MuJoCo plugin API reference

API functions that mainly serve to inject MuJoCo XML code

simMujoco.addFlexcomp

Description	Adds or injects a flexcomp to the MuJoCo world
Lua synopsis	int injectionId=simMujoco.addFlexcomp(map info)
Lua arguments	**info**: an information field that contains: **element**: the name of the XML element where the flexcomp XML code should be injected. Can be empty if shapeHandle is not empty **shapeHandle**: the handle of the shape where the flexcomp XML code should be injected. Can be empty if element is not empty **name**: the flexcomp name element. Make sure the name is unique, and possibly derived from the related shape **type**: the flexcomp type element. Only grid is currently supported **count**: the flexcomp count element **spacing**: the flexcomp spacing element **radius**: the flexcomp radius element **mass**: the flexcomp mass element **pose**: the flexcomp pose element **pin**: the flexcomp pin element (array of indices) **extraFlexcompXml**: additional xml code to be added to the flexcomp element (as attribute) **extraXml**: additional xml code to be added as a flexcomp element node **callback**: a callback function that is called when the MuJoCo world gets regenerated. This allows to adjust the flexcomp on-the-fly
Lua return values	**injectionId**: the id of the injection
Python synopsis	int injectionId=simMujoco.addFlexcomp(dict info)

simMujoco.composite

Description	Adds or injects a composite to the MuJoCo world
Lua synopsis	int injectionId=simMujoco.composite(string xml,map info)
Lua arguments	**xml**: the xml code corresponding to a composite object **info**: additional information about the composite: **element**: the name of the XML element where the composite XML code should be injected. Can be empty if shapeHandle is not empty **shapeHandle**: the handle of the shape where the composite XML code should be injected. Can be empty if element is not empty **prefix**: the composite prefix string. Make sure the prefix is unique, and possibly derived from the related shape **type**: the composite type. In newest Mujoco version only grid, and cable are supported types **count**: the size of the composite **respondableMask**: the composite respondable mask **grow**: the amount the composite nodes should grow for visuals **callback**: a callback function that is called when the MuJoCo world gets regenerated. This allows to adjust the composite code on-the-fly
Lua return values	**injectionId**: the id of the injection
Python synopsis	int injectionId=simMujoco.composite(string xml,dict info)

simMujoco.getCompositeInfo

Description	Retrieves data about a composite
Lua synopsis	map info=simMujoco.getCompositeInfo(int injectionId,int what)
Lua arguments	**injectionId**: the injection ID returned by simMujoco.composite **what**: the type of requested data: **0**: the positions of the composite nodes **1**: the poses of the composite nodes **2**: the triangles to render the composite **3**: the grown triangles to render the composite
Lua return values	**info**: the requested data
Python synopsis	dict info=simMujoco.getCompositeInfo(int injectionId,int what)

simMujoco.getFlexcompInfo

Description	Retrieves data about a flexcomp
Lua synopsis	map info=simMujoco.getFlexcompInfo(int injectionId,int what)
Lua arguments	**injectionId**: the injection ID returned by simMujoco.addFlexcomp **what**: the type of requested data: **0**: the positions of the flexcomp nodes **1**: the triangles to render the flexcomp
Lua return values	**info**: the requested data
Python synopsis	dict info=simMujoco.getCompositeInfo(int injectionId,int what)

simMujoco.getInfo

Description	Retrieves general information
Lua synopsis	string info=simMujoco.getInfo(string what)
Lua arguments	**what**: the information type. Currently only 'nameAndIds' and 'bodies' is supported
Lua return values	**info**: the requested information
Python synopsis	string info=simMujoco.getInfo(string what)

simMujoco.addInjection

Description	Adds or injects XML code into a MuJoCo world
Lua synopsis	int injectionId=simMujoco.addInjection(map info)
Lua arguments	**info**: information field containing: **shapeHandle**: the handle of the shape where the XML code should be injected. Can be empty if element is not empty **element**: the name of the XML element where the XML code should be injected. Can be empty if shapeHandle is not empty **xml**: the XML code to inject **callback**: a callback function that is called when the MuJoCo world gets regenerated. This allows to adjust the xml code on-the-fly, if needed

Lua return values	**injectionId**: the id of the injection
Python synopsis	int injectionId=simMujoco.addInjection(dict info)

simMujoco.removeInjection

Description	Removes XML code from the MuJoCo world
Lua synopsis	simMujoco.removeInjection(int injectionId)
Lua arguments	**injectionId**: the injection ID returned by simMujoco.addFlexcomp, simMujoco.composite, or simMujoco.addInjection
Lua return values	
Python synopsis	simMujoco.removeInjection(int injectionId)