MDSANIMA Python Pacage

Release 0.1.1

May 25, 2021

MDSANIMA

Welcome to MDSANIMA Python Packages Documentation!

This is a simple Python Packages developed by Marcin Różewski aka MDSANIMA

CHAPTER 1

Welcome to MDSANIMA Python Package

The package contains modules that will help in calculating rendering time. The package also includes a module for print your text with an animation counting up or down.

1.1 Install Package

Warning: `MDSANIMA` package only works in `Python 3.6` or later.

```
python -m pip install mdsanima                    # install latest version
python -m pip install mdsanima==0.1.1             # install specific version
```

1.2 Upgrade Package

```
python -m pip install --upgrade mdsanima          # upgrade to latest version
python -m pip install --force-reinstall msdsanima # upgrade to latest version
```

1.3 Uninstall Package

```
python -m pip uninstall mdsanima                  # uninstall package
```

1.4 Usage

Module `render_time` returns print of render stats.

```
>>> from mdsanima import render_time
>>> render_time(512, 1, 0, 24, 128)
```

Module `count_down` and `count_up` returns print text and count animation.

```
>>> from mdsanima import count_down
>>> from mdsanima import count_up
>>> count_down('Your text ', 20, 0.1)
>>> count_up('Your text ', 500, 0.001)
```

1.5 Documentation

Full `MDSANIMA` package documentation can be found here documentation

1.6 Languages and Tools

This tools and languages we're using every day.

1.7 3D Stuff

These application we're using every day.

1.8 Latest YouTube Videos

- 4K Sky Timelaps Starlink SpaceX Satellite
- Sharp Old Radio 3D Model Turntable
- Lego Architect 3D Model Turntable
- Lego Mindstorms Reptar 3D Model Turnable
- Lego Mindstorms Grip3r 3D Model Turnable

1.9 Latest Blog Post

- Starlink SpaceX Satellite
- Starlink SpaceX Elon Musk Satellite Photography
- Fruits Dance Man Mograph Animation

- Strawberry Dance Man Animation
- Mograph Animation

1.10 Connect With Me

Hi there, I'm Marcin Różewski aka MDSANIMA. These are all my social media and websites, check it out please. Thanks.

My Social Media and Websites.

1.11 License

Mdsanima is released under the terms of MIT License

1.11.1 Python Package

This is a Documentation of MDSANIMA Python Package

Animation Module

Terminal print output animation.

mdsanima.animation.**anim_ascii**()
Print ascii foto animation.

Returns Ascii animation.

Return type print

Usage:

```
anim_ascii()
```

mdsanima.animation.**count_down**(*print_text*, *count*, *sleep*)
Print text and animation count down.

Parameters

- **print_text** (*str*) – Enter the text to be printed.
- **count** (*int*) – Counter number.
- **sleep** (*float*) – How fast the counts is supposed to be.

Returns Count down animation.

Return type print

Usage:

```
count_down('Your text ', 20, 0.1)
```

mdsanima.animation.**count_up**(*print_text*, *count*, *sleep*)

Print text and animation count up.

Parameters

- **print_text** (*str*) – Enter the text to be printed.
- **count** (*int*) – Counter number.
- **sleep** (*float*) – How fast the counts is supposed to be.

Returns Count up animation.

Return type print

Usage:

```
count_up('Your text ', 500, 0.001)
```

Render Time Module

Simple Render Time Calculator.

mdsanima.render.**render_time**(*frame*, *rt_hours*, *rt_min*, *rt_sec*, *node=1*)

Simple Render Time Calculator.

Parameters

- **frame** (*int*) – How many frame you have to calculate render time.
- **rt_hours** (*int*) – Hours render time one frame.
- **rt_min** (*int*) – Minutes render time one frame.
- **rt_sec** (*int*) – Seconds render time one frame.
- **node** (*int*) – How many node you have. Default 1 node.

Returns Your render time stats.

Return type print

Usage:

```
render_time(512, 1, 0, 24, 128)
```

1.11.2 Changelog

Version 0.1.1

Date October 13, 2020

Some fixes in documentation and other files. Here we have the update steps for `0.1.1` for people to follow.

- @mdsanima: Added new function `anim_ascii` on `animation` module.
- @mdsanima: Fixed some documentation file.
- @mdsanima: Added sphinx documentation.

- @mdsanima: Added keywords in `setup.py`.
- @mdsanima: Added classifiers in `setup.py`.
- @mdsanima: Added extra requires in `setup.py`.
- @mdsanima: Changed `README.rst` file insted of `README.md` file.
- @mdsanima: Changed `CHANGELOG.rst` file insted of `CHANGELOG.md` file.
- @mdsanima: Fixed errors in the documentation, function `render_time`, `count_down` and `count_up`.

Version 0.1.0

Date October 8, 2020

First release `MDSANIMA` Package. Here we have the update steps for `0.1.0` for people to follow.

- @mdsanima: Added function `render_time` on `render` module.
- @mdsanima: Added function `count_down` on `animation` module.
- @mdsanima: Added function `count_up` on `animation` moudle.

1.11.3 About Me

I've been working on VFX and Game Development for the past 10 years now. I make games, cinematic, commercial, cross platform application Windows Linux Android MacOS and iOS.

3D Animation

I love Houdini Sidefx so much. With this great application I create visual effect like smoke, water, cloth simulation, procedural 3D modeling, digital asset for Unreal Engine 4 and much more. Digital asset allow to tweak parameter directly in game engine. In this way, you can make perfect game with only this two excellent software.

Of course I love Blender too. In these software, I make 3D models, sculpt and use this for render my animation. I have experienced on Substance Painter UDIM workflow, Zbrush, Coat 3D, Nuke, Black Magic Davinci Resolve, Black Magic Fusion.

I recently discovered a great software for rendering and layout called Clarisse iFX. This software can handle off the limit the biggest 3D scenes ever, without any display problems. Handle over 10 trylion polygon and no more lagging on 3D viewport, no more LOD - level of detail.

I really like command line rendering, so much fun.

App Development

Our studio also experienced in App Development. We coding in Python, VEX, C#, Java, PHP, HTML and SQL. We working on Windows, Linux Ubuntu, WSL, WSL2 and Cloud Services.

Development Framework Xamarin, Kivy, KivyMD, PySide2, PyQt5 and more. Our favorite coding software is Visual Studio Code and GIT.

We expanded our experience in Cloud Computing like Amazon Cloud AWS, S3, Spot Instance, Azure, Oracle, Google Cloud, Real Time DataBase in Firebase.

Note: If you have any questions please send me messages at this email address.

1.11.4 Showreel

Short Film "The Waiter"

Air Intake Set

Terrace System

Short Film "City"

Tip: More animation you can see in my YouTube Channel and Vimeo Channel.

1.11.5 Subscribe

We will keep to let you know about our new products, special offer, a limited time promotion. After subscribing to the list, you will get access to exclusive 3D models library.

Note: Subscribe our Mailing List. Just follow this link MDSANIMA Subscribe and enter your email.

1.11.6 Developer Help

In this section are **important information**, coding tips, Windows and Linux tips, links etc.

Check it out if you don't know something. I will write here various things that may be useful one day.

1.11.7 Linux Help

In this section you will find some useful information about using `Linux Ubuntu 18.04`. Of course you can also use WSL or WSL2.

Create Swap File

Quick tutorial how to create a swap file. Swap File Tutorial.

```
sudo fallocate -l 1G /swapfile
sudo dd if=/dev/zero of=/swapfile bs=1024 count=1048576
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
sudo swapon --show
sudo free -h
remove swap
sudo rm /swapfile
```

WLS GUI Config

Quick tutorial how to create a WSL GUI. Config GUI Linux Tutorial on YouTube.

Config GUI

```
sudo apt update &&  sudo apt upgrade
sudo sed -i 's/^PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd_
↪config
sudo /etc/init.d/ssh restart
sudo passwd ubuntu
sudo apt install xrdp xfce4 xfce4-goodies tightvncserver
echo xfce4-session$ /home/ubuntu/.xsession
echo xfce4-session> /home/ubuntu/.xsession
sudo cp /home/ubuntu/.xsession /etc/skel
sudo sed -i '0,/-1/s//ask-1/' /etc/xrdp/xrdp.ini
sudo service xrdp restart
sudo apt install -y ubuntu-desktop xrdp         # additional package
sudo reboot
```

Connect Putty

```
tunel:  secure port:5901
    destination -> private ip -> 111.11.11.111:3389
```

Connect WSL

```
echo xfce4-session > /.xsession
sudo /etc/init.d/xrdp start
```

RDP Connection

```
RDP remote desktop connection
Computer 127.0.0.1:3390
Computer localhost:3390
```

1.11.8 Sphinx Auto Documentation

Python Sphinx package automatically generates technical documentation for python modules `.py` `.rst` and `.md` files.

Install Sphinx

```
python -m pip install sphinx                    # install sphinx package
python -m pip install sphinx-autoapi            # install sphinx extensions package
python -m pip install sphinx-tabs               # install sphinx extensions package
python -m pip install sphinx-prompt             # install sphinx extensions package
python -m pip install sphinx-bootstrap-theme    # install sphinx theme package
python -m pip install sphinx-rtd-theme          # install sphinx theme package
python -m pip install recommonmark              # install markdown pacage
```

Create Sphinx Documentation

```
mkdir docs
cd docs
sphinx-quickstart
```

Select create documentation with build and source separation. Edit `conf.py` file. Add extension and change the themes.

Windows PowerShell

```
.\make.bat html
```

Linux

```
make html
```

Add directives to the .rst file to make your docs quickly. All directives avaiable in Sphinx Documentation.

```
.. include:: ../../CHANGELOG.rst

.. automodule:: mdsanima.animation
    :members:

.. automodule:: mdsanima.render
    :members:

.. toctree::
    :maxdepth: 2

    developer

.. tip::

    Tip text

.. tabs::

    .. tab: First Tab

        Text or directives as you like

    .. tab: Last Tab

        Text or directives as you like
```

Add your files .rst as you liked, check that everything is correct in the documentation and then type:

Windows PowerShell

```
.\make.bat clean
```

Linux

```
make clean
```

Add to Read The Docs

Quickstart for GitHub Hosted Projects. By the end of this quickstart, you will have a new project automatically updated when you push to GitHub.

- Create an account on Read the Docs. You will get an email verifying your email address which you should accept within 7 days.
- Log in and click on `Import a Project`.
- Click `Connect to GitHub` in order to connect your account's repositories to GitHub.
- When prompted on GitHub, give access to your account.
- Click `Import a Repository` and select any desired repository.
- Change any information if desired and click `Next`.
- All done. Commit away and your project will auto update.

1.11.9 Send to PyPi

Instructions on how to create your own python packages and send it to PyPi.

My First Pacage

Make sure your directory package structure looks like this.

```
my_package
    __init__.py
    module_1.py
    module_2.py
```

Code in `__init__.py` should look like this.

```
from .module_1 import function_1
from .module_2 import function_2
```

Now if you want to call your `function_1` in a new `main.py` file, outside the `my_package` module.

```
main.py
my_package
    __init__.py
    module_1.py
    module_2.py
```

Your code should look like this.

```
from my_package import function_1
```

Instead of.

```
from my_package.module_1 import function_1
```

Now your code is shorter. It's time to write you first function.

```
def function_1(x, y, z=10):
    mult = x * y * z

    return value
```

Pacage Docstring

Now enter the documentation into the code, the VS code extension will help us Python Docstring Generator `njpwerner.autodocstring`, or write it yourself.

Each function in the your packages should have its own docstring documentation. - Describe in plain language what your function does. - Explain what parameters and types function expects. - Explain what return the function gives. - You can write a simple example of how to use.

Now you just need to position the cursor after the colon, hit enter. Cursor must be on the line directly below the definition to generate full auto populated docstring.

Press enter after opening docstring with triple quotes (`""" or '''`) Keyboard shortcut `ctrl + shift + 2` or `cmd + shift + 2` for mac user. Documentation is almost ready, just complete the summary, types and description.

```
def function_1(x, y, z=10):
    """Sum of the multiplication x, y and z.

    Args:
        x `int`: Number.
        y `int`: Another number.
        z `int` optional: Another number. Defaults to 10.

    Returns:
        int: sum
    Examples: function_1(64, 128, 256)
    """
    mult = x * y * z

    return value
```

Add README.md or README.rst file. Add LICENSE file. Create the blank `setup.py` file.

Now the directory structure of your package must be as follows.

```
my_package
    LICENSE
    README.rst
    setup.py
    my_package
        __init__.py
        module_1.py
        module_2.py
```

Setup File

Open the `setup.py` file and paste the code there. You can also add other features to it. Check the pypi documentation.

```
'''
MDSANIMA Setup
```

(continues on next page)

(continued from previous page)

```
'''

import sys
import pathlib
from setuptools import setup, find_packages

HERE = pathlib.Path(__file__).parent

CURRENT_PYTHON = sys.version_info[:2]
REQUIRED_PYTHON = (3, 6)

# This check and everything above must remain compatible with Python 2.7.
if CURRENT_PYTHON < REQUIRED_PYTHON:
    sys.stderr.write("""==========================
Unsupported Python Version
==========================
This version of MDSANIMA requires Python {}.{}
but you're trying to install it on Python {}.{}
""".format(*(REQUIRED_PYTHON + CURRENT_PYTHON)))
    sys.exit(1)

VERSION = '0.1.0'
PACKAGE_NAME = 'my_package'
AUTHOR = 'my name'
AUTHOR_EMAIL = 'myemail@address.com'
URL = 'https://github.com/mdsanima/mdsanima'

LICENSE = 'MIT License'
DESCRIPTION = 'The package contains modules that will help in calculating rendering␣
↪time.'
LONG_DESCRIPTION = (HERE / "README.rst").read_text()
LONG_DESC_TYPE = "text/x-rst"

INSTALL_REQUIRES = [
        'humanfriendly'
]

KEYWORDS = [
        'mdsanima',
        'render time',
]

setup(name=PACKAGE_NAME,
        version=VERSION,
        description=DESCRIPTION,
        long_description=LONG_DESCRIPTION,
        long_description_content_type=LONG_DESC_TYPE,
        author=AUTHOR,
        license=LICENSE,
        author_email=AUTHOR_EMAIL,
        url=URL,
        install_requires=INSTALL_REQUIRES,
        packages=find_packages(),
        extras_require={
            "docs": [
                'sphinx',
                'sphinx-autoapi',
```

(continues on next page)

(continued from previous page)

```
                'sphinx-rtd-theme',
                'sphinx-bootstrap-theme',
                'sphinx-prompt',
                'sphinx-tabs',
                'recommonmark'
                ],
        },
        python_requires='>=3.6',
        keywords=KEYWORDS,
        classifiers=[
            'Development Status :: 1 - Production/Stable',
            'Intended Audience :: Developers',
            'License :: OSI Approved :: MIT License',
            'Natural Language :: English',
            'Programming Language :: Python :: 3',
            'Programming Language :: Python :: 3.5',
            'Programming Language :: Python :: 3.6',
            'Programming Language :: Python :: 3.7',
        ],
    )
```

Warning: Every time you release a new version of PyPI, you must have to bump version number.

Create Acconut on PyPi

We're almost there. Just create an account here Master PyPi. You will need it to ship your package around the world. This is your main account. Log in and receive an email with verification.

We will need a second account, which we set up here Test PyPi. This is your test account. It will be needed to test the package, if everything is fine we send the package to the main account. We set up an account with exactly the same data as the main account. We log in and receive an email with the verification.

Build Package

We're almost done. We just need to put commands in to terminal and which will create an installable Python Package from your `code` and send it to `PyPi`.

Install Twine

```
python -m pip install twine
```

Build Setup

```
python setup.py sdist bdist_wheel
```

It created some new directories for us, such as `dist`, `build` and `your_package.egg-info`.

Tip: Add these directories to the `.gitignore` file to prevent installation files from being pushed into the repository.

Check the Build

```
twine check dist/*
```

Send to Test PyPi

```
twine upload --repository-url https://test.pypi.org/legacy/ dist/*
```

Go to test.pypi.org and check if everything is ok, if yes, we can finally send our package to PyPi.

Finally Send To PyPi

```
twine upload dist/*
```

Tip: That's it. Now anyone can install your package with `python -m pip install my_package`.

Good job! Keep Coding! `We LOVE PYTHON`

Bash Script

A bash script that builds, checks and deploys the entire package at a time, with only one command.

Creat a new file named `build_deploy.sh` in ths same direcotry as `setup.py`. Put this code on this file.

```
rm -r dist ;
python setup.py sdist bdist_wheel ;
if twine check dist/* ; then
    if [ "$1" = "--test" ] ; then
        twine upload --repository-url https://test.pypi.org/legacy/ dist/*
    else
        twine upload dist/* ;
    fi
fi
```

You can also add commands to this script to generate documentation, before making the package or other actions as you like.

Now run this command.

```
./build_deploy.sh
```

or

```
./build_deploy.sh --test
```

Warning: When you run the script for the first time, give execute permission `chmod + x build_deploy.sh`.

Write Article

Done, great job! Now your package can be installed by anyone around the world, but people don't know it yet. Write an article on your blog, on social media, informing people about this package. It can be really useful for someone.

1.11.10 Terminal Command

This is a list of useful Terminal Command. Works on Windows PowerShell Terminal, Linux Ubuntu and WSL Windows Subsystem for Linux.

Python

List of useful Python Terminal Command.

```
python -m pip install mdsanima                          # install package
python -m pip install mdsanima==0.1.1                   # install specific version
python -m pip install --upgrade mdsanima                # upgrade to latest version
python -m pip install --force-reinstall mdsanima        # force reinstall
python -m pip uninstall mdsanima                        # uninstall package
python -m pip list                                      # list of installed packages
nohup python -u script.py > output.log &                # run python script background␣
↪save log
```

Linux

List of useful Linux Terminal Command.

```
ls -l -a -h                                 # show list hide file show MB
sudo curl ifconfig.me                       # show ip
df -H /                                     # disc spaces
free -h                                     # ram info
neofetch                                    # system info
glances                                     # system info
htop                                        # system stats
nvidia-smi                                  # show gpu stats
sudo ubuntu-drivers autoinstall             # install gpu drivers
watch -n 0.5 nvidia-smi                     # show gpu stats and refresh
cat file.txt                                # print file
nano file.txt                               # edit file
sudo rm -r file*                            # delete all files
pkill "python*"                             # kill all python process
ssh-keygen -C "your@email.com"              # create ssh key
aliases                                     # show all your aliases
aliases my_ali="sudo rm -r file*"           # create aliases
history                                     # command line history
sudo apt-get update && apt-get upgrade      # update and upgrade
tmux --help                                 # multi window show help
```

1.11.11 VS Code Extension

List of Visual Studio Code useful Extension. Name and Extension Id.

- Python `ms-python.python`.
- Python Docstring Generator `njpwerner.autodocstring`.
- reStructuredText `lextudio.restructuredtext`.
- GitHub `knisterpeter.vscode-github`.
- Git History `donjayamanne.githistory`.
- Remote SSH `ms-vscode-remote.remote-ssh`.
- Remote SSH: Editing Configuration `ms-vscode-remote.remote-ssh-edit`.
- Remote WSL `ms-vscode-remote.remote-wsl`.
- Remote Development `ms-vscode-remote.vscode-remote-extensionpack`.
- Bracket Pair Colorizer 2 `coenraads.bracket-pair-colorizer-2`.
- SQLite `alexcvzz.vscode-sqlite`.
- SFTP `liximomo.sftp`.
- vscode-icons `vscode-icons-team.vscode-icons`.
- Todo Tree `gruntfuggly.todo-tree`.

1.11.12 Azure DevOps CI

AWS agent pool configuration for Azure DevOps CI self hosted agent. Tutorial on YouTube.

First step is create a new personal access token on Azure.

Click `Show All Scopes`, select `Agent Pools Read & Manage` options, choose `Deployment Groups Read & Manage` option and finnaly click `Create`.

Next step go in to Azure setting and `Get agent`. On this step everything is explained in documentation on this page.

Run instance on AWS with Windows or Linux. If you need GPU compute power, select Elastic GPU in the instance setting. Open terminal and put the command you saw earlier in the documentation `Get agent` pages. Tested on `t2.xlarge` instance with GPU.

Everything is explained in the video tutorial above. Check it out if you have any problems.

You can also use your computer as an agent, then you don't need to run the machine on AWS or any other cloud computing provider. Just install the agent and it's ready.

Continuous Integration CI also available in GITHUB ACTION.

1.11.13 Changelog Info

Keep a changelog. We're using Semantic Versioning. Don't let your friends dump git logs into changelogs. All notable changes to this project will be documented in CHANGELOG.rst file. More information on versioning can be found here Semantic Versioning and here Keep a Changelog. Check out these links if you want to know more.

For the version number `MAJOR.MINOR.PATCH` increment the:

- **`MAJOR` version when you make incompatible API changes.** For example with version `1.11.0` to version `2.0.0`.

- **`MINOR` version when you add functionality in a backwards compatible manner.** For example with version `0.10.2` to version `0.11.0`.
- **`PATCH` version when you make backwards compatible bug fixes.** For example with version `0.10.1` to version `0.10.2`.

Additional markings for pre-release or build metadata are available as extensions of `MAJOR.MINOR.PATCH` format.

1.11.14 Useful Links

Here I will post links that may be useful for someone.

- Blog Post Workflow GitHub Action
- GitHub Readme Stats Card
- Create Badge And Change Styles
- Readme Gallery Creator for GitHub
- Cron Job Simple Editor
- Stats Download PyPi Package Badge
- Read The Docs Documentation Pages
- Read The Docs GitHub Pages

1.11.15 Blender Rendering

How to Blender command line rendering works. Let me show you an example on an instance with AWS Spot. Of course, you can use any other provider as you like.

Create AWS Spot Instance

- Log in to your `AWS` account.
- Go to `EC2 Dashboard`.
- Then go to `Instances`.
- Select `Launch Instances`.
- Choose an Amazon Machine Image AMI.
- Select this one `Ubuntu Server 18.04 LTS`.
- Next choose an Instance Type.
- For testing choose `t2.large`.

Tip: The best instance for rendering is `Compute Optimized`.

- Click `Next: Configure Instance Details`.
- Now configure your instance as you like.
- Select Purchasing option `Request Spot Instance`

Warning: This option is several times cheaper than a regular instance. But it can happen that you get disconnected from the instance. Spot instances are the unused computing power of Amazon's cloud. If they need that power again, you'll be disconnected. Relax, your data will not be deleted, because we will do EFS Elastic File Storage and connect to this instance.

- Click `Next: Add Storage`.
- Default its fine.
- Click `Next: Add Tags`.
- Default its fine.
- Click `Next: Configure Security Group`
- Default ist fine too for testing.
- Finnaly click `Review and Launch`.
- Click `Launch`.
- Create new SSH Key. Save this file on your machine.

At this point we have created one spot instance and next we are going to connect `EFS` Elastic File System.

Now we wait about 2 minutes until our instance is fully operational.

Connect to Spot Instance

At this point, we can connect to the instance, we can do it in 2 ways.

Connect with PuTTy on Windows

The first time we need to convert your SSH key so that PuTTYy can read it.

- Click RMB on the PuTTy icon in your system toolbar.
- Choose the PuTTY Key Generator.
- Load your SSH Key.
- Click `Save public key` and choose your location.
- File should be a name like this `ssh-key-conv.ppk`.

At this point, we have the key converted. All you need to do now is fire up PuTTy Configuration.

- Fill it up `Host Name`. Should look like this `ec2-1-11-111-111.us-east-2.compute.amazonaws.com` or IP address `1.11.111.111`.
- In the `Category` choose `Connection` and `SSH`. Next choose `Auth`.
- Then enter the path to the converted ssh key in `Private key file for authentication:`.
- Finally click `Open`. That's it, now we are logged in to the spot instance.

Conncet with SSH on Linux WSL

This option is much simpler. Only put the command in to terminal.

```
ssh -i "ssh-key.pem" ubuntu@ec2-1-11-111-111.us-east-2.compute.amazonaws.com
```

Done.

EFS Elastic File System

- Go to `Amazon Elastic File System EFS` in your AWS dashboard.
- Create file system.
- Save the `IP address` on your `Availability Zone` where your spot instance works. Should look like this `222.22.2.222`.

Now go to the terminal and enter this command.

```
apt-get update && apt-get install nfs-common -y
cd /mnt
mkdir efs
ls
sudo mount -t nfs4 222.22.2.222:/ /mnt/efs
```

Everything is ready, now you can mount this EFS to multiple instances. Just put the command showed up above for each instance. YouTube Tutorial EFS

Command Line Rendering

Now we can start the actual rendering of our scenes in the Blender and Cycles render engine. We will use `Blender Command Line Rendering`.

Download Blender

In the terminal put this command.

```
wget https://your_serwer.com/aws/blender-2.90.0-linux64.tar.xz
wget https://your_serwer.com/aws/your_blender_scene_v01_aws.tgz
```

If you don't have your own server where you store files, you can use this command.

Note: You must be in the folder where you have the files to send `./` or just put path.

In Microsoft PowerShell Terminal.

```
pscp -i "C:\path\ssh-key-conv.ppk" ./blender-2.90.0-linux64.tar.xz ubuntu@1.11.111.
↪111:/mnt/efs/
pscp -i "C:\path\ssh-key-conv.ppk" ./your_blender_scene_v01_aws.tgz ubuntu@1.11.111.
↪111:/mnt/efs/
```

In Linux WSL Terminal.

```
scp -i "/home/name/path/ssh-key.pem" ./blender-2.90.0-linux64.tar.xz ubuntu@1.11.111.
↪111:/mnt/efs/
scp -i "/home/name/path/ssh-key.pem" ./your_blender_scene_v01_aws.tgz ubuntu@1.11.111.
↪111:/mnt/efs/
```

Tip: SSH key in Windows should be in the `C:\Users\name\.ssh\` directory. SSH key in Linux should be in the `/home/name/.ssh/` directory.

These commands allow you to upload files from your computer to the AWS server. You can also reverse the operations and download files from the server. You just need to change the path. I'll show it in a moment how to download the files.

All downloaded files should be in `/mnt/efs/`.

```
cd /mnt/efs/                                    # go to direcotry
tar -xf ./blender-2.90.0-linux64.tar.xz         # unpack Blender file
mv ./blender-2.90.0-linux64 ./b290    # move Blender to directory
tar -xf ./your_blender_scene_v01_aws.tgz        # unpack Scene file
mv ./your_blender_scene_v01_aws ./blend         # move Scene to directory
```

Now Blender is almost ready to go.

Install Pacage

Now update your system and install the necessary packages to run Blender properly.

```
sudo apt-get update && sudo apt-get dist-upgrade
sudo apt-get install libboost-all-dev
sudo apt-get install libgl1-mesa-dev
sudo apt-get install libglu1-mesa libsm-dev
sudo apt-get install libxi6
sudo apt-get install libxrender1
```

Blender is ready to go.

Render Your Scene

When everything installed properly, we can start rendering.

CPU One Frame

This command allow to render only one frame on CPU.

```
sudo /mnt/efs/b290/blender -b '/mnt/efs/blend/sc_aws.blend' -o '/mnt/efs/blend/01/' -
↪f 1
```

CPU All Frame

This command allow to render all frame on CPU.

```
sudo /mnt/efs/b290/blender -b '/mnt/efs/blend/sc_aws.blend' -o '/mnt/efs/blend/01/' -
↪s 1 -e 26 -a
```

GPU All Frame

This command allow to render all frame on GPU. You must have a special instance with Elastic GPU, or dedicated e.g. with 8 GPU Nvidia v100. Check the Amazon EC2 On-Demand Pricing or Spot Instance Pricing. All links at the bottom of the page.

You must be in `/mnt/efs/`. You must also have a python script `setgpu.py` on your server in destination `/mnt/efs/setgpu.py`. You can send it at the very beginning or create this file right now, just put this command on the terminal.

```
cd /mnt/efs/
nano setgpu.py
```

You have just created an empty python file. Now put this code in to the setgpu.py file. Copy it, and paste it into the nano editor by right clicking on the terminal AWS machine.

```
import re
import bpy
scene = bpy.context.scene
scene.cycles.device = 'GPU'
prefs = bpy.context.preferences
prefs.addons['cycles'].preferences.get_devices()
cprefs = prefs.addons['cycles'].preferences
print(cprefs)
# Attempt to set GPU device types if available
for compute_device_type in ('CUDA', 'OPENCL', 'NONE'):
    try:
        cprefs.compute_device_type = compute_device_type
        print('Device found',compute_device_type)
        break
    except TypeError:
        pass
# Enable all CPU and GPU devices
for device in cprefs.devices:
    if not re.match('intel', device.name, re.I):
        print('Activating',device)
        device.use = True
```

Now just save the file and close it `crtl + x -> y -> enter`.

Great now you can run this command to activate gpu rendering.

```
cd /mnt/efs/
sudo ./b290/blender -P setgpu.py -b '/mnt/efs/blend/sc_aws.blend' -o '/mnt/efs/blend/
↪01/' -s 1 -e 26 -a
```

Tip: You can use this command `/mnt/efs/b290/blender --help` to show help.

The first part `sudo /mnt/efs/b290/blender -b` of the command.

- Allow to start blender in the background `-b`.

Second part `'/mnt/efs/blend/sc_aws.blend'` of the command.

- Your file to be rendered.

Last part `-o '/mnt/efs/blend/01/' -s 1 -e 26 -a` of the command.

- Output directory `-o`.
- Frame to render `-f 1`.
- Frame start `-s 1`.
- Frame end `-e 26`.
- Animation playback `-a`.
- Run the given Python script file `-P setgpu.py`.

Check out the Blender Ducumentation for more info.

Download Ouptut File

Now it's time to download your render file output. We do the same as with *Download Blender*, we just swap paths like this.

In Microsoft PowerShell Terminal.

```
cd C:\your_blender_output\              # download all the output file in this␣
↪directory
pscp -i "C:\path\ssh-key-conv.ppk" ubuntu@1.11.111.111:/mnt/efs/blend/01/* ./
```

In Linux WSL Terminal.

```
cd /home/name/your_blender_output/      # download all the output file in this␣
↪directory
scp -i "/home/name/path/ssh-key.pem" ubuntu@1.11.111.111:/mnt/efs/blend/01/* ./
```

Tip: Right after rendering done in your instance you can exit on the instance and delete it. Files that rendered are on EFS so they will not be deleted. Now just turn on cheaper instance and download all files.

Now you know how to render Blender file in the cloud with command line rendering. So just now use my Render Time Package and calculate render time for your scenes.

Just type in the terminal.

```
python -m pip install mdsanima
```

More Instances

Now you can create more instances with more processors. You already have a blender installed on the EFS, so you won't have to repeat this step for each instance. You only need to run an instance, connect to it and select files for rendering. Enter the command to connect EFS and enter the command for rendering.

Of course it is also possible to run instances on AWS with Linux Ubuntu Elastic GPU or dedicated GPU and activate GUI to have a live view. Rather, it only works with a small scene and one On-Demand instance or Spot Instance. You would have to use this method Developer Config GUI

You can also use the Python AWS API to run an On-Demand Instance or Spot Instance in the command line and python script. You will save time and pay less for rendering.

In the future, I will write a some Python Package that will run the given number of instances on AWS. Run the necessary commands to update the servers. It will connect all instances from EFS Elastic File System at once and run rendering on all instances simultaneously. After rendering is finished, it will automatically delete instances and create one small instance to download all files.

AWS Useful Links

Here I will post useful links to Amazon AWS Cloud.

- Amazon EC2 On-Demand Pricing
- Amazon EC2 P2 Instances
- Amazon EC2 P3 Instances
- Amazon EC2 G3 Instances
- Amazon Elastic Inference Pricing
- Amazon Elastic Graphics
- Amazon Elastic Graphics Documentation
- Recommended GPU Instances
- AWS Pricing Calculator

1.11.16 Houdini Rendering

How to Houdini command line rendering works. Let me show you an example on an instance with AWS Spot. Of course, you can use any other provider as you like.

First things we know to do. You have to do them to start rendering.

- Create AWS Spot Instance
- Connect to Spot Instance
- Create EFS Elastic File System

Command Line Rendering

Now we can start the actual rendering of our scenes in the Houdini and Mantra render engine. We will use `Houdini Command Line Rendering`.

Download Houdini

I will use this Python Script to Download Houdini. Auto install Houdini script for Windows and Linux developed by @paulwinex

Just clone this repository to our AWS Linux Ubuntu Instance. Put this code in the terminal.

```
cd /mnt/efs/
git clone https://github.com/paulwinex/houdini_install_script.git
```

Then run this script like this.

```
cd /mnt/efs/
sudo python ./houdini_install.py -u username -p password -i /opt/houdini -s y
```

Replace `username` and `password` with your login details from SideFx Website

Now upload your houidni files to the server you want to render. Use this method of uploading files to the server.

Houdini is Installed and then running this command.

```
source houdini_setup
hkey
```

Install Houdini License

The next steps is to install the Houdini License on the server. Put this command in the terminal.

```
sesictrl -L
```

Follow the instructions to download the Server Licenses.

Render Your Scene

When everything installed properly, we can start rendering on this server.

You need to create a file like this.

```
cd /mnt/efs/
nano render.cmd
```

Then put this code in to the `render.cmd` file.

```
# Rendering From the Command Line
# Open the Hip File
mread your_scene_aws.hiplc

# Bake FEM Simulation
#render -V /obj/roberto/filecache1/render

# Turn on Load From Disk
#opparm /obj/roberto/filecache1/ loadfromdisk on
#python -c 'hou.parm("/obj/roberto/filecache1/loadfromdisk").set(1)'

# Render Camera
render -V cam01

# quit
```

Now you can start rendering by typing.

```
hscript render.cmd
```

That's it, now you are waiting for the result.

Warning: Remember your paths in the `.hip` file must be relative path. Otherwise it won't render properly.

Download Output File

Now it's time to download your render file output. Use this method of downloading files from the server.

That's it, exit the server and delete it. Now you know how to render Houdini file in the cloud with command line rendering. So just now use my Render Time Package and calculate render time for your scenes.

Just type in the terminal.

```
python -m pip install mdsanima
```

Now go into this step

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

```
WE LOVE PYTHON THIS IS THE BEST PROGRAMING LANGUAGE IN THE ENTIRE WORLD!
```

Python Module Index

m

Index