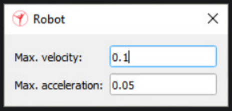
User config callback functions

A customization script that includes a user config callback function (which is one of many system callback functions), displays a configuration dialog icon in the scene hierarchy:

[Configuration dialog icon]

For convenience, the icon is also displayed for the customization script's parent. When double-clicked, the user config callback function is triggered. This can be used as a convenient way of displaying a custom user interface, that is specific to the object/model the customization script refers to. User data can be read and written to objects with custom data properties for instance:


Robot

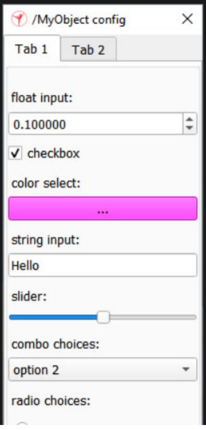
Max. velocity: 0.1
Max. acceleration: 0.05

[Custom configuration dialog]

Python | Lua

```python
#python

def sysCall_init():
    sim = require('sim')
    simUI = require('simUI')
    self.modelHandle = sim.getObject('..')

def sysCall_userConfig():
    xml = '''<ui title="Robot" closeable="true" modal="true" layout="form" on-close="customUiClosed">
            <label text="Max. velocity:" />
            <edit id="1" value="-" on-editing-finished="velocityChanged"/>
            <label text="Max. acceleration:" />
            <edit id="2" value="-" on-editing-finished="accelerationChanged"/>
            </ui>'''
    ui = simUI.create(xml)
    data = readData()
    simUI.setEditValue(ui, 1, str(data['maxVel']))
    simUI.setEditValue(ui, 2, str(data['maxAccel']))

def customUiClosed(ui):
    simUI.destroy(ui)

def velocityChanged(ui, id, val):
    data = readData()
    val = float(val)
    if val:
        if val < 0.1:
            val = 0.1
        if val > 0.5:
            val = 0.5
        data['maxVel'] = val
    simUI.setEditValue(ui, id, str(data['maxVel']))
    writeData(data)

def accelerationChanged(ui, id, val):
    data = readData()
    val = float(val)
    if val:
        if val < 0.01:
            val = 0.01
        if val > 0.2:
            val = 0.2
        data['maxAccel'] = val
    simUI.setEditValue(ui, id, str(data['maxAccel']))
    writeData(data)

def readData():
    data = sim.unpackTable(sim.getBufferProperty(self.modelHandle, 'customData.RobotParams'))
    if data == [] or data == {}:
        data = {}
        data['maxVel'] = 0.2
        data['maxAccel'] = 0.05
    return data

def writeData(data):
    sim.setBufferProperty(self.modelHandle, 'customData.RobotParams', sim.packTable(data))
```

You may also use the *configUI* module for quicker set-up of custom configuration dialog, that will also automatically write custom data to objects:


/MyObject config

Tab 1 | Tab 2

float input: 0.100000

✓ checkbox

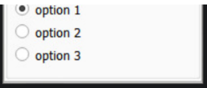
color select: ...

string input: Hello

slider:

combo choices: option 2

radio choices:


[✓ option 1]
option 2
option 3

[Custom configuration dialog based on *configUI* module]

```lua
--lua

require('configUi')

function sysCall_init()
    local schema = {
        floatInput = {
            default = 0.1,
            maximum = 4,
            minimum = 0.05,
            name = "float input",
            type = "float",
            ui = {
                control = "spinbox",
                tab = "Tab 1", order = 1, col = 1
            },
        },
        intInput = {
            default = 1,
            maximum = 10,
            minimum = 0,
            name = "int input",
            type = "int",
            ui = {
                control = "spinbox",
                tab = "Tab 1", order = 2, col = 2
            },
        },
        checkbox1 = {
            type = 'bool',
            name = 'checkbox1',
            default = true,
            ui = {tab = "Tab 1", order = 3, col = 1},
        },
        checkbox2 = {
            type = 'bool',
            name = 'checkbox2',
            default = false,
            ui = {tab = "Tab 1", order = 4, col = 2},
        },
        colorselect = {
            type = 'color',
            name = 'color select',
            default = {1, 0, 1},
            ui = {tab = "Tab 2", order = 5, col = 1},
        },
        stringInput = {
            default = "Hello",
            name = "string input",
            type = "string",
            ui = {tab = "Tab 2", order = 6, col = 2},
        },
        floatSlider = {
            default = 0.5,
            maximum = 1,
            minimum = 0,
            name = "float slider",
            type = "float",
            ui = {
                control = "slider",
                tab = "Tab 2", order = 7, col = 1
            },
        },
        intSlider = {
            default = 1,
            maximum = 10,
            minimum = 0,
            name = "int slider",
            type = "int",
            ui = {
                control = "slider",
                tab = "Tab 2", order = 8, col = 2
            },
        },
        choices1 = {
            default = 2,
            choices = {"option 1", "option 2", "option 3"},
            name = "combo choices",
            type = "choices",
            ui = {
                control = "combo",
                tab = "Tab 3", order = 9, col = 1
            },
        },
        choices2 = {
            default = 1,
            choices = {"option 1", "option 2", "option 3"},
            name = "radio choices",
            type = "choices",
            ui = {
                control = "radio",
                tab = "Tab 3", order = 10, col = 1
            },
        },
    }
    sim.setBufferProperty(sim.getObject('..'), 'customData.__schema__', sim.packTable(schema))
    configUi = ConfigUI('myModelType', nil, modificationCallback)
    configUi.getObjectCallback = function() return sim.getObject '..' end
end

function modificationCallback(config)
    local objectHandle = sim.getObject('..')
    local txt = "Object '"..sim.getObjectAlias(objectHandle,5).."' just changed.\nNew parameters are:"
    print(txt)
    print(config)
end
```