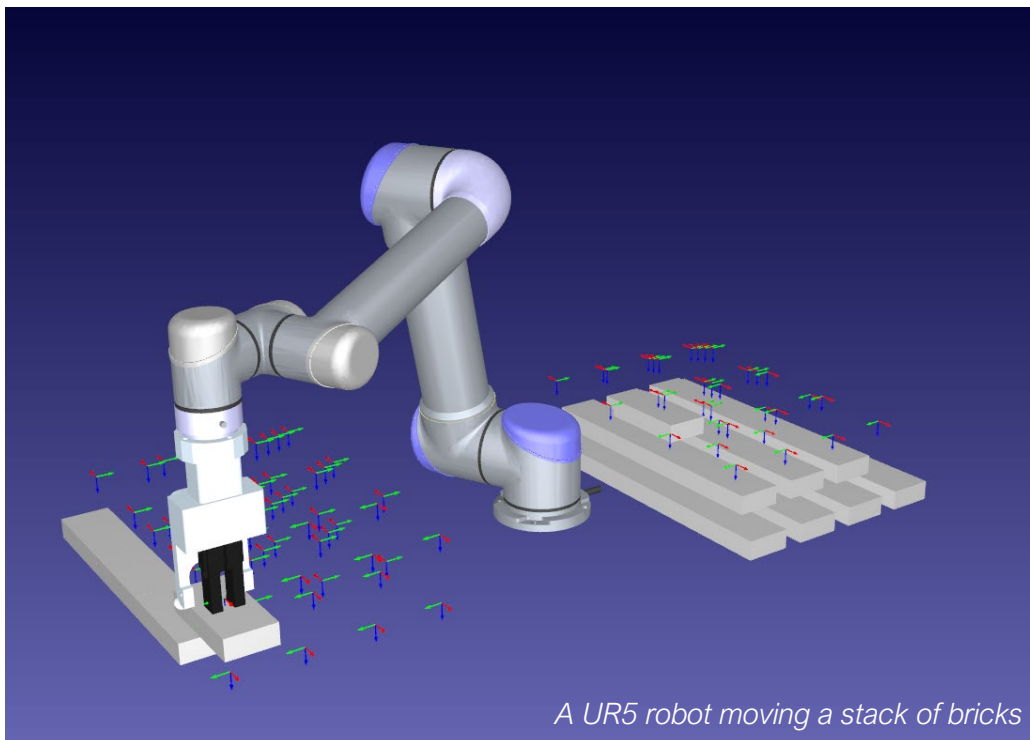


User guide

From Grasshopper to RoboDK

Developed by Job de Vogel

Problem statement: During the Technoledge Design Informatics course, you will learn how to make use of Grasshopper and RoboDK to control a Robot. To help you in this process, several scripts have already been developed for you to help you in your design process. This manual explains you step-by-step how to use them.



This user manual contains:

- Explanation of the process;
- Quick-start Grasshopper to RoboDK;
- In-depth explanation scripts (for Python developers);
- Bug fixing (help... my script doesn't work)







Preparation:

- Install RoboDK, it is highly recommended to install the software in the *default C:/* directory. This will make it easier to use the presented scripts.
- Download the zip-file that contains all the scripts from this tutorial. Your browser may give you a security alert because of the fact that it cannot check the Python scripts for safety issues. After downloading, extract the zip-file. Save the extracted folder to your preferred directory.
- To make use of all the functions, DO NOT move files from the zip folder to other locations. Make sure that *all files are in the same folder*.
- If you are planning to make your own/change Python scripts, it may be useful to download Visual Studio Code.

Zip-folder:

The downloaded zip folder (from Brightspace?) should contain the following files:

- Grasshopper script;
- RoboDK file with prepared Robot setup;
- Python script to transfer geometry from Grasshopper/Rhino to RoboDK;
- Python script to generate a Robot program based on a CSV file.

Name	Status	Date modified	Type ^
 User guide.pdf	✓	10/29/2021 5:29 PM	Adobe Acrobat D...
 Grasshopper_script.gh	✓	10/29/2021 5:33 PM	Grasshopper Defin...
 Generate_csv_program.py	✓	10/29/2021 5:15 PM	Python Source File
 Import_Rhino_geometry.py	✓	10/28/2021 12:12 PM	Python Source File
 UR5_station_default.rdk	✓	10/29/2021 5:32 PM	RoboDK Station M...
 README.txt	✓	10/29/2021 5:35 PM	Text Document

Keep you files in the same directory

Process

The process of working with the provided scripts can be easily explained with the following steps. First we will have to generate some data in Grasshopper. In our case, this will be a **stack of bricks**. Our goal is to use a Robot arm to move the stack of bricks from one side to the other side of a table.

1. Create bricks in Rhino/Grasshopper, each brick has its own origin plane;
2. Based on the origin plane and the drop plane (where the robot should move the brick to), we have to generate 6 different planes in the following order:

Location A: approach plane above brick
Robot moves to a plane above the brick

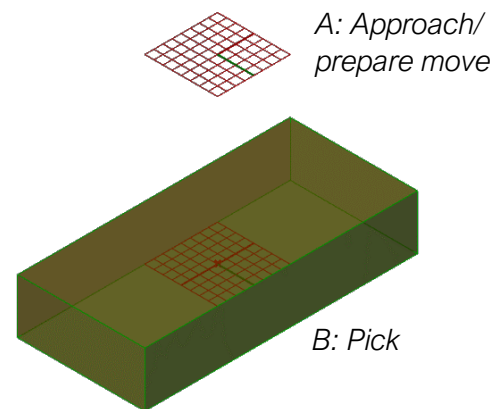
Location B: plane to pick the brick
Robot moves to the location of the brick
Gripper is closed

Location A: plane to prepare the move
Robot moves back to pick approach plane

Location C: approach plane to drop
Robot moves to approach plane above drop location

Location D: plane to drop the brick
Robot move to the drop location
Gripper is opened

Location C: plane to move to next brick
Robot moves back to drop approach plane



3. Specify other data for each movement like speed, subprograms or acceleration;
4. Convert all the data to a .csv (*comma separated values*) file. This file can be read by other programs, or easily by humans in Excel;
5. Convert all the geometry data of the bricks to a .csv file;
6. Read the geometry .csv in RoboDK and rebuild all the bricks;
7. Generate a program in RoboDK based on the .csv data;
8. Execute the program in RoboDK with the Robot.

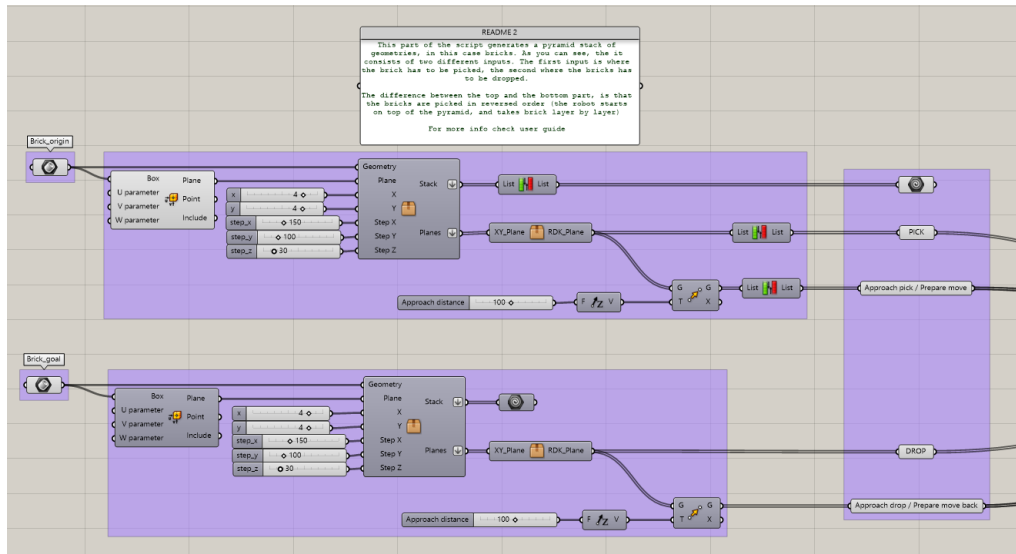
Although this may sound a bit vague so far, you will get to understand all the steps in the following chapter.

Quick Start (30 minutes)

This chapter will give you a quick overview of how all the scripts work. Make sure you follow it step-by-step. If you need an extensive review on how all the different functions work, take a look at the next chapter.

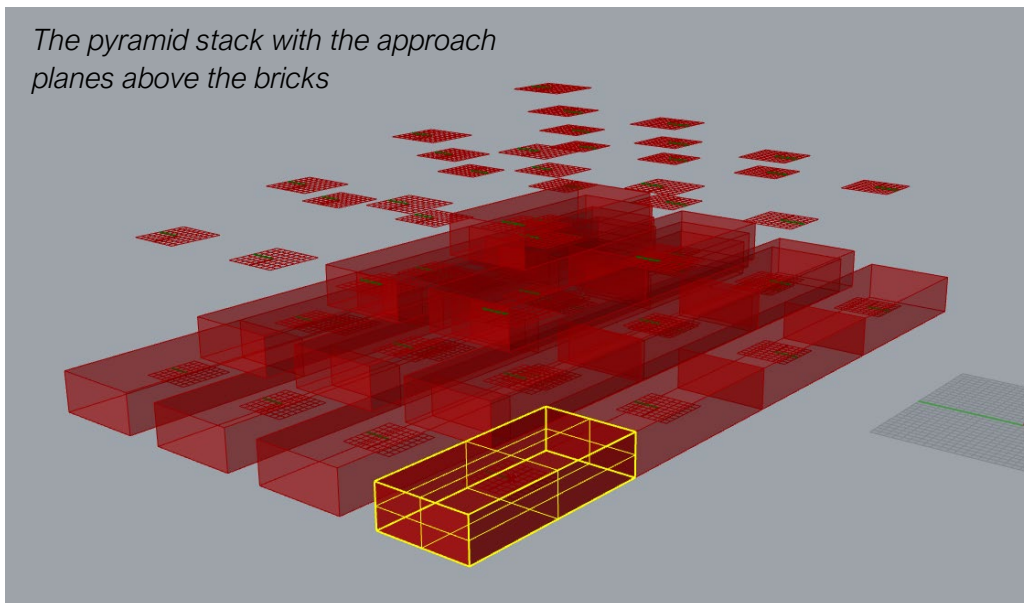
1. Open the Grasshopper file

As you can see in the script, the bricks already have been internalized in the inputs (you do not need to select objects in Rhino). The Grasshopper file contains several functions. First, a pyramid stack of bricks is created, with the corresponding planes. Based on the location of the bricks, we define the pick, drop and approach planes.



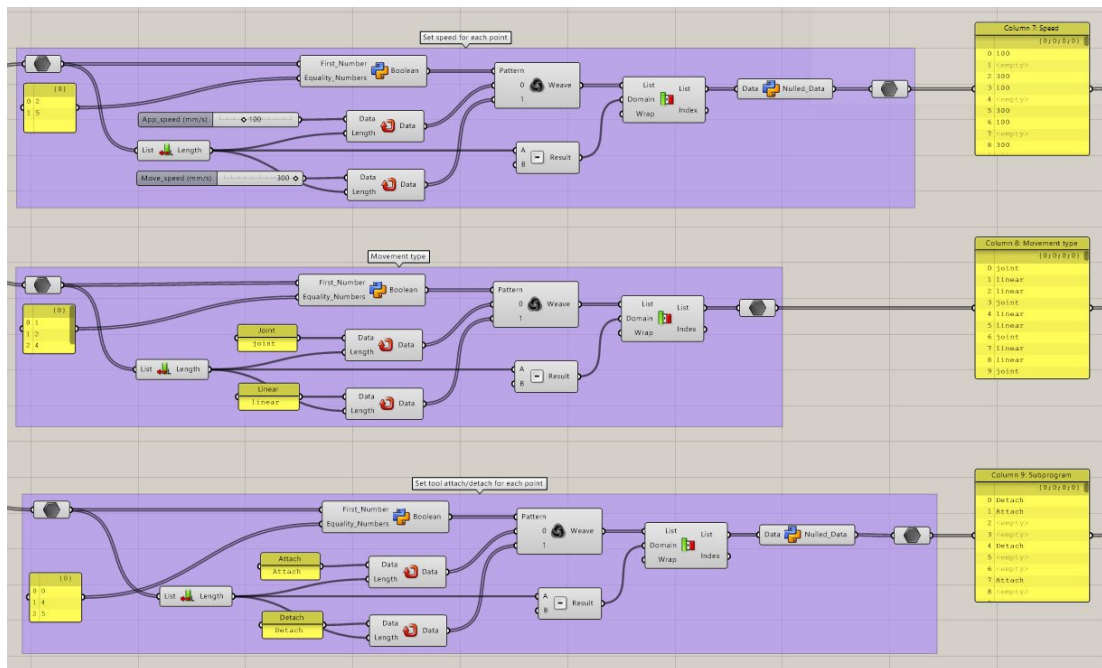
It is easy to change the shape or location of the pyramid stack of bricks with the number sliders. One important aspect to note: In your project, you may want to also rotate the brick, for example 45 degrees. To make sure the Robot gripper also rotates in the right direction you *will have to make sure the Plane is in the right direction*. Therefore you will have to rotate the plane before the Plane input.

The pyramid stack with the approach planes above the bricks



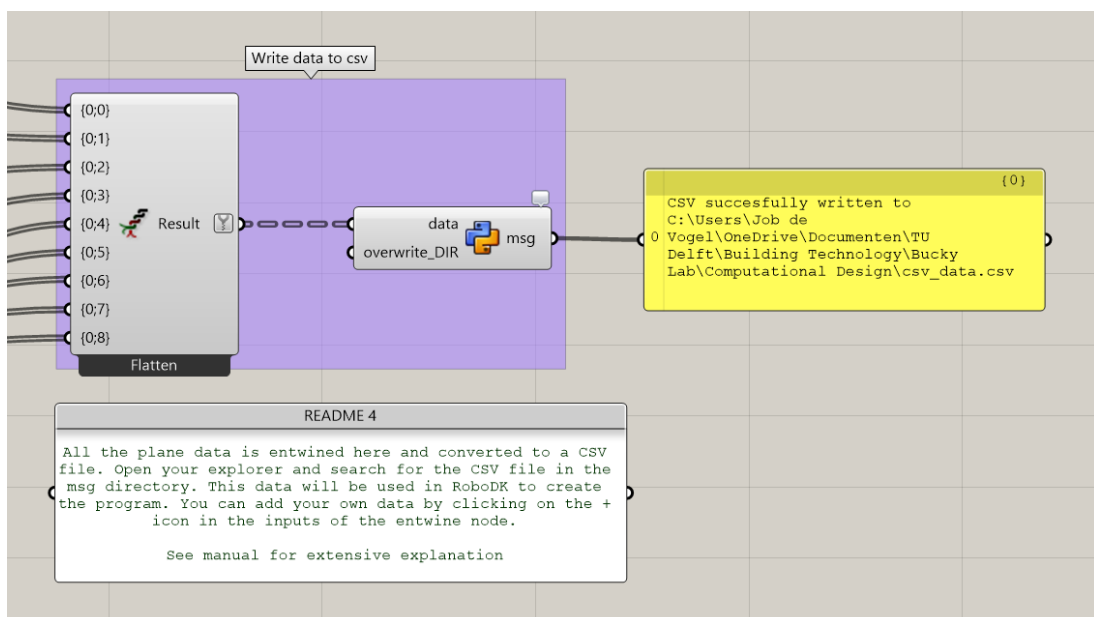
2. Other robot instructions

Next to the location of the planes and the order, we will also need to specify other instructions for the Robot, for example: the speed, the movetype (linear or joint) and the step at which the gripper should attach or detach a brick. This is done in the bottom part of the Grasshopper script.

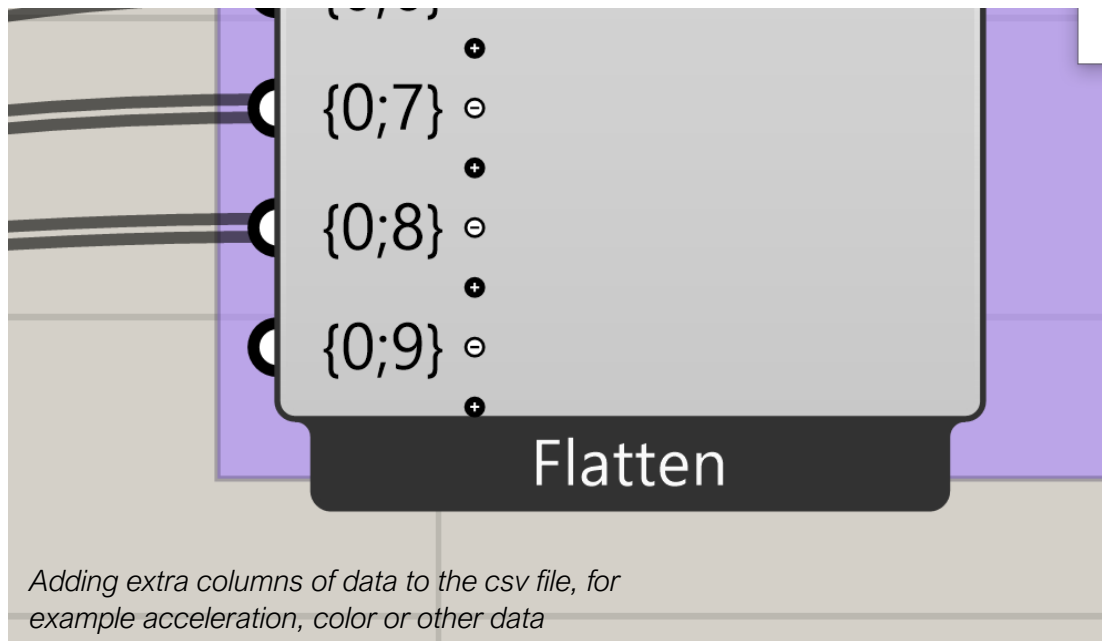


3. Generate the data CSV file*

At the end of the Grasshopper script, the generated data is converted to a CSV file. Every time you change something in Grasshopper, this script will automatically update the CSV file. By default the CSV file is saved in the same folder as your Grasshopper file. You can add extra columns with data by clicking on the small + icon at the inputs of the entwine node.



* You could also use Lunchbox to create a CSV. However this may be a bit unintuitive to use and requires the Lunchbox plugin.



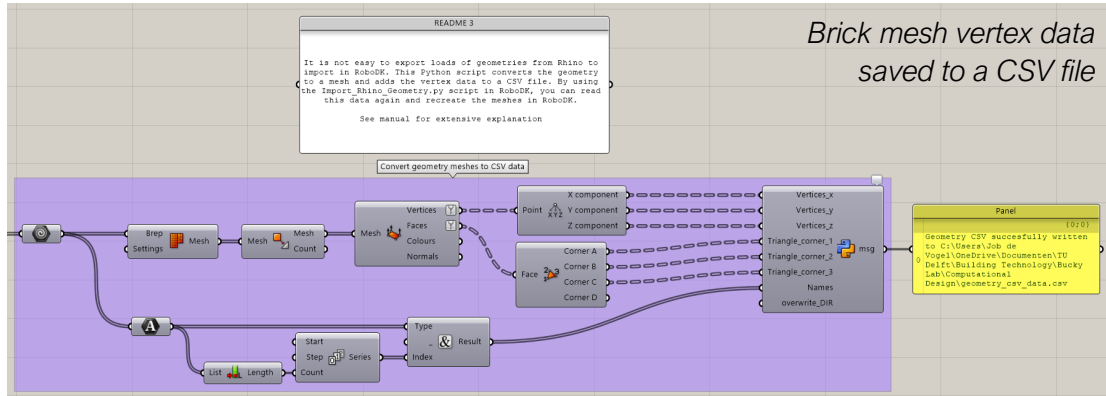
4. Interpreting the CSV file

Open the `csv_data` CSV file in Excel. The CSV file contains several columns, based on the order in which you add the data in Grasshopper. The first six columns are the x, y, z, w, p, r values. After that, you can see a column with the robot speed (both linear and joint movements). Next, there is a column that defines the move type: linear or joint. Finally the csv contains a column with subprograms that the robot should execute. Column G to I are based on the outputs of the script in step 2.

	A	B	C	D	E	F	G	H	I
1	-26	357	183	180	0	180	100	joint	Detach
2	-26	357	83	180	0	180		linear	Attach
3	-26	357	183	180	0	180	300	linear	
4	-301	-567	93	180	0	180	100	joint	
5	-301	-567	-7	180	0	180		linear	Detach
6	-301	-567	93	180	0	180	300	linear	
7	49	407	153	180	0	0	100	joint	
8	49	407	53	180	0	0		linear	Attach
9	49	407	153	180	0	0	300	linear	
10	-151	-567	93	180	0	180	100	joint	
11	-151	-567	-7	180	0	180		linear	Detach
12	-151	-567	93	180	0	180	300	linear	
13	-101	407	153	180	0	180	100	joint	
14	-101	407	53	180	0	180		linear	Attach
15	-101	407	153	180	0	180	300	linear	
16	-1	-567	93	180	0	180	100	joint	
17	-1	-567	-7	180	0	180		linear	Detach

5. Generate the geometry CSV

Next to the robot instruction data, Grasshopper also creates a csv file containing the geometry data. This file is also saved in your Grasshopper folder by default. Open the geometry csv file in Excel. The file contains four columns: the datatype (v: vertex, o: vertex order, n: name) and the coordinates, vertex order or name. You can now close the csv file.

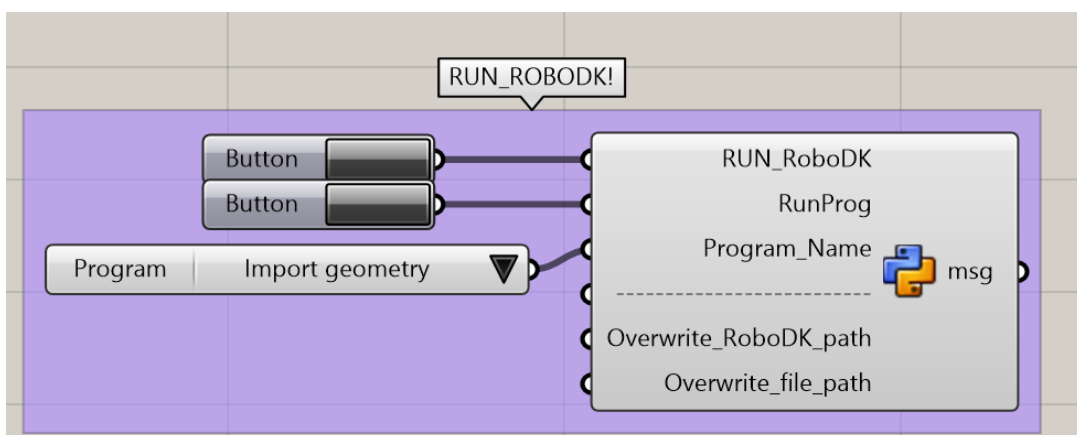


	A	B	C	D
1	v	49	392	68
2	v	49	392	98
3	v	-101	392	68
4	v	-101	392	98
5	v	-101	392	68
6	v	-101	392	98
7	v	-101	322	68
8	v	-101	322	98

27	o	9	8	10
28	o	13	12	14
29	o	17	19	18
30	o	21	20	22
31	o	1	2	3
32	o	5	6	7
33	o	9	10	11
34	o	13	14	15
35	o	17	18	16
36	o	21	22	23
37	n	Closed Brep_0		
38	v	124	442	38

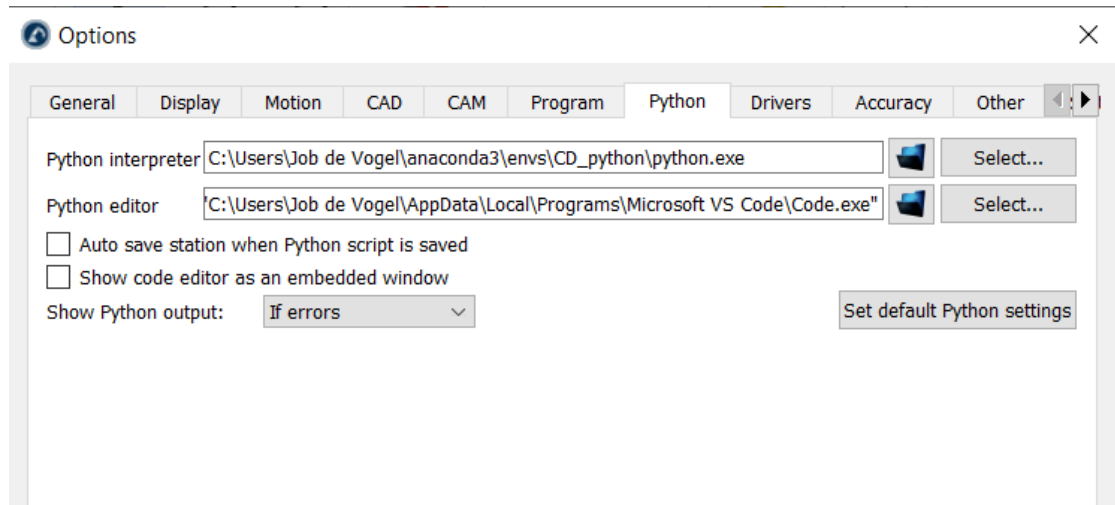
6. Start RoboDK using Grasshopper

In Grasshopper, click on the Run RoboDK button. If you installed RoboDK properly in the C:/ directory of your computer, this should work. If not, you may have to overwrite the installation directory (one of the inputs of the Grasshopper node). Grasshopper should now start RoboDK, specifically the provided RoboDK file.



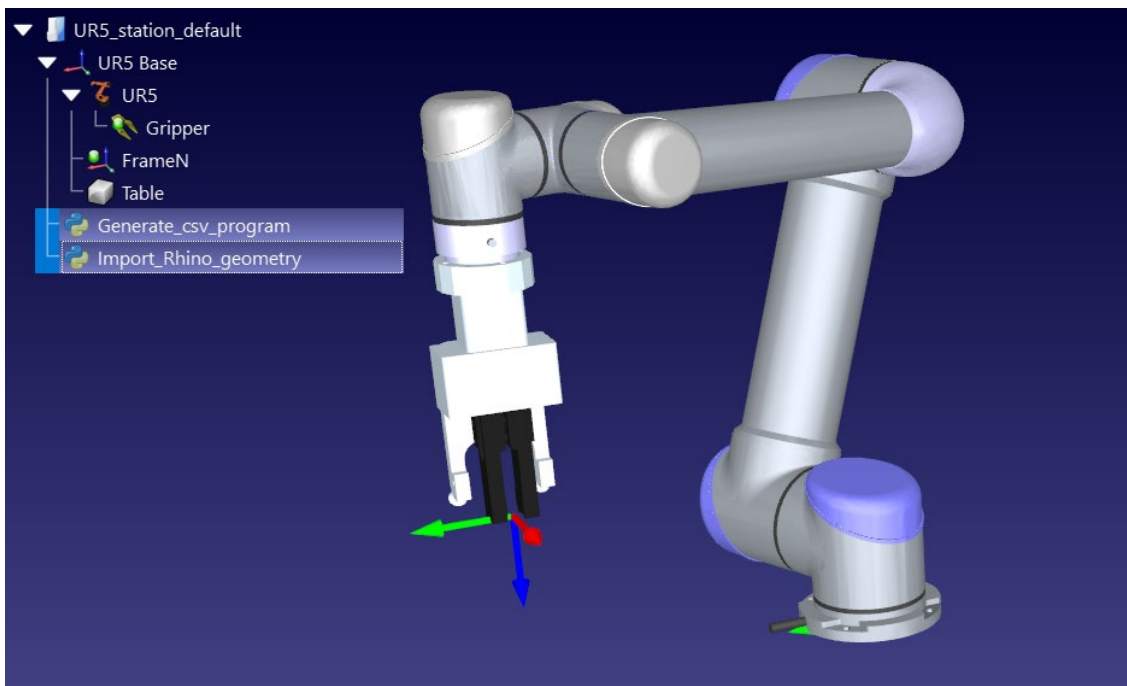
7. (Check your Python interpreter)

Open Tools > Options > Python in RoboDK. Check your Python interpreter. If you are planning to make changes in the Python scripts, it is recommended to set your Python editor to Visual Studio Codium (installed in the RoboDK installation directory) or Visual Studio Code.



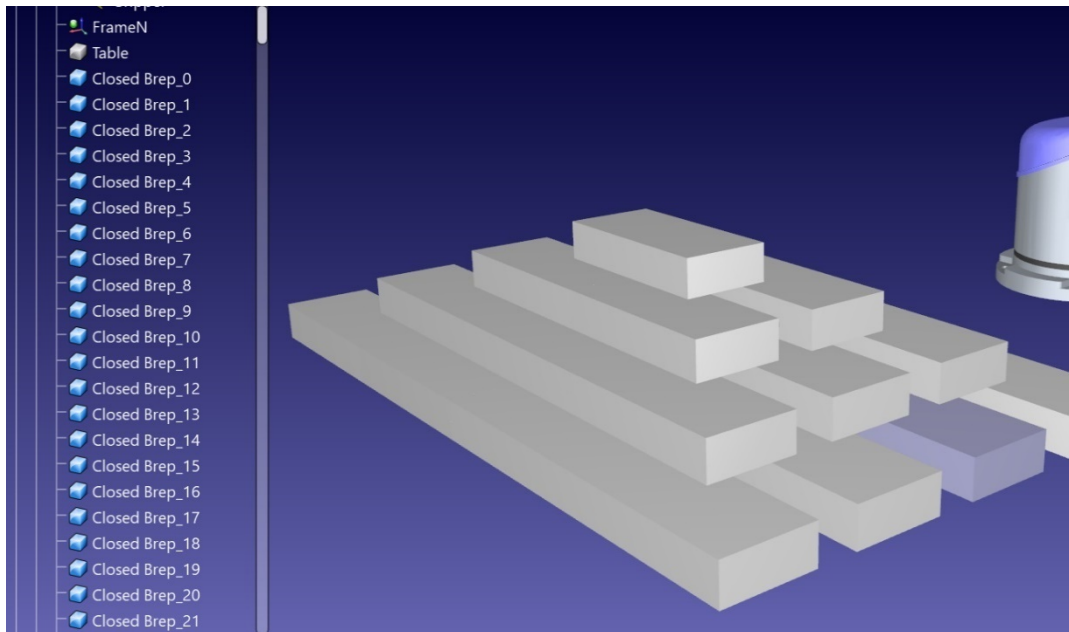
8. Add the Python scripts to your RoboDK file

In RoboDK, click on file > Open file. Open both provided Python scripts: Generate_csv_program.py and Import_Rhino_geometry.py. These scripts are provided in the zip file.



9. Import the Rhino Geometry

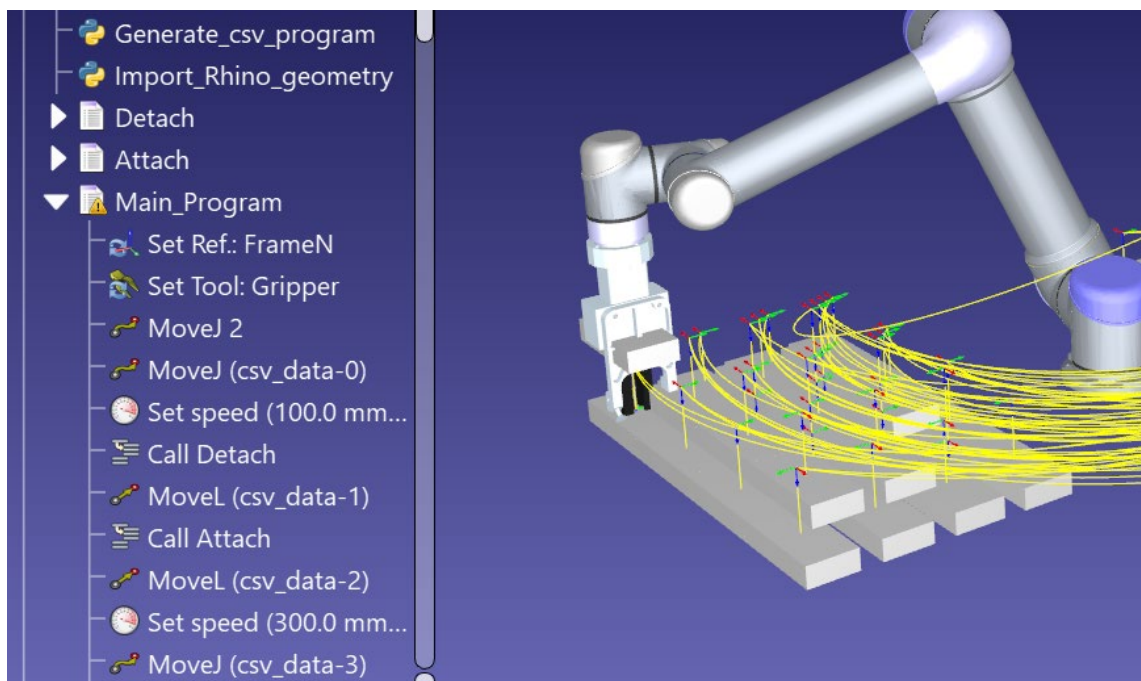
If you double click LMB on the Import_Rhino_geometry program in the tree, RoboDK will import your data. Follow the instructions from the message boxes.



10. Generate the program

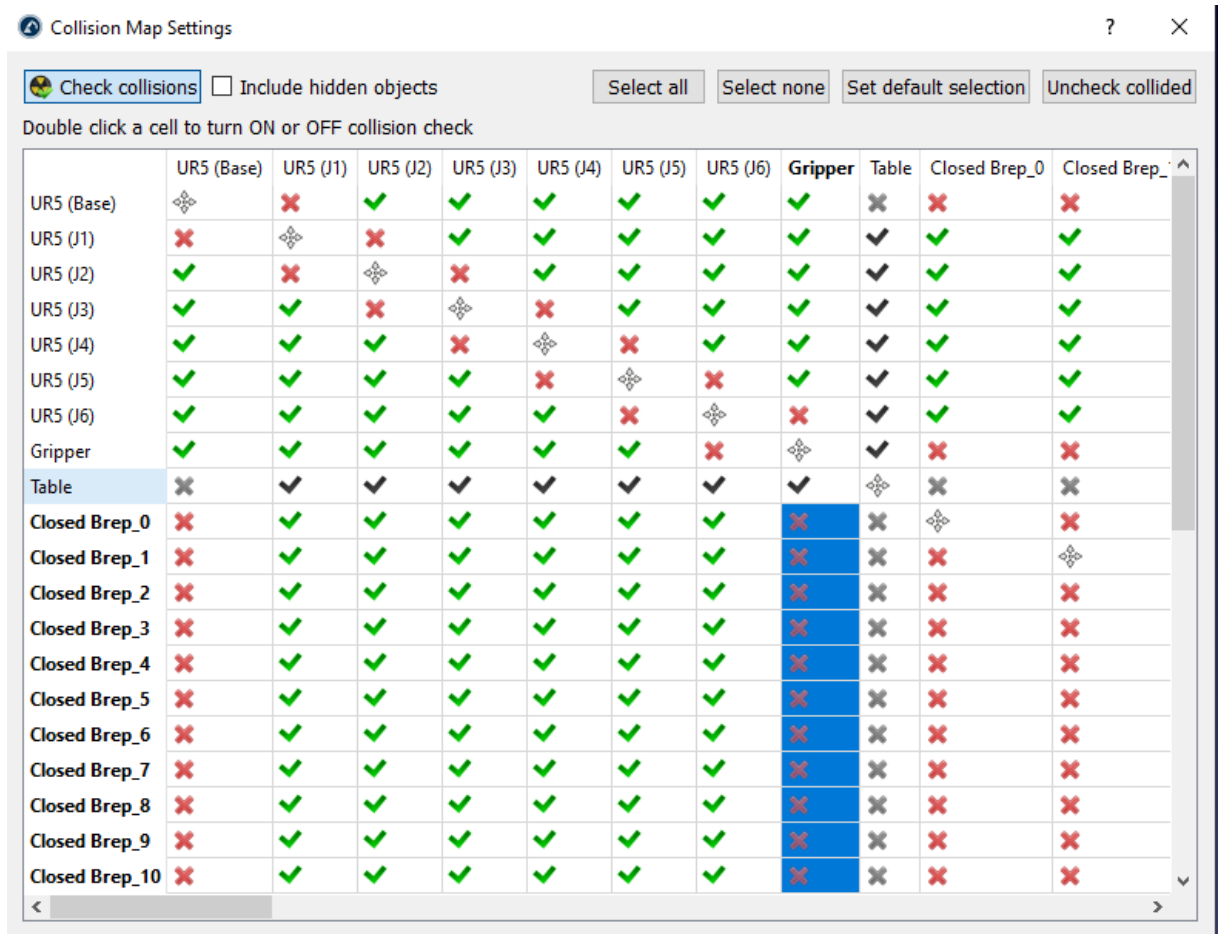
If you double click LMB on Generate_csv_program in the tree, RoboDK can generate or execute your program from your data CSV. Just follow the message boxes. In this process, you will have to select the data CSV file in your explorer. The instructions from your CSV will be added from the left columns (the locations) to the right columns (the subprograms). After adding the program, the Robot will execute the instructions directly from the tree.

The RoboDK window may turn black for a while if you are trying to import a great amount of instructions.



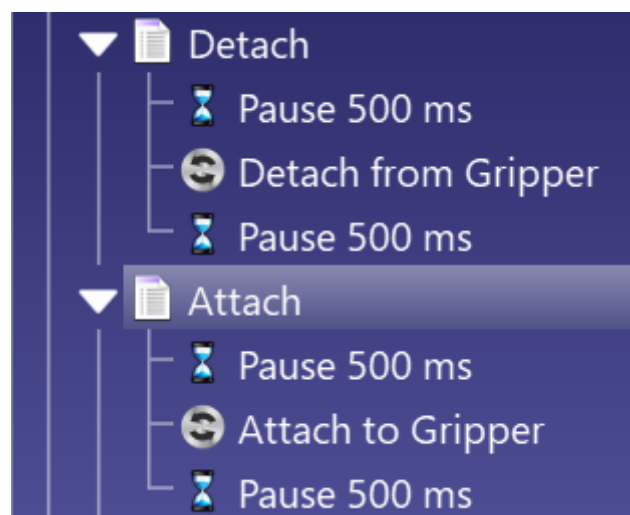
11. Set collision detection

Although all targets and objects are now added, collisions may still occur with the bricks and gripper when you turn on collision detection. Make sure you turn off collision detection between the gripper and the bricks (shift + x).



12. Generate subprograms

As you can see, several subprograms have also been added to the tree (Attach and Detach). You will have to add instructions to these subprograms in RoboDK yourself. If you run the Generate_csv_program script again and update the program, subprograms are not overwritten if they already exist.



In-depth explanation

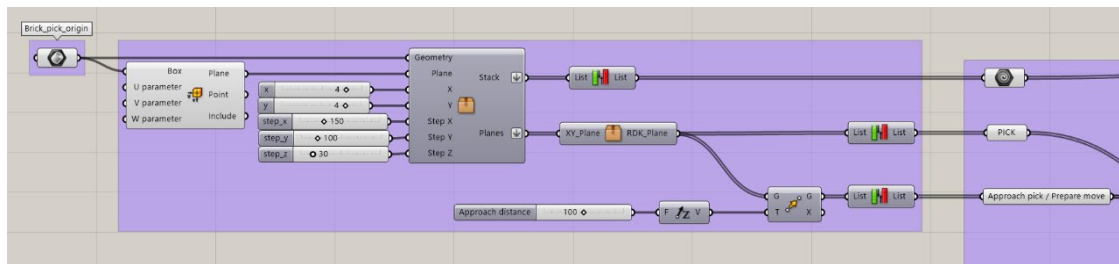
This chapter will dive deeper into the different functions written in both Grasshopper and Python. Unfortunately, it is not possible to understand *all* functions without a basic understanding of programming. However, this information is only relevant if you are interested in how the scripts were developed.

1. Grasshopper - Pyramid stack script

The pyramid brick stack script was developed to make it easy to create a large amount of bricks that the robot can move. Two different brick inputs are used, so that you are able to specifically define where the “pick” locations and the “drop” locations should be.

As you may notice, the pyramid stack cluster also has a plane input. This plane input is very important, because it defines in what direction the gripper of the robot should attach the brick. If you are using a different location for the first brick of the pyramid, make sure, you also define the plane appropriately.

After creating the pyramid, you can see a plane conversion cluster. The reason for that is that the world axes of the gripper in RoboDK are different from the world axes of Rhino. For example, the z-axis is always positive downwards in RoboDK, in contradiction to Rhino. The script is based on the assumption that the Robot is in the world origin of Rhino. Finally, the order of the pick locations are reversed, because the robot should start with picking the top brick of the pyramid, and finalize with the bottom layer.



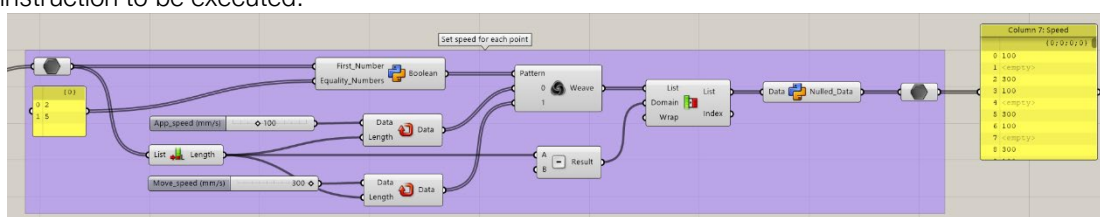
2. Grasshopper - Speed/movetype/subprogram

Next to the pick/drop locations, the script also makes list for several other instructions. Currently the instructions are read as followed:

RoboDK script reads the data CSV:

- The movetype (linear/joint) of the Robot is read;
- Robot arm moves to the x, y, z, w, p, r location using the movetype;
- The speed is set for the next movement;
- The Robot executes a subprogram.
- Next CSV row...

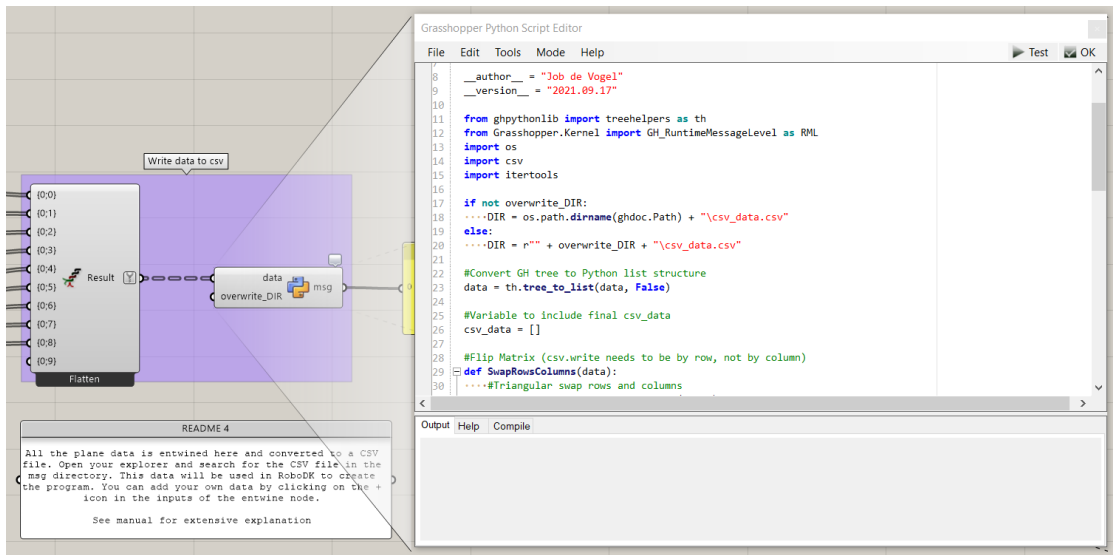
All this data is first generated in Grasshopper. Based on the goal of a certain CSV row (pick/approach/drop/move) we can specify a speed, movetype or subprogram that has to be executed. Since Grasshopper does not have a function to check if one value is equal to one of the other values, a Python script was written for that. Furthermore, at the end of some instruction scripts, you can see that repeating values are removed, since they do not require a new instruction to be executed.



Finally, the panels containing all the data are added to the CSV write Python script.

3. Grasshopper - generate data CSV script

To be able to generate a program in RoboDK, you will need to specify CSV data for the different instructions and positions. In the Grasshopper file, you can see that the data is first entwined and then given to a Python script. If you would like to add extra data (for example: acceleration speed), just add an input to the entwine node. The directory in which the CSV file is saved, can be changed in the second input of the Python script node. To write data to a file from Python, it is not possible to have it opened in another window. Always first close your CSV file, and then update the data from Grasshopper.

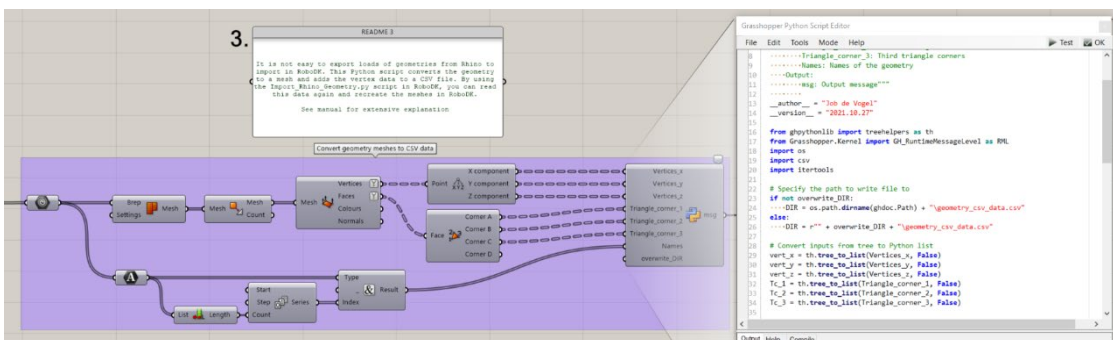


To change the CSV file name, or specific functionalities of the generate CSV script, just open the Python editor of the Grasshopper node. Remember however, that this may lead to problems in the Python scripts of RoboDK. Both scripts have been specifically coded to interact with each other as default. The script has been written using the build-in csv module for Python.

4. Grasshopper - generate geometry CSV script

You may be wondering why we are not just importing geometry data as an .OBJ or .STL format to RoboDK using standard import/export options. Unfortunately, this is not possible because these file formats always generate one object in RoboDK, instead of making multiple objects. For example, exporting 1000 bricks from Rhino to one OBJ file, will result in only one object in RoboDK, containing 1000 bricks. We need 1000 individual objects in RoboDK.

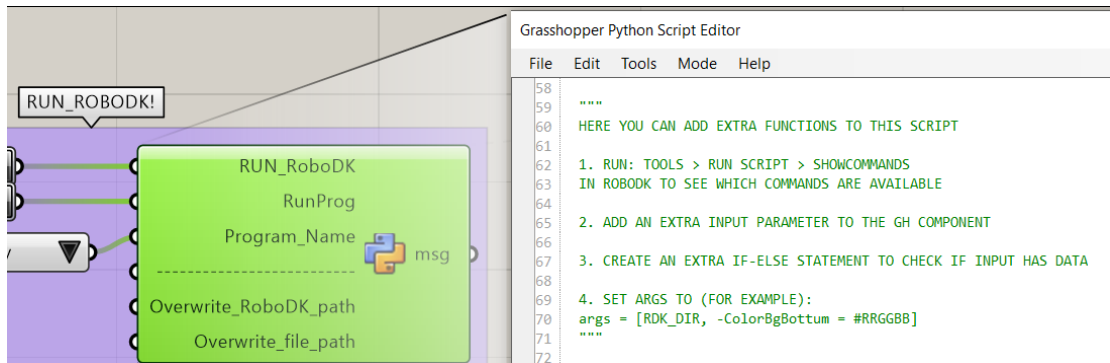
This problem is solved using the following script. In Grasshopper, the geometry is first converted to triangular meshes. Next, the vertex data and order is extracted. Finally, a name is added for each object to the "names" input.



Be careful making changes to this Python script in Grasshopper. The `Import_Rhino_geometry.py` script in RoboDK expects vertices (v), orders (o) and a name (n) for every Rhino object. To edit the script in Grasshopper, double click on the Python node.

5. Grasshopper – RoboDK connection script

In Grasshopper you will find a node that is able to start RoboDK. This includes a function that runs a specific program in RoboDK (for example: `main_program` or `Import_Rhino_geometry.py`) That way, you are able to execute RoboDK programs without even touching RoboDK. The programs however, must already have been added to the Robot tree. If you add new programs to RoboDK, you can add the names of these programs to the value list next to the `RunProg` input in Grasshopper. This will enable you to also run your own programs from Grasshopper.



The Python script is based on a standard Python module called `subprocess`. It runs a separate application or command through `CMD`. For the documentation of all available RoboDK commands and extra arguments (like `runProg` or `QUIT`), take a look at <https://robodk.com/doc/en/RoboDK-API-Command-Line-Options.html>. Some `CMD` options are not included in the documentation. You can find them by running RoboDK > Tools > Run Script (shift + s) > Show Commands.

6. RoboDK - `Import_Rhino_geometry.py`

The `Import_Rhino_geometry.py` script reads the CSV data that was generated in step 4. After reading, it rebuilds all the meshes based on the data in the CSV file.

If you have a basic understanding of Python, this script should speak for itself. Simply said, the script first reads the CSV file line by line. Then it checks what the datatype is and adds that data to a corresponding dictionary by object. Finally, based on the order of the vertices, the script rebuilds the shapes based on the vertices.

As explained before, it is possible to run this script from Grasshopper using the Grasshopper – Run RoboDK! node. If you want to automate the interaction between Grasshopper and RoboDK even more, you can specify the geometry .csv path in the RoboDK Python script. This avoids having to select the geometry csv in your explorer every time you run it.

WARNING: when you import a Python script to RoboDK, it is saved to a new folder (`Appdata/local/temp`). From that moment, all the changes you make are saved to the file in the temp folder. If you want to do changes and save it to your original file, open the file in an external editor like Visual Studio Code (or Codium). Of course, it is also possible to save the file manually.

7. RoboDK – `Generate_csv_program.py`

The `Generate_csv_program.py` script is the core of the Grasshopper to RoboDK process. First the script reads the data csv file with all the instructions. After that, based on the user input in message boxes, a new program is generated, or the instructions are just executed.

The script contains the following important functions:

- `Load_targets`: This function reads the csv data and returns them in several lists.

- Load_program_csv: The script reads the targets and instructions, returned from Load_targets. Based on these settings, the instructions are added to the RoboDK tree.
- Run_simulation: This function executes the instructions of the csv, returned from the load_targets function. If the instructions are already added to the RoboDK tree, they will be executed directly. If this is not the case, the script will try to execute the instructions without adding them to the tree. Important to mention however, is that the script is not able to use any form of collision detection. Therefore, it is likely that the robot will not be able to execute the script. Only run the simulation without adding it to the program, if you are sure collisions are not a problem.

If you are planning to add extra instructions to the Robot, this script is where you have to be to make changes. Within the script, several lines have been added with instructions how to do that. Basic changes like initial simulation speed or the name of the main program can be set in the “main” part of the script.

```

257 ##### -- MAIN -- #####
258 # Force just moving the robot after double clicking:
259 #load_targets_move(CSV_FILE)
260 #quit()
261
262 # Specify file codec
263 codec = 'utf-8' #'ISO-8859-1'
264
265 # SPECIFY YOUR CSV FILE HERE:
266 CSV_FILE = r""
267 SELECT_CSV = mbox('Please select your csv file, you can also specify your csv directory in the generate_csv_program script (line 263)', 'Select')
268
269 if SELECT_CSV and not CSV_FILE:
270     # No CSV has been specified in line 263, select manually
271     CSV_FILE = getOpenFile(RDK.getParam('PATH_OPENSTATION'))
272 elif not SELECT_CSV and not CSV_FILE:
273     # User wants to cancel
274     quit()
275
276 # PROGRAM/SIMULATION SETTINGS:
277 MAKE_GUI_PROGRAM = False
278 SIMULATION_SPEED = 1
279 PROGRAM_NAME = "Main_Program"
280
281 ROBOT.setFrame(FRAME)
282 ROBOT.setTool(TOOL)
283
284 RDK.setSimulationSpeed(SIMULATION_SPEED)

```

8. RoboDK – Subprograms

After reading the subprogram names from the CSV file, you can see that a subprogram is added to the RoboDK tree. These subprograms do not contain any instructions. If you add a subprogram name to the CSV file from Grasshopper, for example “Pause_1500ms”, the subprogram will be added, but you will have to assign instructions manually.

Subprograms are *never* overwritten. Therefore it is save to add instructions and importing new targets, speeds etc. If a subprogram already exists, it will not be added to the RoboDK tree.

On the Google Drive, you can find the following bug fixes. If you're project is not working because of a bug, please add a comment on Drive, so that I get a notification.

Bug fixing Grasshopper

RoboDK is not running when I click the RUN button in Grasshopper

Make sure you RoboDK is installed in the C:/ folder or overwrite the install location

If you are using Mac, we cannot guarantee that this script works.

My program in RoboDK is not running when I press the button in Grasshopper

See previous problem

Make sure you defined the correct name for the program in the value list.

My overwrite_DIR input is not working

Only add the directory to the input, not the name of the csv file. This will be added automatically.

Make sure you use the correct Grasshopper syntax to add a raw string to an input.

The CSV data is not saving to the file

Make sure you file is not opened in another window.

Bug fixing RoboDK

I made changes to the script in RoboDK, but the changes are not saved:

If you open the script from RoboDK, it will be loaded from a temp folder. Always make sure you save as... or open the script in an external editor.

My gripper is not opening/closing

Yes, that is true, you will need to create an alternative gripper for the robot. Check the following video to find out how to do that: [RoboDK Webinar - Mechanisms - YouTube](#)

My robot program leads to collisions:

Check if the collision detection is turned on, and all relations between objects and the robot are set properly.

Make sure the program is first added to the tree, before executing the program, otherwise collision maps/detection cannot be used. (This problem can be avoided by using the Collision Avoidance API: <https://robodk.com/forum/Thread-Collision-Avoidance?pid=1189#pid1189>)

Generate a Collision-free map in RoboDK: <https://robodk.com/doc/en/Collision-Avoidance-Collision-Free-Motion-Planner.html>

The Robot gripper is in the wrong direction

Rotate the XY plane from the Plane input in the pyramid stack script of Grasshopper (see step 1 Quick Start).