OMI_physics_joint

Contributors

- Aaron Franke, The Mirror Megaverse Inc.

Status

Open Metaverse Interoperability Group Stage 1 Proposal

Dependencies

Written against the glTF 2.0 spec.

Depends on the `OMI_physics_body` spec, which depends on the `OMI_physics_shape` spec.

Overview

This extension allows defining a glTF node as a physics joint for constraining rigid bodies.

Each physics joint node is a separate node that references the two bodies it joints together. At least one of the bodies must be a rigid body (`OMI_physics_body` "rigid" or "vehicle") for the joint to function. One or zero of the connected nodes may be a solid body that joints cannot move (`OMI_physics_body` "static", "kinematic", "character"). A joint cannot be connected to a trigger body, and cannot be connected to a non- `OMI_physics_body` glTF node.

Each physics joint node must reference one or more joint constraints defined in the document-level joint constraints array.

A joint should be on its own glTF node, it should not be on the same node as a mesh, camera, light, physics body, physics shape, etc.

Example:

This example defines 2 rigid bodies that are connected with a joint that constrains linearly on a fixed point, also known as a "pin" joint. This example JSON is a subset of the examples/simple_joint.gltf file.

```
{
    "asset": {
        "version": "2.0"
    },
    "extensionsUsed": [
        "OMI_physics_body",
        "OMI_physics_joint",
        "OMI_physics_shape"
    ],
    "extensions": {
        "OMI_physics_joint": {
            "constraints": [
                {
                    "linearAxes": [0, 1, 2]
                }
            ]
        }
    },
    "nodes": [
```

```
        {
            "extensions": {
                "OMI_physics_joint": {
                    "constraints": [0],
                    "nodeA": 1,
                    "nodeB": 2
                }
            },
            "name": "PinJoint",
            "translation": [-0.23, 0.6, 0.0]
        },
        {
            "extensions": {
                "OMI_physics_body": {
                    "type": "rigid"
                }
            },
            "name": "BodyA",
            "rotation": [0.0, 0.0, -0.17364804446697, 0.984807789325714],
            "translation": [-0.45, 0.68, 0.0]
        },
        {
            "extensions": {
                "OMI_physics_body": {
                    "type": "rigid"
                }
            },
            "name": "BodyB",
            "translation": [0.0, 0.6, 0.0]
        }
    ]
}
```

More example assets can be found in the examples/ folder.

glTF Schema Updates

This extension consists of three new data structures for defining joints. A glTF file with joints should have a document-level `"OMI_physics_joint"` object added to the document `"extensions"` which contains an array of joint constraints. Each joint constraint is an object defined as the below spec. Then, the key `"OMI_physics_joint"` can be added to the node-level `"extensions"` of a glTF node to define that node as a physics joint. Each joint node has an array of constraints that references the document-level constraints array by index, and each joint node also references 2 physics body nodes.

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.

Joint Constraint Property Summary

	Type	Description	Default value
linearAxes	`number[0..3]`	The axes to constrain. Can only contain 0 (X), 1 (Y), or 2 (Z).	[] (empty array)
angularAxes	`number[0..3]`	The axes to constrain. Can only contain 0 (X), 1 (Y), or 2 (Z).	[] (empty array)
lowerLimit	`number`	The lower limit of the constraint, in meters or radians.	0.0
upperLimit	`number`	The lower limit of the constraint, in meters or radians.	0.0

	Type	Description	Default value
stiffness	`number`	The stiffness of the limits, how hard to spring back when beyond.	Infinity
damping	`number`	When beyond the limits, the damping of the springing back motion.	1.0

A joint constraint should define at least one linear axis or one angular axis, otherwise the joint constraint will not perform any constraints.

The reason that joints constraints are defined at the document-level and referenced on nodes is because it's very common to have many joints with the same constraints. For example, a long rope using many bodies each with pin joints (linear axes constrained) would have the same joint constraints repeated many times, but with different nodes attached.

Linear Axes

The `"linearAxes"` property defines a list of linear axes to constrain. Can only contain 3 possible values, 0 (X), 1 (Y), or 2 (Z). If empty or not specified, this joint constraint does not constrain linearly.

Angular Axes

The `"angularAxes"` property defines a list of angular axes to constrain. Can only contain 3 possible values, 0 (X), 1 (Y), or 2 (Z). If empty or not specified, this joint constraint does not constrain angularly.

Lower Limit

The `"lowerLimit"` property defines the lower limit of the constraint in meters or radians, depending on whether the constraint is linear or angular. If not specified, the default value is 0.0.

The lower limit must be less than or equal to the upper limit, otherwise the constraint is invalid. If the lower and upper limits are equal, the constraint is fixed on that value.

Upper Limit

The `"upperLimit"` property defines the upper limit of the constraint in meters or radians, depending on whether the constraint is linear or angular. If not specified, the default value is 0.0.

The upper limit must be greater than or equal to the lower limit, otherwise the constraint is invalid. If the lower and upper limits are equal, the constraint is fixed on that value.

Stiffness

The `"stiffness"` property defines how soft or hard the limits are, and how much force should be applied to the bodies to spring back once they surpasses the limits. Must be positive. If not specified, the default value is infinity.

A lower value indicates a softer limit, while a higher value indicades a harder limit. A value of infinity means the limits should be completely hard and may not be surpassed. Implementations are expected to clamp this to within the limits of the physics engine. For example, if a physics engine cannot handle an infinite stiffness, it should be set to a high value, since an extremely stiff joint is close enough to an infinitely stiff joint. For more information, see the Wikipedia article on stiffness.

Damping

The `"damping"` property defines the amount of damping the bodies should experience while the bodies surpasses the limits and are springing back to be within them. Damping must not be applied when within the limits. Must be non-negative. If not specified,

the default value is 1.0.

Node Property Summary

	Type	Description	Default value
constraints	`number[]`	Array of indices in the document-level constraint array. Must be integers.	Required, no default
nodeA	`number`	Node index of one physics body used by the joint. Must be an integer.	Required, no default
nodeB	`number`	Node index of one physics body used by the joint. Must be an integer.	Required, no default

Constraints

The `"constraints"` property defines the joint constraints to use for this joint. Must be an array of integers which are the index of the constraint in the document-level array.

If a joint node has multiple joint constraints that overlap each other, later constraints should override the previous constraints.

Node A and Node B

Each of these properties defines one of two bodies used by the joint. Must be an integer which is the index of the glTF node, and that node must be a physics body defined using the `OMI_physics_body` spec.

At least one of the connected nodes must be a physics body using rigid body physics, meaning `OMI_physics_body` set to "rigid" or "vehicle". If neither node is using rigid body physics, the joints will not work. When physics body nodes are joined, they should not collide with each other. One or zero of the nodes can be a solid body that cannot be moved by joints, meaning `OMI_physics_body` set to "static", "kinematic" or "character". A connected node cannot be null, cannot be a non-body, and cannot be a trigger body.

The position on the body on which the joint operates is determined by the position of the joint relative to each body node. As such, the initial position of the joint node does matter.

Document Property Summary

	Type	Description	Default value
constraints	`object[]`	Array of joint constraint objects.	Required, no default

Special Cases

While a generic joint type is very useful and extensible, it's also worth defining when a generic joint is equivalent to a special-purpose joint. These mappings can be used to convert to more optimized joint types if supported in a physics engine, or as simply a friendly name to display to a user.

In this chart, "fixed" means "constrained with lower and upper limits set to zero", and "free" means "not constrained".

Special case name	Description

Special case name	Description
Fixed / Weld Joint	All axes fixed.
Pin Joint	All linear axes fixed, all angular axes free.
Hinge Joint	One angular axis constrained with non-equal lower and upper limits, the rest are fixed.
Slider Joint	One linear axis and optionally the angular axis around it constrained with non-equal lower and upper limits, the rest are fixed.

JSON Schema

See schema/joint_constraint.schema.json, schema/node.OMI_physics_joint.schema.json, and schema/glTF.OMI_physics_joint.schema.json for the schemas.

Known Implementations

- Godot Engine add-on

Resources:

- Godot Joint3D https://docs.godotengine.org/en/latest/classes/class_generic6dofjoint3d.html
- Unity Joints https://docs.unity3d.com/Manual/Joints.html
- Wikipedia Stiffness https://en.wikipedia.org/wiki/Stiffness