrealengine.com/4.27/en-US/InteractiveExperiences/Physics/Collision/Overview/
- Godot Physics Body: https://docs.godotengine.org/en/stable/classes/class_physicsbody.html
- Godot Area: https://docs.godotengine.org/en/stable/classes/class_area.html
- Godot RigidBody3D: https://docs.godotengine.org/en/latest/classes/class_rigidbody3d.html
- Wikipedia Moment of Inertia and Inertia Tensor https://en.wikipedia.org/wiki/Moment_of_inertia#Inertia_tensor