V-RAY CLOUD - BEYOND THE BUTTON

A practical guide to automated content creation

In this guide, we will show you how to automate the process of generating and rendering product images from different viewpoints for a catalog or webpage.

Let's say we have a line of armchairs, tables, lamps or any other product, that needs to be shown from different viewpoints - left, right, front, back, top. Rendering these views by hand will be daunting. Instead, a bit of scripting can ease the work and save time.

In this example, we'll generate five different views for each of the products (five armchairs) and then, submit these different views for rendering on V-Ray Cloud. Once rendered, all of the views for each armchair will be downloaded automatically to our PC, at the press of a button.

To do that, we need to export:

- One .vrscene file for each armchair model (five models)
- One .vrscene file for each viewpoint (five views)

Then, we'll execute a script that will take the exported armchair models, create new ready to render .vrscene files for all the different views, and save them in a new folder. This will allow us to easily generate all of the render views for each armchair.

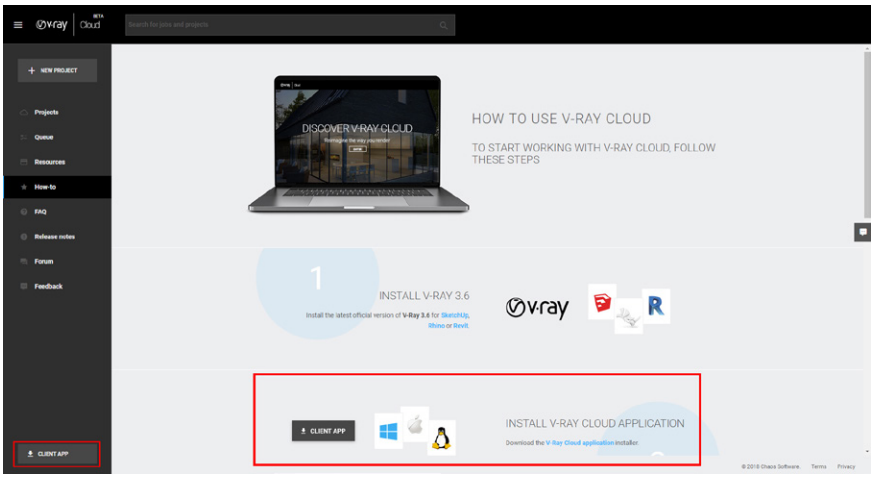
A second script will take all these newly generated .vrscene files and simultaneously send them to be rendered by the V-Ray Cloud. The resulting images will be saved on our PC automatically, and are suitable for use in presentations, websites or application thumbnails.

1 REQUIREMENTS

Software: Rhino, V-Ray, V-Ray Cloud Client App
Programming language: Python

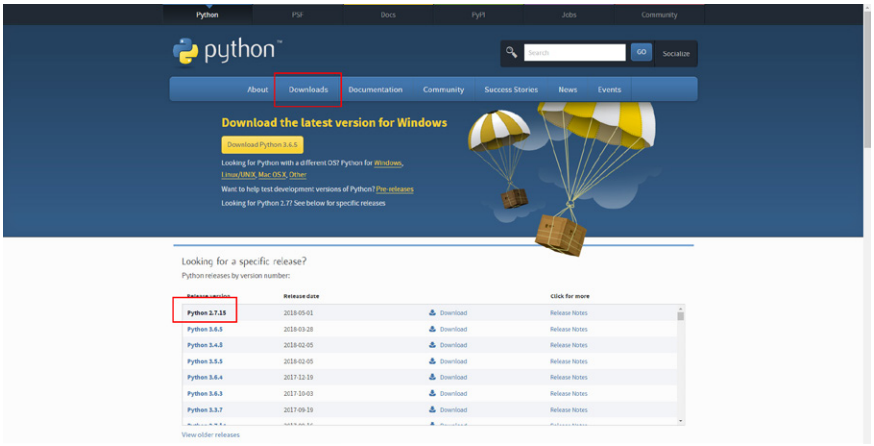
1.1 Install V-Ray Cloud Client App

Go to https://vray.cloud/ and install CLIENT APP.



1.2 Download and install Python (free)

Go to www.python.org, click “Downloads,” then find Python for your operation system. Download and install the software.



1.3 Download "ChaosGroup – ACC"

Download "ChaosGroup – ACC.zip," and extract it to your local machine.The folder will contain:

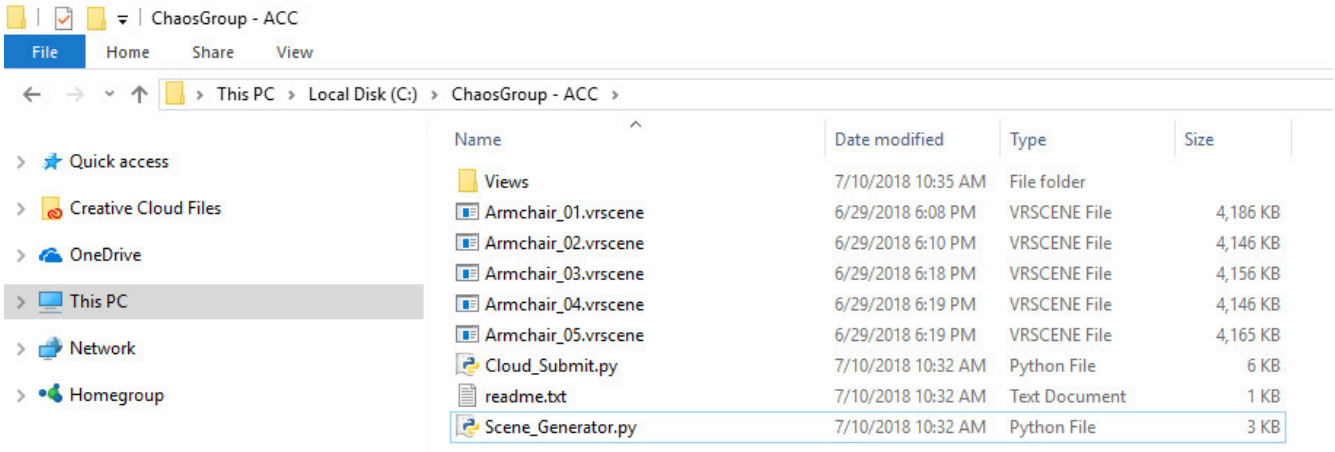
- **Views** - The folder where the five different views of each armchair will be saved.
- **Scene_Generator.py** – This script generates five views for each armchair. The result will be saved as new .vrscene files in a new folder called "Scenes".
- **Cloud_Submit.py** – A script that sends all .vrscene files from the "Scenes" folder to V-Ray Cloud to be rendered with the settings specified in V-Ray for Rhino, or with custom resolution. Once the renders are ready, you can download them with the press of a button.
- **readme.txt** - A text file with a short explanation of the procedure.

2 FOLDER STRUCTURE

The necessary files will be stored in the "ChaosGroup - ACC" folder. However, you'll also need to prepare few files for your own product beforehand. To do so, open your Rhino scene and export a few .vrscene files. These will be divided into two groups:

- **Products** - Here you'll need to export a single .vrscene for each product in the line (Example: Armchair_01, Armchair_02...) and save in the root folder (ChaosGroup - ACC). Note! The render view doesn't matter, since they'll be exported separately in a moment.
- **Views** - Here you'll need to export a .vrscene file for each viewpoint (Example: Left, Right, Front, Back, Top) and save it in the "Views" folder.

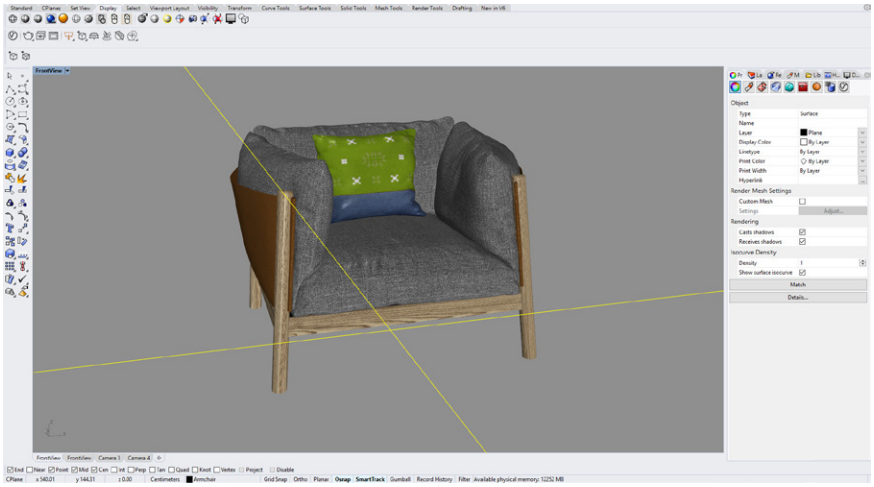
Also, in the root folder, you'll find the "Scene_Generator.py". This script will create the new ready to render files and will save them in a folder called "Scenes". Finally, all of the rendered images will be saved in this location as well. "Cloud_Submit.py" will send the prepared scenes to the V-Ray Cloud.



3 EXPORT SCENES

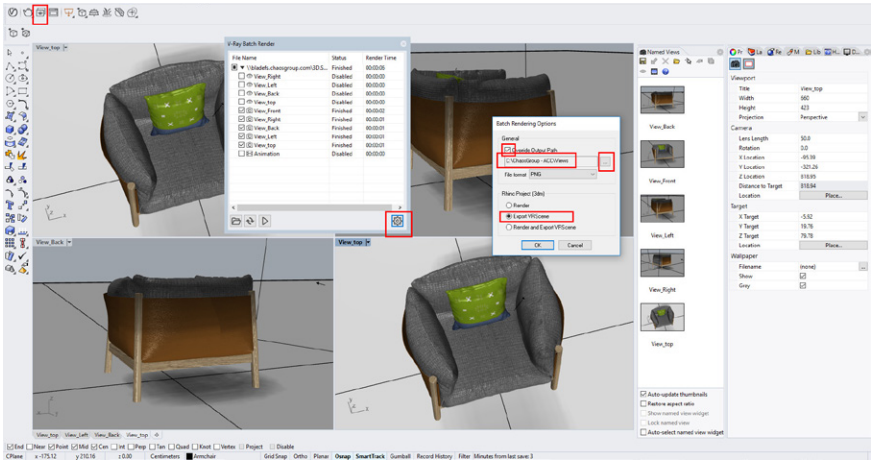
3.1 Model

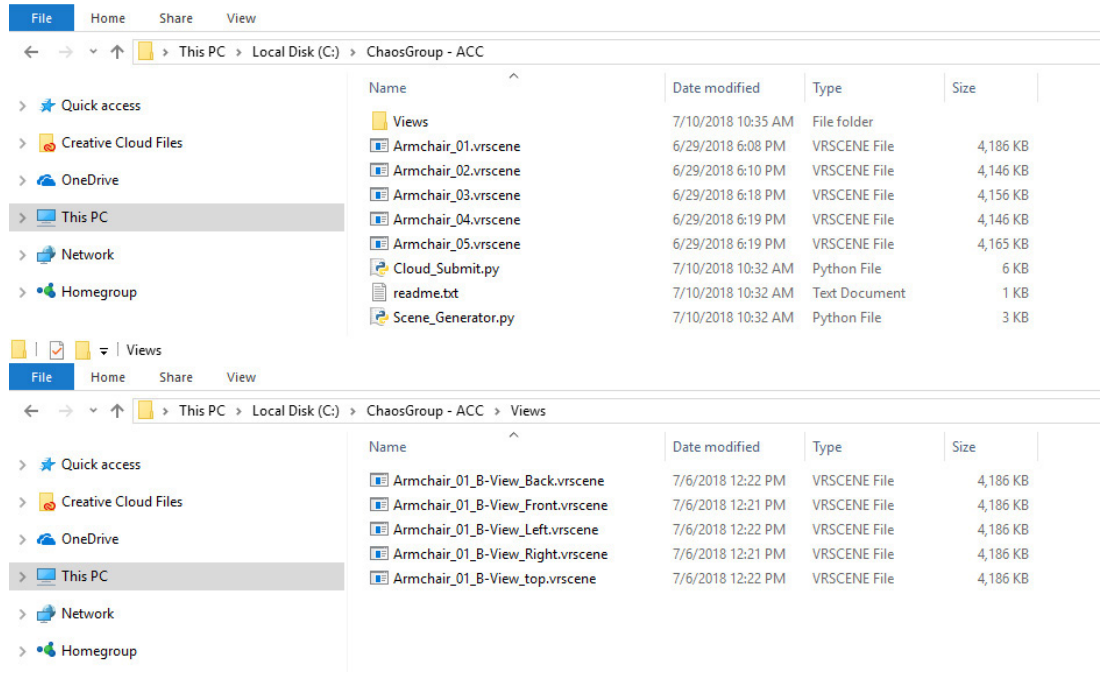
For the purposes of this project we used a simple armchair model from the Design Connected store. The lighting setup uses three V-Ray Rectangle Lights.



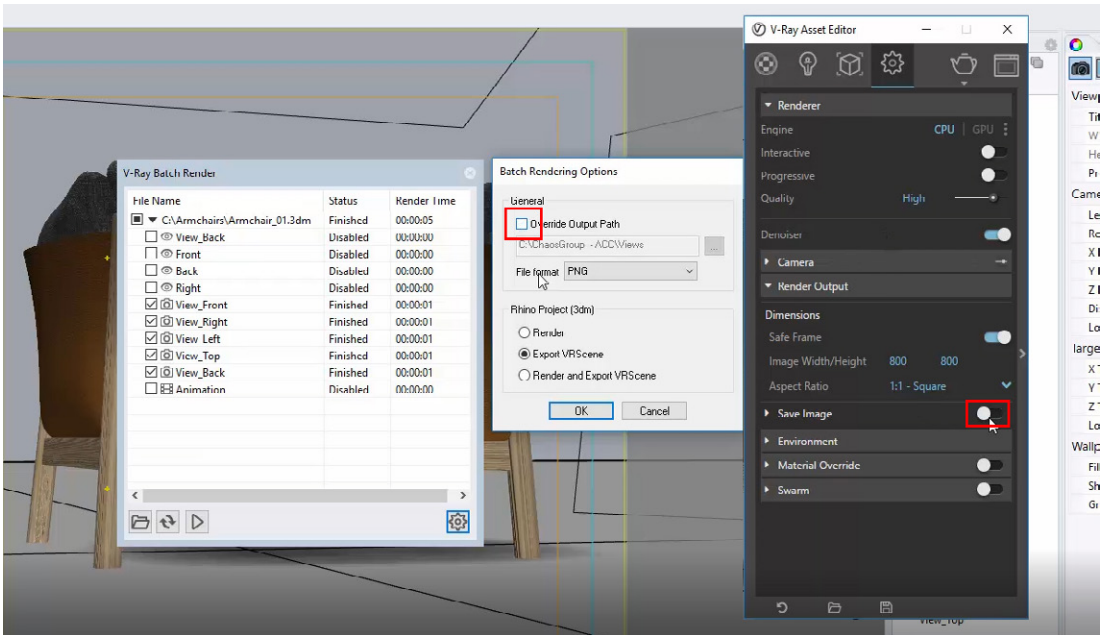
3.2. Export the views

The next step is to create and export five views (or more) – front, left, right, back and top view. Save the views as .vrscene files in the “Views” folder. Later they will be applied by the “Scene_Generator.py” script to the other armchairs in the product line. The geometry is not important here, only the views. You can use V-Ray Batch Render to export all of them. Each .vrscene exported by V-Ray for Rhino will hold information of a single View - the one active at the time of the export.





After that you should uncheck “Override path” in the Batch Rendering Options, and check “Save image” in the V-Ray Asset Editor.

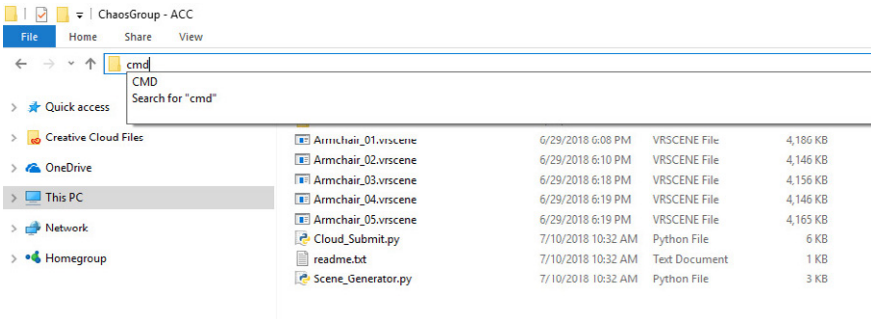


If you want to save your render elements do not uncheck “Save Image” in render asset, just change the path again.

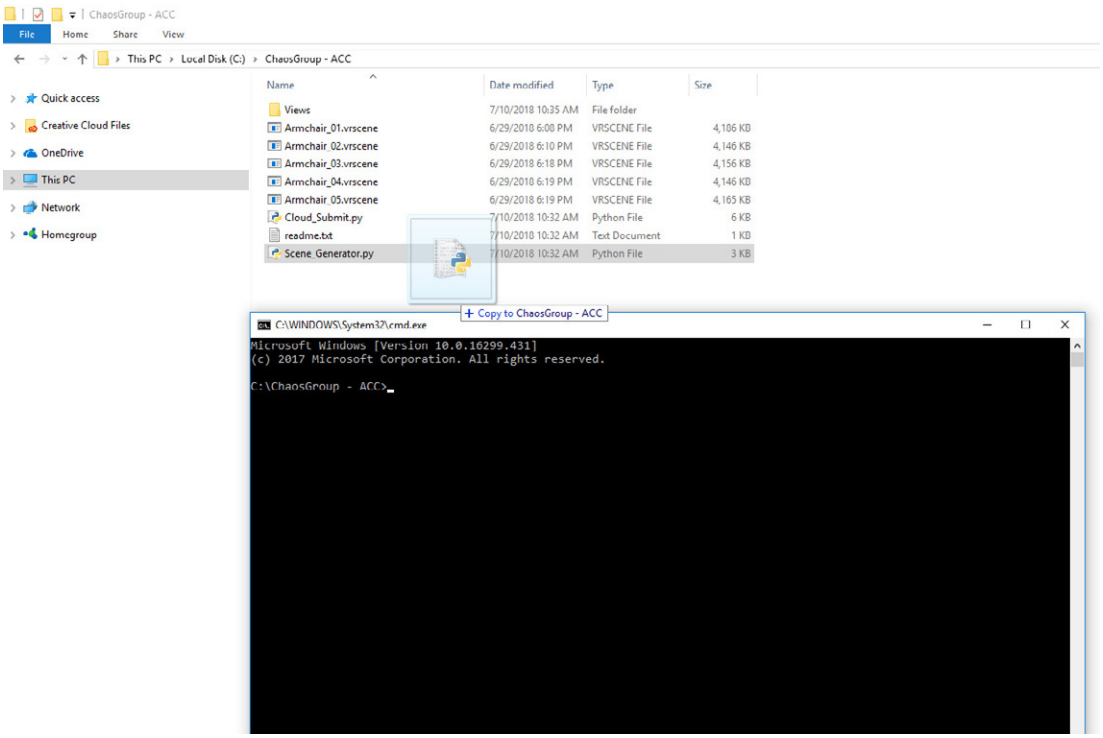
4 SCENE GENERATOR

1.Create views automatically with “Scene_Generator.py”

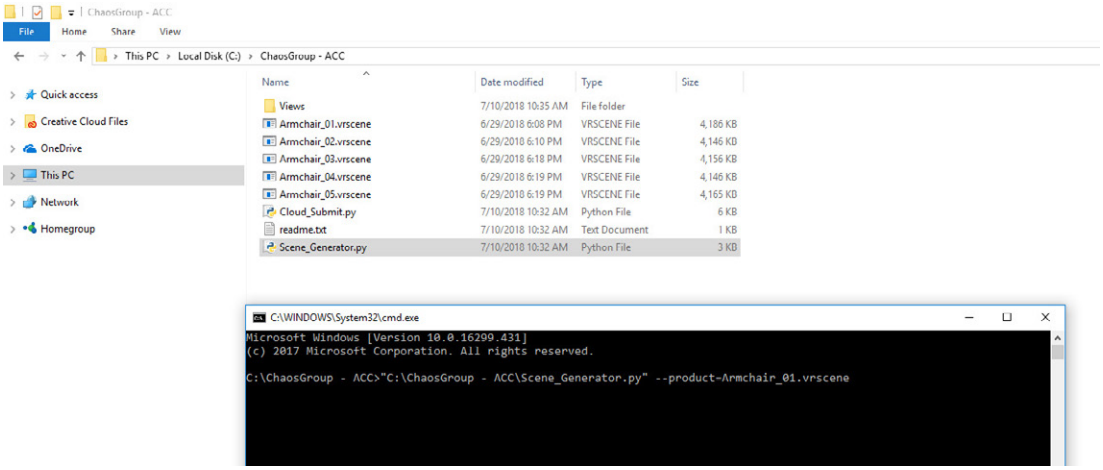
This simple script will use the files in the “Views” folder to create all necessary scenes for rendering. If you have to render a series of objects such as chairs, sofas, or tables you just need to set up your render views for one of these products, and then you can produce multiple views for each of the products. Open “Scene_Generator.py” with a text editor.



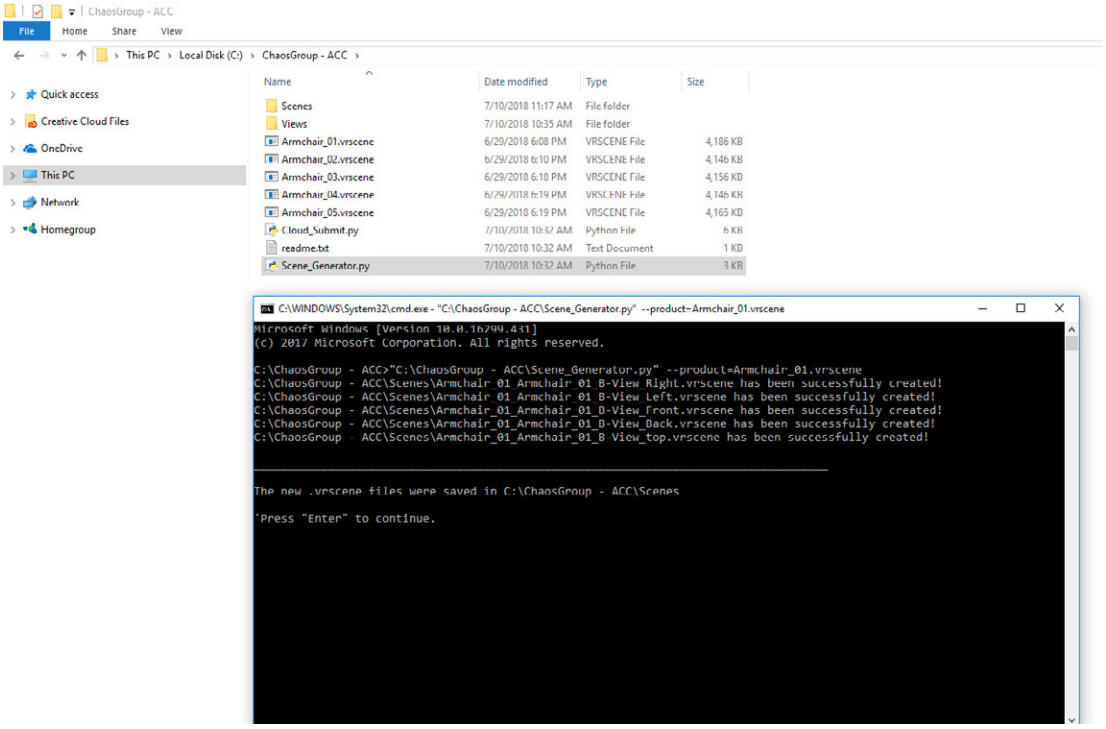
Drag and drop “Scene_Generator.py” to the Command Prompt.
Open the Command Prompt (CMD for Windows).



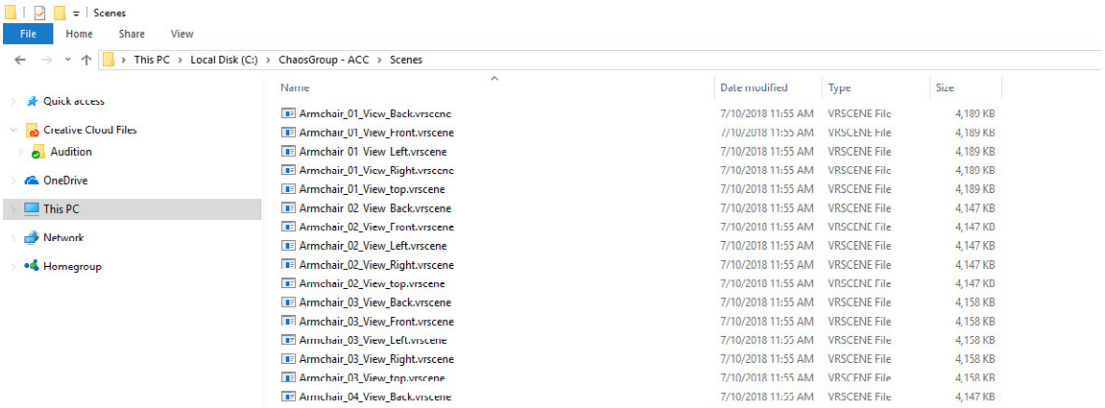
Type " --product=" and the name of the first product.



Hit "Enter".



"Scene_Generator.py" will create a new folder called "Scenes," and automatically create and save the required .vrscene files in it.



Follow the same steps for each of the products in "ChaosGroup - ACC" folder (Example: Armchair_01, Archair_02…).

Another solution is to open "Scene_Generator.py" with a text editor and fill in the name of the product between the apostrophes. Save and run the script.

```
import os
import re
import sys
import argparse

############################## INPUT PARAMETERS ##############################

# Fill the name of the .vrscene product file. It should be in the base folder.
Product='Armchair_01.vrscene'

##############################################################################

basePath = os.path.dirname(os.path.abspath(__file__))
scenesPath = os.path.join(basePath, 'Scenes')
viewsPath = os.path.join(basePath, 'Views')

if not os.path.isdir(scenesPath):
    os.makedirs(scenesPath, 0o755)

argumentsParser = argparse.ArgumentParser()

argumentsParser.add_argument('--product', nargs='?', default=Product)
args = argumentsParser.parse_args()

Product = args.product

```

5 AUTOMATED RENDERING WITH V-RAY CLOUD

Once we have all the needed files in the "Scenes" folder, we can easily send them to V-Ray Cloud for rendering. First we'll render the thumbnails to make sure that everything works properly. If all goes well, we can proceed with rendering the final images with the size and quality specified in Rhino.

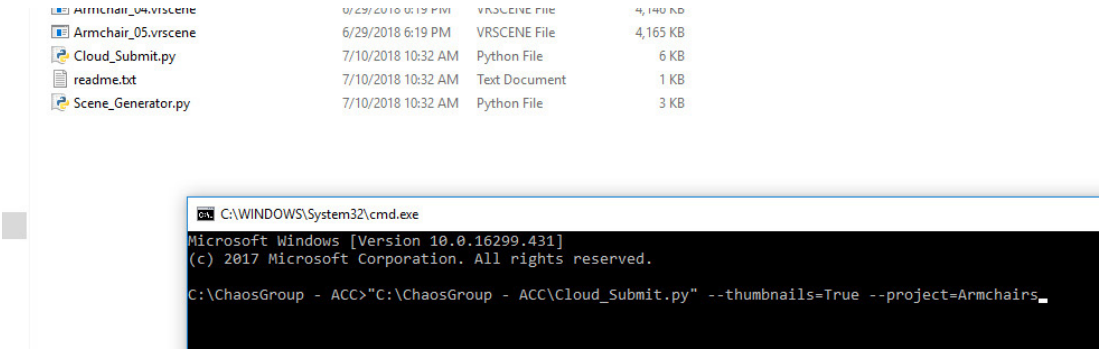
5.1 Render thumbnail images

Open the CMD:

Drag and drop "Cloud_Submit.py" to the command line and type:

" --thumbnails=True".

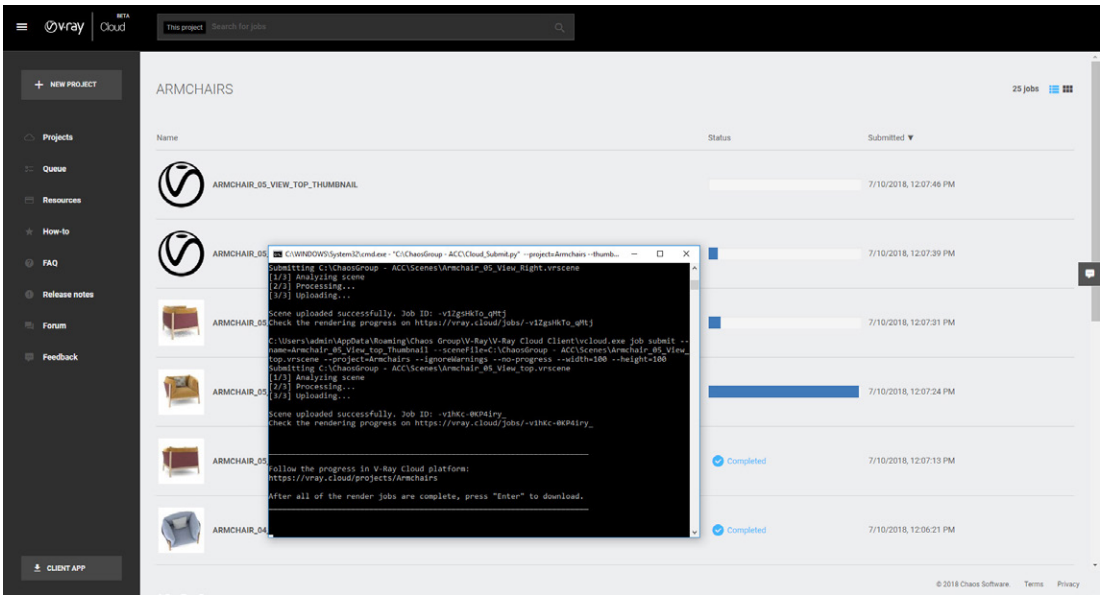
" --project=" and the name of the project (Armchairs in this case).

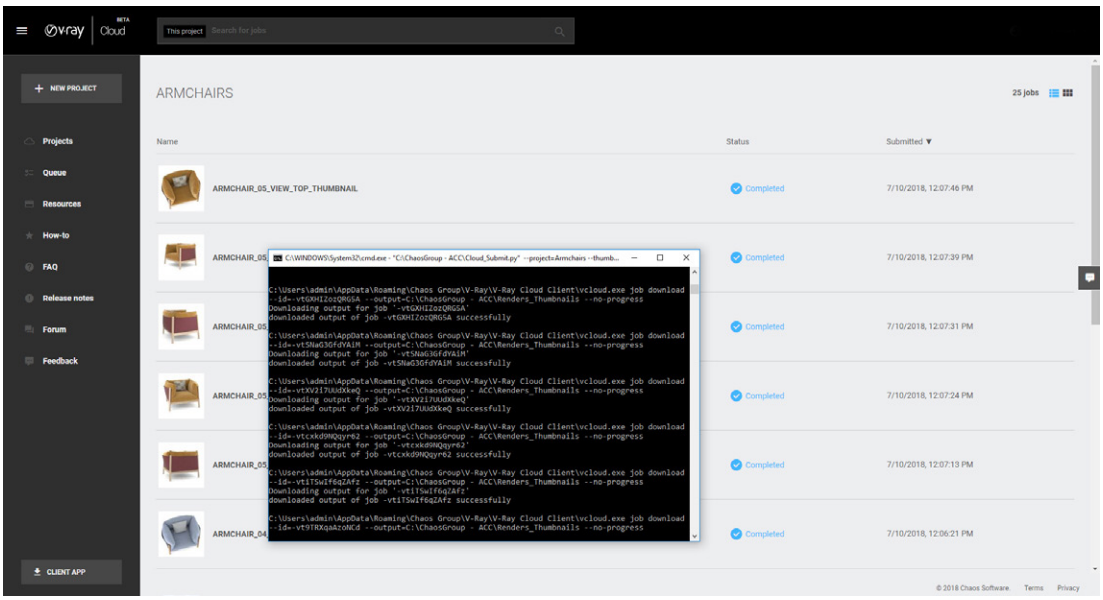


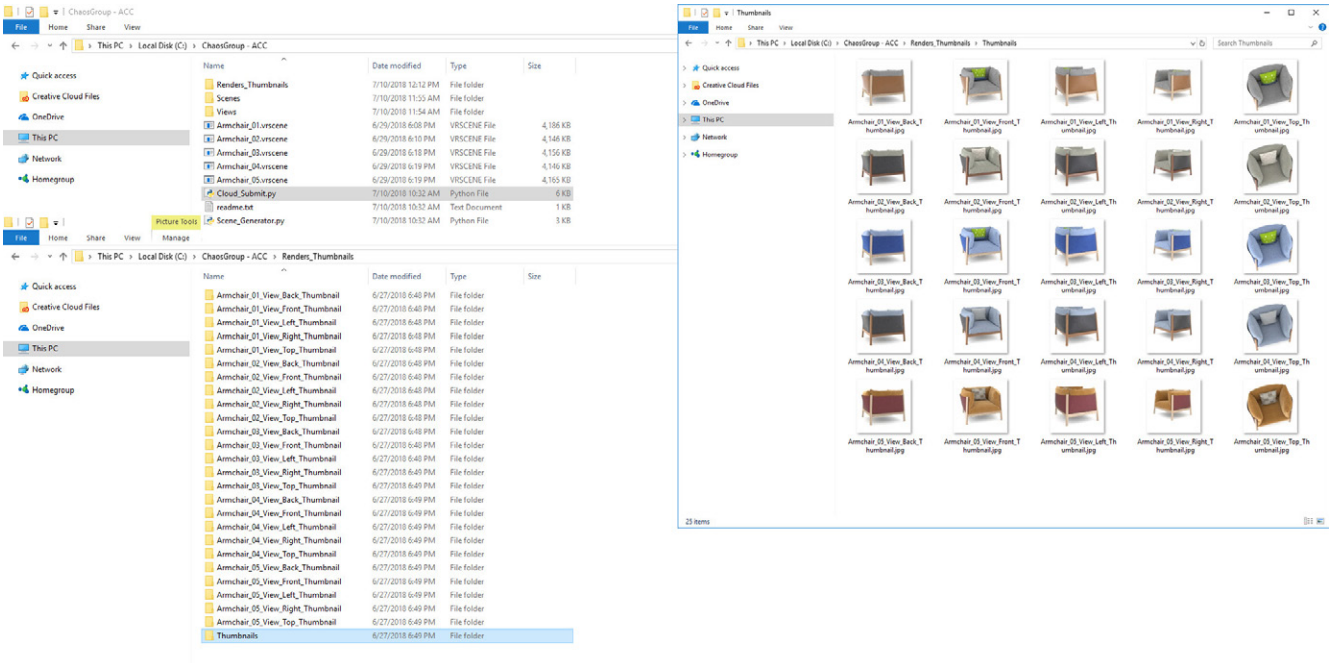
Press "Enter". The script will submit the files from the "Scenes" folder to V-Ray Cloud and it'll start rendering them afterwards. Another way to specify the thumbnail size and project name is to open the script with a text editor and complete the information:

```
import os
import platform
import re
import subprocess
import shutil
import sys

############################## INPUT PARAMETERS ##############################

# Give a name for the project
ProjectName = "Armchair_1"

# Should the renderers be with thumbnail size. Set to True if thumbnails are desired
ThumbnailsOnly = True

# Size of the thumbnails. Used if ThumbnailsOnly is set to True
ThumbnailWidth = '100'
ThumbnailHeight = '100'

##############################################################################

argumentsParser = argparse.ArgumentParser()
argumentsParser.register('type', 'bool', (lambda x: x.lower() in ("yes", "true", "t", "1")))
argumentsParser.add_argument('--project', nargs='?', default=ProjectName)
argumentsParser.add_argument('--thumbnails', nargs='?', type=bool, default=ThumbnailsOnly)
args = argumentsParser.parse_args()
```

- Give a name to the project.
- Change “ThumbnailsOnly” from “False” to “True”. If it’s true, the size will be reduced; if it’s false, the size will be the same as the Rhino file.
- Save.
- Run the script.

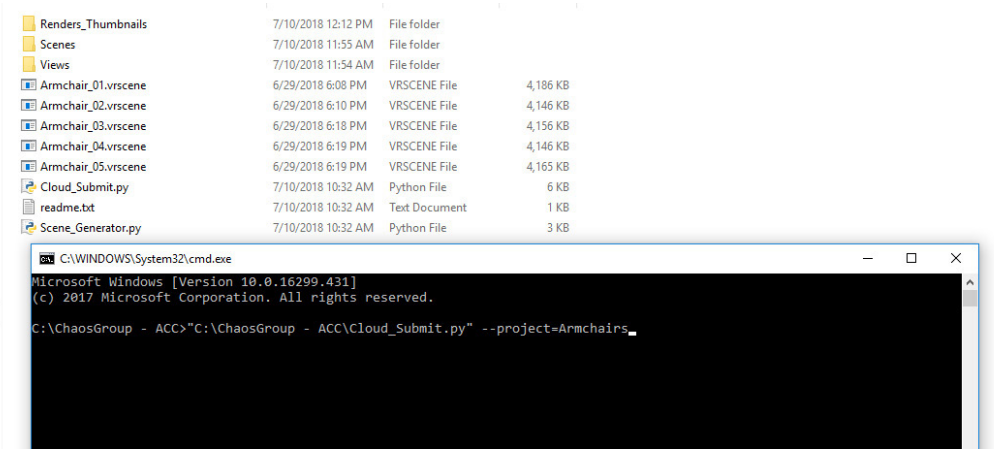






5.2 Render high resolution images

Now let's render the images in high resolution. Follow the Thumbnails steps again, but this time don't type "--Thumbnails=True". Just give a name to the project.

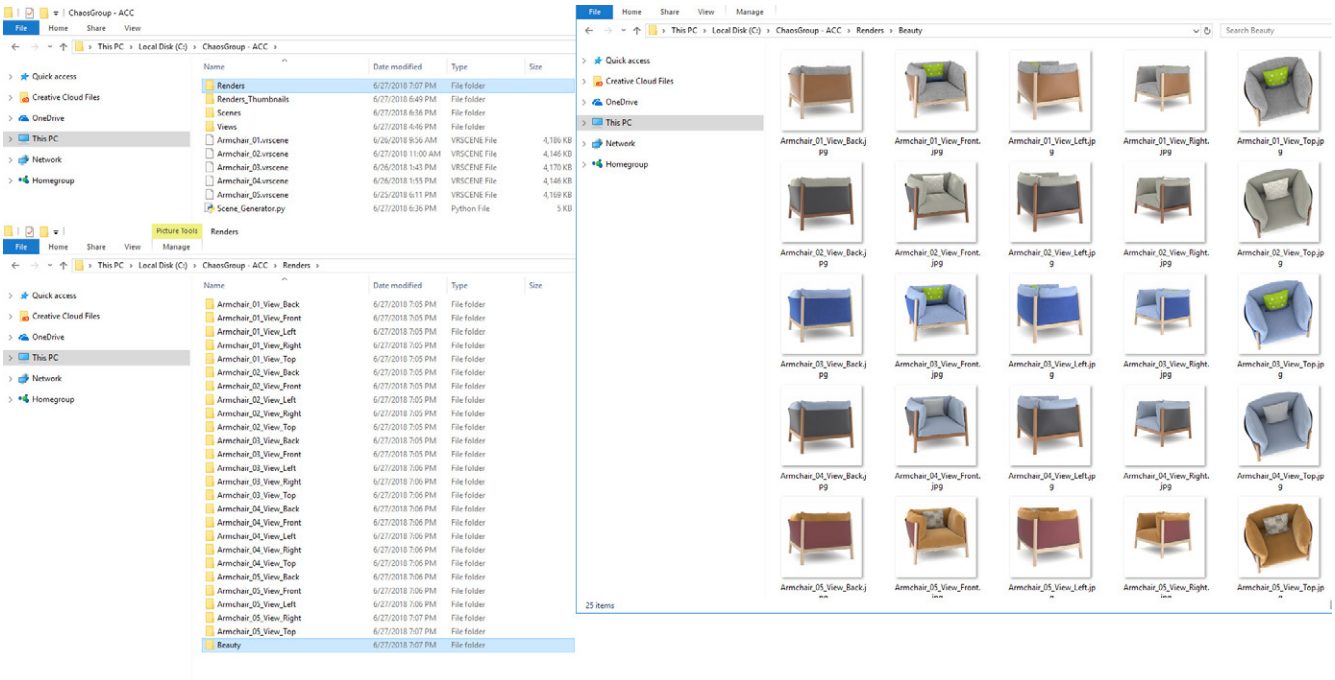


Another way to give a name of the project is to open the script with a text editor.

- Type the name of the project between the apostrophes.
- The "ThumbnailsOnly=" should be "False".
- Save.

```python
import argparse
import glob
import os
import platform
import re
import subprocess
import shutil
import sys

############################## INPUT PARAMETERS ##############################

# Give a name for the project
ProjectName = "Armchair_1"

# Should the renderers be with thumbnail size. Set to True if thumbnails are desired
ThumbnailsOnly = False

# Size of the thumbnails. Used if ThumbnailsOnly is set to True
ThumbnailWidth = '100'
ThumbnailHeight = '100'

##############################################################################

argumentsParser = argparse.ArgumentParser()
argumentsParser.register('type', 'bool', (lambda x: x.lower() in ("yes", "true", "t", "1")))
argumentsParser.add_argument('--project', nargs='?', default=ProjectName)
argumentsParser.add_argument('--thumbnails', nargs='?', type=bool, default=ThumbnailsOnly)
args = argumentsParser.parse_args()

ProjectName = args.project
ThumbnailsOnly = args.thumbnails
```

Run the script, follow the progress, wait until all images are rendered and after that press “Enter”. The renders will be saved in a new folder called “Renders”.



Now all of the chairs from the series are automatically rendered and saved.