Simulace v programu v-rep ovládané přes Spyder v jazyce python.

Předmět: Projekt

Paralelka: Úterý 15:00

Dne: 15. 11. 2016, Praha

Autoři: Aleš Tomek, Jaroslav Petráš, Tomáš Hron

Obsah

Úvod

Tato práce je součástí předmětu Projekt, ve kterém se náš tým společnými silami má seznámit se simulačním počítačovým programem v-rep. Dále s programem Spyder, který nám slouží jako programovací prostředí pro psaní ovládacího scriptu v programovacím jazyce Python.

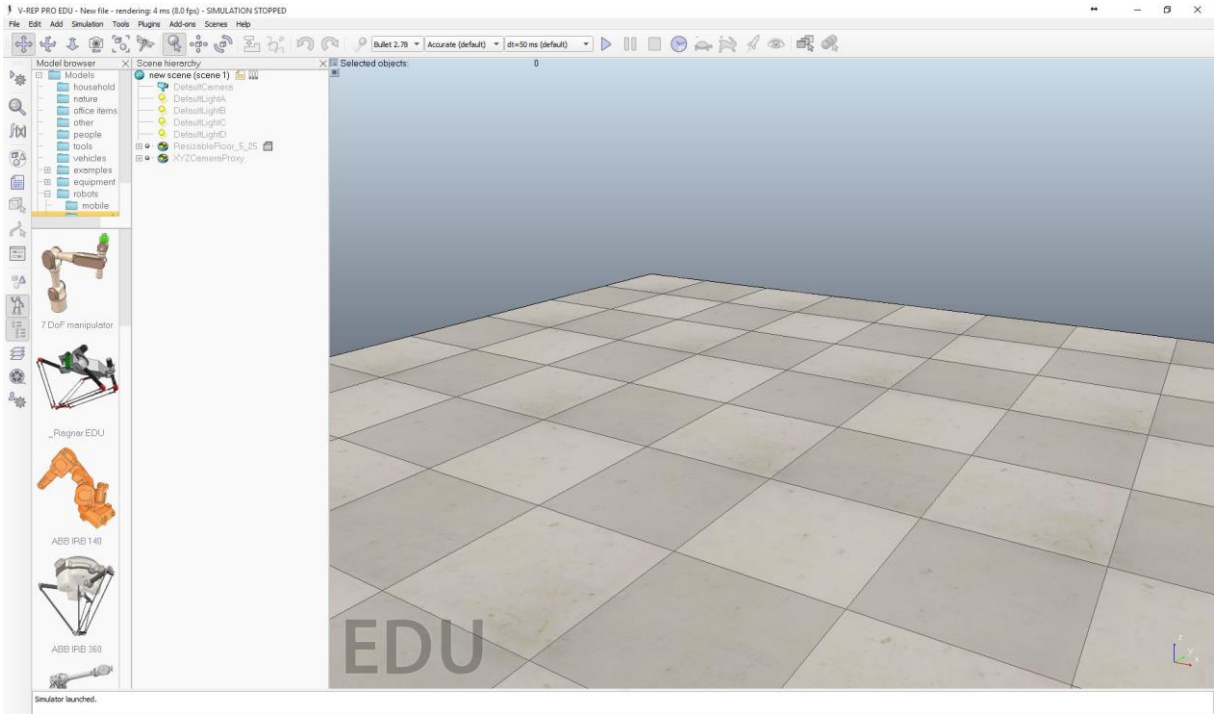
Prvotním úkolem této práce bude propojení obou programů a to tak, aby se vytvořená simulace v programu v-rep pohybovala ne podle vlastního nastavení, avšak podle nastavení nakódovaném ve Spyderu. Dalšími úkoly, kterým se v této práci budeme věnovat, je popis samotného scriptu a dalších, pro funkčnost potřebných úkonů, aby dané akční členy fungovaly tak, jak budeme požadovat.

Díky této práci by měl být každý schopen ovládat elementární akční členy, pomocí kterých se dají ovládat nejrůznější stroje a jiné mechanické prvky libovolné výroby. Dále je našim cílem popsat všechny naše problémy, se kterými jsme se během projektu setkali a jejichž řešení nám trvalo mnohdy i několik hodin, z důvodu případného vyvarování se jím.

Programy

Prvním používaným programem je 3D simulační program v-rep, kde je bohatá škála vymodelovaných robotů, senzorů, pracovníků a jiných, ať už pohyblivých nebo nepohyblivých, objektů. Z těchto objektů lze sestavit téměř libovolnou virtuální realitu, která je v našem případě přizpůsobená rozsahu tohoto předmětu. Nicméně je v tomto programu možné sestavit například celou výrobní linku nebo dokonce výrobní halu pro danou firmu. Lze zde ovládat objekty pomocí předpřipravených skriptů přímo v programu, s čímž se ale zabývat nebudeme.

Tento program je ve verzi Education volně ke stažení na webových stránkách výrobce (odkaz ve zdrojích). Jeho vizualizační prostředí můžeme vidět na obrázku [1].

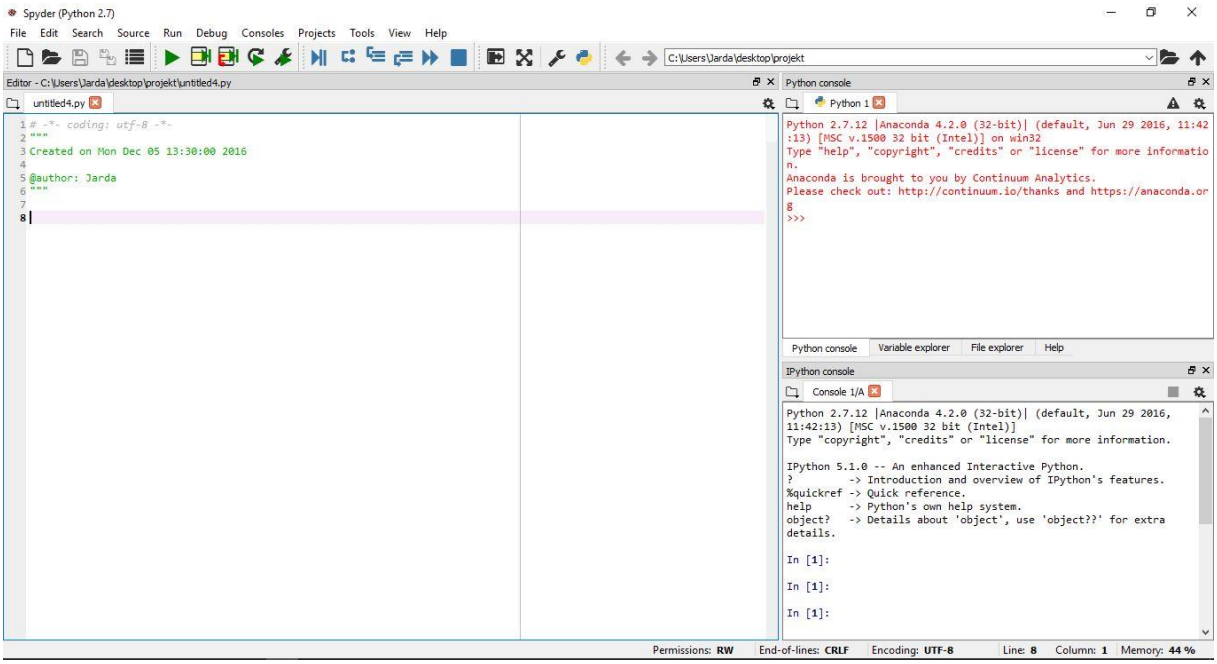


Obrázek [1]: Prostředí v-repu

Druhým, pro tuto simulaci nezbytným, programem je tzv. „open source“ program Spyder, který je součástí programu Anaconda, ze kterého se čerpají veškeré knihovny a jiné potřebné prvky pro samotný chod jazyku Python.

Zde je však velmi důležité, aby v-rep a Anaconda byly ve stejné bitové verzi, jinak není možné vzájemné propojení obou programů. Tato neznalost byla důvodem několikahodinového zdržení. Protože pro operační systém Windows, ve kterém jsme pracovali, je k dispozici pouze 32-bitová verze programu v-rep, bylo za potřebí nainstalovat i Anacondu ve stejné verzi.

Samotné uživatelské rozhraní programu Spyder, který je ke stažení v rámci Anacondy v odkazu ve zdrojích, můžeme vidět níže na obrázku [2].



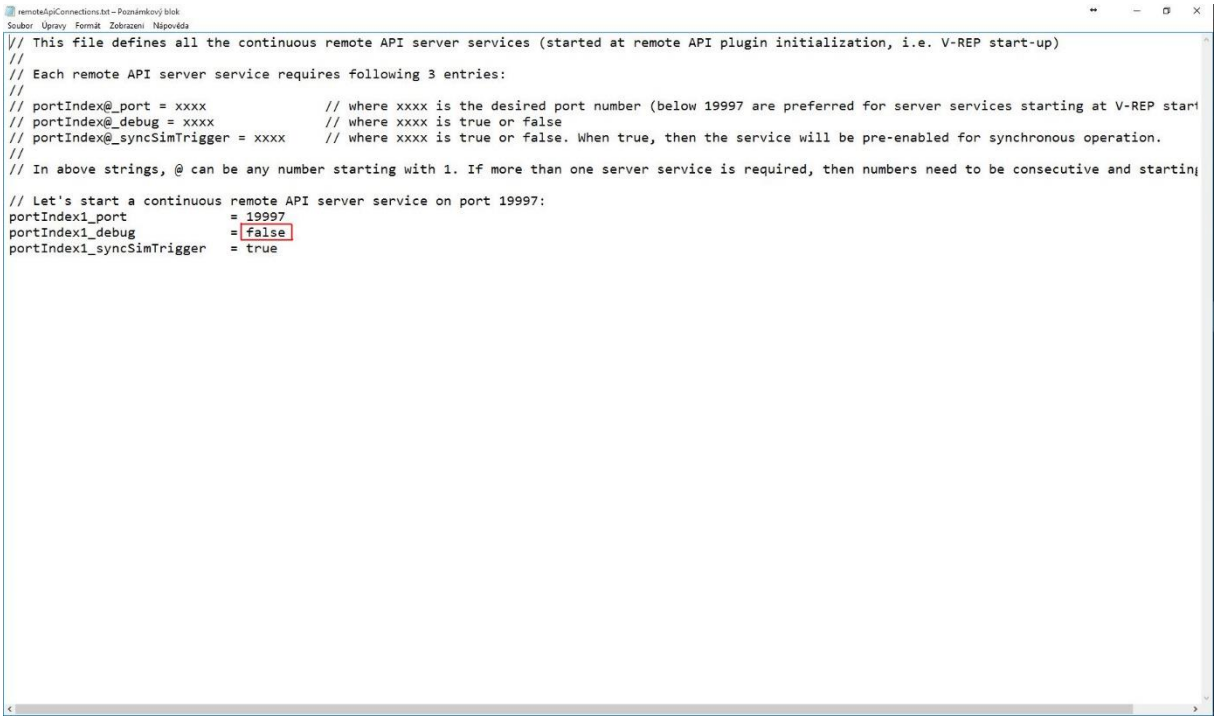
Obrázek [2]: Prostředí Spyderu

Propojení

Aby Spyder uměl pracovat s knihovnami, týkající se objektů ve v-repu, je zapotřebí si vytvořit novou složku, v našem případě s názvem „Projekt“, do které se budou ukládat všechny skripty.py a právě do ní překopírovat obsah z nainstalované složky C:/Program Files (x86)/V-REP3/V-REP_PRO_EDU/programming/remoteApi-Bindings/python/python.

Následně je nutné povolit samotné spojení a to tak, že se ve složce s nainstalovaným v-repem otevře jako textový soubor „remoteApiConnections.txt“, který je vidět na obrázku [3]. Zde se musí místo červeně označeného slova „false“ napsat slovo „true“ a soubor uložit. Tato malá změna způsobí to, že

se po zapnutí programu v-rep otevře druhé okno, které bude čekat na spojení a následně bude zobrazovat komunikaci obou programů.



Obrázek [3]: Textový dokument pro provedení změny

Výše zmíněná komunikace se spustí následným scriptem.py, který si také uložíme do naší složky s názvem „Projekt“.

```
1 import vrep
2 import sys
3  # Propojení s klientem V-REP
4 # close any open connections
5 vrep.simxFinish(-1)
6 # Connect to the V-REP continuous server
7 clientID = vrep.simxStart('127.0.0.1', 19997, True, True, 500, 5)
8
9 if clientID != -1: # if we connected successfully
10     print 'Connected to remote API server'
11
12 else:
13         print 'Connection not successful'
14         sys.exit('Could not connect')
15
```

Pokud spojení proběhne úspěšně, komunikační okno zobrazí zprávu „Connected to remote API server“.

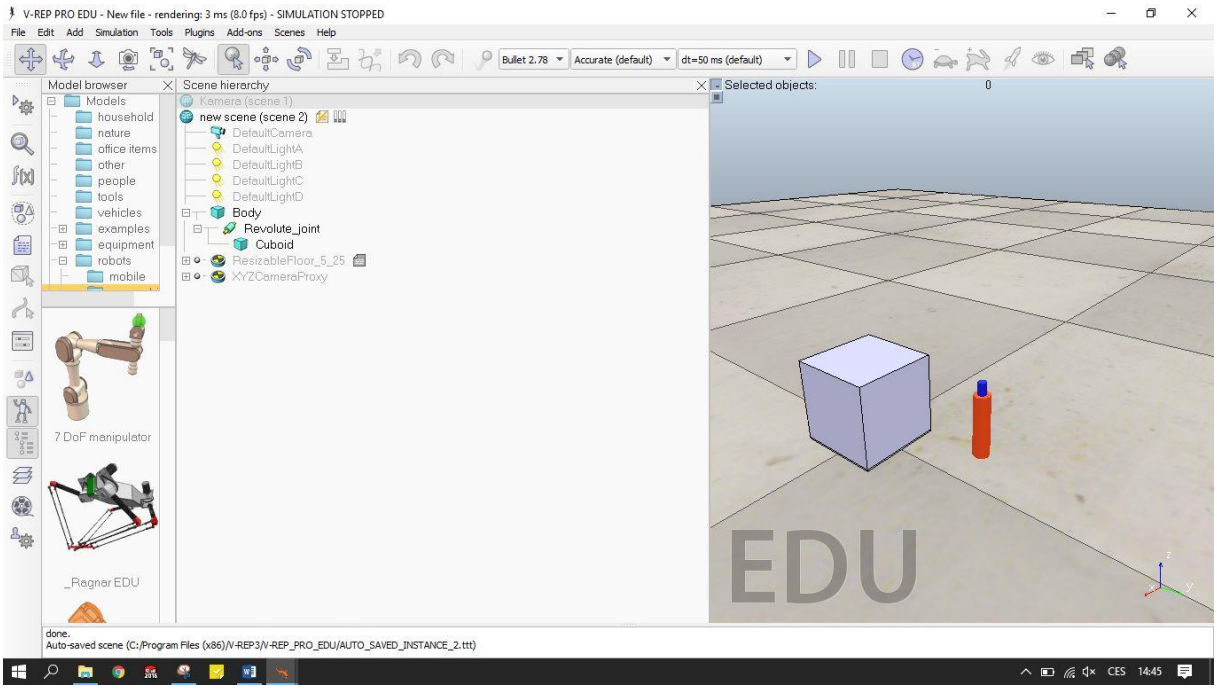
Pohybování objekty

Po sladění a následného propojení obou programů, přišlo na řadu pohybování elementárními prvky, které jsou součástí jakéhosi toolboxu přímo ve v-repu, pomocí rotačního a lineárního kloubu.

Rotační kloub – revolute joint

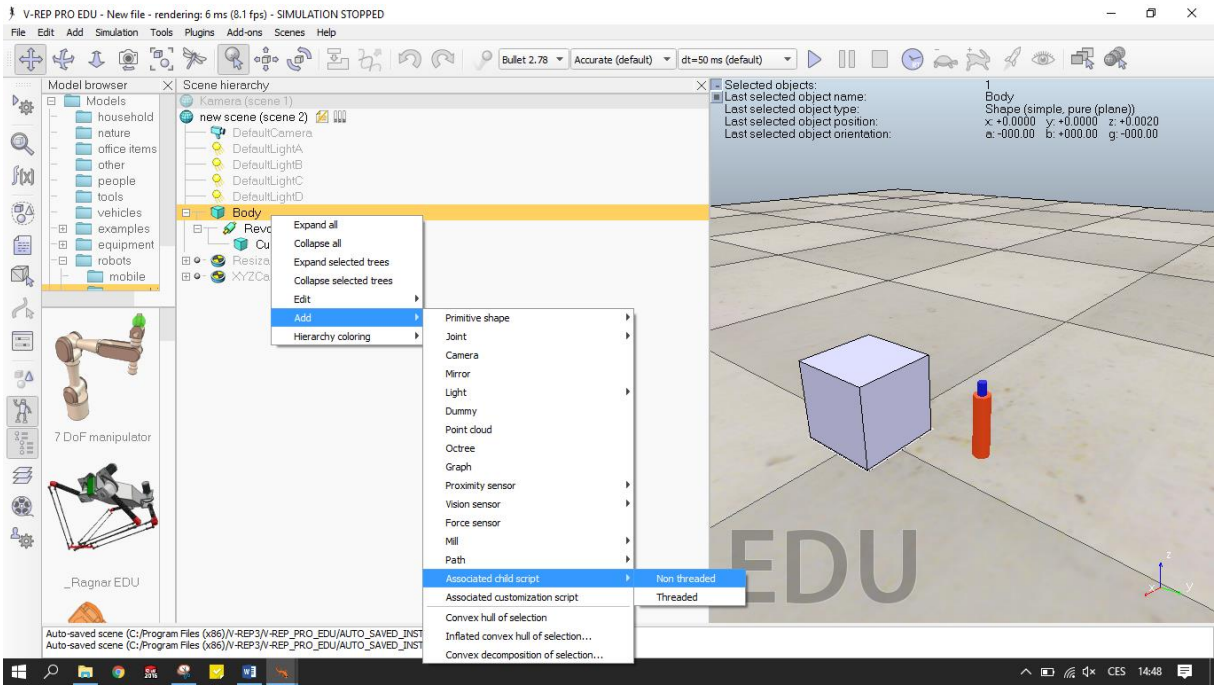
Pro správnou funkci rotace objektů je nutné dodržovat korektní hierarchii stromového adresáře. Nejprve říkáme „čím otáčíme“ (revolute joint) a pak říkáme „co otáčíme“ (cuboid). To souvisí s tím co do čeho vložit v adresáři samotného v-repu, který můžeme vidět na obrázku [4].

Dobré je podotknout, že osa otáčení záleží na vložení otočného kloubu do prostředí v-repu (do osy x y nebo z) my máme motor v ose z, takže se nám kostka bude točit „na podlaze“.

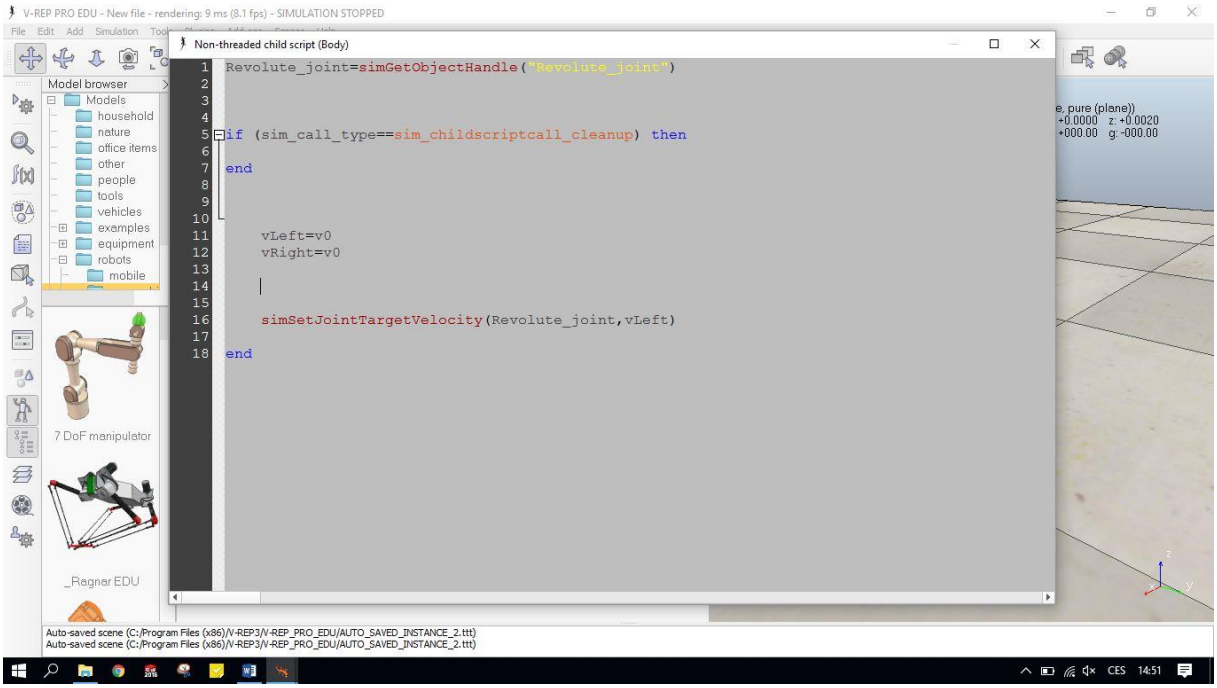


Obrázek [4]: Ukázka vložení a přiřazení rotačního kloubu kostce

Dále je nutné nadefinovat hlavní script pro celou soustavu. Viz obrázek [5], kde vidíme cestu k otevření scriptu z obrázku [6]. Názvy jednotlivých prvků musí souhlasit s názvy ve stromovém adresáři.

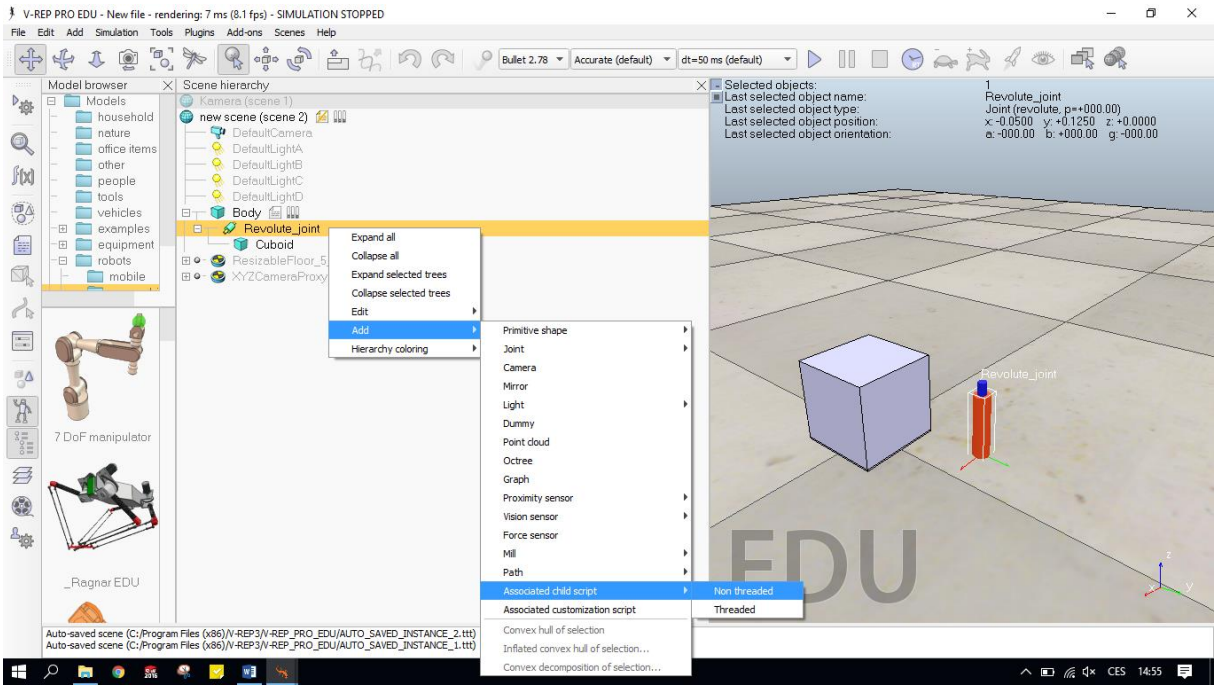


Obrázek [5]: Cesta k v-repovému scriptu

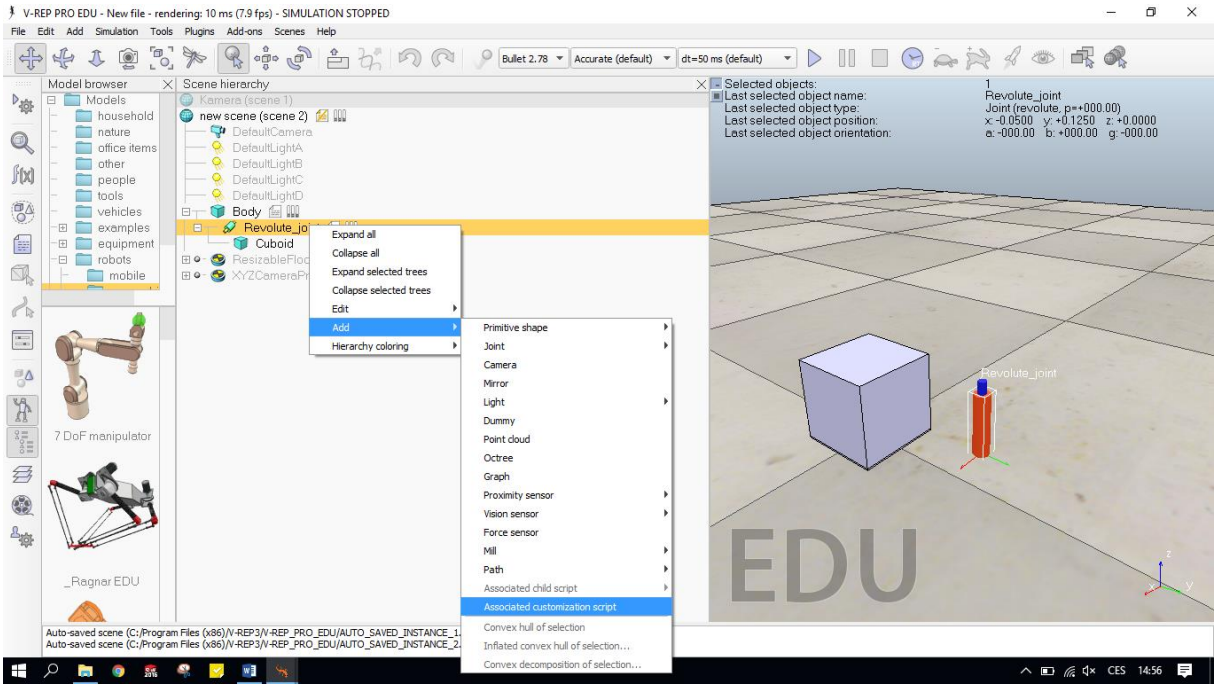


Obrázek [6]: V-repový script rotačního kloubu

Dále je nutné vložit scripty i pro použité rotační klouby. Postup můžeme vidět na obrázku [7] a [8]. V tuto chvíli jsou nadefinované scripty, se kterými se může dále pracovat v pythonu.

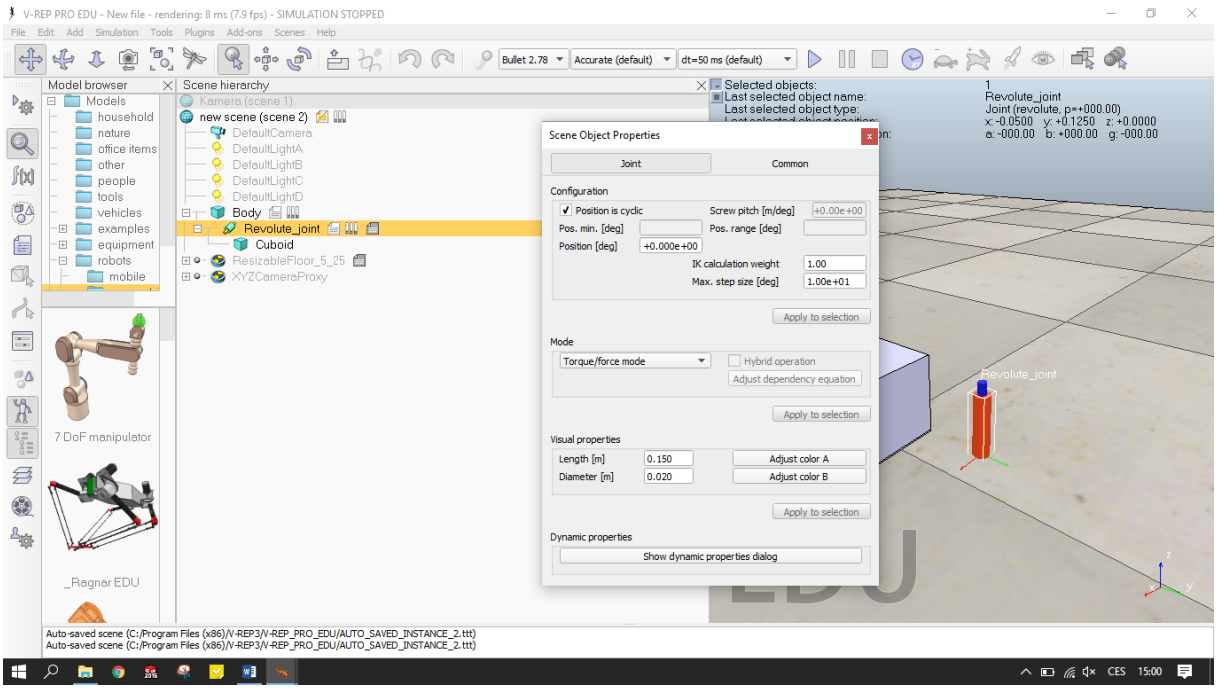


Obrázek [7]: Přiřazení scriptu, který se následně zakáže

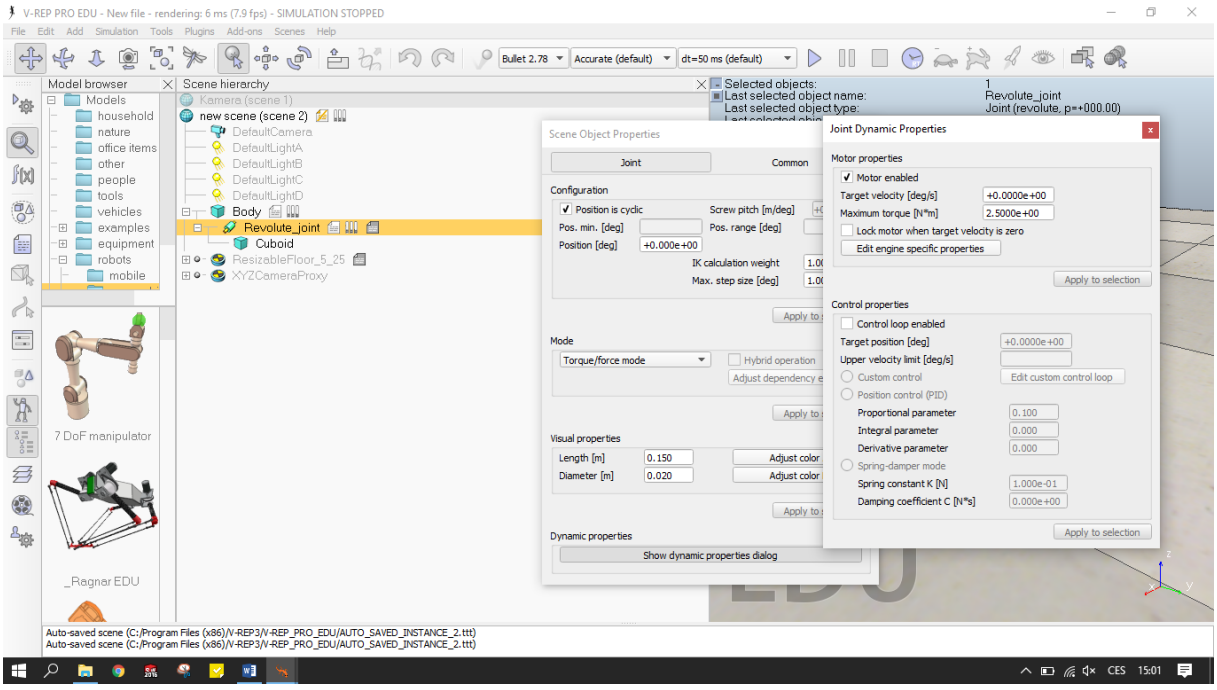


Obrázek [8]: Přiřazení dalšího scriptu, který se také následně zakáže

Nyní je zapotřebí rotační kloub zaktivovat pomocí dynamických vlastností. Nejprve se dvojklikem na motor v adresáři zobrazí tabulka viz obrázek [9]. Zde se klikne na „Show dynamic properties dialog“ a následně zaškrtne políčko „motor enabled“, což můžeme vidět na obrázku [10].

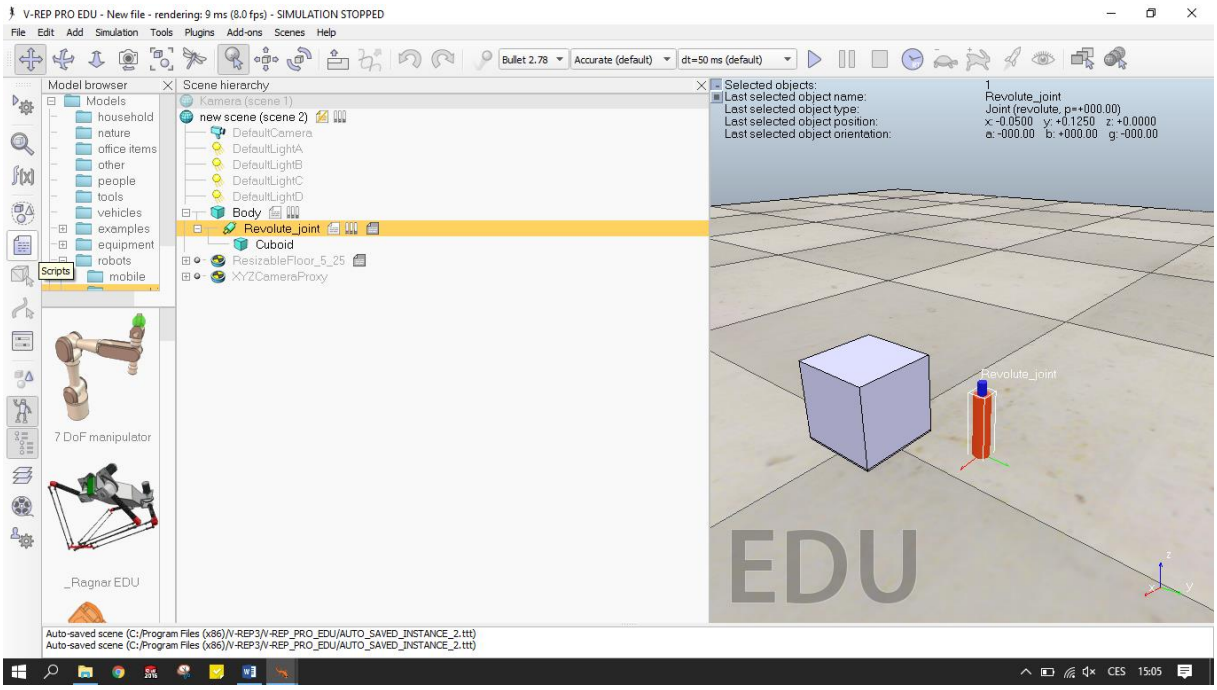


Obrázek [9]: Cesta k aktivaci kloubu

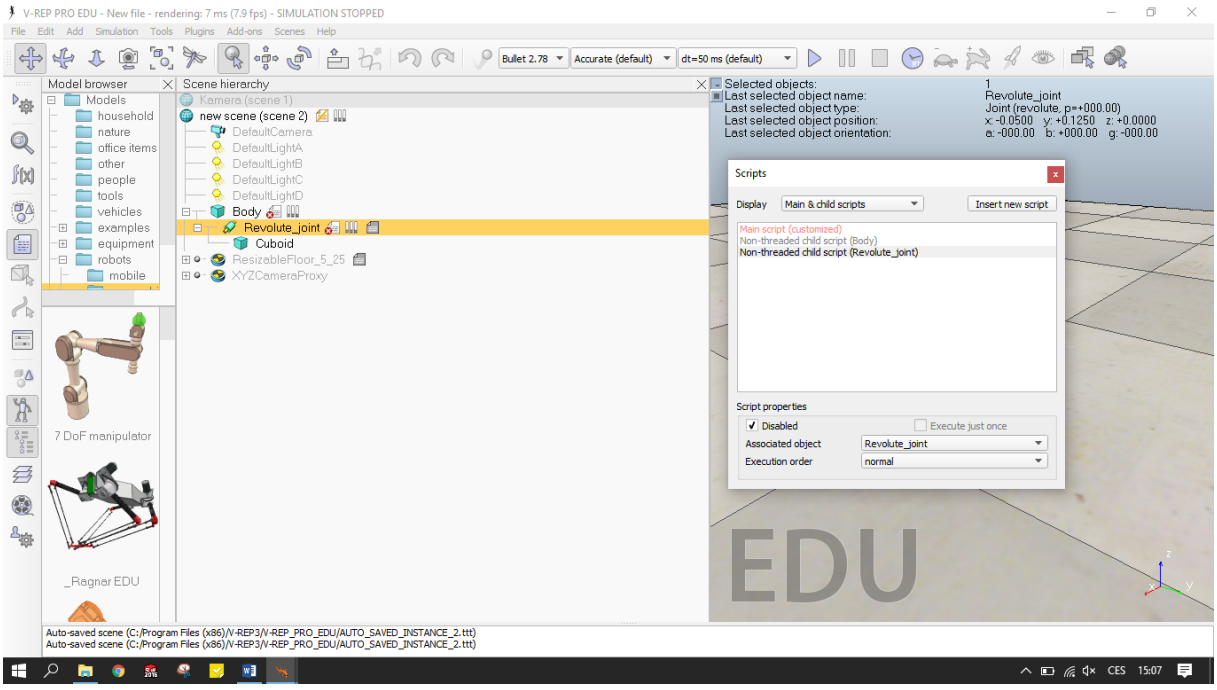


Obrázek [10]: Samotná aktivace kloubu

Nyní je třeba jako poslední krok ve v-repu zakázat scripty, které ovládají defaultní pohyby objekty, které chceme ovládat pomocí pythonu. V levém panelu klikneme na „Scripts“ viz obrázek [11]. následně zakážeme scripty v tabulce pro „Body“ a pro „revolute_joint“ zaškrtnutím tlačítka „disabled“ viz obrázek [12]. Pro kontrolu by se po správné deaktivaci scriptu měli v adresáři u daného prvku objevit červené křížky.



Obrázek [11]: Postup k zakázání scriptu



Obrázek [12]: Zakázání scriptu

Nyní je ve v-repu vše připraveno a jde se do Spyderu. Scripty se navážou následně za předpřipravený script ohledně propojení v-rep - Spyder viz strana [5] tohoto dokumentu.

Pro inicializaci rotační vazby z v-repu se použije následující script:

```
errorCode,Revolute_joint_handle=vrep.simxGetObjectHandle(clientID,'Revolute_joint',vrep.simx_opmode_oneshot_wait)
```

Je opět nutné, aby ve scriptu souhlasily názvy objektů tak, jak jsou uvedeny ve v-repu, jak ve stromovém adresáři, tak v hlavním scriptu.

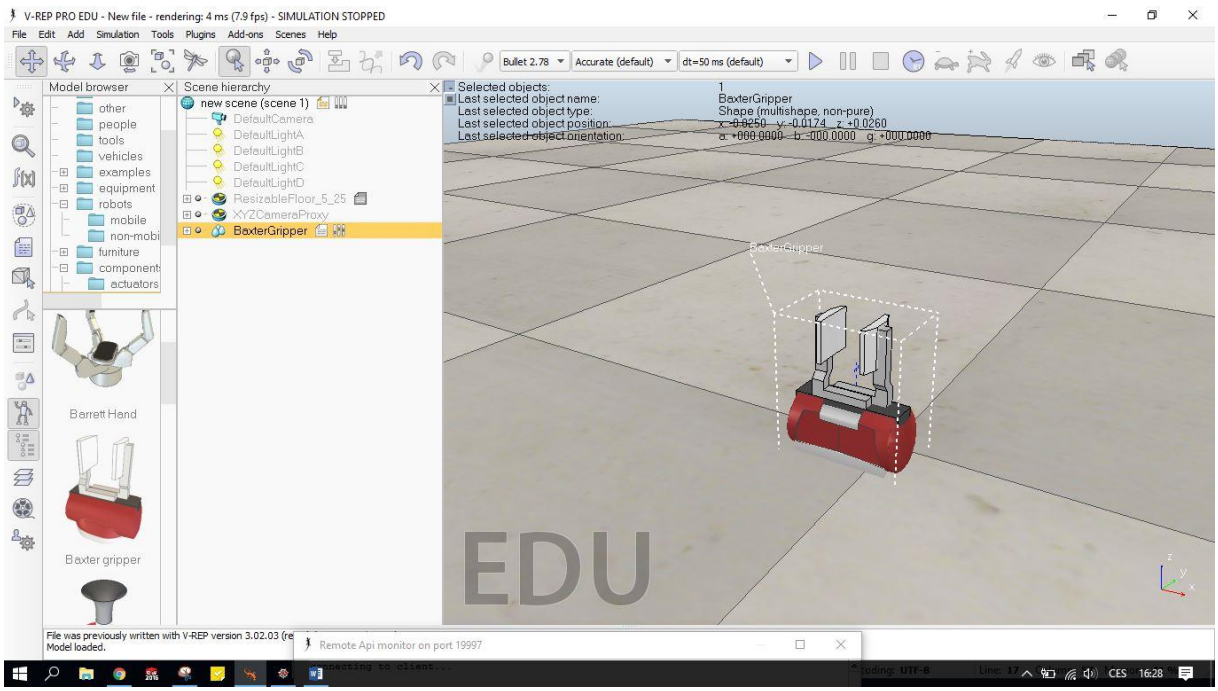
Pro rozpohybování, určení rychlosti pohybujících se objektů pomocí rotace následuje script:

```
vrep.simxSetJointTargetVelocity(clientID,Revolute_joint_handle,2.0, vrep.simx_opmode_streaming)
```

Opět je nutné ve scriptu dodržovat stejné názvy objektů viz předchozí text. Rychlost otáčení se vypisuje v tomto scriptu jedinou číselnou hodnotou. Jednotky deg/s.

Lineární posuv – prismatic joint

Opět jsme vybrali elementární objekt (kostku), které jsme přiřadili lineární posuv. Pomocí pythoního scriptu jsme se s kostkou pokoušeli pohybovat po ose, avšak bez úspěchu. Následně jsme do virtuálního prostředí vložili robotické kleště, které už takovýto posun měly v sobě přiřazený. Použili jsme opět náš script na posuvný pohyb, který zde náhle fungoval. Ovládané kleště můžeme vidět na obrázku [13].

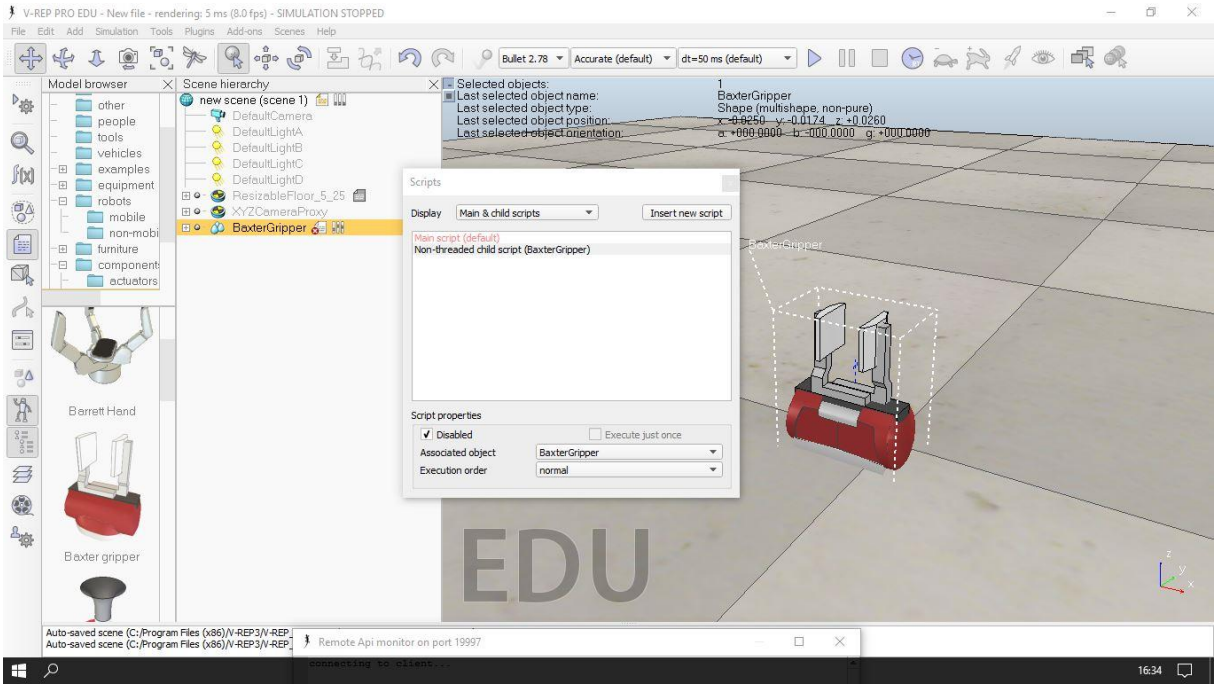


Obrázek [13]: Vložené robotické kleště z toolboxu s přiřazeným lineárním posuvem

Na následujícím obrázku (obrázek [14]) se můžeme podívat na ovládací script, který ve v-repu ovládá robotické kleště a je ho potřeba opět zakázat, aby ovládání mohlo probíhat přes náš program spyder. Toto zakázání můžeme vidět na obrázku [15].

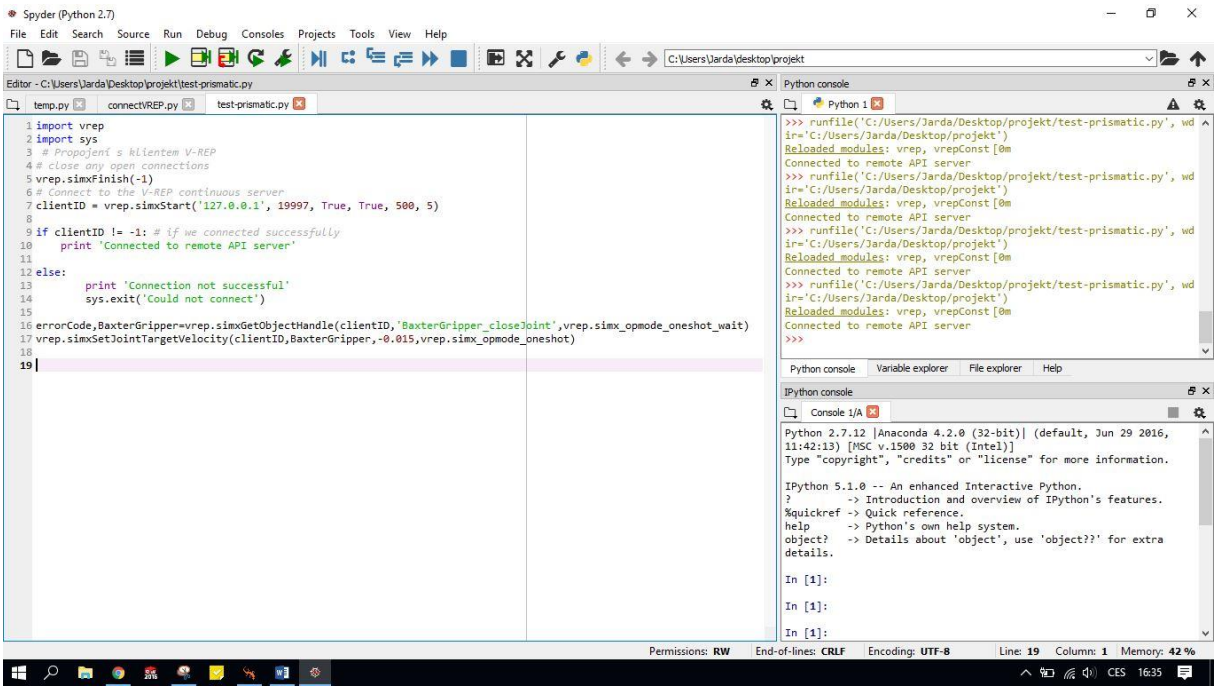
```
if (sim_call_type==sim_childscriptcall_initialization) then
    gripperHandle=simGetObjectHandle('BaxterGripper')
    objectName=simGetObjectName(gripperHandle)

    motorHandle=simGetObjectHandle('BaxterGripper_closeJoint')
    -- the 'BaxterGripper_centerJoint' is handled by a joint callback script
    openedGap=simGetScriptSimulationParameter(sim_handle_self,'openedGap')
    closedGap=simGetScriptSimulationParameter(sim_handle_self,'closedGap')
    interval={0.0362-openedGap,openedGap-closedGap}
    simSetJointInterval(motorHandle,false,interval)
end

if (sim_call_type==sim_childscriptcall_cleanup) then

end

if (sim_call_type==sim_childscriptcall_actuation) then
    close=simGetIntegerSignal(objectName..'_close')

    if (close==1) then
        simSetJointTargetVelocity(motorHandle,0.005)
    else
        simSetJointTargetVelocity(motorHandle,-0.005)
    end
```

Obrázek [14]: V-repový script na ovládání lineárního pohybu kleští



Obrázek [15]: Zakázání v-repového scriptu

Na dalším obrázku (obrázek [16]) se opět vracíme do programu spyder, kde se můžeme podívat na ovládací script. Ten se skládá opět z propojovacího scriptu na komunikaci mezi oběma programy a dále z příkazu na náš posuvný pohyb.



Obrázek [16]: Ovládací pythoní script

Inicializace pohonu, která je vidět v předchozím obrázku:

```
errorCode,BaxterGripper=vrep.simxGetObjectHandle(clientID,'BaxterGripper_closeJoint',vrep.simx_opmode_oneshot_wait)
```

Nadefinování rychlostí pohybu je obsaženo zde:

```
vrep.simxSetJointTargetVelocity(clientID,BaxterGripper,-0.015,vrep.simx_opmode_oneshot)
```

Měníme-li znaménko u hodnoty, která nám definuje rychlost pohybu, měníme tím smysl pohybu.

Import vlastních modelů

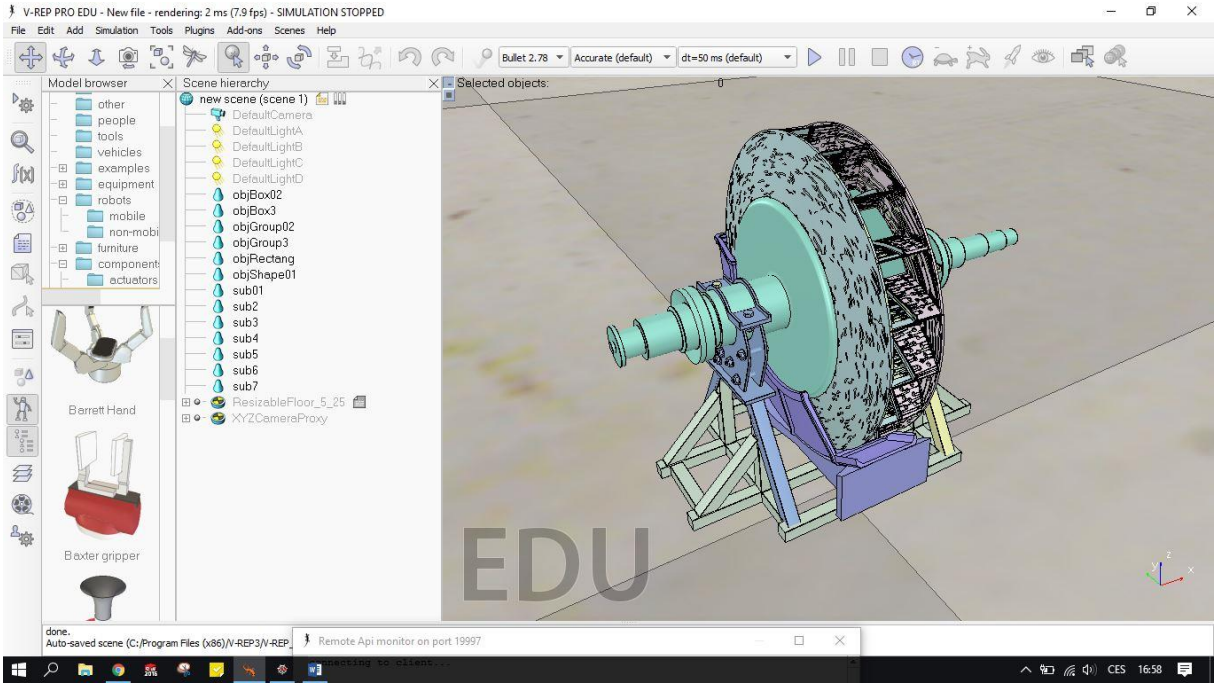
Naprostým základem pro importování vlastních modelů do programu v-rep je typ souboru, pro které je virtuální prostředí naprogramováno. Mezi tyto typy patří například .obj, .dxf, .stl nebo .3ds.

Na základě otestování jsme zjistili že pro práci s objekty je nejvýhodnější používat formát typu 3ds. Je to soubor, který můžeme získat z programu 3DS max. V našem případě, protože jsme tento program neměli, jsme si soubor tohoto typu stáhli z internetu.

Nevýhodou je však ale složitost modelů. Pokud máme složitý model, kde je hodně prvků ve smyslu zaoblení, zkosení atd., nelineárně narůstá počet elementů, které vytvoří po celém modelu polygonální síť. To však ale markantně zpomaluje chod programu, protože se při každém pootočení přepočítává celá sít.

Výhodné je model rozložit na elementární „podmodely“ sloučit je a zjednodušit tak výsledný model. (ideální je vše skládat z elementárních těles typu kostka ,válec, koule,….).

Ukázka naimportovaného modelu je vidět zde na obrázku [17]. Jedná se o lopatkové kolo, kterému jsme přiřadili rotační kloub, pomocí kterého jsme s kolem otáčeli.

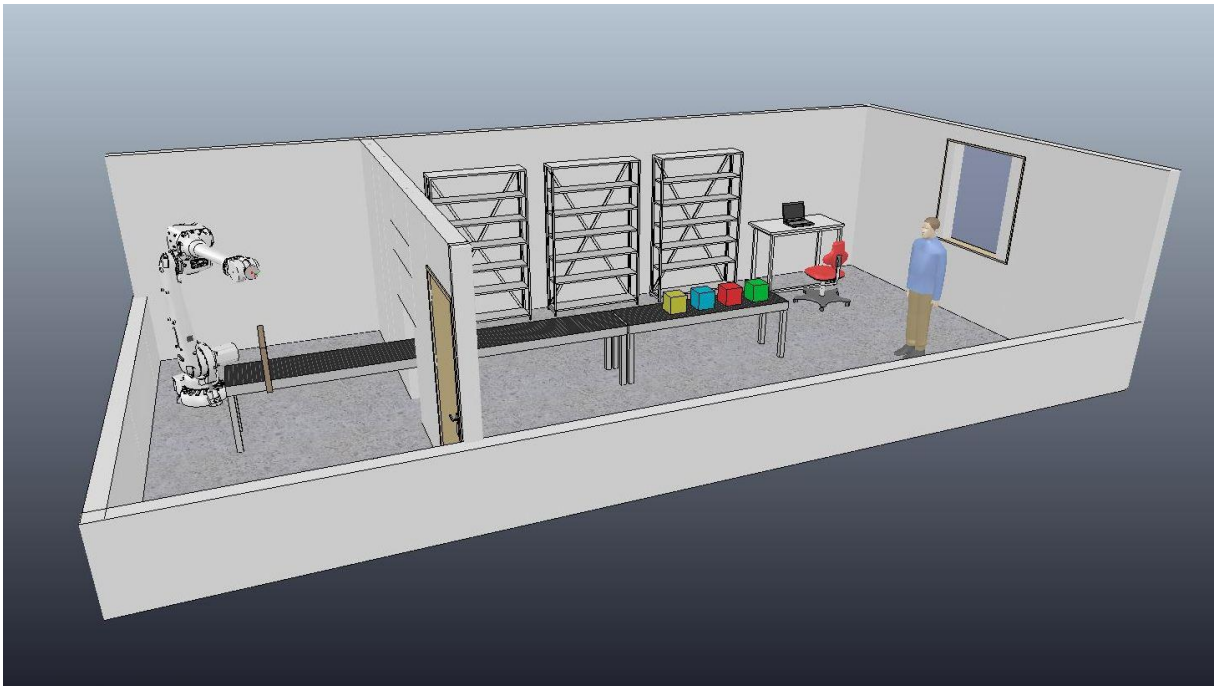


Obrázek [17]: Naimportovaný vlastní model

Závěr

Shrnutím předešlého popisu by se dalo říct, že po úspěšném propojení v-repu se spyderem jsme byli schopni objektu přiřadit rotační kloub a ovládat ho pomocí příkazu z pythonu. Avšak po přiřazení lineárního posuvu jsme pythoním příkazem takovýto objekt nedokázali ovládat. Když jsme ale vybrali nějaký objekt z toolboxu v-repu, který už takovýto posun měl přiřazen, tak tento script fungoval a objektem pohyboval. Z toho jde usoudit, že jsme schopni ovládat rotačním pohybem jak objekt z toolboxu, tak objekt vlastní vložený do v-repu. Lineárním posuvem jsme však ale bohužel schopni ovládat pouze objekt, který už ve v-repu je.

V závěru bychom chtěli ukázat možnou průmyslovou situaci, kde se vyskytuje pásový dopravník, po kterém mohou jezdit výrobky, které v našem případě zjednodušeně představují různě barevné „kostičky“. Na konci tohoto pásu je umístěna jakási laserová brána, která by zaznamenáním výrobku na dané pozici zastavila pás, odkud by ho už přebíral manipulační robot a mohl ho například pokládat někam vedle na zem. Výsledný model dané situace můžeme vidět na obrázku [18].



Obrázek [18]: Možná průmyslová situace

Zdroje

Link na stažení v-rep verze „pro edu“ ..... http://www.coppeliarobotics.com/downloads.html

Link na stažení Anacondy (32 bit) ..... https://www.continuum.io/downloads