CG TD Pipeline

发布 *1.0.4*

2022 年 03 月 11 日

Contents

CHAPTER 1

前言

1.1 流程

学习和掌握 CG 项目管道流程（Pipeline&Workflow）对于 TD 来说是首要职责，下面有张很有趣的管道流程图，是照明娱乐公司关于小黄人电影制作全流程，包含了一部影片从故事板到大荧幕的所有环节，大家可以找找属于自己的小黄人。

- 微信：13851709904
- Email：huweiguo@do-vfx.com
- 微信公众号：CGRnDStudio
- GitHub：https://github.com/CGRnDStudio

1.5 致谢

劲爆羊、薛定谔是铲屎官、徐国梁、臧长龙、鹅大、机器猫、A · Better、热冬、Taka

1.6 赞赏

文字如果对你有所启发，就赞赏一下 up 主加个鸡腿补补身子吧，后面会有更多更好的内容更新 ~(3)~


andyvfx
“非学无以广才，非志无以成学！”
DO-VFX.COM - Andy 的赞赏码

CHAPTER 2

Python

Python 在 CG 行业中应用广泛，大部分 DCC 软件，比如 Houdini、Maya、Nuke、Katana、C4D 等等都用 Python 作为接口编程语言，因其开发效率高，代码简洁容易上手的特性，为没有系统学习过计算机原理的艺术家也可以编写自己的工具插件以提高日常工作效率，因此掌握 Python 知识体系是每个艺术家的当务之急。

Contents:

2.1 Python 2.x VS 3.x

- 3.x 版本中取消 print 空格打印的写法，只能用内置函数 print()

```
# 2.x
>>> print 100
100
>>>
```

- 3.x 版本中新的解包写法，2.x 中会语法异常

```
# 3.x
>>> a, *b = [1, 2, 3, 4]
>>> a
1
>>> b
```

(下页继续)

(续上页)

```
[2, 3, 4]
>>>
```

- 3.x 版本中默认使用 utf-8 编码，下面的写法在 3.x 中是合法的，但禁止使用

```
# 3.x
>>> 变量 = 100
>>> 变量
100
>>>
```

- 3.x 版本中取消 xrange() 写法，range() 返回生成器

```
# 2.x
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> xrange(10)
xrange(10)
>>> type(xrange(10))
<type 'xrange'>
>>>

# 3.x
>>> range(10)
range(0, 10)
>>> xrange(10)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'xrange' is not defined
>>> type(range(10))
<class 'range'>
>>>
```

- 3.x 版本中除法的精度变浮点数

```
# 3.x
>>> 1 / 2
0.5
>>>
```

- 3.x 版本中 map，filter 从内置函数变成了类，得到的结果是迭代对象

```
# 3.x
>>> map
<class 'map'>
>>> filter
<class 'filter'>
```

- 3.x 版本中 reduce 从内置函数挪到 functools 模块中

```
# 3.x
>>> from functools import reduce
>>> reduce
<built-in function reduce>
>>>
```

- 3.x 版本中没有保留 raw_input()，使用 input()，在 2.x 版本中 raw_input() 将所有输入作为字符串看待，返回字符串类型，input() 接收实例对象的输入，在 3.x 中 input() 接收任意输入，将所有输入默认当字符串处理，并返回字符串类型。

```
# 3.x
>>> var = input("Enter something:")
Enter something:10
>>> type(var)
<class 'str'>
>>> var = input("Enter something:")
Enter something:andy
>>> type(var)
<class 'str'>
>>>
```

- 3.x 版本中字典方法去掉了 has_key()、iteritems()、iterkeys()、itervalues()、viewitems()、viewkeys() 和 viewvalues()，其中 items()、keys()、values() 返回迭代器

```
# 3.x
>>> dir(dict())
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__
↪', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__
↪init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__
↪new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setitem__', '__
↪sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'items',
↪'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']
>>>
# 2.x
```

(下页继续)

(续上页)

```
>>> dir(dict())
['__class__', '__cmp__', '__contains__', '__delattr__', '__delitem__', '__doc__', '__eq__
↪', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__
↪init__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__reduce__',
↪'__reduce_ex__', '__repr__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__
↪subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'has_key', 'items', 'iteritems',
↪'iterkeys', 'itervalues', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values',
↪'viewitems', 'viewkeys', 'viewvalues']
>>>
```

- 3.x 版本中去掉了 <> 不等于的写法，统一用!=

```
# 2.x
>>> 100 <> 200
True
>>>
```

- 3.x 版本中添加海象运算符
- 3.x 版本中添加 F-Strings 格式化字符串方法
- 3.x 版本中将 reload 加入到 importlib 模块中，不能直接使用
- ___builtin___ VS builtins

2.2 Python 高级功能

- Python 小整数池
- Python 异常处理
- Python 生成器 yield 以及迭代器
- Python 函数式编程
- Python 高阶函数，匿名函数，闭包函数以及装饰器
- Python 递归函数
- Python anaconda
- IPython
- Jupyter lab
- Jupyter notebook
- import this

- import ___hello___
- import antigravity
- Python 3 f-string 格式化
- ___builtin___
- ___file___ & ___path___
- ___name___
- Python 列表推导
- Python 字典推导
- Python 生成器表达式
- locals & globals

2.3 Python Anaconda

- 什么是 anaconda?
- 什么是 miniconda?
- 什么是 conda?

miniconda2 安装 jupyter 遇到错误可以先执行下面的指令

```
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple ipykernel==4.10.0
```

```
https://docs.conda.io/projects/conda/en/4.6.0/_downloads/
↪52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf
https://conda.io/projects/conda/en/latest/user-guide/getting-started.html
```

```
https://www.anaconda.com/
```

下载默认安装，启用 Anaconda Navigator。

2.4 Python 匿名函数

匿名函数并不是说函数没有名字，而是 def 自定义函数如果简单到只用一句表达式就可以描述的时候，就可以使用 lambda 关键字来替代。

lambda 函数基本语法规则:

```
lambda arguments: expression
```

lambda 函数也支持缺省参数、可变参数以及关键字参数，和 def 自定义函数参数规则完全相同。

```
>>> add = lambda x, y=0: x + y
>>> add(100, 200)
300
>>> add(100)
100
```

Python 函数式编程很容易涉及到匿名函数，高阶函数一般接收函数对象作为参数，而 lambda 表达式在合适不过。

```
>>> help(map)
Help on built-in function map in module __builtin__:

map(...)
    map(function, sequence[, sequence, ...]) -> list

    Return a list of the results of applying the function to the items of
    the argument sequence(s).  If more than one sequence is given, the
    function is called with an argument list consisting of the corresponding
    item of each sequence, substituting None for missing values when not all
    sequences have the same length.  If the function is None, return a list of
    the items of the sequence (or a list of tuples if more than one sequence).

>>>
>>> map(lambda x: x % 2 or x, range(10))
[0, 1, 2, 1, 4, 1, 6, 1, 8, 1]
>>>
>>> help(filter)
Help on built-in function filter in module __builtin__:

filter(...)
    filter(function or None, sequence) -> list, tuple, or string

    Return those items of sequence for which function(item) is true.  If
    function is None, return the items that are true.  If sequence is a tuple
    or string, return the same type, else return a list.
>>>
>>> filter(lambda x: not x % 2, range(10))
```

(下页继续)

(续上页)

```
[0, 2, 4, 6, 8]
>>>
```

lambda 函数中如何使用 if, else 以及 elif?

在 lambda 函数中使用 if else 有点棘手，基本语法规则:

```
lambda <arguments> : <Return Value if condition is True> if <condition> else <Return␣
↪Value if condition is False>
```

比如我们创建一个 lambda 函数来检查给定值是否在-5 到 5 之间，可以这样:

```
>>> filterFunc = lambda x: True if (x > -5 and x < 5) else False
>>> filterFunc(-2)
True
>>> filterFunc(5)
False
>>> filterFunc(10)
False
>>>
```

在这里，我们在 lambda 函数中使用 if else，如果给定值在-5 到 5 之间，则它将返回 True，否则它将返回 False。

实际在 lambda 函数中，我们可以避免使用 if else 的写法，下面是更 pythonic 的写法，仍然可以获得相同的结果。

```
>>> filterFunc = lambda x: x > -5 and x < 5
>>>
>>> filterFunc(-2)
True
>>> filterFunc(5)
False
>>> filterFunc(10)
False
>>>
```

高阶函数 filter 和 lambda 函数一起使用会发生微妙的变化，filter 高阶函数接受 callback 函数，内置循环遍历每一个元素，如果 callback 函数返回 True，则将元素添加到新列表。

```
>>> help(filter)
Help on built-in function filter in module __builtin__:

```

(下页继续)

(续上页)

```
filter(...)
    filter(function or None, sequence) -> list, tuple, or string

    Return those items of sequence for which function(item) is true.  If
    function is None, return the items that are true.  If sequence is a tuple
    or string, return the same type, else return a list.
>>>
>>> filter(lambda x: not x % 2, range(10))
[0, 2, 4, 6, 8]
>>>
```

那么如果多个条件如何在 lambda 函数中使用 if, elif 和 else 呢?

我们不能在 lambda 函数中直接使用 elif，可以使用 if else 加括号的方式。基本语法规则:

```
lambda <args> : <return Value> if <condition> (<return value> if <condition> else
↪<return value>)
```

```
>>> converter = lambda x: x * 2 if x < 10 else (x * 3 if x < 20 else x)
>>> converter(5)
10
>>> converter(13)
39
>>> converter(23)
23
>>>
```

sort 排序使用

re 正则替换使用

2.5 Python API

掌握 API 是一种编程技能，API 是应用程序接口（Application Programming Interface），通俗来讲就是别人写的代码库，提供接口给我们使用。

2.6 Python 内置函数

Python 内置函数也叫内建函数（built-in function），大概有 60 来个内置函数，版本不同个数不同，功能也有所更新，具体 Python 环境具体分析（以下内置函数是基于 Python 3.8 版本）。

```
import __builtin__
print(dir(__builtin__))
```

abs()	# 求绝对值
all()	# 容器中所有元素都为真返回 True，否则返回 False
any()	# 容器中所有元素都为假返回 False，否则返回 True
ascii()	# 是 ascii 码中元素返回该值，不是返回 u””
bin()	# 二进制强制类型转换
bool()	# 布尔类型强制类型转换，用来测试一个对象的真假状态
breakpoint()	#
bytearray()	# 返回一个新字节数组
bytes()	# 将字符串转化成 bytes 类型
callable()	# 判断对象是否可调用
chr()	# ASCII 编码转字符
classmethod()	# 类相关
compile()	# 将字符串类型代码编码
complex()	# 返回一个复数

```
>>> abs(-100)
100
>>>
```

```
for i, item in enumerate(range(10, 20, 2)):
    print(i, "-->", item)
```

delattr()	# 类相关
dict()	# 字典容器强制转换
dir()	# 返回对象的属性和方法
divmod()	# 返回商和余数
enumerate()	# 索引遍历
eval()	# 将一个字符串当成一个表达式来执行，返回表达式执行后的结果。
exec()	# 将一个字符串当成程序来执行。
filter()	# 高阶函数，将容器中元素通过函数过滤成新的容器
float()	# 浮点型强制类型转换
format()	# 字符串格式化函数，通常都使用字符串方法或者% 的格式化
frozenset()	# 转换为不可变集合
getattr()	# 类相关
globals()	# 返回全局作用域中的名称空间
hasattr()	# 类相关

```
filter(bool, [1, 0, 2, "", [], 3])
```

```
>>> format(1/2.43, "0.4f")
'0.4115'
>>> "{:0.4f}".format(1/2.43)
'0.4115'
>>>
```

hash()	# 获取对象的哈希值
help()	# 获取对象具体属性或方法的帮助文档
hex()	# 十六进制强制类型转换
id()	# 返回对象的内存地址
input()	# 标准输入
int()	# 整型强制类型转换
isinstance()	# 判断对象是否是某一类型
issubclass()	# 类相关
iter()	# 迭代器函数
len()	# 返回容器元素个数
list()	# 将一个可迭代对象转换成列表
locals()	# 返回当前作用域中的名称空间
map()	# 高阶函数，将容器中元素通过函数映射成新的容器
max()	# 返回容器中最大元素

```
>>> iter(range(3))
<listiterator object at 0x031A03F0>
>>> iterator = iter(range(3))
>>> next(iterator)
0
>>> next(iterator)
1
>>> dir(iterator)
['__class__', '__delattr__', '__doc__', '__format__', '__getattribute__', '__hash__', '__
↪init__', '__iter__', '__length_hint__', '__new__', '__reduce__', '__reduce_ex__', '__
↪repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'next']
>>> iterator.next()
2
```

```
>>> map(bool, [None, 0, "", u"", list(), tuple(), dict(), set(), frozenset()])
[False, False, False, False, False, False, False, False, False]
```

(下页继续)

(续上页)

```
>>>
```

memoryview()	#
min()	# 返回容器中最小元素
next()	# 迭代器向下执行一次
object()	# 类相关
oct()	# 八进制强制类型转换
open()	# 上下文管理器
ord()	# ASCII 字符转编码
pow()	# 求次方
print()	# 打印任何对象，用于调试代码
property()	# 类相关
range()	# 返回整数列表
repr()	# 将对象字符串化
reversed()	# 反转，和列表方法 reverse() 不同之处是生成新的列表
round()	# 四舍五入求整

```
import hou

print(kwargs)
print(type(kwargs))
print(repr(kwargs))
print(type(repr(kwargs)))
```

```
>>> round(3.14)
3.0
>>> round(3.6)
4.0
>>> round(3.5)
4.0
>>> round(3.4999)
3.0
>>>
```

set()	# 元组容器强制转换
setattr()	# 类相关
slice()	# 列表的切片
sorted()	# 排序，和列表方法 sort() 不同之处是生成新的列表
staticmethod()	# 类相关
str()	# 字符串强制类型转换
sum()	# 求和
super()	# 类相关
tuple()	# 将一个可迭代对象转换成元组
type()	# 返回对象的类型
vars()	#
zip()	# 将两个相同元素个数的列表打包成一个键值对的元组列表
___import___()	# 用于动态加载类和函数

```
>>> print(123)
123
>>> type(1)
<type 'int'>
>>> type("1")
<type 'str'>
>>> type(3 / 2.0)
<type 'float'>
>>> type(3 / 2)
<type 'int'>
>>> isinstance("1", int)
False
>>> int("123")
123
>>> bool(8)
True
>>>
>>> str(123)
'123'
>>> int("123")
123
>>> bin(17)
'0b10001'
>>> int("0b10001", 2)
17
>>> oct(20)
```

(下页继续)

(续上页)

```
'024'
>>> int("024", 8)
20
>>> hex(22)
'0x16'
>>> int("0x16", 16)
22
>>> str(0.9)
'0.9'
>>> float("0.9")
0.9
>>> str([0, 1, 2])
'[0, 1, 2]'
>>> eval("[0, 1, 2]")
[0, 1, 2]
>>>
>>> reduce(lambda x, y: x + y, range(10))
45
>>>
>>> l1 = range(10)
>>>
>>> sl = slice(1, 3, 2)
>>> l1[sl]
range(1, 3, 2)
>>>
```

```
>>> keys = ["name", "age"]
>>> values = ["Andy", 30]
>>> zip(keys, values)
[('name', 'Andy'), ('age', 30)]
>>> dict(zip(keys, values))
{'age': 30, 'name': 'Andy'}
>>>
```

2.7 Python 内置模块：abc

抽象基类（Abstract Base Classes）

- 抽象基类不能实例化

- 子类需要实现基类指定的抽象方法

2.8 Python 内置模块：calendar

日历模块 calendar

```
>>> import calendar
>>> type(calendar)
<class 'module'>
>>> calendar.__file__
'C:\\Python38\\lib\\calendar.py'
>>> dir(calendar)
['Calendar', 'EPOCH', 'FRIDAY', 'February', 'HTMLCalendar', 'IllegalMonthError',
↪'IllegalWeekdayError', 'January', 'LocaleHTMLCalendar', 'LocaleTextCalendar', 'MONDAY',
↪ 'SATURDAY', 'SUNDAY', 'THURSDAY', 'TUESDAY', 'TextCalendar', 'WEDNESDAY', '_EPOCH_ORD
↪', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__
↪name__', '__package__', '__spec__', '_colwidth', '_locale', '_localized_day', '_
↪localized_month', '_monthlen', '_nextmonth', '_prevmonth', '_spacing', 'c', 'calendar',
↪ 'datetime', 'day_abbr', 'day_name', 'different_locale', 'error', 'firstweekday',
↪'format', 'formatstring', 'isleap', 'leapdays', 'main', 'mdays', 'month', 'month_abbr',
↪ 'month_name', 'monthcalendar', 'monthrange', 'prcal', 'prmonth', 'prweek', 'repeat',
↪'setfirstweekday', 'sys', 'timegm', 'week', 'weekday', 'weekheader']
>>>
```

```
import calendar

for i in range(1, 13):
    print(calendar.month_name[i])

import calendar

d={}
for i in range(1, 13):
    d[calendar.month_name[i]] = i

print(d)

import calendar
from pprint import pprint
s2 = "February January  May October August September April  November July March December"
```

(下页继续)

(续上页)

```
d = {}
for i in range(1, 13):
    d[calendar.month_name[i]] = i

def sorter(elem):
    return d[elem]

pprint(sorted(s2.split(), key=sorter))

import calendar
month_names = [calendar.month_name[i] for i in range(1, 13)]
print(month_names)
print(sorted(s2.split(), key=month_names.index))
```

2.9 Python 内置模块：copy

内置模块 copy 的存在主要为了深度拷贝，那么什么是深度拷贝呢?

深度拷贝会递归拷贝所有深度成员。

```
>>> import copy
>>> type(copy)
<type 'module'>
>>> dir(copy)
['Error', 'PyStringMap', '_EmptyClass', '__all__', '__builtins__', '__doc__', '__file__',
↪ '__name__', '__package__', '_copy_dispatch', '_copy_immutable', '_copy_inst', '_copy_
↪with_constructor', '_copy_with_copy_method', '_deepcopy_atomic', '_deepcopy_dict', '_
↪deepcopy_dispatch', '_deepcopy_inst', '_deepcopy_list', '_deepcopy_method', '_deepcopy_
↪tuple', '_keep_alive', '_reconstruct', '_test', 'copy', 'deepcopy', 'dispatch_table',
↪'error', 'name', 't', 'weakref']
>>> help(copy.deepcopy)
Help on function deepcopy in module copy:

deepcopy(x, memo=None, _nil=[])
    Deep copy operation on arbitrary Python objects.

    See the module's __doc__ string for more info.

>>>
```

存在深度拷贝主要原因是可变数据类型可能会触碰到这个问题。

```
>>> l1 = [1, [2, 3], 4]
>>> l2 = l1[:]
>>> l1
[1, [2, 3], 4]
>>> l2
[1, [2, 3], 4]
>>> id(l1)
52764264
>>> id(l2)
52762824
>>> id(l1[1])
52786560
>>> id(l2[1])
52786560
>>> l1.append(5)
>>> l1
[1, [2, 3], 4, 5]
>>> l2
[1, [2, 3], 4]
>>> l1[1].append(100)
>>> l1
[1, [2, 3, 100], 4, 5]
>>> l2
[1, [2, 3, 100], 4]
```

上面代码是一个浅拷贝的例子，不经意间你的代码就会出现问题，如果用 copy 模块呢?

```
>>> import copy
>>> l1
[1, [2, 3, 100], 4, 5]
>>> l3 = copy.deepcopy(l1)
>>> id(l1)
52764264
>>> id(l3)
52786520
>>> id(l1[1])
52786560
>>> id(l3[1])
52788600
>>> l1[1].pop()
```

(下页继续)

(续上页)

```
100
>>> l1
[1, [2, 3], 4, 5]
>>> l3
[1, [2, 3, 100], 4, 5]
>>>
```

2.10 Python 内置模块：csv

2.11 Python 内置模块：datetime

```
>>> import datetime
>>> type(datetime)
<class 'module'>
>>> datetime.__file__
'C:\\Python38\\lib\\datetime.py'
>>> dir(datetime)
['MAXYEAR', 'MINYEAR', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__',
↪ '__name__', '__package__', '__spec__', 'date', 'datetime', 'datetime_CAPI', 'sys',
↪'time', 'timedelta', 'timezone', 'tzinfo']
>>>
>>> datetime.datetime(2020, 8, 18, 18, 48)
datetime.datetime(2020, 8, 18, 18, 48)
```

2.12 Python 内置模块：functools

2.13 Python 内置模块：getpass

```
import getpass

user = getpass.getuser()
```

2.14 Python 内置模块：importlib

importlib 是 Python 3 加入的模块，里面有个重要的方法 reload 在 Python 2 中是内置函数。

```
>>> import importlib
>>> importlib.__file__
'C:\\Python37\\lib\\importlib\\__init__.py'
>>> dir(importlib)
['_RELOADING', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__import_
↪_', '__loader__', '__name__', '__package__', '__path__', '__spec__', '_bootstrap', '_
↪bootstrap_external', '_imp', '_r_long', '_w_long', 'abc', 'find_loader', 'import_module
↪', 'invalidate_caches', 'machinery', 'reload', 'sys', 'types', 'util', 'warnings']
>>>
>>> import os
>>> importlib.reload(os)
<module 'os' from 'C:\\Python37\\lib\\os.py'>
>>>
```

2.15 Python 内置模块：inspect

isfunction()

ismethod()

2.16 Python 内置模块：itertools

```
>>> import itertools
>>>
>>> list(itertools.dropwhile(lambda i: i < 4, range(10)))
[4, 5, 6, 7, 8, 9]
>>> list(itertools.takewhile(lambda i: i < 4, range(10)))
[0, 1, 2, 3]
>>> list(itertools.ifilter(lambda x: x % 2, range(10)))
[1, 3, 5, 7, 9]
>>> list(itertools.ifilterfalse(lambda x: x % 2, range(10)))
[0, 2, 4, 6, 8]
>>> list(itertools.imap(lambda x, y: x + y, (2, 3, 10), (5, 2, 3)))
[7, 5, 13]
>>> list(itertools.starmap(lambda x, y: x + y, [(1, 2), (10, 20)]))
[3, 30]
```

2.17 Python 内置模块：json

json (JavaScript Object Notation) 是一种轻量级的数据交换格式。

```
>>> import json
>>> json.__file__
'C:\\Python27\\lib\\json\\__init__.pyc'
>>> type(json)
<type 'module'>
>>> dir(json)
['JSONDecoder', 'JSONEncoder', '__all__', '__author__', '__builtins__', '__doc__', '__
↪file__', '__name__', '__package__', '__path__', '__version__', '_default_decoder', '_
↪default_encoder', 'decoder', 'dump', 'dumps', 'encoder', 'load', 'loads', 'scanner']
>>>
```

- json.dump 字典转文件
- json.dumps 字典转字符串
- json.load 文件转字典
- json.loads 字符串转字典

```
import json

data = {
    "name": "Andy",
    "age": 29,
    "weight": 55.5
}

jsonStr = json.dumps(data)

jsonData = json.loads(jsonStr)

# Writing JSON data
with open("data.json", "w") as f:
    json.dump(data, f)

# indent 用法
with open("data.json", "w") as f:
    json.dump(data, f, indent=4)
```

(下页继续)

(续上页)

```
with open("data.json", "r") as f:
    data = json.load(f)
```

2.18 Python 内置模块：logging

```
>>> import logging
>>> type(logging)
<class 'module'>
>>> logging.__file__
'C:\\Python38\\lib\\logging\\__init__.py'
>>> dir(logging)
['BASIC_FORMAT', 'BufferingFormatter', 'CRITICAL', 'DEBUG', 'ERROR', 'FATAL',
↪'FileHandler', 'Filter', 'Filterer', 'Formatter', 'Handler', 'INFO', 'LogRecord',
↪'Logger', 'LoggerAdapter', 'Manager', 'NOTSET', 'NullHandler', 'PercentStyle',
↪'PlaceHolder', 'RootLogger', 'StrFormatStyle', 'StreamHandler', 'StringTemplateStyle',
↪'Template', 'WARN', 'WARNING', '_STYLES', '_StderrHandler', '__all__', '__author__', '_
↪_builtins__', '__cached__', '__date__', '__doc__', '__file__', '__loader__', '__name__
↪', '__package__', '__path__', '__spec__', '__status__', '__version__', '_acquireLock',
↪'_addHandlerRef', '_checkLevel', '_defaultFormatter', '_defaultLastResort', '_
↪handlerList', '_handlers', '_levelToName', '_lock', '_logRecordFactory', '_loggerClass
↪', '_nameToLevel', '_register_at_fork_reinit_lock', '_releaseLock', '_removeHandlerRef
↪', '_showwarning', '_srcfile', '_startTime', '_str_formatter', '_warnings_showwarning',
↪ 'addLevelName', 'atexit', 'basicConfig', 'captureWarnings', 'collections', 'critical',
↪ 'currentframe', 'debug', 'disable', 'error', 'exception', 'fatal', 'getLevelName',
↪'getLogRecordFactory', 'getLogger', 'getLoggerClass', 'info', 'io', 'lastResort', 'log
↪', 'logMultiprocessing', 'logProcesses', 'logThreads', 'makeLogRecord', 'os',
↪'raiseExceptions', 're', 'root', 'setLogRecordFactory', 'setLoggerClass', 'shutdown',
↪'sys', 'threading', 'time', 'traceback', 'warn', 'warning', 'warnings', 'weakref']
>>>
```

```
import logging

def main():
    logging.basicConfig(
        filename="app.log",
        level=logging.ERROR
    )
    hostname = "www.python.org"
```

(下页继续)

(续上页)

```
    item = "spam"
    filename = "data.csv"
    mode = "r"

    logging.critical("Host %s unknown", hostname)
    logging.error("Couldn't find %r", item)
    logging.warning("Feature is deprecated")
    logging.info("Opening file %r, mode=%r", filename, mode)
    logging.debug("Got here")

if __name__ == "__main__":
    main()
```

level 是一个过滤器，critical error warning info debug 代表不同的严重级别

```
logging.basicConfig(
    filename="app.log",
    level=logging.WARNING,
    format="%(levelname)s:%(asctime)s:%(message)s"
)
```

2.19 Python 内置模块：math

```
>>> import math
>>> type(math)
<class 'module'>
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin',
↪ 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees
↪', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod',
↪'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan
↪', 'isqrt', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'perm',
↪'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau
↪', 'trunc']
>>>
```

math.ceil()	# 向上取整
math.floor()	# 向下取整

2.20 Python 内置模块：operator

2.21 Python 内置模块：os

os.listdir os.walk os.path

模块大概分类常用的内置模块 import os import sys import glob import json import yaml import shutil import re 正则表达式 import time import datetime import platform import random import subprocess import xml.etree.ElementTree as ET import smtplib from email.header import Header from email.mime.text import MIMEText from email.utils import parseaddr, formataddr

内置模块 DCC 软件自带模块第三方模块自定义模块

Python 模块搜索路径当前工作路径 + sys.path

os os.path os.listdir os.walk os.environ

os.path module

import os

path = "C:/Users/huweiguo/Documents/maya/2018/maya.env" os.path.basename(path) os.path.dirname(path) os.path.join("tmp" , "data" , os.path.basename(path)) path = "~/maya/2018/maya.env" os.path.expanduser(path) os.path.splitext(path)

尽可能使用 os.path 来操作文件路径的问题，最好不要使用字符串来构造自己的代码，主要是为了代码的可移植性，它可以很好的处理 linux、mac 以及 windows 文件路径的差异

```
#!/usr/bin/env python3.7
import os

def findFile(start, name):
    for relpath, dirs, files in os.walk(start):
        if name in files:
            fullPath = os.path.join(start, relpath, name)
            print(os.path.normpath(os.path.abspath(fullPath)))

if __name__ == "__main__":
    findFile(sys.argv[1], sys.argv[2])
```

从文件夹中查找文件

```
with open("somefile", "wt") as f:
    f.write("Hello\n")
```

(下页继续)

(续上页)

```
with open("somefile", "xt") as f:
    f.write("Hello\n")

import os
if not os.path.exists("somefile"):
    with open("somefile", "wt") as f:
        f.write("Hello\n")
else:
    print("File already exist!")
```

```
import os
houVersion = "17.0"
version = 2

###### Do not do this !!!!! ######
# filepath = "C:\Users\huweiguo\Documents\houdini" + houVersion + "\tmp" + "\example_v0"␣
↪+ str(version) + ".py"
# filepath = os.path.join("C:\Users\huweiguo\Documents\houdini", houVersion, "tmp",
↪"example", str(version), ".py")
####################################
# print(filepath)
#
# filepath = os.path.expanduser("~/Documents/houdini{0}/tmp/example_v{1:02}.py".
↪format(houVersion, version))
# filepath = os.path.normpath(filepath)
# print(filepath)
# print("This is a file ? :", os.path.isfile(filepath))
# print("This is a directory ? :", os.path.isdir(r"c:\temp"))
#
#
# tempFolder = 'temp_2'
# cacheType = 'bgeo'
# cacheName = 'waterSplash.bgeo.gz'
# filepath = os.path.join(r"C:\nrojects/bla", tempFolder, cacheType, cacheName)
# print(filepath)
# print(os.path.normpath(filepath))

# print(os.path.split(filepath))
```

(下页继续)

(续上页)

```
# print(os.path.dirname(filepath))

# print(os.path.basename(filepath))

#### PATH SEPARATOR #####
HOUDINI_OTLSCAN_PATH = os.pathsep.join([os.path.expanduser('~/houdini12.1/otls'),
                                        '/houdini_install/houdini/otls',
                                        '/mnt/repo/houdini/otls',
                                        '/mnt/projects/xyzproject/otls'])
print HOUDINI_OTLSCAN_PATH

# t = "D:/Program"
# l = []
# print(os.listdir(t))

# for f in os.listdir(t):
#     l.append(os.path.normpath(os.path.join(t, f)))

# print l
```

```
# Python2 中默认编码是 ASCII, Python3 中默认编码是 Unicode
s1.decode("gb2312") # 将 gb2312 编码的字符串转换成 unicode
s2.encode("gb2312") # 将 unicode 编码的字符串转换成 gb2312

from urllib.request import urlopen
# 正常网页是 utf-8, 所以要转 unicode
import json
u = urlopen(网址)
resp = json.loads(u.read().decode("utf-8"))
from pprint import pprint
pprint(resp)
```

```
import os
import sys
import shutil
import random
import datetime
import module as xxx
from module import xxx as xx
```

(下页继续)

(续上页)

```
os.listdir()
os.getcwd()
os.mkdir()
os.makedirs()
r"\\\"自然字符串
os.remove()
os.rmdir()
os.path 模块功能操作文件夹
os.path.isdir()
os.path.isfile()
os.path.split()
os.path.splitext()
os.path.splitdrive()
os.path.join()
os.path.normpath()
```

2.22 Python 内置模块：pprint

```
>>> import pprint
>>> type(pprint)
<class 'module'>
>>> dir(pprint)
['PrettyPrinter', '_StringIO', '__all__', '__builtins__', '__cached__', '__doc__', '__
↪file__', '__loader__', '__name__', '__package__', '__spec__', '_builtin_scalars', '_
↪collections', '_perfcheck', '_recursion', '_safe_key', '_safe_repr', '_safe_tuple', '_
↪sys', '_types', '_wrap_bytes_repr', 'isreadable', 'isrecursive', 'pformat', 'pp',
↪'pprint', 're', 'saferepr']
>>> pprint.__file__
'C:\\Python38\\lib\\pprint.py'
>>> import sys
>>> pprint.pprint(sys.path)
['',
'C:\\Python38\\python38.zip',
'C:\\Python38\\DLLs',
'C:\\Python38\\lib',
'C:\\Python38',
'C:\\Python38\\lib\\site-packages']
>>>
```

2.23 Python 内置模块：py_compile

2.24 Python 内置模块：random

```
>>> import random
>>> type(random)
<class 'module'>
>>> random.__file__
'C:\\Python38\\lib\\random.py'
>>> dir(random)
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST', 'SystemRandom',
↪'TWOPI', '_Sequence', '_Set', '__all__', '__builtins__', '__cached__', '__doc__', '__
↪file__', '__loader__', '__name__', '__package__', '__spec__', '_accumulate', '_acos',
↪'_bisect', '_ceil', '_cos', '_e', '_exp', '_inst', '_log', '_os', '_pi', '_random', '_
↪repeat', '_sha512', '_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_warn',
↪'betavariate', 'choice', 'choices', 'expovariate', 'gammavariate', 'gauss',
↪'getrandbits', 'getstate', 'lognormvariate', 'normalvariate', 'paretovariate', 'randint
↪', 'random', 'randrange', 'sample', 'seed', 'setstate', 'shuffle', 'triangular',
↪'uniform', 'vonmisesvariate', 'weibullvariate']
>>>
```

random.choice()	# 从序列中随机挑选一个元素
random.randint()	# 给定范围随机整数
random.random()	# 随机 0~1 之间小数

2.25 Python 内置模块：re

正则表达式模块 re

```
import re
from pprint import pprint

files = ["tank_1_color_v0.rat",
         "tank_2_color_v5.rat",
         "tank_1_color_v3.rat",
         "tank_3_color_v1.rat",
         "tank_4_color_v2.rat",
         "tank_4_color_v4.rat",
```

(下页继续)

(续上页)

```
        "tank_5_color_v1.rat",
        "tank_6_color_v6.rat"]

pat_num = re.compile("\D+_(\d+)_")
pat_ver = re.compile("(\d+)\D+$")

def sorter_num(elem):
    res = re.search(pat_num, elem)
    return res.groups()[0]

def sorter_ver(elem):
    res = re.search(pat_ver, elem)
    return res.groups()[0]

# pprint(sorted(files, key=sorter_num))
pprint(sorted(files, key=sorter_ver))
```

2.26 Python 内置模块：shutil

```
shutil.copy()
shutil.copy2()
shutil.rmtree()
```

2.27 Python 内置模块：subprocess

```
>>> import subprocess
>>> type(subprocess)
<type 'module'>
>>> dir(subprocess)
['CREATE_NEW_CONSOLE', 'CREATE_NEW_PROCESS_GROUP', 'CalledProcessError', 'MAXFD', 'PIPE',
↪ 'Popen', 'STARTF_USESHOWWINDOW', 'STARTF_USESTDHANDLES', 'STARTUPINFO', 'STDOUT',
↪'STD_ERROR_HANDLE', 'STD_INPUT_HANDLE', 'STD_OUTPUT_HANDLE', 'SW_HIDE', '__all__', '__
↪builtins__', '__doc__', '__file__', '__name__', '__package__', '_active', '_args_from_
↪interpreter_flags', '_cleanup', '_demo_posix', '_demo_windows', '_eintr_retry_call', '_
↪subprocess', 'call', 'check_call', 'check_output', 'errno', 'gc', 'list2cmdline',
↪'msvcrt', 'mswindows', 'os', 'pywintypes', 'signal', 'sys', 'threading', 'traceback',
↪'types']
```

(下页继续)

(续上页)

```
>>>
```

```
import subprocess

exePath = "C:/Python27/python.exe"

pyFile = "D:/andy/tests_subprocess.py"

commands = '"{0}" "{1}"'.format(exePath, pyFile)

subprocess.check_call(commands)
```

```
import subprocess

exePath = "C:/Program Files/Nuke10.5v1/Nuke10.5.exe"

commands = '"{0}" --nukex'.format(exePath)

subprocess.check_call(commands)
```

subprocess.check_call-> 会卡主程序

subprocess.check_output-> 会卡主程序，会返回一些结果

subprocess.Popen-> 独立程序，不会卡主程序

2.28 Python 内置模块：sys

内置模块 sys 是用来描述与 Python 解释器密切相关的一些功能，而非操作系统的功能，操作系统相关功能都在内置模块 os 中，这也是两个模块主要区别。

```
>>> import sys
>>>
>>> type(sys)
<type 'module'>
>>> dir(sys)
['__displayhook__', '__doc__', '__excepthook__', '__name__', '__package__', '__stderr__',
↪ '__stdin__', '__stdout__', '_clear_type_cache', '_current_frames', '_getframe', '_git
↪', 'api_version', 'argv', 'builtin_module_names', 'byteorder', 'call_tracing',
↪'callstats', 'copyright', 'displayhook', 'dllhandle', 'dont_write_bytecode', 'exc_clear
↪', 'exc_info', 'exc_type', 'excepthook', 'exec_prefix', 'executable', 'exit', 'exitfunc
↪', 'flags', 'float_info', 'float_repr_style', 'getcheckinterval', 'getdefaultencoding',
↪ 'getfilesystemencoding', 'getprofile', 'getrecursionlimit', 'getrefcount', 'getsizeof
↪', 'gettrace', 'getwindowsversion', 'hexversion', 'last_traceback', 'last_type', 'last_
↪value', 'long_info', 'maxint', 'maxsize', 'maxunicode', 'meta_path', 'modules', 'path',
↪ 'path_hooks', 'path_importer_cache', 'platform', 'prefix', 'ps1', 'ps2', 'py3kwarning
```

(下页继续)

(续上页)

```
>>>
>>> sys.api_version
1013
>>> sys.executable
'C:\\Python27\\python.exe'
>>> sys.platform
'win32'
>>> sys.version
'2.7.14 (v2.7.14:84471935ed, Sep 16 2017, 20:19:30) [MSC v.1500 32 bit (Intel)]'
>>> sys.version_info
sys.version_info(major=2, minor=7, micro=14, releaselevel='final', serial=0)
```

- sys.argv
- sys.path
- sys.modules
- reload

2.29 Python 内置模块：math

```
>>> import time
>>> type(time)
<class 'module'>
>>> dir(time)
['_STRUCT_TM_ITEMS', '__doc__', '__loader__', '__name__', '__package__', '__spec__',
↪'altzone', 'asctime', 'ctime', 'daylight', 'get_clock_info', 'gmtime', 'localtime',
↪'mktime', 'monotonic', 'monotonic_ns', 'perf_counter', 'perf_counter_ns', 'process_time
↪', 'process_time_ns', 'sleep', 'strftime', 'strptime', 'struct_time', 'thread_time',
↪'thread_time_ns', 'time', 'time_ns', 'timezone', 'tzname']
>>>
```

2.30 Python 闭包函数

闭包函数指的是在函数内定义的内部函数。

2.31 Python 推导

Python 四种内建容器刚好对应有四种推导方式。

- 列表推导

```
myList = []

for i in range(10):
    myList.append(i * 2)
```

```
myList = [i * 2 for i in range(10)]
```

- 生成器表达式
- 集合推导
- 字典推导

2.32 Python 容器：字典 {key: value}

容器是对专门用来装其它对象的数据类型的统称。在 Python 中，有四种最常见的内建容器类型：列表（list）、元组（tuple）、字典（dict）和集合（set）。Python 语言内部实现细节也与这些容器息息相关。比如 Python 的类实例属性、全局变量 globals() 等都是通过字典类型来存储的。

- 字典 dict {key: value}
- 无序的（散列表）
- 可迭代的

字典的迭代本质上并不是字典本身迭代，而是列表的迭代。

```
for k, v in dict.items():
for k in dict.keys():
for v in dict.values():
for k, v in dict.iteritems():
for k in dict.keys():
for v in dict.values():
```

- 字典的方法

```
>>> type({})
<type 'dict'>
>>> dir({})
```

（下页继续）

(续上页)

```
['__class__', '__cmp__', '__contains__', '__delattr__', '__delitem__', '__doc__', '__eq__
↪', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__
↪init__', '__iter__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__reduce__',
↪'__reduce_ex__', '__repr__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__
↪subclasshook__', 'clear', 'copy', 'fromkeys', 'get', 'has_key', 'items', 'iteritems',
↪'iterkeys', 'itervalues', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values',
↪'viewitems', 'viewkeys', 'viewvalues']
>>>
```

2.33 Python 容器：列表 []

容器是对专门用来装其它对象的数据类型的统称。在 Python 中，有四种最常见的内建容器类型：列表 (list)、元组（tuple)、字典（dict）和集合（set)。Python 语言内部实现细节也与这些容器息息相关。比如 Python 的类实例属性、全局变量 globals() 等都是通过字典类型来存储的。

- 列表 list []
- 有序的
- 索引取值
- 索引改值
- 切片
- 可变的
- 可迭代的
- 列表的方法

```
>>> type([])
<type 'list'>
>>> dir([])
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__delslice__', '_
↪_doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__
↪getslice__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '__
↪le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex_
↪_', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__setslice__
↪', '__sizeof__', '__str__', '__subclasshook__', 'append', 'count', 'extend', 'index',
↪'insert', 'pop', 'remove', 'reverse', 'sort']
>>>
```

- 列表解包

```
>>> a, *b = range(4)
>>> a
0
>>> b
[1, 2, 3]
>>>
>>> a, *b, c = range(4)
>>> a
0
>>> b
[1, 2]
>>> c
3
>>>
```

```
>>> l = []
>>> l.append(1)
>>> print(l)
[1]
>>> l.extend([1, 2, 3])
>>> print(l)
[1, 1, 2, 3]
>>> l.count(1)
2
>>> l.index(3)
3
>>> l.pop()
3
>>> print(l)
[1, 1, 2]
>>>
>>> l2 = [1, 2, 3, 4, 5]
>>> print(l2)
[1, 2, 3, 4, 5]
>>> l2.reverse()
>>> print(l2)
[5, 4, 3, 2, 1]
>>> l2.sort()
>>> print(l2)
[1, 2, 3, 4, 5]
```

(下页继续)

(续上页)

```
>>> l3 = [35, 50, 17, 30]
>>> print(l3)
[35, 50, 17, 30]
>>> sorted(l3)
[17, 30, 35, 50]
>>> print(l3)
[35, 50, 17, 30]
>>> print(sorted(l3))
[17, 30, 35, 50]
>>>
```

```
>>> s1 = set([1, 2, 3, 4, 5])
>>> s2 = set([3, 5, 7, 8, 9])
>>> print(s1.union(s2))
set([1, 2, 3, 4, 5, 7, 8, 9])
>>> s1 = set(["Andy", "Tommy", "Ben", "Gloria", "Dinna"])
>>> s2 = set(["John", "Tommy", "Ben", "Gloria", "Dinna"])
>>> print(s1.difference(s2))
set(['Andy'])
>>> d = {"name": "Andy", "age": 28, "country": "China"}
>>> print(d)
{'country': 'China', 'age': 28, 'name': 'Andy'}
>>> print(d["name"])
Andy
>>> d["name"] = "Gloria"
>>> print(d)
{'country': 'China', 'age': 28, 'name': 'Gloria'}
>>> d2 = d
>>> print(d)
{'country': 'China', 'age': 28, 'name': 'Gloria'}
>>> print(d2)
{'country': 'China', 'age': 28, 'name': 'Gloria'}
>>> d2["name"] = "Andy"
>>> print(d)
{'country': 'China', 'age': 28, 'name': 'Andy'}
>>> print(d2)
{'country': 'China', 'age': 28, 'name': 'Andy'}
>>>
>>> d2 = d.copy()
```

(下页继续)

(续上页)

```
>>> print(d)
{'country': 'China', 'age': 28, 'name': 'Andy'}
>>> print(d2)
{'country': 'China', 'age': 28, 'name': 'Andy'}
>>> d2["name"] = "John"
>>> print(d)
{'country': 'China', 'age': 28, 'name': 'Andy'}
>>> print(d2)
{'country': 'China', 'age': 28, 'name': 'John'}
>>> print(id(d))
58866256
>>> print(id(d2))
58867408
>>> l2 = [9, 5, 45, 2, 1]
>>> l3 = l2
>>> print(l2, l3)
([9, 5, 45, 2, 1], [9, 5, 45, 2, 1])
>>> l2.append(100)
>>> print(l2, l3)
([9, 5, 45, 2, 1, 100], [9, 5, 45, 2, 1, 100])
>>> l4 = l2[:]
>>> print(l2, l4)
([9, 5, 45, 2, 1, 100], [9, 5, 45, 2, 1, 100])
>>> l2.append(200)
>>> print(l2, l4)
([9, 5, 45, 2, 1, 100, 200], [9, 5, 45, 2, 1, 100])
>>> print(d2)
{'country': 'China', 'age': 28, 'name': 'John'}
>>> print(d2["age"])
28
>>> print(d2["ages"])
>>> print(d2.get("key", None))
None
>>> print(d2.has_key("age"))
True
>>> print(d2.items())
[('country', 'China'), ('age', 28), ('name', 'John')]
>>> print(d2.keys())
['country', 'age', 'name']
>>> print(d2.values())
```

(下页继续)

(续上页)

```
['China', 28, 'John']
>>> print(d2)
{'country': 'China', 'age': 28, 'name': 'John'}
>>> d2.pop("age")
28
>>> print(d2)
{'country': 'China', 'name': 'John'}
>>> d2.update(dict(age=29))
>>> print(d2)
{'country': 'China', 'age': 29, 'name': 'John'}
>>> range(5, 0, -1)
[5, 4, 3, 2, 1]
>>> g = xrange(1000000)
>>> print(g)
xrange(1000000)
>>> print(g[1000])
1000
>>>
```

- 列表推导

```
>>> intList = [1, 2, 3]
>>> print(intList)
[1, 2, 3]
>>> intList == [1, 2, "3"]
False
>>> print(intList)
[1, 2, 3]
>>> print(type(intList))
<type 'list'>
>>> nameList = ["andy", "dovfx"]
>>> print(nameList)
['andy', 'dovfx']
>>> complexList = [99, "hello", [1, 2], nameList]
>>> print(complexList)
[99, 'hello', [1, 2], ['andy', 'dovfx']]
>>> type(complexList)
<type 'list'>
>>> print(type(complexList))
<type 'list'>
```

(下页继续)

(续上页)

```
>>> print(range(1, 100))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,␣
↪25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46,
↪ 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,␣
↪68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
↪ 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]
>>> print(range(100))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23,␣
↪24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45,
↪ 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66,␣
↪67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,
↪ 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99]
>>> print(range(1, 100, 5))
[1, 6, 11, 16, 21, 26, 31, 36, 41, 46, 51, 56, 61, 66, 71, 76, 81, 86, 91, 96]
>>>
```

```
>>> numbers = range(10)
>>> print(numbers)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> print(numbers[1])
1
>>> print(numbers[0])
0
>>> print(numbers[-1])
9
>>> myList = [numbers, ["a", "b", "c"]]
>>> print(myList[1][1])
b
>>> myList[-1][1] = "bb"
>>> print(myList)
[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9], ['a', 'bb', 'c']]
>>> print(numbers[1:7])
[1, 2, 3, 4, 5, 6]
>>> print(numbers[3:])
[3, 4, 5, 6, 7, 8, 9]
>>> print(numbers[:3])
[0, 1, 2]
>>> print(numbers[-4:-1])
[6, 7, 8]
```

(下页继续)

(续上页)

```
>>> print(numbers[3:-2])
[3, 4, 5, 6, 7]
>>> print(numbers[1:7:2])
[1, 3, 5]
>>> print(dir(numbers))
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__delslice__', '_
↪_doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__
↪getslice__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '__
↪le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex_
↪_', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__setslice__
↪', '__sizeof__', '__str__', '__subclasshook__', 'append', 'count', 'extend', 'index',
↪'insert', 'pop', 'remove', 'reverse', 'sort']
>>>
>>> numbers.append(100)
>>> print(numbers)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 100]
>>> print(numbers.count(100))
1
>>>
>>> a = ["a", "b", "c"]
>>> numbers.extend(a)
>>> print(numbers)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 100, 'a', 'b', 'c']
>>> numbers = range(10)
>>> numbers.append(a)
>>> print(numbers)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ['a', 'b', 'c']]
>>>
>>> print(a.index("b"))
1
>>> a.append("d")
>>> a.insert(0, "d")
>>> a.pop(0)
'd'
>>> a.remove("a")
>>>
>>> numbers.reverse()
>>> print(numbers)
[['b', 'c', 'd'], 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
>>>
```

(下页继续)

(续上页)

```
>>> numbers.sort()
>>> print(numbers)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ['b', 'c', 'd']]
>>>
```

```
#!/usr/bin/env python
import os

d = dict(spoon=4,
        fork=4,
        plate=7,
        cup=6,
        knife=2,
        teapot=1)

print d.keys()
print d.values()
print d.items()

# for k,v in d.items():
#     print k,v
#
new_env = {'HOUDINI_PATH': '/mnt/share/repos/hou', 'EDITOR': 'vim', 'TEMP': 'C:/TMP'}
# for k,v in os.environ.items():
#     print k,v


# os.environ.update(new_env)
# for k, v in sorted(os.environ.iteritems()):
#     print k, v
```

```
# -*- coding: UTF-8 -*-

#NOTE: For more advanced text formating see textwrap module.
s0 = '           Python 2.6; Python 2.7; Python 3.0; Python 3.3'
s1 = "Java is a programming language that lets you work more quickly\n" \
    "and integrate your systems more effectively. You can learn to use Java\n" \
    "and see almost immediate gains in productivity and lower maintenance costs"
```

(下页继续)

(续上页)

```
s2 = "January February April March May June July August September October November␣
↪December"

s3 = "Popular Names : Girls:{Lauren Isabella Ava Lily Zoe Chloe Mia Layla Emily Lucy} " \
    "Boys:{Aiden Jackson Ethan Liam Mason Noah Lucas Jacob Jayden Jack Alexander Ryan}"
s4 = 'Escape this worlds: \never \try \this'


print "Lower case :", s0.lower()
print s0.count('Python')
print s0.split(';')
print s0.lstrip()
print "2.7" in s0
print s4 #raw string


# print s1.replace("Java", "Python")
# print " ??? ".join(s2.split())

# grlnames = s3[s3.find('Girls:') + len('Girls:') + 1 : s3.find('}')]
# # print grlnames
# # print [name for name in grlnames.split(" ") if name.startswith('L')]
# for name in grlnames.split():
#      if name.startswith('L'):
#          print name
```

2.34 Python 容器：集合 {}

容器是对专门用来装其它对象的数据类型的统称。在 Python 中，有四种最常见的内建容器类型：列表 (list)、元组（tuple)、字典（dict）和集合（set)。Python 语言内部实现细节也与这些容器息息相关。比如 Python 的类实例属性、全局变量 globals() 等都是通过字典类型来存储的。

- 集合 set {}

集合与字典都是使用花括号 {} 来定义，区别是字典是通过键值对的方式存储。

- 元素唯一性
- 集合方法

```
>>> type(set())
<type 'set'>
>>> dir(set())
['__and__', '__class__', '__cmp__', '__contains__', '__delattr__', '__doc__', '__eq__',
↪'__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__iand__', '__init__
↪', '__ior__', '__isub__', '__iter__', '__ixor__', '__le__', '__len__', '__lt__', '__ne_
↪_', '__new__', '__or__', '__rand__', '__reduce__', '__reduce_ex__', '__repr__', '__ror_
↪_', '__rsub__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__', '__
↪subclasshook__', '__xor__', 'add', 'clear', 'copy', 'difference', 'difference_update',
↪'discard', 'intersection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset
↪', 'pop', 'remove', 'symmetric_difference', 'symmetric_difference_update', 'union',
↪'update']
>>>
```

2.35 Python 容器：元组 ()

容器是对专门用来装其它对象的数据类型的统称。在 Python 中，有四种最常见的内建容器类型：列表 (list)、元组（tuple）、字典（dict）和集合（set）。Python 语言内部实现细节也与这些容器息息相关。比如 Python 的类实例属性、全局变量 globals() 等都是通过字典类型来存储的。

- 元组 tuple ()
- 不可变的

元组的不可变是相对的。

- 有序的
- 索引 & 切片
- 可迭代的
- 元组的方法

```
>>> type(())
<type 'tuple'>
>>> dir(())
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__
↪', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__
↪gt__', '__hash__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '_
↪_ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__
↪', '__sizeof__', '__str__', '__subclasshook__', 'count', 'index']
>>>
```

2.36 Python 自定义类

私有属性获取与修改可以通过方法，这样可以在修改属性之前做些 check。

```
class Student(object):
    def __init__(self, name, score):
        self.__name = name
        self.__score = score

    def getName(self):
        return self.__name

    def getScore(self):
        return self.__score

    def setName(self, name):
        self.__name = name

    def setScore(self, score):
        if score >= 0 and score <= 100:
            self.__score = score
        else:
            print("value wrong")
```

```
class Student(object):
    def __init__(self, name, score):
        self.__name = name
        self.__score = score

    @property
    def score(self):
        return self.__score

    @score.setter
    def score(self, value):
        if value >= 0 and value <= 100:
            self.__score = value
        else:
            print("value wrong")
```

(下页继续)

(续上页)

```
s1 = Student("Andy", 90)

print(s1.score)

s1.score = 80

print(s1.score)
```

自定义类也称自定义数据类型，下面是自定义类的一些标识。

- class
- 类命名首字母大写
- self
- ___init___

自定义类的语法

```
class CustomClass(object):
    pass
```

理解类的一些概念

- 类封装
- 类继承
- 类多态
- 实例属性和实例方法
- 类属性和类方法
- 静态方法

```
class MyClass(object):

    def __init__(self, var1, var2):
        self.a = var1
        self.b = var2

    def sum(self):
        return self.a + self.b

a = MyClass(200, 300)
```

(下页继续)

(续上页)

```
print(a)
print(a.sum())
```

理解特殊方法 ___repr___，特殊方法也称魔法方法或者双下方法。

```
class Car(object):
    def __init__(self, name, eng, year):
        self.name = name
        self.eng = eng
        self.year = year

    def __repr__(self):
        return "My car name is %s" % self.name

car = Car("Jili", 120, 2019)
print(car)
```

类继承

```
class Car(object):
    def __init__(self, name, eng, year):
        self.name = name
        self.eng = eng
        self.year = year

    def __repr__(self):
        return "My car name is %s" % self.name

    def orderParts(self, *args):
        print("Connecting to server...")
        print("Ordering parts %s: for car %s" % (args, self.name))
        print("Checking status")

class Truck(Car):
    def __init__(self, name, eng, year):
        Car.__init__(self, name, eng, year)

truck = Truck("Benz", 800, 2008)
truck.orderParts("Wheels", "Silencer")
```

类继承分单继承和多继承，注意 ___init___ 的用法。

- 如果子类没有定义初始化函数，父类的初始化函数默认被调用。
- 如果子类定义了自己的初始化函数，但没有显示调用父类的初始化函数，则父类属性不会被初始化。
- 如果子类定义了自己的初始化函数，在子类中显示调用父类，子类和父类的属性都会被初始化。

初始化方案

```
# python 2.x
def __init__(self, args):
    super(ClassName, self).__init__(args)

# python 3.x
def __init__(self, args):
    super().__init__(args)

def __init__(self, args):
    ClassName.__init__(args)

# PyQt 中
# python 2.x
def __init__(self, parent=None):
    super(ClassName, self).__init__(parent)

# python 3.x
def __init__(self, parent=None):
    super().__init__(parent)
```

自定义向量类型

```
class Vector(object):
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

    def __repr__(self):
        return "Vector(%f, %f, %f)" % (self.x, self.y, self.z)

    def __add__(self, other):
        return Vector(self.x + other.x, self.y + other.y, self.z + other.z)

v1 = Vector(2, 1.5, 3.2)
```

(下页继续)

(续上页)

```
v2 = Vector(3, 4, 5)
print(v1)
print(v1 + v2)
```

```
import math
from __future__ import division

class Vector(object):
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

    def __repr__(self):
        return "Vector(%f, %f, %f)" % (self.x, self.y, self.z)

    def __add__(self, other):
        return Vector(self.x + other.x, self.y + other.y, self.z + other.z)

    def __sub__(self, other):
        return Vector(self.x - other.x, self.y - other.y, self.z - other.z)

    def __mul__(self, other):
        return Vector(self.x * other.x, self.y * other.y, self.z * other.z)

    def __div__(self, other):
        return Vector(self.x / other.x, self.y / other.y, self.z / other.z)

    def __getitem__(self, item):
        if item == 0:
            return self.x
        elif item == 1:
            return self.y
        elif item == 2:
            return self.z
        else:
            raise IndexError("There is no vector index: %d" % item)

    def __setitem__(self, key, value):
```

(下页继续)

(续上页)

```
        if key == 0:
            self.x = value
        elif key == 1:
            self.y = value
        elif key == 2:
            self.z = value
        else:
            raise IndexError("There is no vector index: %d" % key)

    def dot(self, other):
        return self.x * other.x + self.y * other.y + self.z * other.z

    def cross(self, other):
        return Vector(self.x * other.x, self.y * other.y, self.z * other.z)

    def length(self):
        return math.sqrt(pow(self.x, 2) + pow(self.y, 2) + pow(self.z, 2))

v1 = Vector(2, 1.5, 3.2)
v2 = Vector(3, 4, 5)
print(v1)
print(v1 + v2)
print(v1.dot(v2))
print(v1.length())
print(v1[2])
v1[2] = 10
print(v1)
```

实例方法、类方法和静态方法

```
import string

def getAllChars():
    all_letters = string.ascii_lowercase
    result=[]
    for letter in all_letters:
        result.append([letter, all_letters.find(letter)])
    return result

def generateChars():
```

(下页继续)

(续上页)

```
    all_letters = string.ascii_lowercase
    for letter in all_letters:
        yield letter, all_letters.find(letter)

for i in generateChars():
    print("Letter: {0} - Index: {1}".format(*i))
```

语法糖与装饰器

- @property
- @classmethod
- @staticmethod

```
def check_args(func):
    def wrap(*args):
        args = filter(bool, args)
        func(*args)

    return wrap


@check_args
def test(*args):
    print(args)


print(test)
test(1, 0, 2, "", [], 3)
```

装饰器不一定非得是个函数返回包装对象，也可以是个类，通过 ___call___ 完成目标调用

```
class CheckArgs(object):
    def __init__(self, func):
        self._func = func

    def __call__(self, *args):
        args = filter(bool, args)
        self._func(*args)
```

(下页继续)

(续上页)

```
@CheckArgs
def test(*args):
    print(args)


print(test)
test(1, 0, 2, "", [], 3)
```

为 class 提供装饰器

```
def singleton(cls):
    def wrap(*args, **kwargs):
        o = getattr(cls, "__instance__", None)

        if not o:
            o = cls(*args, **kwargs)
            cls.__instance__ = o

        return o

    return wrap


@singleton
class A(object):
    def __init__(self, x):
        self.x = x

print(A)
a, b = A(1), A(2)
print(a is b)
```

类专属的装饰器

```
class Artist(object):
    _hits = ["John"]

    def __init__(self, name):
        self._name = name

    @property
```

(下页继续)

(续上页)

```
    def name(self):
        return self._name

    @name.setter
    def name(self, name):

        if name not in CUSTOM_ARTIST:
            raise ValueError("%s is not a custom artist" % name)

        self._name = name

    @staticmethod
    def random_artist():
        return Artist(random.choice(CUSTOM_ARTIST))

    @classmethod
    def hits(cls):
        return cls._hits


# rr = Artist("Andy Hu")
# print(rr.name)
# print(type(rr.name))
# rr.name = "Andy"
# print(rr.name)
# rr2 = Artist.random_artist()
# print(rr2.name)
# print(Artist.hits())
# print(Artist._hits)
rr = Artist("Andy")
print(rr.random_artist())
# print(rr.hits())
```

私有属性与私有方法

在定义方法前面加 ___method 即声明私有方法，理论上私有方法是不能被继承的，只能在当下定义的类中被调用，但 Python 的私有方法只是约定上的私有，实际是可以通过类名来访问。

- 类属性和类方法可以被实例对象来调用，也可以通过类名直接调用，一般是通过类名调用
- 静态方法可以被实例对象来调用
- 实例属性和实例方法只能通过实例对象来调用，不能通过类名直接调用

- 静态方法和类方法的区别是类方法可能需要访问类属性，和类还有那么点关系，静态方法是访问不了任何类属性或者实例属性的

2.37 Python 自定义函数

函数是 Python 内建的一种封装，我们把大段代码拆成函数，通过一层一层的函数调用，就可以把复杂的任务分解成简单的任务，这种分解称为面向过程的程序设计。函数就是面向过程的程序设计的基本单元。Python 提供了很多内置函数，比如 print()，你也可以自己创建函数，这被叫做自定义函数。

函数的几个概念

- 基本语法

```
def 函数名(形式参数):
    函数体
    return 返回值
```

形式参数和返回值不是必须的，但函数都是有返回值的，如果没有使用关键字 return 关键字，则函数默认返回 None。

- 形式参数 (形参)
- 实际参数 (实参)
- 返回值

形参的几种形式

- 缺省参数，又叫默认参数
- 可变参数 *args

*args 表示可变参数，就是传入的参数个数是可变的，可以是 1 个，2 个到任意个或者 0 个。其实就是将传入的一堆参数打包成元组使用。

- 关键字参数 **kwargs

**kwargs 将传入的 0 个或者多个含参数名的参数打包成字典使用。

- 不同形式参数之间组合

作用域

在函数外部，locals() 和 globals() 作用完全相同。而当在函数内部调用时，locals() 则是获取当前函数堆栈中的名字空间，其中存储的是函数参数、局部变量等信息。

```
>>> locals()
{'__builtins__': <module '__builtin__' (built-in)>, '__name__': '__main__', '__doc__':␣
↪None, '__package__': None}
```

(下页继续)

(续上页)

```
>>> globals()
{'__builtins__': <module '__builtin__' (built-in)>, '__name__': '__main__', '__doc__':␣
↪None, '__package__': None}
>>> locals() is globals()
True
>>> def func(a):
...     b = a % 2
...     print(locals())
...     print(locals() is globals())
...
>>> func(5)
{'a': 5, 'b': 1}
False
>>> locals() is globals()
True
```

```
def f(arg1, arg2, arg3):
    print(arg1, arg2, arg3)

def f(*args):
    print(type(args))
    print(args)

f(3, 2, 1)

def f(**kwargs):
    print(type(kwargs))
    print(kwargs)

f(name="Andy", lang="Chinese")

def f(*args, **kwargs):
    print(args)
    print(kwargs.items())

f(1, 2, name="Andy", lang="Chinese")

def setParms(**kwargs):
```

(下页继续)

(续上页)

```
    for k,v in kwargs.items():
        print("Setting parameter {0} to {1}".format(k,v))

setParms(streng=13, resistance=0.7, weigh=100)

return

def double_parm(**kwargs):
    return kwargs["weigh"] * 2

print(double_parm(streng=13, resistance=0.7, weigh=100))
```

函数赋值

```
def double_parm(**kwargs):
    return kwargs["weigh"] * 2, kwargs["streng"]

new_func = double_parm

print(new_func(streng=13, resistance=0.7, weigh=100))
```

函数中套函数

```
def double_parm(**kwargs):

    def check(weight):
        if weight < 100:
            return False
        else:
            return True

    if check(kwargs["weigh"]):
        return kwargs["weigh"] * 2, kwargs["streng"] * 2
    else:
        return kwargs["weigh"] * 2, kwargs["streng"] * 4

new_func = double_parm

print(new_func(streng=13, resistance=0.7, weigh=60))

import os
```

(下页继续)

(续上页)

```
from sys import version

PATH = "/tmp/folder/name"


# def localFunc():
#     global version
#     version = 13.3
#     print("Local version", version)

# localFunc()
# print(version)


# def getTempContent():
#     tempdir = os.listdir("C:/")

# getTempContent()
# print(tempdir)


def func1():
    print(mvar)
    print(PATH)


def funcBase():
    mvar = 15
    func1()

funcBase()
```

默认值（缺省值）对函数重载的作用

len 多态函数

len(“andy”) len(range(10))

运算符重载多态性 100 + 200 “hello ” + “python”

Python 函数没有重载的概念主要是因为动态语言特性以及缺省值

```
def function(args):
    code
    return

def foo():
    print("this is function")

foo()

# 形式参数
def sayHello(name):
    print("hello, ", name)

sayHello("andy")

# 缺省参数

def sayHello(name="andy"):
    print("hello, ", name)

sayHello()

# 形式参数> 缺省参数>*args>**kwargs
# 可变参数
def sayHello(*names):
    print(names)

sayHello("andy", "tommy")

# 顺序传参, 关键字传参
def foo(a, b, c):
    print("a is ", a)
    print("b is ", b)
    print("c is ", c)

foo(1, 2, 3)
foo(a=1, b=2, c=3)
foo(b=2, c=3, a=1)
foo(1, c=3, b=2)
```

(下页继续)

(续上页)

```
def sayHello(**names):
    print(names, type(names))

sayHello(name="andy", age=30)

def foo(a, b=1, *args, **kwargs):
    pass

# 返回值
def foo():
    return 5

a = foo()
print(a)

def foo(a):
    if a < 0:
        return
    return 100 + a

foo(9)
foo(-9)
```

2.38 Python 自定义模块

- 几种常见的模块导入方式
- 模块导入的搜索机制 sys.path
- 缓存区 sys.modules、pyc 以及 reload
- 查询模块路径 ___file___
- 模块中 ___name___ 的作用
- 模块帮助文档 docstring（___doc___）

几种常见的模块导入方式

Python 允许开发者通过导入外部程序块的方式来扩展自己的程序，这些可以被导入（import）并使用的程序块就是模块（module）

模块的基本单元是一个.py 文件，Python 的包在广义上也被称为模块（module）

以内置模块 datetime 为例

```
import datetime
import datetime as dt
print(dt.datetime.now())
from datetime import datetime
print(datetime.now())
from datetime import *
from datetime import datetime as ddt
print(ddt.now())
```

模块导入的搜索机制 sys.path，模块导入搜索路径是由多个目录路径组成的列表，第一个路径默认是当前模块所在的路径，模块搜索路径 = 当前工作路径 + sys.path 列表内的所有路径。

- 何为环境变量 PYTHONPATH?
- sys.path 在环境变量中起什么作用?
- sys.path 添加路径的两种方案以及区别?
- DCC 软件如何管理 sys.path?

```
import sys

path in sys.path or sys.path.insert(0, path)
```

- 缓存区 sys.modules、pyc 以及 reload
- import 可以是属性方法类型，reload 只能是模块
- 查询模块路径 ___file___

___file___ 属性是一个 Python 模块隐藏的默认属性，它描述了一个模块的完整路径。

- 模块中 ___name___ 的作用

___name___ 属性是所有 Python 模块自带的一个隐藏属性，用于标注模块在不同执行环境下的名称。当一个模块作为主模块运行（被 Python 解释器直接运行）时，___name___ 的值是” ___main___”，否则，该模块的 ___name___ 属性值为此模块的名称。

- 模块帮助文档 docstring（___doc___）

双三引号，docstring 写在什么位置? help() 内置函数与 docstring 的关系?

myFirstModule.py

```
# -*- coding: utf-8 -*-
#!/usr/bin/python

"""
```

(下页继续)

(续上页)

```
this is a doc string
"""

a = 5

def foo():
    print("this is foo function")

print("hello, this is my first module")

if __name__ == "__main__":
    print("this string is under main")
```

main.py

```
# -*- coding: utf-8 -*-
#!/usr/bin/python

print("before import")
import myFirstModule as mfm

print("after import")
print("my first module attr a is: ", mfm.a)
print("my first module method foo is: ",  mfm.foo)
print(mfm.__name__)
print(mfm.__file__)
print(mfm.__doc__)
```

2.39 Python 自定义包

- 包的唯一标识 ___init___.py
- 查询包路径 ___path___
- 包的层级结构
- 通配导入 ___all___
- 包的相对路径导入

包（package）是将一些模块组织在一起构成一个更大规模的模块，也是 Python 中对模块的更高一级的抽象。Python 允许用户把目录当成模块看待，目录中不同模块文件，就变成了包里面的子模块。一个包可以由

一个或多个模块或子包构成的模块，包目录下不但可以包含作为子模块的 py 文件，还可以包含子目录，这些子目录也可以是 Python 的包。

- ___init___.py 文件的作用?

当一个文件夹中存在一个 ___init___.py 文件时，这个文件夹就会被 Python 解释器识别为一个包。___init___.py 除了标识一个文件夹是包的作用以外，它还可以执行一些初始化操作。在使用 import 导入一个模块的时候，___init___.py 文件会首先被执行。因此 ___init___.py 中可以写一些初始化的代码，比如导入其他依赖的 Python 模块。

- 如何验证 ___init___.py 是优先被导入的?

绝大部分时候让 ___init___.py 文件空着就好。

- 查询包路径 ___path___

___path___ 是一个包自带的隐藏属性，它描述一个包的完整路径，也是包的一个标识。

- 包的层级结构

封装成包是很简单的。在文件系统上组织你的代码，并确保每个目录都定义了一个 ___init___.py 文件。

```
chooseTask/
    __init__.py
    ui/
        __init__.py
        resource.py
    config/
        __init__.py
        user.py
        sg.py
    mainWin.py
```

有了上面的层级结构，你可以执行各种 import 语句

包的相对路径导入

Python 自定义包的相对导入问题?

Python 2.x 和 3.x 包导入的区别

相对导入对执行方式是有一定的要求，不同执行方式可能会报一种不是错误的错误

假设现在有一个 myPackage 的包

```
myPackage/
    __init__.py
    aaa/
        __init__.py
        spam.py
```

(下页继续)

(续上页)

```
        grok.py
    bbb/
        __init__.py
        bar.py
```

如果模块 myPackage.aaa.spam 要导入同目录下的模块 grok

```
# myPackage/aaa/spam.py
from . import grok
```

如果模块 myPackage.aaa.spam 要导入不同目录下的模块 bbb.bar

```
# myPackage/aaa/spam.py
from ..bbb import bar
```

```
# -*- coding: utf-8 -*-
#!/usr/bin/python

"""
this is a doc string of test pack
"""

a = 1
b = 2

def foo():
    print("fool")
```

2.40 Python 守护线程

2.41 Python 数据类型：布尔值 & 空值

- 布尔值 True & False

比较运算符

==, <, >, !=, <=, >=, in, not in, is, is not

逻辑运算符

```
and
or
not
```

Python 逻辑运算符的短路规则

如果你了解二进制以及逻辑电路的知识，对逻辑运算符应该不会陌生。十进制数有加减乘除等算术运算，二进制作为另一种进制规则，自然也会有自己的运算方法，这种运算方法叫做逻辑运算。在 Python 中逻辑运算符有三个 and、or 和 not，对应逻辑电路里的与、或、非门。

短路规则，又称最小化求值。是一种逻辑运算符的求值策略。只有当第一个运算数的值无法确定逻辑运算的结果时，才对第二个运算数进行求值。例如，当 and 的第一个运算数的值为 False 时，其结果必定为 False；当 or 的第一个运算数为 True 时，最后结果必定为 True，在这种情况下，就不需要知道第二个运算数的具体值。

那么 Python 中如何判定一个对象的真假呢？下面列举的是 Python 中的默认为假值的对象。

```
None
False
0
0.0
0j
Decimal(0)
Fraction(0, 1)
[] - an empty list
{} - an empty dict
() - an empty tuple
'' - an empty str
b'' - an empty bytes
set() - an empty set
```

一个对象的真假状态可以通过内置函数 bool() 来做检测，在写 if 条件的时候可以通过对象的真假状态来直接判断，而无需比较。

不建议的代码：

```
if var == None:
if var == 0:
if var == []:
if var == "":
```

建议的代码：

```
if not var:
```

用 and 和 or 实现三元表达式（也叫三目运算符）

```
x = 5
y = "A" if x > 0 else "B"
```

用 or 提供默认值

```
path in sys.path or sys.path.insert(0, path)
```

基于这种惰性求值方法，尽可能将需要求值时间短的表达式放前面。

- 空值 None

None 是 Python 中一个特殊的值，虽然它不表示任何数据，但仍然具有重要的作用。自定义函数如果没有使用 return，则默认返回值是 None。

2.42 Python 数据类型：整型 & 浮点型

整型即整数（int），Python 在内存中有一个小整数缓存池：[-5, 257)。

浮点型也称浮点数，即小数，在内存中存储的精度比整型要高。

```
>>> 1 + 2
3
>>> 1.0 + 2
3.0
>>> 10 / 3
3
>>> 10 / 3.0
3.3333333333333335
>>> 10.0 // 3.3
3.0
>>> 1.1 % 0.3
0.20000000000000012
>>> "hello " + "world"
'hello world'
>>> "hello" * 5
'hellohellohellohellohello'
>>> True + True
2
>>> 5 / False
```

```
>>> a = 5
>>> a
5
>>> a = 3.3
>>> a
3.3
>>> a = "andy"
>>> a
'andy'
>>> a = True
>>> a
True
>>> b = False
>>> b
False
>>> a == b
False
>>> a != b
True
>>> s = "My Name is Andy"
>>> s2 = "Hello "
>>> s2 + s
'Hello My Name is Andy'
>>> s - "Name"
>>> dir(s)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__
↪', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__
↪gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__
↪ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '_
↪_setattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split
↪', '_formatter_parser', 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith
↪', 'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower',
↪ 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition',
↪'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
↪'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill
↪']
>>>
>>>
>>> s.capitalize()
'My name is andy'
```

(下页继续)

(续上页)

```
>>> s.upper()
'MY NAME IS ANDY'
>>> s.endswith("y")
True
>>> s.find("n")
12
>>> "My name is %s" % "Andy"
'My name is Andy'
>>> "My name is {0}".format("Andy")
'My name is Andy'
>>> "My name is {0}, age is {1}".format("Andy", 28)
'My name is Andy, age is 28'
>>> "My name is %s, age is %d".format("Andy", 28)
'My name is %s, age is %d'
>>> "My name is %s, age is %d" % ("Andy", 28)
'My name is Andy, age is 28'
>>>
>>> print("流浪地球")
流浪地球
>>> print(u"流浪地球")
🄵梅🄵🄵碌🄵🄵貌
>>> print("My name is Andy.\nMy age is 28.")
My name is Andy.
My age is 28.
>>> print(r"My name is Andy.\nMy age is 28.")
My name is Andy.\nMy age is 28.
>>> s3 = "Hello Andy  "
>>> s3
'Hello Andy  '
>>> s3.strip()
'Hello Andy'
>>> s4 = "Name1 Name2 Name3"
>>> s4.split()
['Name1', 'Name2', 'Name3']
>>> s4.startswith("N")
True
>>>
>>> print("hello world")
hello world
>>> print("123")
```

(下页继续)

(续上页)

```
123
>>> print(123)
123
>>> print("how are you?")
how are you?
>>> print("I'm OK!")
I'm OK!
>>> print("""
... this is a multi-line string.
... it has two lines.
... """)

this is a multi-line string.
it has two lines.

>>> print('''
... this is a multi-line string.
... it has two lines.
... ''')

this is a multi-line string.
it has two lines.

>>>
>>> print(1 + 2)
3
>>> print(1.0 + 2)
3.0
>>> print(10 / 3)
3
>>> print(10 / 3.0)
3.33333333333
>>> print(10.0 / 3.3)
3.0303030303
>>> print(10.0 // 3.3)
3.0
>>> print(10 / 3)
3
>>> print(1.1 % 0.3)
0.2
```

(下页继续)

(续上页)

```
>>> print("hello" + "world")
helloworld
>>> print("hello" * 5)
hellohellohellohellohello
>>> print(True + True)
2
>>>
>>> print(3  > 4)
False
>>> print(4 < 5)
True
>>> print(5 == 6)
False
>>> print(5 is 5)
True
>>>
```

2.43 Python 数据类型：字符串

字符串（str）是编程语言中最基本也是最重要的一种数据类型，在 Python 中可以说无时无刻不会跟字符串打交道。

- 字符串定义

三引号正常用于多行注释 Docstring。

```
"andy"
'andy'
"""andy"""
'''andy'''
```

- 字符串格式化
  - %
  - .format()
  - F-Strings
  - format()
- 字符串索引取值
- 字符串不可变性

- 字符串切片
- 字符串循环迭代
- 字符串方法

掌握字符串方法是非常重要的一项技能，特别是常用的 format、join、split 和 replace 四个方法。

```
>>> type("python")
<type 'str'>
>>>
>>> dir("python")
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__
↪', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__
↪gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__
↪ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '_
↪_setattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split
↪', '_formatter_parser', 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith
↪', 'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower',
↪ 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition',
↪'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
↪'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill
↪']
>>>
```

- 转义字符
  - n
  - %%
  - \
- 字符编码
  - ASCII
  - GB2312
  - GBK
  - Unicode
  - UTF-8

2.44 Python 数据库操作

2.45 Python 装饰器

装饰器在 Python 中也称为语法糖，使用到的知识点是高阶函数与闭包函数，而高阶函数的依据是 Python 中一切皆对象，函数也是对象，所以装饰器也是以此为依据。

```
def myFunc():
    l = []

    for i in range(1000000):
        l.append(i)

    return l

if __name__ == "__main__":
    myFunc()
```

```
import time


def myFunc():
    l = []

    for i in range(1000000):
        l.append(i)

    return l

if __name__ == "__main__":
    print("start...")
    start = time.time()
    myFunc()
    end = time.time()
    print("end...")
    print("execute time: %.6f" % (end - start))
```

```
import time


```

(下页继续)

(续上页)

```
def myFunc():
    print("start...")
    start = time.time()

    l = []

    for i in range(1000000):
        l.append(i)

    end = time.time()
    print("end...")
    print("execute time: %.6f" % (end - start))

    return l

if __name__ == "__main__":
    myFunc()
```

```
import time


def measureTime(fun):
    def wrapped():
        print("start...")
        start = time.time()

        result = fun()

        end = time.time()
        print("end...")
        print("execute time: %.6f" % (end - start))

        return result

    return wrapped


def myFunc():
    l = []
```

(下页继续)

(续上页)

```
    for i in range(1000000):
        l.append(i)

    return l

if __name__ == "__main__":
    myFunc = measureTime(myFunc)
    myFunc()
```

```
import time


def measureTime(fun):
    def wrapped():
        print("start...")
        start = time.time()

        result = fun()

        end = time.time()
        print("end...")
        print("execute time: %.6f" % (end - start))

        return result

    return wrapped


@measureTime # Python 会自动执行 measureTime(myFunc)，执行完之后会将结果给到 myFunc，相当于
myFunc = measureTime(myFunc)
def myFunc():
    l = []

    for i in range(1000000):
        l.append(i)

    return l
```

(下页继续)

(续上页)

```
if __name__ == "__main__":
    yFunc()
```

```
import time


def measureTime(fun):
    def wrapped(*args, **kwargs):
        print("start...")
        start = time.time()

        result = fun(*args, **kwargs)

        end = time.time()
        print("end...")
        print("execute time: %.6f" % (end - start))

        return result

    return wrapped


@measureTime
def myFunc(num):
    l = []

    for i in range(num):
        l.append(i)

    return l

if __name__ == "__main__":
    yFunc(1000000)
```

```
import time


def measure_time(func):
```

(下页继续)

(续上页)

```
    def wrapped(*args, **kwargs):
        start = time.time()

        try:
            return func(*args, **kwargs)
        finally:
            runtime = time.time() - start
            print("Execution time: %.6f seconds" % runtime)

    return wrapped


@measure_time
def f():
    l = []
    for i in range(1000000):
        l.append(i)

if __name__ == "__main__":
    f()
```

```
import logging


def wrap_func_log(level="info"):
    LOG_LEVELS = {
        "debug": logging.debug,
        "info": logging.info,
        "warning": logging.warning,
        "error": logging.error
    }
    log_command = LOG_LEVELS.get(level) or logging.info

    def dec_wrapper(func):
        def wrapper(*args, **kwargs):
            log_command(func.__name__ + " is running...")
            return func(*args, **kwargs)
        return wrapper
    return dec_wrapper
```

(下页继续)

(续上页)

```
@wrap_func_log(level="error")
def add1(n):
    print(n + 1)

add1(10)
```

如果装饰器本身需要传入参数，则需要像下面这样处理

```
def log(text):
    def decorator(func):
        def wrapper(*args, **kwargs):
            print("%s %s():" % (text, func.__name__))
            return func(*args, **kwargs)
        return wrapper
    return decorator


@log("execute")
def now():
    print("2021/01/14")
```

装饰器的缺点是会改变原来函数的一些属性，比如 ___name___

```
import functools

def log(func):
    @functools.wraps(func)
    def wrapper(*args, **kwargs):
        print("call %s():" % func.__name__)
        return func(*args, **kwargs)
    return wrapper
```

2.46 Python 开发环境

解决两个问题 CG 软件中在哪里可以执行代码? 学习 print() 函数独立环境如何执行代码?

独立执行环境 DCC 软件中的执行环境 IDE 中的执行环境 print 函数 关键字 转义字符 注释

VS Code 安装、配置以及使用下载地址：https://code.visualstudio.com/ Python 2.7.15 安装下载地址：https:

//www.python.org/downloads/ Git 管理代码库 Git 下载地址：https://git-scm.com/downloads TortoiseGit 下载地址：https://tortoisegit.org/ 掌握 Markdown 语法撰写文档 Git Bash 结合 ipython 使用 ipython 安装：pip install ipython pip install PySide Python 第三方库路径 C:Python27Libsite-packages

Markdown 语法 Linux 操作指令 VS Code 高级使用技巧 PEP8 规范 Git 指令 ipython 高级使用技巧

TD 日常工作内容 撰写流程规范、软件工具说明文档等 根据需求开发相应的软件工具或插件 检查并修正生产过程中出现的错误文件 维护公共服务系统，如渲染农场、存储设备、公共服务器等 向软件工具或插件的使用者演示其功能及操作方式 管理和维护生产素材、制作文件、输出序列、生产元数据

TD 日常工作流程选择集成开发环境（IDE） VS Code 轻量跨平台免费 VS Code 基本配置以及用途 VS Code Python 开发环境 VS Code 调试以及断点调试 print 是最简单的一种 debug 方法 VS Code 实用快捷键

Ctrl+F2：批量选择相同字段并修改 Ctrl+D：逐一加选相同字段并修改 Alt+Z：自动换行查看 Ctrl+/：注释与反注释 Tab：Shift+Tab：Alt+Shift+ 左键：列模式编辑 Ctrl+H：替换 Ctrl+F：查找 F5：Debug Ctrl+Shift+ 方向键：复制代码

PEP8 代码规范 - 安装

https://pypi.org/project/pep8/

pip install pep8

pip install –upgrade pep8 - 向导

https://www.python.org/dev/peps/pep-0008/

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

""" 写个简单统计运行时间的函数
"""

import time

def measure_time(func):

    def wrapped(*args, **kwargs):
        start = time.time()

        try:
            return func(*args, **kwargs)
        finally:
            runtime = time.time() - start
            print("Execution time: %.6f seconds" % runtime)
```

(下页继续)

(续上页)

```
    return wrapped

@measure_time
def f(n):
    l = []
    for i in range(n):
        l.append(i)

if __name__ == "__main__":
    f(1000000)
```

其他 IDE 简单介绍 VS Code Eclipse PyCharm Atom …

文本编辑器 Vim Emacs Sublime Notepad++ ….

管理自己的代码 - Git & TortoiseGit - Github & Gitlab - Gitlab 创建私有代码仓库 - Gitlab Release 版本 - Git 最常用的操作指令

```
git clone URL
git config --global user.name "Your Name"
git config --global user.email "Your Email"
git status
gitk
git add .
git commit -m "Your Comment" -a
git push origin master
git pull origin master
```

- TortoiseGit 界面化提交代码
- Github 拉取代码并使用
- 反编译 pyc 文件 Easy Python Decompiler
- IPython & Gitbash

```
# 更新 pip
python -m pip install --upgrade pip
# 安装 ipython
pip install ipython
pip install *.whl
```

import this import antigravity dir(__builtin__)

解决一个中心问题：在哪里可以执行 Python 代码？认识第一句代码，内置函数 print?

Python 2.7 VS Python 3.7 https://www.liaoxuefeng.com/ Google 文档 https://www.python.org/ https://docs.python.org/zh-cn/3.9/

python -V pip 安装第三方模块

pip install ipython Path 环境变量

Anaconda https://www.anaconda.com/distribution/#download-section jupyter notebook IPython help() cls

```
a = 100
if isinstance(a, int):
    print("a is int")
else:
    print("a is not int")

help(isinstance)
help(list)

for i in range(1, 11):
    print(i)

array = list()

for i in range(1, 11):
    array.append(i)

print(array)
```

列表生成式 array = [i for i in range(1, 11)]

range(1, 100, 2) range(1, 100)[::2] [i for i in range(1, 100) if i % 2 != 0] [i for i in range(1, 100) if i % 2 != 0 and i < 50]

range(50, 10, -1)

help(map)

help(filter)

%ls %cd D: ? map? exit

jupyter notebook

array = [i for i in range(1, 11) if i % 2 == 0 or i == 1]

def getList(min, max, step): return [i for i in range(min, max, step)]

getList(1, 100, 5)

https://docs.python.org/zh-cn/3.9/

入门教程标准库参考语言参考

 # 注释 缩进

a = u" ccccc" b = "10" c = "" This is a three single quote This is a string %s : %s "" % (a, b)

c = "" This is a three single quote This is a string {0} : {0} "" .format(a, b)

print(type(a)) print(type(b)) print(c)

d = 1000 e = 3.14159 f = 0x400 g = 3.14e-2 print(d, e, f, g)

a = 1000L print(type(a))

print(oct(100)) print(hex(100)) print(bin(100)) print(0.1234) print(.1234) print(3.14e-8)

mySet = {1, 2, 3, 4, 3, 3, 3, 1} mySet.add(10) mySet.add(1) mySet = frozenset(mySet) print(mySet)

生成器 (i for i in range(1, 10))

a = 99 b = 77 print(~a + 1) print(~b + 1)

a = 2

print(a << 2) print(a << 3) print(bin(a << 1)) print(bin(a << 2)) print(bin(a << 3))

cmd 执行代码 debug 代码

Git TortoiseGit

解决一个中心问题：如何存储自己编写的代码?

IPython VS Code Vim notepad++ Sublime Text PyCharm

Python 执行环境 Python IDE(集成开发环境)

注释 #

hello.py hou.py

可以写代码的软件 VS Code

= 赋值 == 等于—> 真假事件

if 真假事件: do something

else: do something

自动补全

代码健壮性

量到质变代码量记忆碎片内功心法条件反射

解决一个中心问题：如何区分 Python 和 VEX? Python 代码有哪些明显的特征?

PEP8 Google 二进制计算机编码简单涉及一下基本数据类型

Python 执行环境 Python IDE Python 基本数据类型 (Pythond 语法规则或者代码规范) PEP8 代码规范 Google 代码规范

Python VS VEX

语句块 Python 有冒号，VEX 没有冒号 VEX 有分号 Python 没有 Python import module

基本数据类型 (五大类) 整型 int 浮点型 float 布尔值 bool True&False 1&0 二进制 01010100 计算机编码 (硬件环境) 0-9 二极管电压高电压低 8 位二进制 == 一个字节 16 位二进制 ==2 个字节 unicode? 字符串 string None 假的事件函数返回值默认是 None bool 函数 id()

0-0 1-1 2-10 3-11 4-100 5-101 6-110 7-111 8-1000 9-1001 10-1010

八进制跟十六进制 0

8GB 1TB 8GB = 8*1024MB = 8*1024*1024KB = 8*1024*1024*1024 字节 =8*1024*1024*1024*8 二进制

注释 # # TODO docstring 如何写 help 帮助文档 Ctrl+/ 注释反注释 Tab 代码缩进多行注释 “”” “””

2.47 Python 注释文档

- docstring

单行代码注释多行代码注释

docstring 规范都是以三引号的字符串自定义模块在开始自定义函数 def 之后自定义 class 之后

```
def sceneViewer():
    """ Returns an existing open Scene Viewer pane if there is one. A
        Context viewer is also acceptable is no dedicated scene viewer
        is found.
    """
```

如何将 docstring 转成 html 文档?

2.48 Python 文件操作

Python 文件操作实际就是内存数据与硬盘文件数据 IO 的一个过程，Python 提供基础的内置函数以及所谓的上下文管理器来读写文件，也提供了很多第三方库来操作文件解析文件。

- 内置函数 open

通过 open() 读写文件一要注意获取的文件对象需要最后通过 close() 方法关闭，以释放内存空间，二要注意编码转换的问题，py3 添加了编码的支持。

```
# py2
open(name[,mode[,buffering]])

# py3
open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None,␣
↪closefd=True, opener=None)
```

```
f = open("somefile.txt", "rt")
print(f)
data = f.read()
print(data)
f.close()
```

- 上下文管理器

上下文管理协议为代码块提供了包含初始化和清理操作的安全上下文环境。即便代码块发生异常，清理操作也会被执行。

通过关键字 with 即可简单实现上下文管理器，上下文管理器 with 在语句块结束之后，文件会自动关闭，无需再编写 close() 方法，因此更推荐使用 with open 的写法。

```
with open("somefile.txt", "rt") as f:
    data = f.read()

with open("somefile.txt", "rt") as f:
    for line in f:
        pass

with open("somefile.txt", "wt") as f:
    f.write(text1)
    f.write(text2)

with open("somefile.txt", "wt") as f:
    print(line1, file=f)
    print(line2, file=f)
```

```
with open("somefile.txt", "rt", encoding="gb2312") as f:
    pass
```

- 文件读写模式

rb

wt

- 文件对象的方法

read()

readlines()

write()

- 第三方库

```
import os
import shutil
import json
import yaml
```

os 模块是 python 与计算机操作系统交互的模块，它提供了一系列访问操作系统的相关功能。

```
import os
# 当前路径
os.getcwd()
os.mkdir()
os.listdir(os.getcwd())
# os.chdir VS os.getcwd
os.chdir()
os.rmdir()
os.makedirs()
os.rename()
os.removedirs()
os.path.join()
os.path.exists()
os.path.dirname()
os.path.split()
os.path.splitext()
os.path.splitdriver()
os.path.isdir()
os.path.isfile()
```

shutil 模块是 python 内置的一个高级文件操作模块，提供了一些针对文件操作和文件采集相关的高级功能。在 os 模块相关功能基础上进行了升级，形成了一系列专门为文件操作而设计的功能集合。

```
import shutil
filePath = "D:/test"
newFilePath = "D:/test_new"
# copy 函数将一个文件拷贝到另一个路径下
```

(下页继续)

(续上页)

```
shutil.copy(filePath, newFilePath)
# copy2 函数除了拷贝文件外，还会将源文件的权限，最后访问时间，最后修改时间等拷贝到目标文件上
newFilePath2 = "D:/test_new2"
shutil.copy2(filePath, newFilePath2)
shutil.move()
shutil.rmtree()
```

json 模块是用来解析字典容器与文件数据之间的模块，用来编写.json 配置文件非常方便以容易处理。

load

dump

loads

dumps

yaml

xml

2.49 Python 流控制语句

Python 流控制语句包括 if 条件判断语句、for 循环语句、while 循环语句以及 try 异常中断语句。

- if 条件语句

if 语句如果条件太多，可以使用字典替代方案。

```
# if 语句的几种常见写法
if condition:
    statement block

if condition:
    statement block
else:
    statement block

if condition:
    statement block
elif condition:
    statement block
...
```

(下页继续)

(续上页)

```
else:
    statement block
```

- for 循环语句

```
# for 语句的写法
for element in iterable:
    statement block

for element in iterable:
    statement block
else:
    statement block
```

break 与 continue 的区别：break 中断整个循环语句，continue 中断本次循环，继续下一次循环。

pass 空语句表示什么都不做。

- while 循环语句

while 循环需要注意避免死循环，语句块中需要有自增或自检以结束循环条件。

```
# while 语句的写法
while condition:
    statement block
```

- try 异常中断

```
# try 语句的写法

try:
    statement block
except:
    statement block

try:
    statement block
except A:
    statement block
except B:
    statement block
except:
    statement block
```

(下页继续)

(续上页)

```
else:
    statement block
finally:
    statement block
```

```
try:
    1 / 0
except ZeroDivisionError as e:
    print(e)
```

异常捕获对于调试代码并不是一件友好的事情，经常在调试代码过程中我们会避免使用 try 语句以获得代码异常的详细信息。异常也可以自定义，可以通过关键字 raise 来抛出异常。

```
def checkAbs(x):

    if not isinstance(x, (int, float)):
        raise TypeError("x only support int or float")

    if x >= 0:
        return x
    else:
        return -x
```

不要这样写

```
if a == False:
if a == 0:
if a == None:
if a == []:
if a == "":
```

建议写成

```
if not a:
```

- 三元赋值法

```
userSays = raw_input("请输入:") or "nothing"

"hello" if True else "world"
"hello" if 1 > 2 else "world"
```

2.50 Python 函数式编程

函数式编程（Functional Programming，简称 FP）理论依据是在 Python 中一切皆对象

Python 函数是一等对象意味着在 Python 中可以赋值给变量引用。

```
>>> def funObject(arg):
...     return arg
...
>>> funObject(100)
100
>>> funTest = funObject
>>> funTest
<function funObject at 0x03239BF0>
>>> funObject
<function funObject at 0x03239BF0>
>>> funTest(99)
99
>>>
```

Python 支持高阶函数意味着函数可以接受其它函数作为参数或者返回值。下面有几个经典的案例。

内置函数 map()、filter()、reduce() 以及 sorted() 都是高阶函数的经典案例。

```
>>> help(map)
Help on built-in function map in module __builtin__:

map(...)
    map(function, sequence[, sequence, ...]) -> list

    Return a list of the results of applying the function to the items of
    the argument sequence(s).  If more than one sequence is given, the
    function is called with an argument list consisting of the corresponding
    item of each sequence, substituting None for missing values when not all
    sequences have the same length.  If the function is None, return a list of
    the items of the sequence (or a list of tuples if more than one sequence).

>>> help(filter)
Help on built-in function filter in module __builtin__:

filter(...)
    filter(function or None, sequence) -> list, tuple, or string
```

(下页继续)

(续上页)

```
    Return those items of sequence for which function(item) is true.  If
    function is None, return the items that are true.  If sequence is a tuple
    or string, return the same type, else return a list.

>>> help(reduce)
Help on built-in function reduce in module __builtin__:

reduce(...)
    reduce(function, sequence[, initial]) -> value

    Apply a function of two arguments cumulatively to the items of a sequence,
    from left to right, so as to reduce the sequence to a single value.
    For example, reduce(lambda x, y: x+y, [1, 2, 3, 4, 5]) calculates
    ((((1+2)+3)+4)+5).  If initial is present, it is placed before the items
    of the sequence in the calculation, and serves as a default when the
    sequence is empty.

>>> help(sorted)
Help on built-in function sorted in module __builtin__:

sorted(...)
    sorted(iterable, cmp=None, key=None, reverse=False) --> new sorted list
```

不管是内置函数 sorted 还是列表方法 sort()，都可以传递 key 参数接受函数来自定义排序方式（通常默认排序方式不满足需求）。

```
import re

files = ["tank_1_color_v0.rat",
         "tank_2_color_v5.rat",
         "tank_1_color_v3.rat",
         "tank_3_color_v1.rat",
         "tank_4_color_v2.rat",
         "tank_4_color_v4.rat",
         "tank_5_color_v1.rat",
         "tank_6_color_v6.rat"]

pat_num = re.compile("\D+_(\d+)_")
pat_ver = re.compile("(\d+)\D+$")
```

(下页继续)

(续上页)

```
def sorter_num(elem):
    res = re.search(pat_num, elem)
    return res.groups()[0]


def sorter_ver(elem):
    res = re.search(pat_ver, elem)
    return res.groups()[0]

print(sorted(files, key=sorter_num))
print(sorted(files, key=sorter_ver))

# ['tank_1_color_v0.rat', 'tank_1_color_v3.rat', 'tank_2_color_v5.rat', 'tank_3_color_v1.
↪rat', 'tank_4_color_v2.rat', 'tank_4_color_v4.rat', 'tank_5_color_v1.rat', 'tank_6_
↪color_v6.rat']
# ['tank_1_color_v0.rat', 'tank_3_color_v1.rat', 'tank_5_color_v1.rat', 'tank_4_color_v2.
↪rat', 'tank_1_color_v3.rat', 'tank_4_color_v4.rat', 'tank_2_color_v5.rat', 'tank_6_
↪color_v6.rat']
```

因为函数可以作为参数传递，匿名函数应用而生，对于简单的函数可以使用 lambda 函数一句话来表达。

```
import re

files = ["tank_1_color_v0.rat",
         "tank_2_color_v5.rat",
         "tank_1_color_v3.rat",
         "tank_3_color_v1.rat",
         "tank_4_color_v2.rat",
         "tank_4_color_v4.rat",
         "tank_5_color_v1.rat",
         "tank_6_color_v6.rat"]

pat_num = re.compile("\D+_(\d+)_")
pat_ver = re.compile("(\d+)\D+$")

print(sorted(files, key=lambda elem: re.search(r"\D+_(\d+)_", elem).groups()[0]))
print(sorted(files, key=lambda elem: re.search(r"(\d+)\D+$", elem).groups()[0]))
```

(下页继续)

(续上页)

```
# ['tank_1_color_v0.rat', 'tank_1_color_v3.rat', 'tank_2_color_v5.rat', 'tank_3_color_v1.
↪rat', 'tank_4_color_v2.rat', 'tank_4_color_v4.rat', 'tank_5_color_v1.rat', 'tank_6_
↪color_v6.rat']
# ['tank_1_color_v0.rat', 'tank_3_color_v1.rat', 'tank_5_color_v1.rat', 'tank_4_color_v2.
↪rat', 'tank_1_color_v3.rat', 'tank_4_color_v4.rat', 'tank_2_color_v5.rat', 'tank_6_
↪color_v6.rat']
```

Python 闭包函数是一种在函数中定义函数的行为，如果将函数以返回值的形式 return，是高阶函数的另一种形式。

Python 装饰器是闭包函数与高阶函数的一种应用案例。

```
import time


def measure_time(func):

    def wrapped(*args, **kwargs):
        start = time.time()

        try:
            return func(*args, **kwargs)
        finally:
            runtime = time.time() - start
            print "Execution time: %.6f seconds" % runtime

    return wrapped


@measure_time
def testFunc(num):
    time.sleep(num)


testFunc(1)
testFunc(3)
testFunc(5)


# Execution time: 1.000000 seconds
# Execution time: 3.000000 seconds
```

(下页继续)

(续上页)

```
# Execution time: 5.001000 seconds
```

2.51 Python 生成器

生成器的本质也是迭代器，惰性迭代，在于减少内存占用。

yield 与 return 的区别

next() 与 ___next___()

```
def generatorFunc():
    a = 1
    yield a

    a = 2
    yield a

    a = 3
    yield a
```

2.52 Python IDE

IDE 的全称是：Integrated Development Environment，简称 IDE，也称为 Integration Design Environment、Integration Debugging Environment，翻译成中文叫做“集成开发环境”，在台湾那边叫做“整合開發環境”。它是一种辅助程序员开发用的应用软件。

IDE 通常包括程式语言编辑器、自动建立工具、通常还包括除错器。有些 IDE 包含编译器/直译器，如微软的 Microsoft Visual Studio，有些则不包含，如 Eclipse、SharpDevelop 等，这些 IDE 是通过调用第三方编译器来实现代码的编译工作的。有时 IDE 还会包含版本控制系统和一些可以设计圆形用🄵界面的工具。许多支援物件导向的现代化 IDE 还包括了类🄵浏览器、物件检视器、物件结构图。虽然目前有一些 IDE 支援多种程式语言（例如 Eclipse、NetBeans、Microsoft Visual Studio），但是一般而言，IDE 主要还是针对特定的程式语言而量身打造（例如 Visual Basic）。

- 集成开发环境
  - Visual Studio Code
  - JetBrains PyCharm
  - Sublime Text 3
- 文本编辑器
  - Notepad++

  - Vim

- 交互式开发环境
  - IDLE
  - DreamPie
  - IPython
- VSCode 配置

```
{
    "python.linting.pylintEnabled": false,
    "python.linting.pep8Enabled": true,
    "editor.renderWhitespace": "all",
    "editor.mouseWheelZoom": true,
    "editor.rulers": [79, 120, 150],
    "editor.tabSize": 4,
    "window.title": "${activeEditorLong}",
    "python.pythonPath": "C:/Python27/python.exe"
}
```

- VSCode 快捷键
  - Ctrl + /：注释、反注释
  - Ctrl + F2：选中所有相同字段以便批量修改
  - Ctrl + D：逐个选中相同字段以便批量修改
  - Shift + Alt + LM：列模式编辑
  - Shift + Alt + ↑/↓：向上、向下复制
- VSCode 好用的插件
  - Python
  - VEX
  - MayaCode
  - Code Runner

2.53 Python 缩进原则

Python 中使用冒号 + 缩进的方式代替 C++ 等语言中的 {} 类表示代码的逻辑结构。

2.54 Python 迭代器

- 迭代器对象
- 可迭代对象

```
from collections import Iterator
from collections import Iterable

print(help(Iterator))
print(help(Iterable))
```

```
list1 = range(5)
iter1 = iter(list1)
print(list1.__next__)
print(type(list1))
print(type(iter1))
```

2.55 Python 知识体系：思维导图

https://www.processon.com/view/link/5e8c1a04e4b0bf3ebcfd2dcc

2.56 Python 模块导入机制

sys.path

sys.modules

reload

2.57 Python 模块名称空间

2.58 Python 命名规则

编程的时候遇到变量、函数、模块以及类的时候都要给它们起名字，起名字不能张三李四这样随便乱起，否则会导致代码的阅读性降低，后期维护变得困难。命名一般遵循以下几条规则：

- 命名要具有描述性。
- 必须以字母或下划线开头，且只能是下划线、字母和数字的组合。

- 不能使用 Python 的关键字，也称保留字。
- 命名是区分大小写的。
- 模块中以下划线开头的名字是模块私有的。
- 以双下划线开头的类成员名字是类私有的。
- 同时以双下划线开头和结尾的名字，叫做特殊方法，也叫双下方法或魔术方法。
- 推荐使用驼峰命名法或下划线命名法。
- 类命名，首字母大写，尽量使用驼峰命名法。
- 驼峰命名法 VS 下划线命名法

```
myFirstVarable = 100
my_first_varable = 100
```

- 关键字

关键字也称保留字，不能用于命名。

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nolcoal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	break
except	in	raise		

2.59 Python 运算符

- 算术运算符
- 比较运算符
- 赋值运算符
- 逻辑运算符
- 位运算符
- 成员运算符
- 身份运算符

Python 算术运算符

+	加 - 两个对象相加	2 + 3 输出结果 5
-	减 - 得到负数或是一个数减去另一个数	2 - 3 输出结果 -1
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	2 * 3 输出结果 6
/	除 - x 除以 y	3 / 2 输出结果 1
%	取模 - 返回除法的余数	3 % 2 输出结果 1
**	幂 - 返回 x 的 y 次幂	2 ** 3 输出结果 8
//	取整除 - 返回商的整数部分（向下取整）	3 // 2 输出结果 1

Python 比较（关系）运算符

==	等于 - 比较对象是否相等	(2 == 3) 返回 False
!=	不等于 - 比较两个对象是否不相等	(2 != 3) 返回 true
<>	不等于 - 比较两个对象是否不相等	(2 <> 3) 返回 true
>	大于 - 返回 x 是否大于 y	(2 > 3) 返回 False
<	小于 - 返回 x 是否小于 y	(2 < 3) 返回 true
>=	大于等于 - 返回 x 是否大于等于 y	(2 >= 3) 返回 False
<=	小于等于 - 返回 x 是否小于等于 y	(2 <= 3) 返回 true

Python 赋值运算符

=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a

Python 位运算符

&	按位与运算符	(60 & 13) 输出结果 12，二进制解释：0000 1100
|	按位或运算符	(60 | 13) 输出结果 61，二进制解释：0011 1101
^	按位异或运算符	(60 ^ 13) 输出结果 49，二进制解释：0011 0001
~	按位取反运算符	(~60) 输出结果 -61，二进制解释：1100 0011
<<	左移动运算符	60 << 2 输出结果 240，二进制解释：1111 0000
>>	右移动运算符	60 >> 2 输出结果 15，二进制解释：0000 1111

Python 逻辑运算符

and	x and y	布尔”与”	(10 and 20) 返回 20
or	x or y	布尔”或”	(10 or 20) 返回 10
not	not x	布尔”非”	not(10 and 20) 返回 False

Python 成员运算符

in	如果在指定的序列中找到值返回 True，否则返回 False。	x in y
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x not in y

Python 身份运算符

is	is 是判断两个标识符是不是引用自一个对象	x is y，类似 id(x) == id(y)
is not	is not 是判断两个标识符是不是引用自不同对象	x is not y，类似 id(x) != id(y)

2.60 Python PEP8

2.61 Python 文件后缀

- .py
- .pyc
- .pyo
- .pyd
- .pyz(w)
- .exe

2.62 Python 递归函数

所谓递归就是在函数内部调用自己的函数被称之为递归，一般递归函数都有终止条件，采用的是后入先出的压栈数据结构。

阶乘函数 n!

```
def factorial(n):

    if n == 1:
        return 1

    return n * factorial(n - 1)
```

斐波那契数列 1, 1, 2, 3, 5, 8...

```
def fibonacci(n):

    if n < 2:
        return n

    return fibonacci(n - 2) + fibonacci(n - 1)
```

从递归函数谈

```
os.walk()
os.listdir()
os.mkdir()
os.makedirs()
os.rmdir()
os.removedirs()
copy.copy()
copy.deepcopy()
```

深浅拷贝

从递归函数谈 reduce

os.listdir 递归查找文件

children 递归查找 filecache

斐波拉契数列 1 1 2 3 5 8

```
def fibonacci(num):
    if isinstance(num, int):
        if num <= 0:
            return 0
        elif num == 1:
            return 1
        else:
            return fibonacci(num - 1) + fibonacci(num - 2)
    else:
        raise ValueError("The input number error!")

def factorial(number):

    if not isinstance(number, int) or number <= 0:
        return
```

(下页继续)

(续上页)

```
    result = 1

    for n in range(1, number + 1)
        result *= n

    return result

factorial(100)

def fact(number):
    if number == 1:
        return number
    return number * fact(number - 1)
```

```
# os.walk()
import os
from pprint import pprint

def walkFolders(folderPath):
    subFileList = []
    for subFileName in os.listdir(folderPath):
        subPath = os.path.join(folderPath, subFileName)
        if os.path.isfile(subPath):
            subFile = {"type": "file", "name": subFileName}
            subFileList.append(subFile)
        else:
            subFolder = {"type": "folder", "name": subFileName, "subFiles":␣
↪walkFolders(subPath)}
            subFileList.append(subFolder)
    return subFileList
```

2.63 Python 正则表达式

正则表达式（Regular Expression）使用单个字符串来描述、匹配一系列符合某个句法规则的字符串。在很多文本编辑器里，正则表达式通常被用来检索、替换那些符合某个模式的文本。

正则表达式规则中特殊符号的作用.^$*+?{}[]|()。

- \d 数字 [0-9]
- \w 字母或数字 a-z A-Z 0-9 _

- \s 空白字符 space
- \D \W \S 与小写相反
- . 任意字符
- | 或
- ^ 非，或者开始位置标记
- $ 结束位置标记
- \b 单词边界
- \B 非单词边界

匹配变长的字符，贪婪匹配。

- ? 表示 0 - 1 次 ab?c 匹配 abc 或者 ac
- + 表示 1 - 无数次 ab+c 匹配 abc abbc abbbbbbc
- * 表示 0 - 无数次 ab*c 匹配 ac abc abbbbbbc
- {m, n} 表示匹配 m 到 n 次 ab{2, 5}c 匹配 abbbc abbbbbbbc
- {m} 表示只匹配 m 次
- {m,} 表示匹配最少 m 次
- {, n} 表示匹配最多 n 次

Python 中提供内置模块 re 处理正则表达式。

```
>>> import re
>>> type(re)
<type 'module'>
>>> re
<module 're' from 'C:\Python27\lib\re.pyc'>
>>> dir(re)
['DEBUG', 'DOTALL', 'I', 'IGNORECASE', 'L', 'LOCALE', 'M', 'MULTILINE', 'S', 'Scanner',
↪'T', 'TEMPLATE', 'U', 'UNICODE', 'VERBOSE', 'X', '_MAXCACHE', '__all__', '__builtins__
↪', '__doc__', '__file__', '__name__', '__package__', '__version__', '_alphanum', '_
↪cache', '_cache_repl', '_compile', '_compile_repl', '_expand', '_locale', '_pattern_
↪type', '_pickle', '_subx', 'compile', 'copy_reg', 'error', 'escape', 'findall',
↪'finditer', 'match', 'purge', 'search', 'split', 'sre_compile', 'sre_parse', 'sub',
↪'subn', 'sys', 'template']
>>>
```

- re.compile # 生成一个匹配器实例，用来匹配，不用重复编译，优化代码执行时间
- re.findall # 找到全部匹配字符串，以列表返回

- re.finditer # 找到全部匹配实例对象，以迭代器返回
- re.match # 匹配字符串开始位置
- re.search # 扫描字符串，找到第一个位置
- re.split # 根据规则切分字符串
- re.sub # 查询替换字符串，返回替换结果
- re.subn # 查询替换字符串，返回替换结果和替换次数

re.match、re.search、re.finditer 返回匹配结果的实例对象，实例对象有如下方法

- instance.group() # 返回匹配的完整字符串
- instance.start() # 匹配的开始位置
- instance.end() # 匹配的结束位置
- instance.span() # 包含起始、结束位置的元组
- instance.groups() # 返回分组信息
- instance.groupdict() # 返回命名分组信息

```
import re

s = "12abc345ab"
m = re.match(r"\d+", s)
print(dir(m))
print(m.group())
print(m.span())

m = re.match(r"\d{3,}", s)
print(m is None)
m = re.search(r"\d{3,}", s)
print(m.group())
print(m.span())

m = re.search(r"\d+", s)
print(m.group())
print(m.span())

ms = re.findall(r"\d+", s)
print(ms)
ms = re.findall(r"\d{5}", s)
print(ms)
```

(下页继续)

(续上页)

```
for m in re.finditer(r"\d+", s):
    print(m.group())
    print(m.span())

for m in re.finditer(r"\d{5}", s):
    print(m.group())
    print(m.span())

m = re.match(r"(\d+)(?P<letter>[abc]+)", s)
print(m.group())
print(m.start())
print(m.end())
print(m.span())
print(m.groups())
print(m.groupdict())
print(m.group(0))
print(m.group(1))
print(m.group(2))
print(m.group(1, 2))
print(m.group(0, 1, 2))
print(m.start(0), m.end(0))
print(m.start(1), m.end(1))
print(m.start(2), m.end(2))
print(m.span(0))
print(m.span(1))
print(m.span(2))

res = re.search("ch_", "proj_ch_dog")
res = re.search("d[aoes]g", "dog dag deg dsg")
res = re.search("d[a-zA-Z0-9]g", "dog dag deg dsg")

res = re.search(r"a\wt", "adfaa_tfagdga")
res = re.search("1[357]\d{9}", "phone number: 13851709904")
```

```
res = re.search("SSNI-\d{3}\.avi", "this is budsfaf SSNI-518.avi dfaganiojfoas")

if res:
    print(res.group(), res.start(), res.end())
else:
```

(下页继续)

(续上页)

```
    print("Not found!")

res = re.findall("[a-zA-Z]+\.html", sss)
```

() 表达式编组，括号里面的表达式作为一个整体逻辑

- (?P<name>)
- (?=name)
- (mov|exr)
- (?=abc) 判断字符后面包含 abc
- (?!abc) 判断字符后面不包含 abc
- (?<=abc) 判断字符前面包含 abc
- (?<!abc) 判断字符前面不包含 abc
- (?#…) 注释

```
match = re.search("(?P<id>\d+), Date: (?P<date>.+)", "ID: 021523, Date: Feb/12/2017")
print(match.group("id"))
print(match.group("date"))

res = re.search("\w+(?=.jpg)", "image.jpg")

re.split(r"\W", "abc,123,x")
re.split(r"(\W)", "abc,123,x")
re.sub(r"[a-z]+", "abc,123,x")
re.sub(r"[a-z]+", "abc,123,x", 1)
re.subn(r"[a-z]+", "*", "abc,123,x")
```

编译标志 (?iLmsux), 可以用 re.I、re.M 等参数，也可以直接在表达式中添加 (?)

- s 单行
- i 忽略大小写
- L 让 w 匹配本地字符，对中文支持不好
- m 多行
- x 忽略多余的空白字符
- u unicode

(?i) 忽略大小写

组操作，小括号即组，分组的概念

```
>>> import re
>>>
>>> s = "%123Abc%45xyz&"
>>> s
'%123Abc%45xyz&'
>>>
>>> re.findall(r"(\d+)(\w+)", s)
[('123', 'Abc'), ('45', 'xyz')]
>>> re.findall(r"(\d+(\w+))", s)
[('123Abc', 'Abc'), ('45xyz', 'xyz')]
>>>
```

命名组 (?P<name>…)

```
>>> import re
>>>
>>> for m in re.finditer(r"(?P<number>\d+)(?P<letter>[a-z]+)", "%123Abc%45xyz&", re.I):
...     print(m.group())
...     print(m.groupdict())
...
123Abc
{'number': '123', 'letter': 'Abc'}
45xyz
{'number': '45', 'letter': 'xyz'}
>>>
```

无捕获组 (?:…)，作为匹配条件，匹配的对象在无名的组中，无需关心

```
>>> import re
>>>
>>> s = "%123Abc%45xyz&"
>>> s
'%123Abc%45xyz&'
>>> print(re.findall(r"(?:\d+)([a-z]+)", s))
['xyz']
>>> print(re.findall(r"(?:\d+)([a-z]+)", s, re.I))
['Abc', 'xyz']
>>>
```

反向引用，通过 <number> 或 (?P=name)，引用前面的组

```
>>> import re
>>>
>>> for m in re.finditer(r"<a>\w+</a>", "%<a>123Abc</a>%<b>45xyz</b>&"):
...     print(m.group())
...
<a>123Abc</a>

for m in re.finditer(r"<\w>\w+</(\1)>", "%<a>123Abc</a>%<b>45xyz</b>&"):
    print(m.group())

>>> for m in re.finditer(r"<(?P<tag>\w)>\w+</(?P=tag)>", "%<a>123Abc</a>%<b>45xyz</b>&"):
...     print(m.group())
...
<a>123Abc</a>
<b>45xyz</b>
```

- (?=…) # 组内容必须出现在右侧
- (?!…) # 组内容不能出现在右侧
- (?<=…) # 组内容必须出现在左侧
- (?<!…) # 组内容不能出现在左侧

re.split 用 pattern 做分隔符切割字符串，如果用 (pattern)，分隔符也会返回

```
>>> import re
>>>
>>> s = "abc,123,x"
>>> print(re.split(r"\W", s))
['abc', '123', 'x']
>>> print(re.split(r"(\W)", s))
['abc', ',', '123', ',', 'x']
>>>
```

re.sub 替换查找到的字符串，返回新的字符串，可指定替换次数

```
>>> import re
>>>
>>> s = "abc,123,x"
>>> print(re.sub(r"[a-z]+", "*", s))
*,123,*
>>> print(re.sub(r"[a-z]+", "*", s, 1))
```

(下页继续)

(续上页)

```
*,123,x
>>>
```

re.subn 和 re.sub 用法一样，只是返回值不同，返回 (新的字符串，被替换的次数)

```
>>> import re
>>>
>>> s = "abc,123,x"
>>> s
'abc,123,x'
>>> print(re.subn(r"[a-z]+", "*", s))
('*,123,*', 2)
>>> print(re.subn(r"[a-z]+", "*", s, 1))
('*,123,x', 1)
>>>
```

用来替换的参数 repl 可以接受自定义函数

```
>>> import re
>>>
>>> def repl(m):
...     print(m.group())
...     return "*" * len(m.group())
...
>>> s = "abc,123,x"
>>>
>>> print(re.sub(r"[a-z]+", repl, s))
abc
x
***,123,*
>>>
>>> print(re.sub(r"[a-z]+", repl, s, 1))
abc
***,123,x
>>>
>>> print(re.subn(r"[a-z]+", repl, s))
abc
x
('***,123,*', 2)
>>> print(re.subn(r"[a-z]+", repl, s, 1))
abc
```

(下页继续)

(续上页)

```
('***,123,x', 1)
>>>
```

结合匿名函数 lambda 可以更简洁

```
>>> import re
>>>
>>> s = "abc,123,x"
>>> print(re.sub(r"[a-z]+", lambda x: "*" * len(x.group()), s))
***,123,*
>>>
>>> s = "magicFireNezhaSpirit"
>>> print(re.sub(r"[A-Z]+", lambda x: "_" + x.group().lower(), s))
magic_fire_nezha_spirit
>>>
```

```
### Various Sorting methods for lists and dicts
import re
import random
import calendar
from pprint import pprint
files = ['tank_1_color_v0.rat',
        'tank_2_color_v5.rat',
        'tank_1_color_v3.rat',
        'tank_3_color_v1.rat',
        'tank_4_color_v2.rat',
        'tank_4_color_v4.rat',
        'tank_5_color_v1.rat',
        'tank_6_color_v6.rat']

pat_num = re.compile('\D+_(\d+)_')
pat_ver = re.compile('(\d+)\D+$')

def sorter_num(elem):
    res = re.search(pat_num, elem)
    return res.groups()[0]

def sorter_ver(elem):
    res = re.search(pat_ver, elem)
    return res.groups()[0]
```

(下页继续)

(续上页)

```
# pprint(sorted(files, key=sorter_num))
# pprint(sorted(files, key=sorter_ver))

s2 = "February January  May October August September April  November July March December"

d = {}
for i in range(1, 13):
    d[calendar.month_name[i]] = i
def sorter(elem):
    return d[elem]
# month_names = sorted(s2.split(), key=sorter)

month_names = [calendar.month_name[i] for i in range(1, 13)]

pprint(sorted(s2.split(), key=month_names.index))
```

2.64 Python 特殊方法: __getitem__ & __setitem__

2.65 Python 特殊方法: __repr__

```
import random

class Point(object):
    def __init__(self, dist_from_origin, weight):
        self.dist = dist_from_origin
        self.weight = weight

    def __repr__(self):
        return "Point dist: %f; weight: %f" % (self.dist, self.weight)

def getPointList(numPoints, maxDist):
    l = []
    n = 0

    while n <= numPoints:
        dist = random.random()
        random.seed()
```

(下页继续)

(续上页)

```
        weight = random.uniform(0.01, 0.1)

        if dist <= maxDist:
            l.append(Point(dist, weight))
            n += 1

    return l

print(getPointList(100, 0.17))
```

```
import random

class Point(object):
    def __init__(self, dist_from_origin, weight):
        self.dist = dist_from_origin
        self.weight = weight

    def __repr__(self):
        return "Point dist: %f; weight: %f" % (self.dist, self.weight)

@measure_time
def getPointList(numPoints, maxDist):
    l = []
    n = 0

    while n <= numPoints:
        dist = random.random()
        random.seed()
        weight = random.uniform(0.01, 0.1)

        if dist <= maxDist:
            l.append(Point(dist, weight))
            n += 1

    return l

@measure_time
def pointGenerator(numPoints, maxDist):
    n = 0
```

(下页继续)

(续上页)

```
    while n <= numPoints:
        dist = random.random()
        random.seed()
        weight = random.uniform(0.01, 0.1)

        if dist <= maxDist:
            n += 1
            yield Point(dist, weight)

print("List: ", getPointList(1000, 0.17))
print("Generator: ", pointGenerator(1000, 0.17))
```

2.66 Python 特殊属性：___dict___

___dict___ 是模块对象的 globals 名称空间。

2.67 Python 特殊属性：___doc___

___doc___ 是模块对象的特殊属性，存储模块文件的 docstring，docstring 实际就是代码的解释文档，如果按规范写成，在使用 help() 的时候就可以获取到模块、函数抑或类的具体帮助文档，甚至可以通过第三方模块直接将代码转成 API 帮助文档。

2.68 Python 特殊属性：___file___

___file___ 是模块对象的特殊属性，它会返回模块文件的完整路径，如果模块是包的话，则返回包的___init___.py 的完整路径。这在工作中非常实用，比如我自定义了一个命名污染的模块 platform.py，在使用的时候一直报错，就可以通过代码查找此模块的路径，就会发现系统已经存在这个模块，是因为导入的不是我写的这个模块导致的错误，很多时候很方便检查代码在什么地方。

```
>>> import platform
>>> platform.__file__
'C:\\Python27\\lib\\platform.pyc'
```

2.69 Python 特殊属性：___name___

___name___ 是模块对象的特殊属性，模块文件在直接执行的时候等于” ___main___”，而在被导入的时候则为模块的名称。所以一般可以在模块中将单元测试代码放在下面的 if 语句块中。

```
if __name__ == "__main__":
    code block
```

正常来说模块文件都不是直接执行，而是通过模块导入名称空间的方式调用模块中具体变量，函数以及类从而达到使用模块的目的，也就是说上面的代码块在导入模块的时候将不起作用，只有在直接执行的时候才会运行，所以常用来作为单个模块单元测试使用。

在导入模块的时候 ___name___ 将会作为 key 的方式存储在 sys.modules 字典中。

2.70 Python 特殊属性：___path___

___path___ 是模块包的特殊属性，它会返回模块包的文件夹完整路径。

```
>>> import PySide
>>> PySide.__file__
'C:\\Python27\\lib\\site-packages\\PySide\\__init__.pyc'
>>> PySide.__path__
['C:\\Python27\\lib\\site-packages\\PySide']
```

2.71 Python 第三方模块：Pillow

处理图片

- https://pillow.readthedocs.io/en/stable/

2.72 Python 第三方模块：python-fire

2.73 Python 第三方模块：yaml

yaml 模块和 json 模块有异曲同工之妙

- yaml.load
- yaml.loads
- yaml.dump

- yaml.dumps

2.74 Python 类型实例概念

Python 中万物皆对象，但是得先有类型，才能生成实例对象。我们操作某一个实例的时候，首先得分析它是什么类型，可以通过 type() 来获取对象的类型，然后分析它有什么属性以及方法，可以通过 dir() 来获取对象的属性和方法，再然后可以通过 help() 来获取具体属性与方法的帮助文档。

```
>>> s1 = "Hello world!"
>>> s1
'Hello world!'
>>> type(s1)
<type 'str'>
>>> dir(s1)
['__add__', '__class__', '__contains__', '__delattr__', '__doc__', '__eq__', '__format__
↪', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getslice__', '__
↪gt__', '__hash__', '__init__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__
↪ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '_
↪_setattr__', '__sizeof__', '__str__', '__subclasshook__', '_formatter_field_name_split
↪', '_formatter_parser', 'capitalize', 'center', 'count', 'decode', 'encode', 'endswith
↪', 'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha', 'isdigit', 'islower',
↪ 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'partition',
↪'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
↪'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill
↪']
>>> help(s1.split)
Help on built-in function split:

split(...)
    S.split([sep [,maxsplit]]) -> list of strings

    Return a list of the words in the string S, using sep as the
    delimiter string.  If maxsplit is given, at most maxsplit
    splits are done. If sep is not specified or is None, any
    whitespace string is a separator and empty strings are removed
    from the result.
>>> s1.split()
['Hello', 'world!']
```

可以看出 type()、dir() 以及 help() 乃是一套通用心法，无论你得到一个什么实例对象，你都可以通过此三个内置函数来获取对于这个对象的初步的文档以及使用方法。

2.75 Python 变量引用

- 变量命名规则
  - 命名要具有描述性。
  - 必须以字母或下划线开头，且只能是下划线、字母和数字的组合。
  - 不能使用 Python 的关键字，也称保留字。
  - 命名是区分大小写的。
- 赋值运算符

在编程语言中，一个等于号是赋值运算符，两个等于号才是比较运算符。

```
>>> a = 100
>>> a
100
>>> b = 100
>>> b
100
>>> a == b
True
>>> a is b
True
>>> a = 257
>>> b = 257
>>> a == b
True
>>> a is b
False
>>>
```

- 变量交换

```
>>> a = 10
>>> b = 20
>>> a
10
>>> b
20
>>> a, b = b, a
>>> a
20
```

(下页继续)

(续上页)

```
>>> b
10
>>>
```

CHAPTER 3

PyQt

PyQt 是 Python 语言的 GUI 编程解决方案之一。可以用来代替 Python 内置的 Tkinter。其它替代者还有 PyGTK、wxPython 等。与 Qt 一样，PyQt 是一个自由软件。

Contents:

3.1 PyQt 使用 bat 伪装应用程序

3.2 PyQt 自定义控件玩法

自定义控件，事件机制，如何让 UI 元素更丰富!

```
from PySide2 import QtWidgets

class QLineEditPath(QtWidgets.QLineEdit):

    def __init__(self, parent=None):
        super(QLineEditPath, self).__init__(parent)

    def dragEnterEvent(self, event):

        if event.mimeData().hasUrls():
            event.accept()
```

(下页继续)

(续上页)

```
        else:
            event.ignore()

    def dropEvent(self, event):

        if event.mimeData().hasUrls():
            url = event.mimeData().urls()[0]
            self.setText(url.toLocalFile())

dialog = QLineEditPath()
dialog.show()
```

```
from PySide2 import QtCore
from PySide2 import QtWidgets

class QLineEditPath(QtWidgets.QLineEdit):

    def __init__(self, parent=None):
        super(QLineEditPath, self).__init__(parent)

    def keyPressEvent(self, event):
        if event.key() == QtCore.Qt.Key_Return:
            self.setText(self.text().replace("\\", "/"))
        else:
            QtWidgets.QLineEdit.keyPressEvent(self, event)

dialog = QLineEditPath()
dialog.show()
```

```
class QPublishedFileItem(QtWidgets.QWidget):

    def __init__(self, file_name, file_type, parent=None):
        super(QPublishedFileItem, self).__init__(parent)

        vblFile = QtWidgets.QVBoxLayout()

        lPublishedFileName = QtWidgets.QLabel(file_name)
        lPublishedFileType = QtWidgets.QLabel(file_type)
        # lPublishedFileName = QtWidgets.QLabel("<span style='font-weight: bold;'>lgt_
↪autosphere_02.v007.nk</span> Version 007 (Task lgt_autosphere_02)")
```

(下页继续)

(续上页)

```
        # lPublishedFileType = QtWidgets.QLabel("<span style='font-size: 10px;'><span␣
↪style='color: #2e93e2;'>Nuke Script</span> by xu tao at 2020-02-05 17:40</span>")
        vblFile.addWidget(lPublishedFileName)
        vblFile.addWidget(lPublishedFileType)

        hblAction = QtWidgets.QHBoxLayout()
        self.pbAction = QtWidgets.QPushButton("Action")
        self.pbAction.hide()
        self.mAction = QtWidgets.QMenu()
        self.pbAction.setMenu(self.mAction)
        hblAction.addLayout(vblFile)
        spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,␣
↪QtWidgets.QSizePolicy.Minimum)
        hblAction.addItem(spacerItem)
        hblAction.addWidget(self.pbAction)

        vblPublishedFile = QtWidgets.QVBoxLayout()
        # vblPublishedFile.addLayout(vblThumbnail)
        vblPublishedFile.addLayout(hblAction)
        self.setLayout(vblPublishedFile)

fileName = "<span style='font-weight: bold;'>lgt_autosphere_02.v007.nk</span> Version␣
↪007 (Task lgt_autosphere_02)"
fileType = "<span style='font-size: 10px;'><span style='color: #2e93e2;'>Nuke Script</
↪span> by xu tao at 2020-02-05 17:40</span>"
dialog = QPublishedFileItem(fileName, fileType)
dialog.show()
```

3.3 PyQt Designer 安装与使用

了解 PyQt 的历史以及协议，Qt 实际是 C++ 的一套 GUI 编程的套件，底层就是 C++ Qt，Python 将其做了封装之后才有了 PyQt 的 UI 库。

- Qt4->PyQt4->PySide
- Qt5->PyQt5->PySide2

PySide 和 PyQt 实际是一回事，只是开源协议有所不同，从我使用的角度来说有些高级功能可能不太一样，比如数据库的支持，比如 uic 的支持，最新版本的 Houdini、Maya 与 Nuke 中内置 PySide2，以前老版本是内置 PySide，内置使用起来就比较方便，当然你非得使用 PyQt5，就得基于具体的软件的 Python 环境去编译一套 PyQt 库来使用。

PyQt5 和 PySide2 可以直接通过 pip 安装即可，如果有特殊版本得费点心思，一般安装好 PySide2，你就可以找到这个 C:Python37Libsite-packagesPySide2designer.exe

PySide2 库的层级结构

- QtCore 核心算法库
- QtGui 界面图形算法库
- QtWidgets 界面控件库
- uic ui 文件到 py 文件的桥梁
- sip C++ 对象到 Python 转换库

QtCore 包含了核心的非 GUI 的功能。主要和时间、文件与文件夹、各种数据、流、URLs、mime 类文件、进程与线程一起使用。

QtGui 包含了窗口系统、事件处理、2D 图像、基本绘画、字体和文字类。

QtWidgets 类包含了一系列创建桌面应用的 UI 元素。QtMultimedia 包含了处理多媒体的内容和调用摄像头 API 的类。QtBluetooth 模块包含了查找和连接蓝牙的类。QtNetwork 包含了网络编程的类，这些工具能让 TCP/IP 和 UDP 开发变得更加方便和可靠。QtPositioning 包含了定位的类，可以使用卫星、WiFi 甚至文本。Engine 包含了通过客户端进入和管理 Qt Cloud 的类。QtWebSockets 包含了 WebSocket 协议的类。QtWebKit 包含了一个基 WebKit2 的 web 浏览器。QtWebKitWidgets 包含了基于 QtWidgets 的 WebKit1 的类。QtXml 包含了处理 xml 的类，提供了 SAX 和 DOM API 的工具。QtSvg 提供了显示 SVG 内容的类，Scalable Vector Graphics (SVG) 是一种是一种基于可扩展标记语言（XML），用于描述二维矢量图形的图形格式（这句话来自于维基百科）。QtSql 提供了处理数据库的工具。QtTest 提供了测试 PyQt5 应用的工具。

一般 PyQt 的代码分手写与 Designer 设计，Designer 设计出来的.ui 文件又分两种使用方法，要么编译成.py 文件去使用，要么直接通过方法 load 这个.ui 文件来使用。按 Python 环境来验证哪种更方便。

```
from PySide2 import QtGui
from PySide2 import QtCore
from PySide2 import QtWidgets

class CustomDialog(QtWidgets.QDialog):
    pass

if __name__ == "__main__":
    dialog = CustomDialog()
    dialog.show()
```

```
import sys

from PySide2 import QtGui
from PySide2 import QtCore
```

(下页继续)

(续上页)

```
from PySide2 import QtWidgets

class CustomDialog(QtWidgets.QDialog):
    pass

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    dialog = CustomDialog()
    dialog.show()
    print(dir(app))
    app.exec_()
```

```
import sys

from PySide2 import QtGui
from PySide2 import QtCore
from PySide2 import QtWidgets

class CustomDialog(QtWidgets.QDialog):
    def __init__(self):
        super().__init__()

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    dialog = CustomDialog()
    dialog.show()
    print(dir(app))
    app.exec_()
```

手写代码

```
from PySide2 import QtGui
from PySide2 import QtCore
from PySide2 import QtWidgets

class CustomDialog(QtWidgets.QDialog):
    def __init__(self, parent=None):
        # super().__init__(parent)
        super(CustomDialog, self).__init__(parent)
```

(下页继续)

(续上页)

```
        # mainLayout = QtWidgets.QVBoxLayout()
        # self.setLayout(mainLayout)
        vblLayout = QtWidgets.QVBoxLayout(self)

        gbSim = QtWidgets.QGroupBox()
        gbSim.setTitle("Sim or Seq:")
        hblLayout = QtWidgets.QHBoxLayout(gbSim)
        rbSim = QtWidgets.QRadioButton("Simulation")
        rbSeq = QtWidgets.QRadioButton("Sequence")

        hblLayout.addWidget(rbSim)
        hblLayout.addWidget(rbSeq)

        pbSubmit = QtWidgets.QPushButton("Submit")
        vblLayout.addWidget(gbSim)
        vblLayout.addWidget(pbSubmit)

if __name__ == "__main__":
    dialog = CustomDialog()
    dialog.show()
```

PySide2 加载.ui 文件模式

```
from PySide2 import QtUiTools
from PySide2 import QtWidgets
from PySide2 import QtGui
from PySide2 import QtCore

class CustomDialog(QtWidgets.QDialog):
    def __init__(self, parent=None):
        super(CustomDialog, self).__init__(parent)

        loader = QtUiTools.QUiLoader()

        self.ui = loader.load(r"D:\2020\test.ui")

        mainLayout = QtWidgets.QVBoxLayout()
        mainLayout.setContentsMargins(0, 0, 0, 0)
        mainLayout.addWidget(self.ui)
        self.setLayout(mainLayout)
```

(下页继续)

(续上页)

```
dialog = CustomDialog()
dialog.show()
```

转 py 文件

```
# C:\Python37\Scripts\pyside2-uic.exe -o D:\2020\mainWin.py D:\2020\test.ui

import sys

path = "D:/2020"

path in sys.path or sys.path.insert(0, path)

from PySide2 import QtWidgets
from PySide2 import QtGui
from PySide2 import QtCore
import mainWin
reload(mainWin)

class CustomDialog(QtWidgets.QDialog, mainWin.Ui_Dialog):
    def __init__(self, parent=None):
        super(CustomDialog, self).__init__(parent)

        self.setupUi(self)

dialog = CustomDialog()
dialog.show()
```

信号与槽（事件）

```
import sys
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import (QWidget, QLCDNumber, QSlider,
    QVBoxLayout, QApplication)


class Example(QWidget):

    def __init__(self):
        super().__init__()
```

(下页继续)

(续上页)

```
        self.initUI()


    def initUI(self):

        lcd = QLCDNumber(self)
        sld = QSlider(Qt.Horizontal, self)

        vbox = QVBoxLayout()
        vbox.addWidget(lcd)
        vbox.addWidget(sld)

        self.setLayout(vbox)
        sld.valueChanged.connect(lcd.display)

        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('Signal and slot')
        self.show()


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

参考文档:

- https://www.cnblogs.com/jakeyChen/articles/4103494.html
- https://wiki.python.org/moin/PyQt4
- https://www.riverbankcomputing.com/static/Docs/PyQt4/classes.html
- https://doc.qt.io/qtforpython/PySide2/QtWidgets/QPushButton.html
- https://www.riverbankcomputing.com/static/Docs/PyQt4/classes.html
- https://doc.bccnsoft.com/docs/PyQt5/class_reference.html
- https://build-system.fman.io/qt-designer-download
- http://zetcode.com/gui/pyqt4/

3.4 PyQt 实现 Maya 中 frameLayout 布局功能

```
import maya.cmds as cmds

cmds.window()
cmds.scrollLayout( 'scrollLayout' )
cmds.columnLayout( adjustableColumn=True )
cmds.frameLayout( label='Buttons', collapsable=1)
cmds.columnLayout()
cmds.button()
cmds.button()
cmds.button()
cmds.setParent( '..' )
cmds.setParent( '..' )
cmds.frameLayout( label='Scroll Bars', collapsable=1)
cmds.columnLayout()
cmds.intSlider()
cmds.intSlider()
cmds.intSlider()
cmds.setParent( '..' )
cmds.setParent( '..' )
cmds.frameLayout( label='Fields', collapsable=1)
cmds.columnLayout()
cmds.intField()
cmds.intField()
cmds.intField()
cmds.setParent( '..' )
cmds.setParent( '..' )
cmds.frameLayout( label='Check Boxes', collapsable=1)
cmds.columnLayout()
cmds.checkBox()
cmds.checkBox()
cmds.checkBox()
cmds.setParent( '..' )
cmds.setParent( '..' )
cmds.showWindow()
```

```
from PySide import QtCore
from PySide import QtGui

```

(下页继续)

(续上页)

```
class FrameWidget(QtGui.QGroupBox):
    def __init__(self, title='', parent=None):
        super(FrameWidget, self).__init__(title, parent)

        layout = QtGui.QVBoxLayout()
        layout.setContentsMargins(0, 7, 0, 0)
        layout.setSpacing(0)
        super(FrameWidget, self).setLayout(layout)

        self.__widget = QtGui.QFrame(parent)
        self.__widget.setFrameShape(QtGui.QFrame.Panel)
        self.__widget.setFrameShadow(QtGui.QFrame.Plain)
        self.__widget.setLineWidth(0)
        layout.addWidget(self.__widget)

        self.__collapsed = False

    def setLayout(self, layout):
        self.__widget.setLayout(layout)

    def expandCollapseRect(self):
        return QtCore.QRect(0, 0, self.width(), 20)

    def mouseReleaseEvent(self, event):
        if self.expandCollapseRect().contains(event.pos()):
            self.toggleCollapsed()
            event.accept()
        else:
            event.ignore()

    def toggleCollapsed(self):
        self.setCollapsed(not self.__collapsed)

    def setCollapsed(self, state=True):
        self.__collapsed = state

        if state:
            self.setMinimumHeight(20)
            self.setMaximumHeight(20)
            self.__widget.setVisible(False)
```

(下页继续)

(续上页)

```
        else:
            self.setMinimumHeight(0)
            self.setMaximumHeight(1000000)
            self.__widget.setVisible(True)

    def paintEvent(self, event):
        painter = QtGui.QPainter()
        painter.begin(self)

        font = painter.font()
        font.setBold(True)
        painter.setFont(font)

        x = self.rect().x()
        y = self.rect().y()
        w = self.rect().width()
        offset = 25

        painter.setRenderHint(painter.Antialiasing)
        painter.fillRect(self.expandCollapseRect(), QtGui.QColor(93, 93, 93))
        painter.drawText(
            x + offset, y + 3, w, 16,
            QtCore.Qt.AlignLeft | QtCore.Qt.AlignTop,
            self.title()
            )
        self.__drawTriangle(painter, x, y)#(1)
        painter.setRenderHint(QtGui.QPainter.Antialiasing, False)
        painter.end()

    def __drawTriangle(self, painter, x, y):#(2)
        if not self.__collapsed:#(3)
            points = [  QtCore.QPoint(x+10,  y+6 ),
                        QtCore.QPoint(x+20, y+6 ),
                        QtCore.QPoint(x+15, y+11)
                        ]

        else:
            points = [  QtCore.QPoint(x+10, y+4 ),
                        QtCore.QPoint(x+15, y+9 ),
                        QtCore.QPoint(x+10, y+14)
```

(下页继续)

(续上页)

```
                        ]

        currentBrush = painter.brush()#(4)
        currentPen   = painter.pen()

        painter.setBrush(
            QtGui.QBrush(
                QtGui.QColor(187, 187, 187),
                QtCore.Qt.SolidPattern
                )
            )#(5)
        painter.setPen(QtGui.QPen(QtCore.Qt.NoPen))#(6)
        painter.drawPolygon(QtGui.QPolygon(points))#(7)
        painter.setBrush(currentBrush)#(8)
        painter.setPen(currentPen)

window = QtGui.QMainWindow()
window.setWindowTitle('Frame Widget Test')

frame = FrameWidget('Frame Title', window)
window.setCentralWidget(frame)

widget = QtGui.QWidget(frame)
layout = QtGui.QVBoxLayout(widget)
frame.setLayout(layout)
for i in range(5):
    layout.addWidget(QtGui.QPushButton('Button %s' % i, widget))

window.show()
```

```
from PySide2 import QtGui
from PySide2 import QtCore
from PySide2 import QtWidgets

class FrameWidget(QtWidgets.QGroupBox):
    def __init__(self, title='', parent=None):
        super(FrameWidget, self).__init__(title, parent)

        layout = QtWidgets.QVBoxLayout()
```

(下页继续)

(续上页)

```
        layout.setContentsMargins(0, 7, 0, 0)
        layout.setSpacing(0)
        super(FrameWidget, self).setLayout(layout)

        self.__widget = QtWidgets.QFrame(parent)
        self.__widget.setFrameShape(QtWidgets.QFrame.Panel)
        self.__widget.setFrameShadow(QtWidgets.QFrame.Plain)
        self.__widget.setLineWidth(0)
        layout.addWidget(self.__widget)

        self.__collapsed = False

    def setLayout(self, layout):
        self.__widget.setLayout(layout)

    def expandCollapseRect(self):
        return QtCore.QRect(0, 0, self.width(), 20)

    def mouseReleaseEvent(self, event):
        if self.expandCollapseRect().contains(event.pos()):
            self.toggleCollapsed()
            event.accept()
        else:
            event.ignore()

    def toggleCollapsed(self):
        self.setCollapsed(not self.__collapsed)

    def setCollapsed(self, state=True):
        self.__collapsed = state

        if state:
            self.setMinimumHeight(20)
            self.setMaximumHeight(20)
            self.__widget.setVisible(False)
        else:
            self.setMinimumHeight(0)
            self.setMaximumHeight(1000000)
            self.__widget.setVisible(True)
```

(下页继续)

(续上页)

```
    def paintEvent(self, event):
        painter = QtGui.QPainter()
        painter.begin(self)

        font = painter.font()
        font.setBold(True)
        painter.setFont(font)

        x = self.rect().x()
        y = self.rect().y()
        w = self.rect().width()
        offset = 25

        painter.setRenderHint(painter.Antialiasing)
        painter.fillRect(self.expandCollapseRect(), QtGui.QColor(93, 93, 93))
        painter.drawText(
            x + offset, y + 3, w, 16,
            QtCore.Qt.AlignLeft | QtCore.Qt.AlignTop,
            self.title()
            )
        self.__drawTriangle(painter, x, y)#(1)
        painter.setRenderHint(QtGui.QPainter.Antialiasing, False)
        painter.end()

    def __drawTriangle(self, painter, x, y):#(2)
        if not self.__collapsed:#(3)
            points = [  QtCore.QPoint(x+10,  y+6 ),
                        QtCore.QPoint(x+20, y+6 ),
                        QtCore.QPoint(x+15, y+11)
                        ]

        else:
            points = [  QtCore.QPoint(x+10, y+4 ),
                        QtCore.QPoint(x+15, y+9 ),
                        QtCore.QPoint(x+10, y+14)
                        ]

        currentBrush = painter.brush()#(4)
        currentPen = painter.pen()
```

(下页继续)

(续上页)

```
        painter.setBrush(
            QtGui.QBrush(
                QtGui.QColor(187, 187, 187),
                QtCore.Qt.SolidPattern
                )
            )#(5)
        painter.setPen(QtGui.QPen(QtCore.Qt.NoPen))#(6)
        painter.drawPolygon(QtGui.QPolygon(points))#(7)
        painter.setBrush(currentBrush)#(8)
        painter.setPen(currentPen)

window = QtWidgets.QMainWindow()
window.setWindowTitle('Frame Widget Test')

frame = FrameWidget('Frame Title', window)
window.setCentralWidget(frame)

widget = QtWidgets.QWidget(frame)
layout = QtWidgets.QVBoxLayout(widget)
frame.setLayout(layout)
for i in range(5):
    layout.addWidget(QtWidgets.QPushButton('Button %s' % i, widget))

window.show()
```

- https://kiwamiden.com/make-mayas-framelayout-with-pyside

3.5 PyQt 在 Houdini 中执行的模板代码

PyQt 的历史 Dialog、Layout 与 Widget 之间的关系 setStyleSheet load

https://build-system.fman.io/qt-designer-download https://github.com/qt

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

"""

"""
```

(下页继续)

(续上页)

```
import os
import sys

import re



try:
    APP = "houdini"
    import hou

    from PySide2 import QtCore
    from PySide2 import QtWidgets as QtGui
except Exception as e:
    raise



class ActiveShotgun(QtGui.QDialog):
    """
    Initial main window
    """
    def __init__(self, parent=None):
        """
        """
        super(ActiveShotgun, self).__init__(parent)

        self._initUI()

    def _initUI(self):

        gbChoose = QtGui.QGroupBox("Choose Asset or Shot:")
        hblChoose = QtGui.QVBoxLayout()
        self.rbAsset = QtGui.QRadioButton("Asset")
        self.rbShot = QtGui.QRadioButton("Shot")
        self.rbShot.setChecked(True)
        hblChoose.addWidget(self.rbAsset)
        hblChoose.addWidget(self.rbShot)
        gbChoose.setFixedHeight(100)
        gbChoose.setLayout(hblChoose)

        # Initial labels
```

(下页继续)

(续上页)

```
        hlLabel = QtGui.QHBoxLayout()
        lProj = QtGui.QLabel("Proj:")
        lTask = QtGui.QLabel("Task:")
        hlLabel.addWidget(lProj)
        hlLabel.addWidget(lTask)

        # Initial choose asset or shot task
        hlListWidget = QtGui.QHBoxLayout()
        self.lwProj = QtGui.QListWidget()

        self.lwTask = QtGui.QListWidget()
        hlListWidget.addWidget(self.lwProj)
        hlListWidget.addWidget(self.lwTask)

        # Initial buttons
        hlActive = QtGui.QHBoxLayout()
        pbActive = QtGui.QPushButton("&Active Shotgun...")
        pbActive.setFixedWidth(120)
        pbActive.setFixedHeight(50)
        hlActive.addStretch(1)
        hlActive.addWidget(pbActive)

        v_box = QtGui.QVBoxLayout()
        v_box.addWidget(gbChoose)
        v_box.addLayout(hlLabel)
        v_box.addLayout(hlListWidget)
        v_box.addLayout(hlActive)

        self.setLayout(v_box)
        self.resize(QtCore.QSize(500, 600))
        self.setWindowTitle("Active Shotgun Options: Hello, %s")
        self.setWindowFlags(QtCore.Qt.WindowStaysOnTopHint)

    def closeEvent(self, event):
        self.done(0)


def showUI(func):
    """
    Show UI instance
```

(下页继续)

(续上页)

```
    """
    app = QtGui.QApplication.instance()

    if not app:
        app = QtGui.QApplication([APP])

    dialog = func()
    dialog.raise_()
    dialog.show()
    dialog.exec_()


def main():
    """
    main function
    """
    showUI(ActiveShotgun)

main()
```

```
import os
import hou

from hutil.Qt import QtCore, QtWidgets, QtUiTools
from commonQt.appQss import qss
path = os.path.dirname(__file__)
print(path)


class AttribManager(QtWidgets.QWidget):
    def __init__(self):
        QtWidgets.QWidget.__init__(self)
        print("Hey I am in a class!!!")

        # load UI file
        loader = QtUiTools.QUiLoader()
        self.ui = loader.load("D:/centralizeTools/houdini/scripts/python/houQt/untitled.
↪ui")
```

(下页继续)

(续上页)

```
        # Layout
        mainLayout = QtWidgets.QVBoxLayout()
        mainLayout.setContentsMargins(0, 0, 0, 0)
        mainLayout.addWidget(self.ui)
        self.setLayout(mainLayout)
        self.setStyleSheet(qss)


def show():
    dialog = AttribManager()
    dialog.setParent(hou.qt.floatingPanelWindow(None), QtCore.Qt.Window)
    dialog.show()


if __name__ == "hou.session":
    show()
```

将.ui 文件转.py 文件

```
C:\Python27\Scripts\pyside-uic.exe -o ???.py ???.ui
```

将.qrc 文件转.py 文件

```
C:\Python27\Lib\site-packages\PySide\pyside-rcc.exe -o ???.py ???.qrc
```

Python 代码将.ui 转.py 文件

```
import pyside2uic

with open("py 文件路径", "w") as f:
    pyside2uic.compileUi("ui 文件路径", f)
```

load ui

3.6 PyQt 在 Maya 中执行的模板代码

坐井观天：本节知识点如何用 PyQt 创建用户界面？如何删除已经存在的用户界面？如何给 button 连接函数传参？

管中窥豹：延伸阅读

```
from PySide2 import QtWidgets as QtGui

class MyDialog(QtGui.QDialog):
    def __init__(self, parent=None):
        super(MyDialog, self).__init__(parent)
        self._initUI()

    def _initUI(self):
        pass

if __name__ == "__main__":
    dialog = MyDialog()
    dialog.show()
```

```
import maya.cmds as cmds
from PySide2 import QtWidgets as QtGui

DIALOGNAME = "My Dialog"

class MyDialog(QtGui.QDialog):
    def __init__(self, parent=None):
        super(MyDialog, self).__init__(parent)
        self._initUI()

    def _initUI(self):
        self.setObjectName(DIALOGNAME)


if __name__ == "__main__":
    if cmds.window(DIALOGNAME, exists=True, query=True):
        cmds.deleteUI(DIALOGNAME)

    dialog = MyDialog()
    dialog.show()
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

try:
    from PySide import QtGui
```

(下页继续)

(续上页)

```
    from PySide import QtCore
except ImportError:
    from PySide2 import QtWidgets as QtGui

class Example(QtGui.QDialog):
    def __init__(self, parent=None):
        super(Example, self).__init__(parent)
        self.initUI()

    def initUI(self):
        self.setGeometry(600, 300, 500, 500)
        self.setWindowTitle("360 Playblast")
        Browse_Button = QtGui.QPushButton("Browse")
        name_label_file = QtGui.QLabel("Open file")
        self.name_line_edit_file = QtGui.QLineEdit()
        name_label_Frame = QtGui.QLabel("From")
        self.name_line_edit_frame = QtGui.QLineEdit()
        name_label_to = QtGui.QLabel("To")
        self.name_line_edit_to = QtGui.QLineEdit()
        name_label_Format = QtGui.QLabel("Format")
        self.ComboBox_format = QtGui.QComboBox()
        self.ComboBox_format.addItems(["avi", "image", "qt", "movie"])
        name_label_resolution = QtGui.QLabel("Resoultion")
        self.ComboBox_resolution = QtGui.QComboBox()
        self.ComboBox_resolution.addItems(["1920*1080", "1080*720", "720*540"])
        Browse_Button_to = QtGui.QPushButton("Browse")
        name_label_to1 = QtGui.QLabel("To")
        self.name_line_edit_file1 = QtGui.QLineEdit()
        name_button_OK = QtGui.QPushButton("OK")
        name_button_Canle = QtGui.QPushButton("Canle")
        baseLayout = QtGui.QGridLayout()
        baseLayout.addWidget(Browse_Button, 0, 2)
        baseLayout.addWidget(name_label_file, 0, 0)
        baseLayout.addWidget(self.name_line_edit_file, 0, 1)
        baseLayout.addWidget(name_label_Frame, 1, 0)
        baseLayout.addWidget(self.name_line_edit_frame, 1, 1)
        baseLayout.addWidget(name_label_to, 1, 2)
        baseLayout.addWidget(self.name_line_edit_to, 1, 3)
        baseLayout.addWidget(name_label_Format, 2, 0)
        baseLayout.addWidget(self.ComboBox_format, 2, 1)
```

(下页继续)

(续上页)

```
        baseLayout.addWidget(name_label_resolution, 3, 0)
        baseLayout.addWidget(self.ComboBox_resolution, 3, 1)
        baseLayout.addWidget(name_label_to1, 4, 0)
        baseLayout.addWidget(self.name_line_edit_file1, 4, 1)
        baseLayout.addWidget(Browse_Button_to, 4, 2)
        baseLayout.addWidget(name_button_OK, 5, 1)
        baseLayout.addWidget(name_button_Canle, 5, 3)
        self.setLayout(baseLayout)

if __name__ == "__main__":
    ex = Example()
    ex.show()
```

```
from PySide2 import QtGui
from PySide2 import QtCore
from PySide2 import QtWidgets
import maya.cmds as cmds

from PySide2.QtWidgets import *
# 命名污染

from shiboken2 import wrapInstance

import maya.OpenMayaUI as omui

def getMayaMainWin():
    pointer = omui.MQtUtil.mainWindow()
    return wrapInstance(long(pointer), QtWidgets.QWidget)

mayaWindow = getMayaMainWin()
mayaWindow.setWindowOpacity(1)

class MyWindow(QtWidgets.QDialog):
    def __init__(self, parent=getMayaMainWin()):
        super(MyWindow, self).__init__(parent)
        self.initUI()

    def initUI(self):
        self.setWindowTitle("My Custom Window")
```

(下页继续)

(续上页)

```
        self.setMinimumSize(QSize(360, 240))
        mainLayout = QtWidgets.QVBoxLayout()
        titleLabel = QLabel("Control Window")
        nameText = QLineEdit()
        execButton = QPushButton("Do It!")
        mainLayout.addWidget(titleLabel)
        mainLayout.addWidget(nameText)
        mainLayout.addWidget(execButton)
        self.setLayout(mainLayout)

        execButton.clicked.connect(self.createCube)

    def createCube(self):
        userInput = nameText.text()
        if not userInput:
            print("Please enter cube name!")
            return 0
        cubeName = cmds.polyCube()
        print("%s has been created" % cubeName)

if "myWin" in globals():
    print("myWin is exists, close it first!")
    myWin.deleteLater()
    del mywin

myWin = MyWindow()
myWin.show()
```

理解类继承查看 PyQt 文档字符串格式化两种方法理解类 self

判断语句

真假事件布尔运算 and or not bool()

PySide 与 PySide2 的兼容性

if int(cmds.about(v=True)) >= 2017: from PySide2 import QtCore from PySide2 import QtGui as QtGui4 from PySide2 import QtWidgets as QtGui from shiboken2 import wrapInstance

else: from PySide import QtGui from PySide import QtCore from shiboken import wrapInstance

loadUi

from functools import partial

def test(a, b, c): print(a, b, c)

f = partial(test, b=2, c=3) f(100) f = partial(test, 1, c=3) f(100)

3.7 PyQt 在 Nuke 中执行的模板代码

3.8 PyQt 重写 QLineEdit 支持拖拽功能

```
class QLineEditPath(QtGui.QLineEdit):

    def __init__(self, parent):
        super(QLineEditPath, self).__init__(parent)

    def dragEnterEvent(self, event):

        if event.mimeData().hasUrls():
            event.accept()
        else:
            event.ignore()

    def dropEvent(self, event):

        if event.mimeData().hasUrls():
            url = event.mimeData().urls()[0]
            self.setText(url.toLocalFile())
```

3.9 PyQt MVC 设计模式

3.10 PyQt 美化界面皮肤 qss

- 认识 qss
- 下载 qss
- 修改 qss

3.11 PyQt 之 Qt.py 的使用

Qt.py 如何使用?

- https://github.com/mottosso

3.12 PyQt rcc 使用

基本概况

Houdini、Maya 与 Nuke 中内置 PySide2

了解 PyQt 的历史以及协议 PySide & PyQt4 & Qt4 PySide2 & PyQt5 & Qt5

https://github.com/mottosso

Qt.py 如何使用?

PySide2 安装 & 部署 PySide2 层级结构 - QtCore 核心算法库 - QtGui 界面图形算法库 - QtWidgets 界面控件库 - uic ui 文件到 py 文件的桥梁 - sip C++ 对象到 Python 转换库

Designer 使用

setModel QThread QSQL

- 配置左下角托盘
- 修改图标
- 如何部署自己的 app
- 编译 qrc 相对路径

创建 resource.qrc "C:Python27Scriptspyside-uic.exe" -o deploy.py deploy.ui "C:Python27Libsite-packagesPySidepyside-rcc.exe" -o resource_rc.py resource.qrc

loadUi

参考文档:

- https://www.riverbankcomputing.com/static/Docs/PyQt4/classes.html
- https://doc.bccnsoft.com/docs/PyQt5/class_reference.html

3.13 PyQt 设置图标的相对路径

- 编译 qrc 相对路径

创建 resource.qrc "C:Python27Scriptspyside-uic.exe" -o deploy.py deploy.ui "C:Python27Libsite-packagesPySidepyside-rcc.exe" -o resource_rc.py resource.qrc

3.14 PyQt 富文本

创建 QLabel 控件，如何让文本丰富起来，比如将其中一部分字体设置为红色该如何操作?

打开 Nuke Script Editor 或者 Maya Script Editor 验证下面的代码。

```
from PySide2 import QtGui
from PySide2 import QtCore
from PySide2 import QtWidgets


widget = QtWidgets.QLabel("<span style='color: red;'>Hello</span> World")
widget.show()
```

```
from PySide2 import QtGui
from PySide2 import QtCore
from PySide2 import QtWidgets


widget = QtWidgets.QLabel("<span style='color: red;'>Hello</span><br> World")
widget.show()
```

红色	<span style=’ color: red;’ >text
绿色	<span style=’ color: green;’ >text
蓝色	<span style=’ color: blue;’ >text
颜色	<span style=’ color: #2e93e2;’ >text
加粗	<span style=’ font-weight: bold;’ >text
字体大小	<span style=’ font-size: 10px;’ >text
换行	
下划线	<u>text</u>
斜体	text
加粗	text
删除线	<s>text</s>

3.15 PyQt 信号与槽事件机制

- 事件无处不在，默认都有一套事件机制，可以重写来修改事件
- 控件都有自己的信号函数和对应的信号处理函数
- 事件比较隐蔽，一般很难察觉

- 在 MVC 编辑代码的时候，事件机制很复杂
- qss 也是，自有的一套事件机制

```
import sys
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import (QWidget, QLCDNumber, QSlider,
    QVBoxLayout, QApplication)


class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()


    def initUI(self):

        lcd = QLCDNumber(self)
        sld = QSlider(Qt.Horizontal, self)

        vbox = QVBoxLayout()
        vbox.addWidget(lcd)
        vbox.addWidget(sld)

        self.setLayout(vbox)
        sld.valueChanged.connect(lcd.display)

        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('Signal and slot')
        self.show()


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

```
import sys
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QWidget, QApplication

class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()


    def initUI(self):

        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('Event handler')
        self.show()


    def keyPressEvent(self, e):

        if e.key() == Qt.Key_Escape:
            self.close()


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

```
import sys
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QWidget, QApplication, QGridLayout, QLabel

class Example(QWidget):

    def __init__(self):
        super().__init__()
```

(下页继续)

(续上页)

```
        self.initUI()


    def initUI(self):

        grid = QGridLayout()
        grid.setSpacing(10)

        x = 0
        y = 0

        self.text = "x: {0},  y: {1}".format(x, y)

        self.label = QLabel(self.text, self)
        grid.addWidget(self.label, 0, 0, Qt.AlignTop)

        self.setMouseTracking(True)

        self.setLayout(grid)

        self.setGeometry(300, 300, 350, 200)
        self.setWindowTitle('Event object')
        self.show()


    def mouseMoveEvent(self, e):

        x = e.x()
        y = e.y()

        text = "x: {0},  y: {1}".format(x, y)
        self.label.setText(text)


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

```
import sys
from PyQt5.QtWidgets import QMainWindow, QPushButton, QApplication


class Example(QMainWindow):

    def __init__(self):
        super().__init__()

        self.initUI()


    def initUI(self):

        btn1 = QPushButton("Button 1", self)
        btn1.move(30, 50)

        btn2 = QPushButton("Button 2", self)
        btn2.move(150, 50)

        btn1.clicked.connect(self.buttonClicked)
        btn2.clicked.connect(self.buttonClicked)

        self.statusBar()

        self.setGeometry(300, 300, 290, 150)
        self.setWindowTitle('Event sender')
        self.show()


    def buttonClicked(self):

        sender = self.sender()
        self.statusBar().showMessage(sender.text() + ' was pressed')


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

自定义信号发射

```
import sys
from PyQt5.QtCore import pyqtSignal, QObject
from PyQt5.QtWidgets import QMainWindow, QApplication


class Communicate(QObject):

    closeApp = pyqtSignal()


class Example(QMainWindow):

    def __init__(self):
        super().__init__()

        self.initUI()


    def initUI(self):

        self.c = Communicate()
        self.c.closeApp.connect(self.close)

        self.setGeometry(300, 300, 290, 150)
        self.setWindowTitle('Emit signal')
        self.show()


    def mousePressEvent(self, event):

        self.c.closeApp.emit()


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

```
import sys

path = "D:/2020"

path in sys.path or sys.path.insert(0, path)

from PySide2 import QtWidgets
from PySide2 import QtGui
from PySide2 import QtCore
import mainWin

class CustomDialog(QtWidgets.QDialog, mainWin.Ui_Dialog):
    def __init__(self, parent=None):
        super(CustomDialog, self).__init__(parent)

        self.setupUi(self)

    @QtCore.Slot()
    def on_pushButton_clicked(self):
        print(100)

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    dialog = CustomDialog()
    dialog.show()
    sys.exit(app.exec_())
```

3.16 PyQt 定义应用程序任务栏图标

3.17 PyQt uic 使用

Houdini、Maya 与 Nuke 中内置 PySide2

了解 PyQt 的历史以及协议 PySide & PyQt4 & Qt4 PySide2 & PyQt5 & Qt5

https://github.com/mottosso

Qt.py 如何使用?

PySide2 安装 & 部署 PySide2 层级结构 - QtCore 核心算法库 - QtGui 界面图形算法库 - QtWidgets 界面控件库 - uic ui 文件到 py 文件的桥梁 - sip C++ 对象到 Python 转换库

Designer 使用

setModel QThread QSQL

- 配置左下角托盘
- 修改图标
- 如何部署自己的 app
- 编译 qrc 相对路径

创建 resource.qrc "C:Python27Scriptspyside-uic.exe" -o deploy.py deploy.ui "C:Python27Libsite-packagesPySidepyside-rcc.exe" -o resource_rc.py resource.qrc

loadUi

3.18 PyQt VS PySide

PyQt 是 GPLv3 协议，大意是你的程序中用了它，你的程序就要开源，如果闭源商用就会违反协议（后果自负，脸皮够厚无所谓）。除非你搞封装动态加载那一套来强行规避。

PySide 是 LGPL 协议，如果你只是作为库用用它，你的程序还是可以闭源商用。

所以很多人喜欢 PySide。如果不做商业项目，强烈建议使用 PyQt，资料多，稳定。需要开发闭源商用软件的就用 PySide。

GPL(General Public License) 和 LGPL(Lesser General Public License) 是 GNU 的两种 License。越来越多的自由软件 (Free Software) 使用 GPL 作为其授权声明，如果对 GPL 一点都不了解，有可能在使用自由软件时违反了 GPL 的授权。如果是个人或不正规的公司倒也无所谓，但如果是有规模的公司，恐怕会有被起诉的风险。

LGPL 是 GPL 的变种，也是 GNU 为了得到更多的甚至是商用软件开发商的支持而提出的。与 GPL 的最大不同是，可以私有使用 LGPL 授权的自由软件，开发出来的新软件可以是私有的而不需要是自由软件。所以任何公司在使用自由软件之前应该保证在 LGPL 或其它 GPL 变种的授权下。

pyside2-uic.exe VS pyuic5.exe

pyside2-rcc.exe VS pyrcc5.exe

pyqtSignal VS Signal

pyqtSlot VS Slot

- https://blog.csdn.net/The_Time_Runner/article/details/89329556
- https://wiki.qt.io/Differences_Between_PySide_and_PyQt/zh
- https://www.zhihu.com/question/21237276
- https://maicss.gitbooks.io/pyqt5/content/
- https://github.com/maicss/PyQt5-Chinese-tutorial

CHAPTER 4

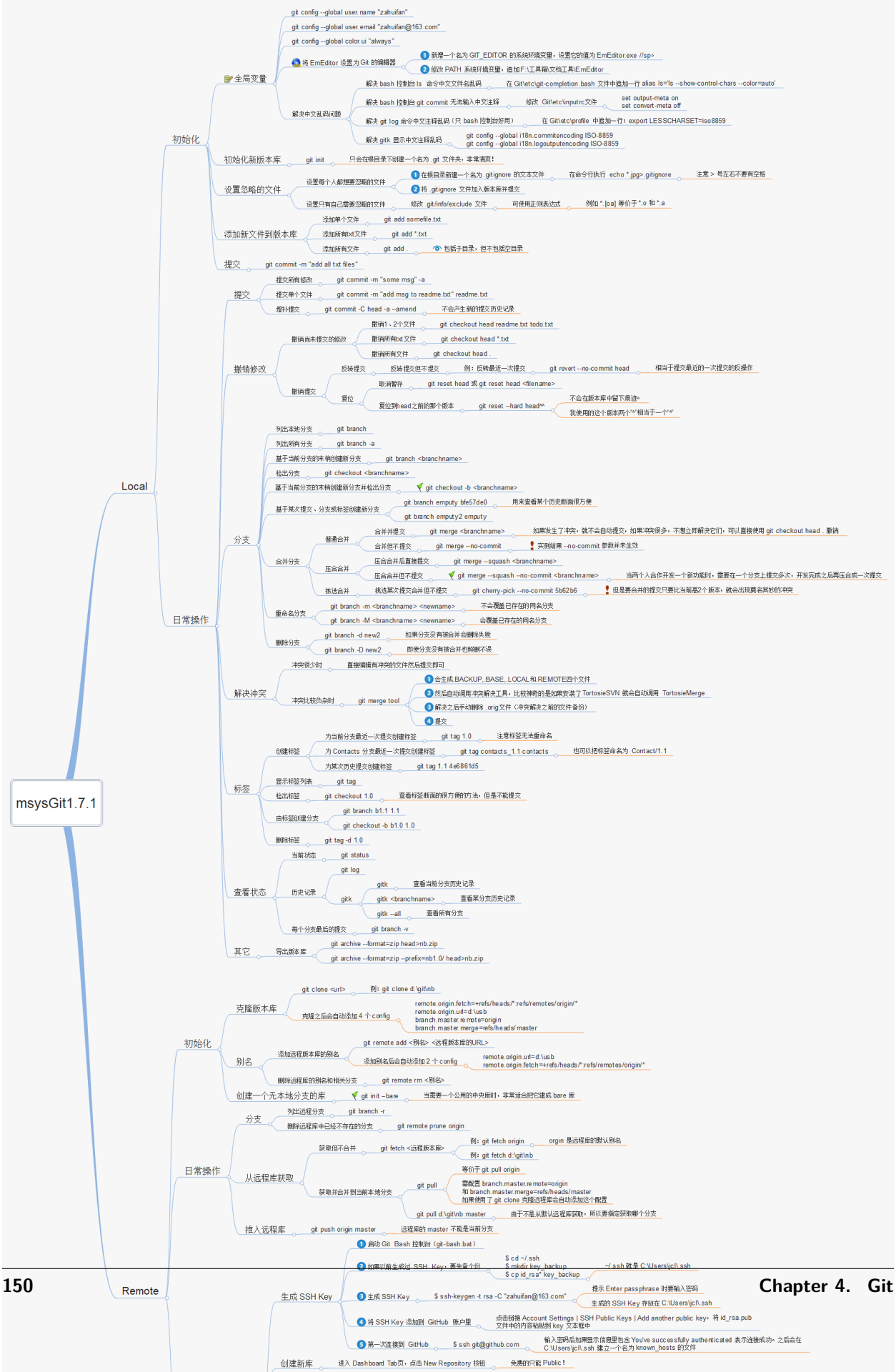
Git

Git 是一个分布式版本控制软件，最初目的是为更好地管理 Linux 内核开发而设计，后来 Git 内核已经成熟到可以独立地用作版本控制。

Contents:

4.1 Git 指令大全

友情提示：图片可以另存为或者新标签放大查看。


msysGit1.7.1
Local
初始化
全局变量
git config --global user.name "zahuifan"
git config --global user.email "zahuifan@163.com"
git config --global color.ui "always"
将 EmEditor 设置为 Git 的编辑器
新增一个名为 GIT_EDITOR 的系统环境变量，设置它的值为 EmEditor.exe //sp-
修改 PATH 系统环境变量，追加F:\工具箱\文档工具\EmEditor
解决中文乱码问题
解决 bash 控制台 ls 命令中文文件名乱码
在 Git\etc\git-completion.bash 文件中追加一行 alias ls='ls --show-control-chars --color=auto'
解决 bash 控制台 git commit 无法输入中文注释
修改 Git\etc\inputrc文件
set output-meta on
set convert-meta off
解决 git log 命令中文注释乱码（只 bash 控制台好用）
在 Git\etc\profile 中追加一行：export LESSCHARSET=iso8859
解决 gitk 显示中文注释乱码
git config --global i18n.commitencoding ISO-8859
git config --global i18n.logoutputencoding ISO-8859
初始化新版本库
git init
只会在根目录下创建一个名为 .git 文件夹，非常满意！
设置忽略的文件
设置每个人都想要忽略的文件
在根目录新建一个名为 .gitignore 的文本文件
在命令行执行 echo *.jpg> .gitignore
注意 > 号左右不要有空格
将 .gitignore 文件加入版本库并提交
设置只有自己需要忽略的文件
修改 .git/info/exclude 文件
可使用正则表达式
例如 *.[oa] 等价于 *.o 和 *.a
添加新文件到版本库
添加单个文件
git add somefile.txt
添加所有txt文件
git add *.txt
添加所有文件
git add .
包括子目录，但不包括空目录
提交
git commit -m "add all txt files"
日常操作
提交
提交所有修改
git commit -m "some msg" -a
提交单个文件
git commit -m "add msg to readme.txt" readme.txt
增补提交
git commit -C head -a --amend
不会产生新的提交历史记录
撤销修改
撤销尚未提交的修改
撤销1、2个文件
git checkout head readme.txt todo.txt
撤销所有txt文件
git checkout head *.txt
撤销所有文件
git checkout head .
撤销提交
反转提交
反转提交但不提交
例：反转最近一次提交
git revert --no-commit head
相当于提交最新的一次提交的反操作
复位
取消暂存
git reset head 或 git reset head <filename>
复位到head之前的那个版本
git reset --hard head^^
不会在版本库中留下痕迹
我使用的这个版本两个"^"相当于一个"~"
分支
列出本地分支
git branch
列出所有分支
git branch -a
基于当前分支的末梢创建新分支
git branch <branchname>
检出分支
git checkout <branchname>
基于当前分支的末梢创建新分支并检出分支
git checkout -b <branchname>
基于某次提交，分支或标签创建新分支
git branch emputy bfe57de0
用来查看某个历史版面很方便
git branch emputy2 emputy
合并分支
普通合并
合并并提交
git merge <branchname>
如果发生了冲突，就不会自动提交，如果冲突很多，不想立即解决它们，可以直接使用 git checkout head . 撤销
合并但不提交
git merge --no-commit
实例结果 --no-commit 参数并未生效
压合合并
压合合并后直接提交
git merge --squash <branchname>
压合合并但不提交
git merge --squash --no-commit <branchname>
当两个人合作开发一个新功能时，需要在一个分支上提交多次，开发完成之后再压合成一次提交
拣选合并
拣选某次提交合并但不提交
git cherry-pick --no-commit 5b62b6
但是要合并的提交只要比当前高2个版本，就会出现莫名其妙的冲突
重命名分支
git branch -m <branchname> <newname>
不会覆盖已存在的同名分支
git branch -M <branchname> <newname>
会覆盖已存在的同名分支
删除分支
git branch -d new2
如果分支没有被合并会删除失败
git branch -D new2
即使分支没有被合并也照删不误
解决冲突
冲突很少时
直接编辑有冲突的文件然后提交即可
冲突比较负杂时
git merge tool
会生成 BACKUP、BASE、LOCAL 和 REMOTE四个文件
然后自动调用冲突解决工具，比较神奇的是如果安装了 TortoiseSVN 就会自动调用 TortoiseMerge
解决之后手动删除 .orig文件（冲突解决之前的文件备份）
提交
标签
为当前分支最近一次提交创建标签
git tag 1.0
注意标签无法重命名
创建标签
为 Contacts 分支最近一次提交创建标签
git tag contacts_1.1 contacts
也可以把标签命名为 Contact/1.1
为某次历史提交创建标签
git tag 1.1.4e6861d5
显示标签列表
git tag
检出标签
git checkout 1.0
查看标签截面的很方便的方法，但是不能提交
由标签创建分支
git branch b1.1 1.1
git checkout -b b1.0 1.0
删除标签
git tag -d 1.0
查看状态
当前状态
git status
历史记录
git log
gitk
查看当前分支历史记录
gitk <branchname>
查看某分支历史记录
gitk --all
查看所有分支
每个分支最后的提交
git branch -v
其它
导出版本库
git archive --format=zip head>nb.zip
git archive --format=zip --prefix=nb1.0/ head>nb.zip
Remote
初始化
克隆版本库
git clone <url>
例：git clone d:\git\nb
克隆之后会自动添加 4 个 config
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
remote.origin.url=d:\usb
branch.master.remote=origin
branch.master.merge=refs/heads/master
别名
添加远程版本库的别名
git remote add <别名> <远程版本库的URL>
添加别名后会自动添加 2 个 config
remote.origin.url=d:\usb
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
删除远程库的别名和相关分支
git remote rm <别名>
创建一个无本地分支的库
git init --bare
当需要一个公用的中央库时，非常适合把它建成 bare 库
日常操作
分支
列出远程分支
git branch -r
删除远程库中已经不存在的分支
git remote prune origin
从远程库获取
获取但不合并
git fetch <远程版本库>
例：git fetch origin
origin 是远程库的默认别名
例：git fetch d:\git\nb
获取并合并到当前本地分支
git pull
等价于 git pull origin
需配置 branch.master.remote=origin
和 branch.master.merge=refs/heads/master
如果使用了 git clone 克隆远程库会自动添加这个配置
git pull d:\git\nb master
由于不是从默认远程库获取，所以要指定获取哪个分支
推入远程库
git push origin master
远程库的 master 不能是当前分支
生成 SSH Key
启动 Git Bash 控制台（git-bash.bat）
如果以前生成过 SSH Key，要先备个份
$ cd ~/.ssh
$ mkdir key_backup
$ cp id_rsa* key_backup
~/.ssh 就是 C:\Users\jc\.ssh
生成 SSH Key
$ ssh-keygen -t rsa -C "zahuifan@163.com"
提示 Enter passphrase 时要输入密码
生成的 SSH Key 存放在 C:\Users\jc\.ssh
将 SSH Key 添加到 GitHub 账户里
点击链接 Account Settings | SSH Public Keys | Add another public key，将 id_rsa.pub 文件中的内容粘贴到 key 文本框中
第一次连接到 GitHub
$ ssh git@github.com
输入密码后如果提示信息里包含 You've successfully authenticated 表示连接成功。之后会在 C:\Users\jc\.ssh 建立一个名为 known_hosts 的文件
创建新库
进入 Dashboard Tab页，点击 New Repository 按钮
免费的只能 Public！

4.2 Git Code Review 流程

4.3 Git 配置代理网络访问

首先我们要用 Vmware+CentOS 搭建一个代理服务器，比如 IP 是 192.168.0.112。

内网 Git 可以通过配置代理访问远程仓库。

```
git config --global http.proxy 192.168.0.112:3128
```

如果内网搭建了域以及 DNS 服务器做了 IP 映射，也可以通过下面的方式来配置。

```
git config --global http.proxy squid.do-vfx.com:3128
```

如果不想使用代理服务器了，可以使用下面的指令移除配置。

```
git config --global --unset https.proxy
```

4.4 Git 使用.gitignore 过滤上传文件

很多时候我们并不想仓库中所有文件都要推送到远程仓库中，此时可以在仓库的根目录上创建一个.gitignore 文件。

记事本打开写入如下内容：

```
*.py[cod]
.vscode
*.nk.autosave
*.nk~
```

4.5 Git 安装配置本机身份验证

从几何时起 Github 单方面取消了 Git 终端使用账号密码访问，从而导致无法提交代码

此时需要安装配置个人身份令牌

https://docs.github.com/en/get-started/quickstart/set-up-git#next-steps-authenticating-with-github-from-git

下载 gh 的 msi 客户端安装

https://github.com/cli/cli/releases/tag/v2.4.0

安装完成管理员打开命令提示符，确保 git.exe 在 Path 环境变量里

输入 gh auth login，然后依次选择下图中选项

在选择 Paste an authentication token 之后，首先你需要网页端登陆 www.github.com

找 settings>Developer settings>Personal access tokens>Generate new token，勾选所有选项创建即可

参考文档

https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token

4.6 Git 以及 TortoiseGit 安装以及使用

SVN 和 Git 都是代码版本控制软件，Git 更是采用分布式版本控制系统，目前市面比较常用，而 TortoiseGit 是 Git 图形化软件，使代码管理更高效。

- 安装 Git

官网下载地址：https://git-scm.com/downloads

- 安装 TortoiseGit

官网下载地址：https://tortoisegit.org/download/

4.7 Git 发布版本

Github 中找到想要发布版本的 repo 代码仓库，找到 Release 选项创建新版本。

Gitlab 中找到想要发布版本的 repo 代码仓库，点击 Tag 标签，New Tag，修改 Tag name，Message 以及 Release notes，Tag name 建议是 v1.1.0 这样的版本号。

- 语义化版本标准：https://semver.org/lang/zh-CN/

4.8 Git 代码仓库托管平台

Git 托管平台推荐使用 Github、Gitlab，国外服务器访问上可能有些丢包延时的问题，也可以使用本地服务器安装虚拟机搭建本地 Gitlab 以满足大型团队协作。

国外平台：

- Github: https://github.com/
- Gitlab: https://gitlab.com
- BitBucket: https://bitbucket.org/
- SourceForge: https://sourceforge.net/

国内平台：

- 码云: https://gitee.com/

参考文档：

- https://www.jianshu.com/p/bd24f3202011

4.9 Git 基本工作流程

下面的命令是日常 Git 工作流必须要使用到的命令，多敲几遍掌握它们。

- git clone repo
- git config –global user.name “Your Name”
- git config –global user.email “email@example.com”
- git status
- gitk
- git pull origin master
- git add .
- git commit -m “comment” -a
- git push origin master

下面这些命令偶尔使用，知道是做什么用的即可。

- git remote -v # 获取远程仓库的目标地址

4.10 Git 国外服务器加速方案

不管是 Github 还是 Gitlab 都是国外服务器托管代码的平台，这就导致了代码推送或者拉取的时候速度受到限制。

那么如何曲线救国呢?

咱以一个开源仓库 qLib 下载作为测试案例。

```
git clone https://github.com/qLab/qLib
```

500M 移动网络拉取代码速度只有 30~50kb 左右，代码仓库小还没什么影响，一旦文件过多基本以失败告终。

上国内代码托管平台码云注册账号，创建仓库的时候选择从 Gitlab/Github 导入仓库。

选择从 URL 导入，填写代码仓库网址。

```
https://github.com/qLab/qLib.git
```

再次尝试拉取仓库，此次使用码云仓库链接，速度变成 10M/s，100M 的仓库 10s 就下载完成啦，牛逼!

```
git clone https://gitee.com/CGRnDStudio/qLib.git
```

此时的码云仓库和国外平台仓库一毛钱关系没有，如果国外平台仓库更新了，需要手动同步才能将更新的部分同步到码云。

那么如何将自建的代码仓库用码云管理的同时，推送代码的时候能自动同步到 Github 或者 Gitlab 呢?

如果你选择码云管理代码仓库，而不需要国外平台，就不没有同步一说。我个人喜欢用 Gitlab，所以想通过 Gitee 来中转。

这里给一个解决方案，git remote 多配置一个远程仓库，在 push 的时候可以同时推送代码到码云和 Gitlab 上，而拉取代码依然是码云仓库。

```
git remote set-url --add --push origin https://gitee.com/CGRnDStudio/blablabla.git
git remote set-url --add --push origin https://github.com/CGRnDStudio/blablabla.git
```

完事之后可以看到.git/config 文件中已添加了 Github 地址。

但我相信应该有更好的解决方案，比如配置 webhook 技术，还没有去测试。

当然想解决自己公司的仓库推送和拉取速度缓慢的问题，可以自己搭建 Gitlab 本地服务器，云时代我还是比较喜欢用云端产品。

CHAPTER 5

Houdini

Houdini 是一款三维计算机图形软件，由加拿大 Side Effects Software Inc.（简称 SESI）公司开发，SESI 公司由 Kim Davidson 和 Greg Hermanovic 创建于 1987 年。Houdini 是在 Prisms 基础上重新开发而来，可运行于 Linux, Windows, Mac OS 等操作系统，是完全基于节点模式设计的产物，其结构、操作方式等和其它的三维软件有很大的差异。Houdini 自带的渲染器是 Mantra，基于 Reyes 渲染架构，因此也能够快速的渲染运动模糊、景深和置换效果。Mantra 是经过产品验证的成熟渲染器，可以满足电影级别的渲染要求。当然，Houdini 也有第三方渲染器的接口，比如：RenderMan、Mental ray、V-Ray 和 Torque 等，可以把场景导出到这些渲染引擎进行渲染。此部分文档主要用来探讨在 Houdini 中所有可行的编程方案以及更好的 Pipeline 解决方案。

Contents:

5.1 Houdini 通过 Python 代码自定义节点参数

```
node = hou.node("/out")

mantra = node.createNode("ifd")

#print(dir(mantra))
#print(mantra.asCode())
#
#hou_parm_template = hou.ToggleParmTemplate("tpreframe", "tpreframe", default_value=True)
#hou_parm_template_group.append(hou_parm_template)
```

(下页继续)

(续上页)

```
#hou_node.setParmTemplateGroup(hou_parm_template_group)

exec_parm = mantra.parm("execute")
print(exec_parm.asCode())
print(exec_parm.parmTemplate().asCode())
```

5.2 Houdini 后台输出那些事儿

几种渲染器介绍

Mantra Karma

Arnold Redshift

V-Ray RenderMan

认识环境变量 HOUDINI_PATH

环境变量可以让所有扩展插件共存，HOUDINI_PATH。

```
# Redshift env
RS_PATH = C:/ProgramData/Redshift
PATH = $PATH;$RS_PATH/bin

HOUDINI_PATH = $RS_PATH/Plugins/Houdini/18.0.348;&
```

```
# htoa env
HTOA = M:/thirdParty/htoa/htoa-5.1.0_r9289183_houdini-18.0.348/htoa-5.1.0_r9289183_
↪houdini-18.0.348
PATH = $PATH;$HTOA/scripts/bin
solidangle_LICENSE = 5053@localhost

# HOUDINI_PATH
HOUDINI_PATH = $HTOA;&
```

渲染的模式

串联与并联的区别

prepost 与 merge 的区别

FrameByFrame 与 RopByRop 区别

```
Help on method render in module houpythonportion:

render(*args, **kwargs) method of hou.RopNode instance
    render(self, frame_range=(), res=(), output_file=None,
    output_format=None, to_flipbook=False, quality=2, ignore_inputs=False,
    method=RopByRop, ignore_bypass_flags=False, ignore_lock_flags=False,
    verbose=False, output_progress=False)
```

后台渲染的三种方案

Houdini 后台输出一般支持所有的 ROP 节点，比如：

- ROP Output Driver(rop_geometry)
- ROP Alembic Output(rop_alembic)
- File Cache(filecache)
- RF Mesh Export(rf_mesh_export)
- RF Particle Export(rf_particle_export)
- Mantra(ifd)
- Arnold(arnold)
- OpenGL(opengl)
- Fetch(fetch)
- Geometry(geometry)
- Prepost(prepost)
- Merge(merge)
- Redshift(Redshift_ROP)
- hbatch | hscript

安装路径 bin 文件夹下的 hscript.exe 和 hbatch 是一样的东西，没有任何区别，正常我们会使用 hbatch.exe。

```
Usage: hbatch [-R][-e name=value][-c <command>][-j nproc][-h][-i][-q][-v][file.hip ...]

hbatch shell.  This is the non-graphical interface to a hip
file.  Type "help" for a list of commands.

Any number of .hip, .cmd, or .otl files may be specified on the
command line.  Multiple .hip files are merged together.

The -e option sets the named enviroment variable to the given
```

(下页继续)

(续上页)

```
        value.  There should be no spaces around the '=' separator between
        the name and value (i.e. -e foo=bar)

The -c option will run the option argument as an hscript command, after
        the specified files have been loaded.

The -f option forces the use of asset definitions found in OTL
        files specified on the command line.

The -j option sets the HOUDINI_MAXTHREADS to the given value.
The -h option shows this message
The -q option prevents the version information from being printed
The -w option suppresses load warnings and errors from being printed
The -v option specifies verbose handling of renders
The -i option uses a simpler interface for reading input
        when running hbatch from other applications (like Pixar's
        Alfred), it may be necessary to use this option.  Use of this
        option will disable several commands (openport and atjob)
The -R option will request a non-graphics token instead
        of a graphical on
```

```
hbatch myscene.hip
Director -> help render
Director -> render mantra1
```

```
hbatch
Director -> mread myscene.hip
Director -> help render
Director -> render mantra1
```

- hrender

hrender 是通过 csh.exe 来调用的，所以得编写 csh 脚本。

```
Usage:

Single frame:   hrender    [options] driver|cop file.hip [imagefile]
Frame range:    hrender -e [options] driver|cop file.hip

driver|cop:     -c /img/imgnet
```

(下页继续)

(续上页)

```
                -c /img/imgnet/cop_name
                -d output_driver

options:        -w pixels       Output width
                -h pixels       Output height
                -F frame        Single frame
                -b fraction     Image processing fraction (0.01 to 1.0)
                -t take         Render a specified take
                -o output       Output name specification
                -v              Run in verbose mode
                -I              Interleaved, hscript render -I

with "-e":      -f start end    Frame range start and end
                -i increment    Frame increment

Notes:  1)  For output name use $F to specify frame number (e.g. -o $F.pic).
        2)  If only one of width (-w) or height (-h) is specified, aspect ratio
            will be maintained based upon aspect ratio of output driver.
```

批量渲染多个 hip 文件，将下面文件保存成.csh 文件

```
#!C:/PROGRA~1/SIDEEF~1/HOUDIN~1.378/bin/csh.exe -f
hrender -e -f 1 5 -v -d /obj/ropnet1/mantra1 D:/test/test1.hip
hrender -e -f 1 10 -v -d /obj/ropnet1/mantra1 D:/test/test2.hip
```

- hython

自从 Houdini 引入 Python 接口之后，逐渐 Python 成为了 HScript 的替代品。

```
usage: hython [hip_files] [options] ... [-c cmd | -m mod | file | -] [arg] ...
Extra options supported by hython:
-b     : enable background openport command processing
-j arg : sets HOUDINI_MAXTHREADS to the given value;
         arg is the max number of threads

Regular Python options:
usage: hython [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-B     : don't write .py[co] files on import; also PYTHONDONTWRITEBYTECODE=x
-c cmd : program passed in as string (terminates option list)
-d     : debug output from parser; also PYTHONDEBUG=x
```

(下页继续)

(续上页)

```
-E     : ignore PYTHON* environment variables (such as PYTHONPATH)
-h     : print this help message and exit (also --help)
-i     : inspect interactively after running script; forces a prompt even
         if stdin does not appear to be a terminal; also PYTHONINSPECT=x
-m mod : run library module as a script (terminates option list)
-O     : optimize generated bytecode slightly; also PYTHONOPTIMIZE=x
-OO    : remove doc-strings in addition to the -O optimizations
-R     : use a pseudo-random salt to make hash() values of various types be
         unpredictable between separate invocations of the interpreter, as
         a defense against denial-of-service attacks
-Q arg : division options: -Qold (default), -Qwarn, -Qwarnall, -Qnew
-s     : don't add user site directory to sys.path; also PYTHONNOUSERSITE
-S     : don't imply 'import site' on initialization
-t     : issue warnings about inconsistent tab usage (-tt: issue errors)
-u     : unbuffered binary stdout and stderr; also PYTHONUNBUFFERED=x
         see man page for details on internal buffering relating to '-u'
-v     : verbose (trace import statements); also PYTHONVERBOSE=x
         can be supplied multiple times to increase verbosity
-V     : print the Python version number and exit (also --version)
-W arg : warning control; arg is action:message:category:module:lineno
         also PYTHONWARNINGS=arg
-x     : skip first line of source, allowing use of non-Unix forms of #!cmd
-3     : warn about Python 3.x incompatibilities that 2to3 cannot trivially fix
file   : program read from script file
-      : program read from stdin (default; interactive mode if a tty)
arg ...: arguments passed to program in sys.argv[1:]

Other environment variables:
PYTHONSTARTUP: file executed on interactive startup (no default)
PYTHONPATH   : ';'-separated list of directories prefixed to the
            default module search path.  The result is sys.path.
PYTHONHOME   : alternate <prefix> directory (or <prefix>;<exec_prefix>).
            The default module search path uses <prefix>\lib.
PYTHONCASEOK : ignore case in 'import' statements (Windows).
PYTHONIOENCODING: Encoding[:errors] used for stdin/stdout/stderr.
PYTHONHASHSEED: if this variable is set to 'random', the effect is the same
as specifying the -R option: a random value is used to seed the hashes of
str, bytes and datetime objects.  It can also be set to an integer
in the range [0,4294967295] to get hash values with a predictable seed.
[Redshift]Closing the RS instance. End of the plugin log system.
```

```
hython myscene.hip
rnode = hou.node("/out/mantra1")
help(rnode)
rnode.render()
```

```
hython
import hou
hou.hipFile.load(myscene.hip)
rnode = hou.node("/out/mantra1")
help(rnode)
rnode.render()
```

```
hython -c "hou.hipFile.load('hip 文件路径'); ropNode = hou.node(' 输出节点路径'); ropNode.
↪render(frame_range=(1, 10), verbose=True)"
hython rRop.py
```

认识 hou 模块

hou 模块可以分为三大类 sub-modules、classes、functions。

- sub-modules：首字母小写，不带括号为 module，module 可能又有 classes 以及 functions。
- classes：首字母大写，不带括号为 class。class 必须实例化使用，class 的属性以及方法必须通过实例化对象调用。
- functions：首字母小写，带括号为 function。

参考文档：

- https://www.sidefx.com/docs/houdini/render/batch.html#hython-and-hbatch

5.3 Houdini 自定义菜单

自定义菜单是一项必备的技能，而 Houdini 中扩展自定义菜单更是非常简单方便，Houdini 中可以自定义菜单的地方有很多，文档举三个比较重要的案例，其它菜单扩展以此类推。

- MainMenuCommon.xml 自定义主菜单
- OPmenu.xml 自定义节点菜单
- PARMmenu.xml 自定义参数菜单

scriptArgs 怎么用

自定义主菜单

```
<MainMenu>
    <menuBar>
        <subMenu>
            <label></label>
            <scriptItem id=>
                <label></label>
                <scriptPath></scriptPath>
                <scriptArgs></scriptArgs>
            </scriptItem>
            <scriptItem id=>
                <label></label>
                <scriptCode></scriptCode>
                <scriptArgs></scriptArgs>
            </scriptItem>
        </subMenu>
    </menuBar>
</MainMenu>
```

在 Windows 主菜单中添加子菜单

```
<subMenu id="windows_menu">
    <subMenu id=>
        <label></label>
        <insertBefore/>
        <titleItem>
            <label></label>
        </titleItem>
        <>
    </subMenu>
</subMenu>
```

可执行代码标签 <scriptPath> 或 <scriptCode>

```
<scriptCode><![CDATA[
hou.ui.displayMessage("Hello World")
]]>
</scriptCode>
```

```
<scriptCode><![CDATA[
def runLater():
    hou.ui.displayMessage("Hello World")
```

(下页继续)

(续上页)

```
import hdefereval
hdefereval.executeDeferred(runLater)
]]>
</scriptCode>
```

参考文档:

- https://www.sidefx.com/docs/houdini/basics/config_menus

5.4 Houdini 自定义分辨率预设

添加一条 HOUDINI_PATH，并创建 FBres 文件，无扩展名

5.5 Houdini 自定义布局

5.6 Houdini 搭建 VS Code 开发环境

- 安装 VS Code
- 安装 Python 插件
- 安装 VEX 插件
- 配置 VS Code

```
{
    "python.linting.pylintEnabled": false,
    "python.linting.pep8Enabled": true,
    "editor.renderWhitespace": "all",
    "editor.mouseWheelZoom": true,
    "editor.rulers": [79, 120, 150],
    "editor.tabSize": 4,
    "window.title": "${activeEditorLong}",
    "python.pythonPath": "C:/Python27/python.exe"
}
```

- 下载 Houdini Expression Editor

```
https://cgtoolbox.com/
```

- 配置 HOUDINI_PATH，安装 Houdini Expression Editor

- 配置扩展编辑器快捷键

```
Ctrl+Shift+Alt+LM
```

CG Toolbox 扩展编辑器配置

将 HoudiniExprEditor 插件配置到 HOUDINI_PATH 中

```
from HoudiniExprEditor import ParmWatcher
reload(ParmWatcher)
ParmWatcher.set_external_editor()
```

5.7 Houdini 中心化理念：环境变量

Houdini 中心化理念又称集成式环境部署，中心化最重要的知识点就是环境变量，HOUDINI_PATH 在 Houdini 中是及其重要的一个环境变量，它管控 Houdini 扩展开发插件的层级结构。

打开菜单 Windows>Shell，输入 hconfig -ap 可以获取 Houdini 中所有环境变量的用途描述，第一个就是 HOUDINI_PATH。

```
C:\Users\huweiguo>hconfig -ap
Common paths are set to:
$JOB = C:/Users/huweiguo
$HIP = C:/Users/huweiguo
$HOUDINI_USER_PREF_DIR = C:/Users/huweiguo/Documents/houdini16.5
$HOME = C:/Users/huweiguo/Documents
$HSITE = C:/PROGRA~1/SIDEEF~1/HOUDIN~1.378/site
$HFS = C:/PROGRA~1/SIDEEF~1/HOUDIN~1.378

HOUDINI_PATH := ""
    The path of directories where Houdini looks for configuration files.

    Directories searched (in order) are:
        1) "$HIP"
        2) "$HOUDINI_USER_PREF_DIR"
        3) "$HFS/houdini"
        4) "$HFS/bin"
```

在 Houdini 中想扩展开发插件，插件的层级结构可参考 `$HFS/houdini` 或者 `$HH` ， `$HFS` 是 Houdini 安装所在的路径。

config/Icons	自定义图标
config/NodeShapes	自定义节点形状
config/.hcs	自定义窗口风格
desktop	自定义窗口布局
dso	自定义 HDK 编译插件
gallery	自定义节点预设
geo	自定义缓存文件
ocio	自定义色彩空间
otls	自定义 otl 文件
presets	自定义节点参数预设
python_panels	自定义 Python 面板
radialmenu	自定义热盒菜单
scripts/123.py	自定义 123.py
scripts/456.py	自定义 456.py
scripts/python	自定义 Python 代码
toolbar	自定义工具架工具
vex/include	自定义 VEX 头文件
display.pref	视图窗口显示配置
ExampleMenu.xml	自定义案例菜单
FBres	自定义分辨率预设
HotkeyOverrides	自定义快捷键
jump.pref	自定义左侧快捷路径
MainMenuCommon.xml	自定义主菜单
OPmenu.xml	自定义节点右键菜单
PARMmenu.xml	自定义节点参数右键菜单
VEXpressions.txt	自定义 VEX 代码片段

还有很多其它可以自定义的配置文件，可以详细研究一下。

认识环境变量 HOUDINI_PATH 如何部署 qLib Shell & hconfig houdini.env .bat 批处理文件管理环境变量伪装快捷方式集中式管理 Arnold，Redshift，V-Ray，Renderman，RFConnect，Denoiser 等集中式管理插件的好处，方便维护，方便版本切换几种常用的环境变量

Shell

hconfig hconfig -a hconfig -ap set 某个环境变量

几种常用的环境变量 HOUDINI_BUFFEREDSAVE 加速 Houdini 网盘文件存储 HOUDINI_ACCESS_METHOD 控制 abc, hip 文件无权限读取 HOUDINI_EXTERNAL_HELP_BROWSER 设置默认谷歌浏览器打开帮助文档 HOUDINI_OTLSCAN_PATH HOUDINI_SPLASH_FILE 修改启动图片

Arnold 4.0.2htoa-4.0.2_r236e192_houdini-17.5.229

Arnold env HTOA = //server/manager/thirdParty/tools/htoa/4.0.2/htoa-4.0.2_r236e192_houdini-17.5.229 PATH = $PATH;$HTOA/scripts/bin solidangle_LICENSE = 5053@farm.do-vfx.com

HOUDINI_PATH HOUDINI_PATH = $HTOA;&

RedshiftPluginsHoudini16.5.268

Redshift env RS_PATH = C:/ProgramData/Redshift PATH = $PATH;$RS_PATH/bin

HOUDINI_PATH HOUDINI_PATH = $RS_PATH/Plugins/Houdini/16.5.268;&

PixarRenderManForHoudini-22.517.5

RenderMan env RMAN = C:/Program Files/Pixar RMANTREE = $RMAN/RenderManProServer-22.5 RFHTREE = $RMAN/RenderManForHoudini-22.5

HOUDINI_PATH HOUDINI_PATH = $RFHTREE/17.5;&

vrayvfh_home

V-Ray env VFH_ROOT=" //server/manager/thirdParty/tools/vray" VRAY_APPSDK=" $VFH_ROOT/appsdk" VRAY_OSL_PATH=" $VRAY_APPSDK/bin" VRAY_UI_DS_PATH=" $VFH_ROOT/ui" VFH_HOME=" $VFH_ROOT/vfh_home" VFH_PATH=" $VFH_HOME/bin;$VRAY_APPSDK/bin" PATH=" $VFH_PATH;$PATH"

HOUDINI_PATH HOUDINI_PATH = $VFH_HOME;&

http://127.0.0.1:48626/basics/config_env http://127.0.0.1:48626/ref/env.html https://github.com/qLab/qLib https://docs.chaosgroup.com/display/VRAYHOUDINI/QuickStart+Guides

5.8 Houdini 表达式函数

https://www.sidefx.com/docs/houdini/expressions/index.html

5.9 Houdini FBX 材质自动赋予工具

5.10 Houdini 自定义快捷键

配置 HotkeyOverrides 文件自定义快捷键 Ctrl+Shift+Alt+ 左键

5.11 Houdini HScript 命令

https://www.sidefx.com/docs/houdini/commands/index.html

5.12 Houdini 添加侧边栏快捷路径

jump.pref

5.13 Houdini 输出进度条不在显示器内的解决方案

偶尔我们在更换过程中会遇到某些窗口跑到显示器外面，如何找回呢?

通常的解决方案是通过 Alt+Tab 选择有问题的窗口，然后 Alt+Space 调出最大化最小化移动的窗口，再用快捷键 M+↑↓←→ 就可以移动窗口了，或者按 X 最大化窗口。

但是 Houdini 这个窗口位置是记录在我的文档里面的，可以通过删除我的文档内容，切记要备份文件。

5.14 Houdini 图标丢失解决方案

删我的文档可以恢复，删除的时候格外小心是否有配置 houdini.env，desktop 或者其它预设，最好备份一下，把需要的文件保留。

5.15 Houdini 模块：hou

Houdini 中有多少 Python 模块可以使用，可以通过下面的代码获取。

```
>>> help("modules")
```

从获取的结果可以看出，Python 可用的模块不止一个 hou，还有其它模块如 hutil、toolutils、husd、kramautils 等，这些模块在帮助文档中并没有提到，只能从源代码 docstring 中查询一些帮助。

hou 模块按功能可以分为三大类子模块（sub-modules）、类（classes）、函数（functions）。

- 子模块 (sub-modules): 首字母小写，不带括号为 module，module 里面可能又有 classes 以及 functions。
- 类（classes）：首字母大写，不带括号为 class。class 必须实例化使用，实例的属性以及方法必须通过实例化对象调用。
- 函数（functions）：首字母小写，带括号为 function。

子模块（sub-modules）

- hou.hipFile

```
import hou


# 获取当前的文件名
```

(下页继续)

(续上页)

```
hou.hipFile.basename()
# 获取文件的完整路径
hou.hipFile.name()
# 新建文件
hou.hipFile.clear()
# 判断当前文件是否有未保存的修改
hou.hipFile.hasUnsavedChanges()
# 保存文件
hou.hipFile.save("D:/scene.hip")
# 将选择节点保存成文件
```

- hou.session
- hou.ui

```
import hou

hou.ui.displayMessage("Hello world!!!")
hou.ui.selectFile(title="Select Obj Directory", file_type=hou.fileType.Directory)
```

- hou.qt
- hou.hotkeys

类（classes）

- hou.Node

```
import hou

hou.node("/obj").createNode("geo")
```

- hou.Parm

函数（functions）

```
import hou

# 获取环境变量值
hou.expandString("$HIPNAME")
# 获取节点实例对象
hou.node("/obj")
# 获取所有选择节点
hou.selectedNodes()
```

(下页继续)

(续上页)

```
# 获取所有的外链资产文件
hou.fileReferences()
# 获取当前节点实例对象
hou.pwd()
# 获取节点参数实例对象
hou.parm()
# 获取文档文件夹路径
hou.homeHoudiniDirectory()
# 获取软件版本
hou.applicationVersion()
# 获取软件版本字符串
hou.applicationVersionString()
>>> hou.setSimulationEnabled(0)
>>> hou.setSimulationEnabled(1)
```

案例代码：

- 获取当前帧范围

```
def getFrameRange(**kwargs):
    """
    getFrameRange will return a tuple of (fin, fout)
    :returns: Returns the frame range in the form (fin, fout)
    :rtype: tuple[int, int]
    """
    currentIn, currentOut = hou.playbar.playbackRange()
    return (currentIn, currentOut)
```

- 设置当前帧范围以及帧速率

```
def setFrameRange(fin=None, fout=None, **kwargs):
    """
    setFrameRange will set the frame range using `fin` and `fout`

    :param int fin: fin for the current context
        (e.g. the current shot, current asset etc)
    :param int fout: fout for the current context
        (e.g. the current shot, current asset etc)
    """
    hou.script("tset `((%s-1)/$FPS)` `(%s/$FPS)`" % (fin, fout))
    hou.playbar.setPlaybackRange(fin, fout)
```

参考文档:

https://www.sidefx.com/docs/houdini/hom/index.html https://www.sidefx.com/docs/houdini/hom/intro.html https://www.sidefx.com/docs/houdini/hom/hou/index.html

5.16 Houdini 模块：stateutils

5.17 Houdini 模块：toolutils

- 查询某一类型所有节点，比如查询所有的 File Cache 节点

```
import toolutils

nodes = toolutils.findAllChildNodesOfType(hou.node("/obj"), "filecache", dorecurse=True)

for node in nodes:
    print(node.path())
```

- 与 Scene View 窗口交互，比如只能选择相机得到结果。

```
import toolutils

scene_viewer = toolutils.sceneViewer()
selected_objects = scene_viewer.selectObjects("Select a camera and press Enter", allowed_
↪types = ["cam"])
print(selected_objects)
```

- 设置当前视图相机

```
>>> import toolutils
>>> scene_view = toolutils.sceneViewer()
>>> viewport = scene_view.curViewport()
>>> viewport.setCamera(hou.node("/obj/cam1/"))
>>>
```

```
import toolutils

kwargs["pane"] = None

toolutils.createOrShowPythonPanel(kwargs, "attribmanager", "Attribute Manager", 4)
```

```
>>> import toolutils
>>>
>>> scene_view = toolutils.sceneViewer()
>>>
>>> scene_view
<hou.SceneViewer panetab1>
>>> scene_view.pwd()
<hou.Node at /obj>
>>> scene_view.pwd()
<hou.ObjNode of type geo at /obj/geo1>
>>> scene_view.curViewport()
<hou.GeometryViewport persp1 of type Perspective>
>>>
>>> viewport = scene_view.curViewport()
>>> viewport
<hou.GeometryViewport persp1 of type Perspective>
>>> settings = viewport.settings()
>>> settings
<hou.GeometryViewportSettings>
>>>
>>> settings.camera()
>>> settings.camera()
<hou.ObjNode of type cam at /obj/cam1>
>>>
```

5.18 Houdini 批量导入工具

```
import os

obj_dir = hou.ui.selectFile(title="Select Obj Directory", file_type=hou.fileType.
↪Directory)
obj_dir_expanded = hou.expandString(obj_dir)

obj_files = os.listdir(obj_dir_expanded)

file_nodes = []
loader = hou.node("/obj").createNode("geo", "OBJ_Loader")

for obj in obj_files:
```

(下页继续)

(续上页)

```
    obj_file_node = loader.createNode("file", obj)
    obj_file_node.parm("file").set(obj_dir + obj)
    obj_file_node.parm("missingframe").set(1)

    file_nodes.append(obj_file_node)

merge_objs = loader.createNode("merge", "OBJ_Merger")

for node in file_nodes:
    merge_objs.setNextInput(node)

loader.layoutChildren()

merge_objs.setDisplayFlag(True)
merge_objs.setRenderFlag(True)
```

5.19 Houdini OpenCL 简单介绍

```
kernel void up(int P_length, global float *P) {
    int idx = get_global_id(0);

    if (idx >= P_length) {
        return;
    }

    float3 pos = vload3(idx, P);

    pos.y += 1;

    vstore3(pos, idx, P);
}
```

OpenCL 不能创建属性。

5.20 Houdini otls：创建以及更新的流程

otl 是一种老的数字资产的扩展名，现在扩展名是 hda（Houdini Digital Assets），因为习惯就沿用了 otl 的名称。

Subnet 自定义 HDA 方法 HOUDINI_PATH & HOUDINI_OTLSCAN_PATH Operator Name & Operator Label 自定义参数齿轮菜单

内嵌资产是指将 otl 保存到 Embedded，这样 otl 只会和场景文件一起存储，不会生成 otl 文件，一般是用于在正式创建资产之前测试数字资产，便于迭代和共享资产。

正常流程 Match Current Definition

Allow Editing of Contents

案例

创建一个 pack cache 的 otl

要求能打包当前文件中所有 filecache 节点缓存，并生成新的 hip 文件

知识点

- 获取所有 filecache 节点
- 获取所有 filecache 节点上缓存路径
- 拷贝文件操作
- 修改所有 filecache 节点上缓存路径
- 保存成新的 hip 文件

```
# 获取某节点下的所有 filecache 节点
import toolutils

rootNode = hou.node("/")

nodes = toolutils.findAllChildNodesOfType(rootNode, "filecache", dorecurse=True)

# 获取所有 filecache 节点上缓存路径
node.parm("file").eval()
node.parm("file").unexpandedString()
```

Subnet 自定义 HDA 方法 HOUDINI_PATH & HOUDINI_OTLSCAN_PATH Operator Name & Operator Label 自定义参数齿轮菜单

参考文档：

https://www.sidefx.com/docs/houdini/assets/index.html

5.21 Houdini otls：自定义参数

otls 上参数创建有三种方案：

- By Type
- From Nodes
- 拖拽参数

Name Label Type Callback Script Invisible File Pattern Horizontally Join to Next Parameter

Houdini 创建 otl 的时候，想要将节点的参数面板做些布局，学会使用参数 Folder 的用法是比较有意思的。

Folder 参数有以下几种类型模式：

- Collapsible # 加减号收缩与伸展
- Simple # 简单的框框
- Tabs # 选项卡式
- Radio Buttons # 和 Tabs 类似，多了一个 Radio 按钮显示
- Import Block
- Multiparm Block (list) # 可以加减控件数量以列表的形式
- Multiparm Block (scrolling) # 可以加减控件数量以下拉滑条的形式
- Multiparm Block (tabs) # 可以加减控件数量以选项卡的形式

Ordered Menu>Menu Script

```
labels = ["Box", "Sphere", "Grid"]
result = []

for i, v in enumerate(labels):
    result.extend([i, v])

return result
```

String>Menu

- Replace (Field + Single Selection Menu)
- Toggle (Field + Multiple Selection Menu)

Action Button

参考文档

https://www.sidefx.com/docs/houdini/assets/index.html

5.22 Houdini otls：可执行代码

hou.pwd().hdaModule()

hou.pwd() & kwargs

Custom Script

Expressions

Python Module

Before First Create

On Created

On Loaded

On Updated

On Deleted

After Last Delete

On Input Changed

On Name Changed

On Install

On Uninstall

Sync Node Version

5.23 Houdini otls：升级的两种方案

Houdini 中 otl 的版本切换可以通过主菜单 Asset>Asset Manager>Configuration 设置 Asset Bar 为 Display Menu of All Definitions ，在每个 otl 参数面板即可看到 Asset Name and Path 的切换信息。

Houdini 中想升级 otl，在已经创建好的 otl 节点上右键选择 Show in Asset Manager…菜单打开 Asset Manager 。

Asset Manager 对话框中在当前选择的节点上右键选择 Duplicate…菜单，可以打开 Copy Asset 对话框。

这里升级 otl 有两种方案，推荐使用第一种。

第一种方案是在 Operator Name 上添加版本号，Operator Label 以及 Save to Library 保持不变。

比如：

```
do::pack_cache::1.0
do::pack_cache::2.0
```

使用这种方案的好处是 otl 文件只有一个，两个版本都存储在此文件中，可以通过 otl 参数面板上的 Asset Name 来切换版本，默认优先使用的都是最新版本。

第二种方案是在 Save to Library 的时候将版本号添加到 otl 文件名上，保持 Operator Name 和 Operator Label 不变。这样做会另外存一个 otl 文件出来，相当于重新做了一个 otl 的方案。因为 Operator Name 是相同的，所以节点也只有一个，可以通过 otl 参数面板上的 Asset Path 来切换版本，默认优先使用的都是最新版本。

参考文档

https://www.sidefx.com/docs/houdini/assets/index.html

5.24 Houdini PDG 相关

5.25 Houdini 中心化部署：Arnold 插件

中心化部署也称集中式部署环境，集中式部署对于多台电脑的局域网环境是非常有利的，只要将 Arnold 渲染器部署到共享位置，通过环境变量来控制插件加载路径即可。

这意味着你可以将 Arnold 渲染器部署到任何位置，比如 D 盘。

在安装 Arnold 是时候可以自定义选择安装路径，尽量不要去拷贝文件夹到别的位置，如果不想改，默认用 C 盘安装路径也无不可。

在我的文档中找到需要安装 Arnold 渲染器的 Houdini 文件夹，比如 houdini18.0，打开 houdini.env 文件，写入如下四句环境变量配置脚本即可。

```
# htoa env
HTOA = M:/thirdParty/htoa/htoa-5.1.0_r9289183_houdini-18.0.348/htoa-5.1.0_r9289183_
↪houdini-18.0.348
PATH = $PATH;$HTOA/scripts/bin
solidangle_LICENSE = 5053@localhost

# HOUDINI_PATH
HOUDINI_PATH = $HTOA;&
```

如果 houdini.env 中配置了别的环境变量，注意将环境变量合并处理。

5.26 Houdini 中心化部署：qLib 插件

5.27 Houdini 中心化部署：Redshift 插件

中心化部署也称集中式部署环境，集中式部署对于多台电脑的局域网环境是非常有利的，只要将 Redshift 渲染器部署到共享位置，通过环境变量来控制插件加载路径即可。

这意味着你可以将 Redshift 渲染器部署到任何位置，比如 D 盘。

Redshift 渲染器默认安装路径 `C:\ProgramData\Redshift` ，安装完可以将 Redshift 文件夹拷贝到 D 盘或者共享位置，如果不想改，默认用 C 盘安装路径也无不可。

在我的文档中找到需要安装 Redshift 渲染器的 Houdini 文件夹，比如 houdini18.0，打开 houdini.env 文件，写入如下三句环境变量配置脚本即可。

```
# Redshift env
RS_PATH = C:/ProgramData/Redshift
PATH = $PATH;$RS_PATH/bin

HOUDINI_PATH = $RS_PATH/Plugins/Houdini/18.0.348;&
```

如果你想用共享位置的 Redshift，假如现在有个 M 盘，我将 Redshfit 文件夹拷贝到 M:thirdPartyRedshift 位置，就可以这样配置 houdini.env。

```
# Redshift env
RS_PATH = M:/thirdParty/Redshift
PATH = $PATH;$RS_PATH/bin

HOUDINI_PATH = $RS_PATH/Plugins/Houdini/18.0.348;&
```

如果 houdini.env 中配置了别的环境变量，注意将环境变量合并处理。

5.28 Houdini 自定义预设

5.29 Houdini 程序化练习：Path Solver

5.30 Houdini 编程语言种类

Houdini 中编程语言的种类 Python & VEX & HScript 之间的区别和联系 Python 的优缺点 VEX 的优缺点 Python 的执行环境种类 Python Shell Python Source Editor & hou.session Shelf Tools & Custom Menu 如何在工具架工具中编写 Python 代码如何在自定义菜单中编写 Python 代码 Event Handler & Startup Scripts 123.py & 456.py & onCreated.py Button Callback & HDA Scripts Python Sops & Parameter Expressions

HScript, VEX, Python, Expression functions, openCL, HDK, C++

切换场景自动与手动更新工具架工具

```
def autoUpdate():
    """ Switch current auto update status
    """
    VIEW_UPDATE = "viewupdate %s %s.%s.world"
```

(下页继续)

(续上页)

```
    currentDesktop = hou.ui.curDesktop()
    desktopName = currentDesktop.name()
    paneType = hou.paneTabType.SceneViewer
    paneTabName = currentDesktop.paneTabOfType(paneType).name()
    currentStatus = hou.hscript(VIEW_UPDATE % ("-c", desktopName, paneTabName))
    currentStatus = str(currentStatus)
    currentStatus = currentStatus[:-8]
    currentStatus = currentStatus[+16:]

    if currentStatus == "always":
        hou.hscript(VIEW_UPDATE % ("-u never", desktopName, paneTabName))
    else:
        hou.hscript(VIEW_UPDATE % ("-u always", desktopName, paneTabName))
```

Houdini Desktop 设置存储默认启动环境变量配置

@pscale = .5; @N = rand(@ptnum);

For-Each Point

```
import os

objectsPath = hou.node(".").parm("import").eval()

folder = os.listdir(objectsPath)

print(folder)

rootNode = hou.node("../python_test")
grid = rootNode.createNode("grid")
wrangler = rootNode.createNode("attrbwrangle")
wrangler.setInput(0, grid)

switch = rootNode.create("switch")
copy = rootNode.createNode("copytopoints")
blockBegin = rootNode.createNode("block_begin", "begin")
blockBegin

blockEnd = rootNode.createNode("block_end", "end")
```

5.31 Houdini 进度条

Houdini Python API 中提供 hou.InteruptableOperation 类来处理进度条，还有一种进度条方案是通过 PyQt 来实现。此文档介绍一下官方提供的接口的使用方法。

如果只有一个循环任务可以使用下面的套路代码，将 time.sleep(1) 替换成任务代码。

```
import time
import hou

num = 5

with hou.InterruptableOperation("Performing Tasks", open_interrupt_dialog=True) as␣
↪operation:

    for i in range(num):
        # Perform task here.
        time.sleep(1)

        # Update operation progress.
        percent = float(i) / float(num)
        operation.updateProgress(percent)
```

如果有两个循环嵌套的子任务，可以用下面的套路代码模板来写，将 time.sleep(1) 替换成任务代码即可。

```
import time
import hou

taskNum = 5
subTaskNum = 5

with hou.InterruptableOperation("Performing", "Performing Tasks", open_interrupt_
↪dialog=True) as operation:

    for i in range(taskNum):
        # Update long operation progress.
        longPercent = float(i) / float(taskNum)
        operation.updateLongProgress(longPercent, long_op_status="Performing Tasks %d%%"
↪% (longPercent * 100))

        # Start the sub-operation.
        with hou.InterruptableOperation("Performing Task %i" % i) as suboperation:
```

(下页继续)

(续上页)

```
            for j in range(subTaskNum):
                # Update sub-operation progress.
                percent = float(j) / float(subTaskNum)
                suboperation.updateProgress(percent)

                # Perform subtask here.
                time.sleep(1)
```

5.32 Houdini 中可执行 Python 代码的环境

- Python Shell

Windows>Python Shell

- Python Source Editor

Window>Python Source Editor

- Shelf Tools
- Custom Menu
- Parameter Expressions
- Python SOPs
- Button Callback
- Event Handler
- Python Panel

5.33 Houdini Python 控制几何体

5.34 Houdini 用户界面

```
import hou

from hutil.Qt import QtCore
from hutil.Qt import QtWidgets
from hutil.Qt import QtUiTools

uiPath = "D:/centralizeTools/houdini/scripts/python/houQt/untitled.ui"
```

(下页继续)

(续上页)

```
class SnippetA(QtWidgets.QDialog):
    # self.ui only Main Window
    def __init__(self, parent=None):
        super(SnippetA, self).__init__(parent)

        # load UI file
        loader = QtUiTools.QUiLoader()
        self.ui = loader.load(uiPath)

        # Layout
        mainLayout = QtWidgets.QVBoxLayout()
        mainLayout.setContentsMargins(0, 0, 0, 0)
        mainLayout.addWidget(self.ui)
        self.setLayout(mainLayout)


def show():
    dialog = SnippetA()
    dialog.show()
    dialog.exec_()


if __name__ == "__builtin__":
    show()
```

```
import hou

from hutil.Qt import QtCore
from hutil.Qt import QtWidgets
from hutil.Qt import QtUiTools

uiPath = "D:/centralizeTools/houdini/scripts/python/houQt/untitled.ui"

class SnippetB(QtWidgets.QDialog):
    def __init__(self, parent=None):
        super(SnippetB, self).__init__(parent)

        # load UI file
```

(下页继续)

(续上页)

```
        loader = QtUiTools.QUiLoader()
        self.ui = loader.load(uiPath)

        # Layout
        mainLayout = QtWidgets.QVBoxLayout()
        mainLayout.setContentsMargins(0, 0, 0, 0)
        mainLayout.addWidget(self.ui)
        self.setLayout(mainLayout)


def show():
    dialog = SnippetB()
    dialog.setParent(hou.qt.floatingPanelWindow(None), QtCore.Qt.Window)
    dialog.show()


if __name__ == "__builtin__":
    show()
```

```
import hou

from hutil.Qt import QtCore
from hutil.Qt import QtWidgets
from hutil.Qt import QtUiTools

uiPath = "D:/centralizeTools/houdini/scripts/python/houQt/untitled.ui"

class SnippetC(QtWidgets.QDialog):
    def __init__(self, parent=None):
        super(SnippetC, self).__init__(parent)

        # load UI file
        loader = QtUiTools.QUiLoader()
        self.ui = loader.load(uiPath)

        # Layout
        mainLayout = QtWidgets.QVBoxLayout()
        mainLayout.setContentsMargins(0, 0, 0, 0)
        mainLayout.addWidget(self.ui)
```

(下页继续)

(续上页)

```
        self.setLayout(mainLayout)


def show():
    dialog = SnippetC(hou.qt.floatingPanelWindow(None))
    dialog.show()


if __name__ == "__builtin__":
    show()
```

```
import hou
from hutil.Qt import QtCore
from hutil.Qt import QtWidgets
from houQt import mainWin
reload(mainWin)


class WindowA(QtWidgets.QMainWindow, mainWin.Ui_MainWindow):
    def __init__(self, parent=None):
        super(WindowA, self).__init__(parent)
        self.setupUi(self)


def show():
    win = WindowA(hou.qt.floatingPanelWindow(None))
    win.show()
    win.exec_()

if __name__ == "__builtin__":
    show()
```

```
from hutil.Qt import QtCore
from hutil.Qt import QtWidgets
from houQt import mainDialog
reload(mainDialog)


class WindowB(QtWidgets.QDialog, mainDialog.Ui_Dialog):
```

(下页继续)

(续上页)

```
    def __init__(self, parent=None):
        super(WindowB, self).__init__(parent)
        self.setupUi(self)


def show():
    win = WindowB(hou.qt.floatingPanelWindow(None))
    win.show()

if __name__ == "__builtin__":
    show()
```

5.35 Houdini Python Panel

Interfaces

Toolbar Menu

Pane Tab Menu

```
from hutil.Qt import QtWidgets

def onCreateInterface():
    widget = QtWidgets.QPushButton("Create Explosion NOW!")
    return widget
```

5.36 Houdini 自定义热盒菜单

配置 radialmenu 文件自定义热盒

5.37 Houdini 工具架工具

Houdini 工具架上点击加号 + > New Shelf ···

toolbar & HOUDINI_PATH

Name & Label

Icon 官方图标加载的两种方案 & 自定义图标 & HOUDINI_PATH

模块化代码管理 sys.path & HOUDINI_PATH & reload & sys.modules & 包与模块

TAB menu

Hotkeys & HotkeyOverrides & HOUDINI_PATH & Ctrl+Shift+Alt+Left Mouse

拖拽工具架工具集中管理

将一套节点拖拽到工具架工具管理

kwargs & ___builtin___ & ___name___

5.38 Houdini Shell 相关

```
hconfig -ap
icp
```

5.39 Houdini stage

Scene Graph Tree 面板控制 USD 层级结构

5.40 Houdini USD 相关

5.41 Houdini VEX：数组函数

数组是存储相同类型数据的容器，数组的特点是有序的，有序性决定了数组具有索引与切片的功能。

- 初始化

数组在变量或属性初始化的时候如果没有给初始值，也不会警告或报错，默认初始值是空数组，数组初始化一般是通过花括号 {} 或者数组函数 array()。

```
int a[];
i[]@b;
int c[] = {1, 2, 3};
i[]@d = {100, 200, 300};
int e[] = array(0, $PI, 0);

printf("%g\n", a);
printf("%g\n", @b);
printf("%g\n", c);
printf("%g\n", @d);
printf("%g\n", e);
```

(下页继续)

(续上页)

```
vector color[] = {{1, 0, 0}, {0, 1, 0}, {0, 0, 1}};
```

- 索引与切片

```
int array1[] = {100, 200, 300, 400, 500};

printf("%g\n", array1[:2]);
printf("%g\n", array1[-2:]);
printf("%g\n", array1[::-1]);
printf("%g\n", array1[-1:-3:-1]);
printf("%g\n", array1[1::2]);
printf("%g\n", array1[1:len(array1):2]);
```

- 数组函数
  - void append(<type>&array[], <type>value)
  - int [] argsort(<type>value[])
  - <type>[] array(…)
  - void insert(string &str, int index, string value)
  - int isvalidindex(<type>&array[], int index)
  - int len(<type>array[])
  - <type> pop(<type>&array[])
  - void push(<type>&array[], <type>value)
  - <type> removeindex(<type>&array[], int index)
  - int removevalue(<type>&array[], <type>value)
  - <type>[] reorder(<type>values[], int indices[])
  - void resize(<type>&array[], int size)
  - <type>[] reverse(<type>values[])
  - <type>[] slice(<type>s[], int start, int end)
  - <type>[] slice(<type>s[], int start, int end, int step)
  - int [] sort(int values[])
- 循环迭代
  - foreach (index, value; array) statement;
  - foreach (int index; element_type value; array) statement;

```
int an_array[] = {1, 2}

foreach (int num; an_array) {
    printf("%d", num);
}

string days[] = { "Mon", "Tue", "Wed", "Thu", "Fri" }
foreach (int i; string name; days) {
    printf("Day number %d is %s", i, name);
}
```

- 类型转换

```
vector pos = {1.0, 2.0, 3.0};
float a[] = set(pos);
vector color = set(a);
```

参考文档:

- http://www.sidefx.com/docs/houdini/vex/arrays.html

5.42 Houdini VEX：属性

创建属性

```
int @num = 100;

i@num = 100;

int @num[];

i[]@num = {100, 200};
```

两种方案的区别有一些，i@num 可以使用表达式运算或者函数返回值，数组属性不简写不能赋初始值。

```
vector @up = {0, 1, 0};
v@up = set(0, 1, 0);
```

内置属性不用指定数据类型，比如 @ptnum，@N 等，自定义属性如果不指定数据类型，默认初始化为浮点型数据。

所以为了和变量创建区别开推荐使用简写创建属性的方案。

i@i = 100; f@pi = 3.14; u@v1 = {1.0, 2.0}; v@v2 = set(1.0, $PI, 2.0); p@ 2@ 3@ 4@ s@

数组属性

i[]@ f[]@ s[]@

函数添加属性

```
addpointattrib(0, "Cd", {0, 0, 0});
addvertexattrib(0, "Cd", {0, 0, 0});
addprimattrib(0, "Cd", {0, 0, 0});
adddetailattrib(0, "Cd", {0, 0, 0});
```

修改属性

```
setpointattrib(0, "Cd", 0, {0, 0, 0});
setvertexattrib(0, "Cd", 0, {0, 0, 0});
setprimattrib(0, "Cd", 0, 0, {0, 0, 0});
setdetailattrib(0, "Cd", {0, 0, 0});
```

获取属性

```
float f1 = @P.y;
float f2 = @opinput1_Cd.y;
float f3 = v@opinput1_v1.y;
vector c1 = point(1, "Cd", @ptnum);
vector c2 = prim(1, "Cd", 0);
vector c3 = vertex(1, "Cd", 0);
vector c4 = detail(1, "Cd", 0);
```

全局属性

```
printf("%g\n", $PI);
printf("%g\n", $E);
printf("%g\n", @Frame);
printf("%g\n", @Time);
printf("%g\n", @Timeinc);
printf("%g\n", @SimFrame);
printf("%g\n", @SimTime);
```

内在属性

Intrinsic 内在属性只有 prim 和 detail 上才会存在

prim() 读取属性 primintrinsic() 读取属性 setprimintrinsic() 设置属性

属性相关函数

5.43 Houdini VEX：内置函数

VEX 内置函数有七八百之多，掌握如何查询与使用的通用心法非常重要，剩下的就是靠项目经验的积累来灵活运用。

我们来查看帮助文档具体分析 VEX 内置函数使用上的需要掌握的知识点。

```
int  addpoint(int geohandle, int point_number)

    Creates a new point with all the attributes and group memberships of the point with␣
↪the given point number.

int  addpoint(int geohandle, vector pos)

    Creates a new point with the given position.
```

写过函数的我们都知道函数最关键的就是传参以及返回值，我们分析一下 addpoint() 这个函数的形式参数和返回值。

首先 addpoint, 这个函数有两种写法，这是一种函数重载的写法，意味着你传入不同的实参的时候 VEX 会自我选择对应函数执行，如果实参类型和所有函数的形参类型都对不上，则会报错。函数重载是 VEX 内置函数的一种常见形态。

- int geohandle 也是 VEX 内置函数常见的一个形参，这个和 Python 类中的 self 很像，指的是当前几何体自身，一般传入 0 或者 geoself()。
- int point_number 表示添加几个点。
- vector pos 表示添加一个点的空间坐标，是个向量类型。

```
vector position = {0, 0, 0};
addpoint(0, position);
```

代码创建一个三角形。

```
int p1 = addpoint(geoself(), {0, 1, 0});
int p2 = addpoint(geoself(), {1, 1, 0});
int p3 = addpoint(geoself(), {1, 1, 1});

int prim = addprim(geoself(), "poly");
addvertex(geoself(), prim, p1);
addvertex(geoself(), prim, p2);
addvertex(geoself(), prim, p3);
```

代码创建一个圆环。

```
float angle = 0;
int num = chi("num");
float segmentAngle = 2 * PI / num;
int prim = addprim(geoself(), "poly");

for (int n = 0; n < num; n++) {
    int p = addpoint(geoself(), set(cos(angle), 0, sin(angle)));
    addvertex(geoself(), prim, p);
    angle += segmentAngle;
}
```

代码实现正弦波。

```
@P.y = sin(@P.x + @Time);
```

代码实现噪波。

```
// size = 5
// offset = 0
// threshold = 0.5
@Cd = {0, 0, 0};
float noseValues = noise(@P*chf("size") + chf("offset"));

if(noseValues > chf("threshold")) {
    @Cd.r = 1;
}
```

VEX 代码可视化，创建一个 line 节点，将点数增加到 1000。

```
@P.y = @P.x;
@P.y = pow(@P.x, 2);
@P.y = sin(@P.x);
@P.y = floor(@P.x);
@P.y = frac(@P.x);
@P.y = abs(@P.x);
@P.y = abs(sin(@P.x));
@P.y = floor(sin(@P.x));
@P.y = clamp(sin(@P.x));
@P.y = pow(frac(@P.x), 2);
@P.y = noise(frac(@P.x));
```

内置函数 set() 可以用来做类型的强制转换，这非常有用，很多时候定义向量，四元素的时候以花括号初始

化固定值，不能是动态的值，我们可以依赖 set() 函数来处理。

```
vector2  set(float v1, float v2)

vector  set(float v1, float v2, float v3)

vector4  set(float v1, float v2, float v3, float v4)

matrix2  set(float v1, float v2, float v3, float v4)

matrix3  set(float v1, float v2, float v4, float v4, float v5, float v6, float v7, float␣
↪v8, float v9)

matrix  set(float v1, float v2, float v3, float v4, float v5, float v6, float v7, float␣
↪v8, float v9, float v10, float v11, float v12, float v13, float v14, float v15, float␣
↪v16)
```

然而 set() 有其更复杂的类型转换机制，可以参考文档 http://www.sidefx.com/docs/houdini/vex/functions/set

VEX 内置函数的分类:

ARRAYS	数组	append, argsort, array, insert, isvalidindex, len, pop, push, removeindex, removevalue, reorder, resize, reverse, slice, sort
ATTRIBUTES AND INTRINSICS	属性	addattrib, adddetailattrib, addpointattrib, addprimattrib, addvertexattrib, attrib, detail, point, prim, primintrinsic, primuv, setpointattrib, setprimattrib, setprimintrinsic, setvertexattrib, vertex
BSDFS		
CHOP	通道	
COLOR	颜色	
CONVERSION	转换	atof, atoi, degrees, radians
CROWDS	群组	

下页继续

表 1 – 续上页

DISPLACE	置换	
FILE I/O	IO	
FUZZY LOGIC		
GEOMETRY	几何体	addpoint, addprim, addvertex, geoself, intersect, intersect_all, nearpoint, nearpoints, npoints, nprimitives, nvertices, pointprims, pointvertex, pointvertices, primpoint, primpoints, primvertex, primvertexcount, removepoint, removeprim, removevertex
GROUPS	组	expandpointgroup, expandprimgroup, expandvertexgroup, inpointgroup, inprimgroup, invertexgroup, npointsgroup, nprimitivesgroup, setpointgroup, setprimgroup, setvertexgroup
HALF-EDGES		
IMAGE PROCESSING		
INTERPOLATION	插值	clamp, fit, fit01, fit10, fit11, lerp
LIGHT		
MATH	数学	abs, acos, asin, atan, atan2, ceil, cos, cross, dot, floor, frac, length, length2, log, log10, max, min, normalize, pow, sin, sqrt, sum, tan pow, sin, squrt, sum
MEASURE	测量	distance, getbbox, getbbox_size, relbbox
METABALL		

下页继续

表 1 – 续上页

NODES	节点	ch, ch2, ch3, ch4, chf, chi, chramp, chs, chu, chv, opfullpath
NOISE AND RANDOMNESS	噪波	anoise, noise, rand, random, snoise, xnoise
NORMALS	法线	
OPEN COLOR IO	色彩空间	
PARTICLES	粒子	
POINT CLOUDS AND 3D IMAGES	点云	pcfilter, pcfind, pcimport, pcopen
SAMPLING	采样	
SENSOR INPUT	传感器	
SHADING AND RENDERING	材质渲染	
STRINGS	字符串	endswith, find, isalpha, isdigit, itoa, join, lstrip, match, re_find, re_findall, re_match, re_replace, re_split, split, sprintf, toupper
SUBDIVISION SURFACES		
TETRAHEDRONS		
TEXTURING	贴图	colormap
TRANSFORMS AND SPACE		
USD	USD	
UTILITY	实用	getcomp, printf, set, setcomp
VOLUME	体积	

```
vector npos = point(1, "P", @ptnum);

int np = addpoint(0, npos);

addprim(0, "polyline", @ptnum, np);
```

参考文档:

- http://www.sidefx.com/docs/houdini/vex/functions/index.html

5.44 Houdini VEX：通道参数

```
float dist = chf("dist");
int num = chi("num");

for (int n = 0; n < num; n++) {
    addpoint(0, set(n * dist, 0, 0));
}
```

通道参数是 Houdini 本身就存在的一种自定义参数的方案，只不过这里用到了 VEX 代码中而已。

这里总结一些 VEX 关于通道的内置函数。

类型	HScript	VEX
int	ch	chi()
float	ch	chf()
string	ch	chs()
vector		chv()
ramp	chramp	chramp()

ch() 通道函数默认类型是 float，一般建议在 VEX 脚本中标明通道参数的具体类型。

创建通道参数

chi() chf() chu() chv() chp() ch2() ch3() ch4() chs()

chramp()

```
// 0-1循环
@P.y = chramp("ramp", @P.x);

float px = fit(@P.x, chf("min"), chf("max"), 0, 1);
@P.y = chramp("ramp", px);

vector(chramp("color", fit(@P.x, -5, 5, 0, 1)));
```

读取通道参数

chf(, 时间)

不谈

- chsraw()
- chexpr()

```
float  chramp(string channel, float ramppos)
float  chramp(string channel, float ramppos, float time)
vector  chramp(string channel, float ramppos)
vector  chramp(string channel, float ramppos, float time)
```

ramppos 值域 [0, 1]，任何可以区别开来的属性都可以借这个值域去映射，比如 @ptnum / (@numpt - 1.0), sin(@ptnum), rand(@ptnum)。

- 创建个 Geometry。
- 创建个 Grid，Rows>100，Columns>100。
- 创建个 Mountain。
- 创建个 Attribute Wrangle。

```
@P.y = fit(@P.y, -0.5, 0.5, 0, chf("heights"));
float ramp = chramp("ramp", fit(@P.x, -5, 5, 0, 1));
@P.y = @P.y * ramp;
```

通道参数不光光可以自定义，还可以直接引用别的节点的参数过来，通过具体的参数路径或者相对路径。

. 当前层级

参考文档:

- http://www.sidefx.com/docs/houdini/ref/expression_cookbook

5.45 Houdini VEX：编译器 pragmas

5.46 Houdini VEX：转换函数

- float atof(string str) // 字符串转浮点数
- int atoi(string str) // 字符串转整数
- float radians(float num_in_degs) // 角度转弧度
- float degrees(float num_in_rads) // 弧度转角度

5.47 Houdini VEX：自定义函数

函数是编程语言中最简单的一种封装，函数最重要的是参数和返回值，参数又有形式参数（简称形参）与实际参数（简称实参）之分。

```
int test(int a, b; string c) {
    if (a > b) {
        printf(c);
    }
}
```

```
function int[] nb(int ptnum) {
    pass
}
```

```
int[] nb(int ptnum) {
    pass
}
```

```
void createGeo(vector pos){
    int p0 = addpoint(geoself(), pos + {1, 0, 1});
    int p1 = addpoint(geoself(), pos + {1, 0, -1});
    int p2 = addpoint(geoself(), pos + {-1, 0, 1});
    int p3 = addpoint(geoself(), pos + {-1, 0, -1});
    int p4 = addpoint(geoself(), pos + {0, 1.5, 0});

    addprim(geoself(), "poly", p0, p1, p3, p2);
    addprim(geoself(), "poly", p2, p4, p0);
    addprim(geoself(), "poly", p0, p4, p1);
    addprim(geoself(), "poly", p1, p4, p3);
    addprim(geoself(), "poly", p3, p4, p2);
}

createGeo(chv("pos"));
```

5.48 Houdini VEX：流控制语句

- 循环语句
  - do statement [while (condition)]
  - for (init; condition; change) statement
  - foreach (type value; array) statement
  - foreach (int index, type value; array) statement

- while (condition) statement

```
int i = 0;

for (float foo = 1; foo <= 128; foo *= 2, i++) {
    vector pos = point(0, "P", i);
    --pos.y;
    setpointattrib(0, "P", i, pos);
}
```

```
for (int n = 0; n < 10; n++) {
    addpoint(0, set(n, 0, 0));
}
```

```
s@a = "andyvfx";

foreach (string s; @a) {
    printf("%g\n", s);
}
```

```
i[]@a = {10, 20, 30, 40, 50, 60};

foreach (int i; @a) {
    printf(itoa(i) + "\n");
}
```

```
i[]@a = {10, 20, 30, 40, 50, 60};

foreach (int i; int v; @a) {
    printf(itoa(i) + "\n");
    printf(itoa(v) + "\n");
}
```

- 条件语句
  - if (condition) statement_if_true [else statement_if_false]
  - if (condition) statement_if_true [else if (condition) statement_if_true] else statement_if_false
- break & continue & return

```
int max(int a, b) {
```

(下页继续)

(续上页)

```
    if (a > b) {
        return a;
    }

    return b;
}
```

```
for (int i = 0; i < sizes; i++) {
    mixamount += getAmount(roughness);

    if (mixamount > 1) {
        break;
    }
}
```

```
foreach (x; myarray) {

    if (x < 10) continue;
    ...
}
```

- 与或非 && || !

```
if (@ptnum % 2 == 0) {
    print("%g\n", @ptnum);
}



if (@ptnum % 2) {
    print("%g\n", @ptnum);
}
```

- 条件表达式

```
@Cd = @ptnum ? 1 : 0;
```

5.49 Houdini VEX：几何体函数

addpoint()	添加点到几何体
addvertex()	添加顶点到几何体上的面
addprim()	添加面到几何体
removepoint()	删除点
removeprim()	删除面
removevertex()	删除顶点

使用 add 创建一个四角锥或者 Attribute Wrangle 创建一个四角锥，注意面法线方向。

```
int p0 = addpoint(geoself(), {1, 0, 1});
int p1 = addpoint(geoself(), {1, 0, -1});
int p2 = addpoint(geoself(), {-1, 0, 1});
int p3 = addpoint(geoself(), {-1, 0, -1});
int p4 = addpoint(geoself(), {0, 1.5, 0});

addprim(geoself(), "poly", p0, p1, p3, p2);
addprim(geoself(), "poly", p2, p4, p0);
addprim(geoself(), "poly", p0, p4, p1);
addprim(geoself(), "poly", p1, p4, p3);
addprim(geoself(), "poly", p3, p4, p2);
```

```
if (@P.x < 0) {
    removepoint(geoself(), @ptnum);
}
```

```
for (int i = 0; i < chi("num"); i++) {
    addpoint(geoself(), set(0, 0, i * chf("size")));
}
```

setprimintrinsic() setvertexpoint(0, 2, 0, 0);

primvertices() primvertex() primvertexcount() primpoints() primpoint()

pointprims() pointvetices() pointvertex() vertexpoint() vertexprev() vertexnext() vertexindex() vertexprim() vertexprimindex()

顶点序号

线性顶点序号

```
int lvt0 = pointvertex(0, 7);
int lvt1 = pointvertex(0, lvt0);
int lvt2 = pointvertex(0, lvt1);

setvertexattrib(0, "Cd", -1, lvt0, {1, 0, 0});
setvertexattrib(0, "Cd", -1, lvt1, {0, 1, 0});
setvertexattrib(0, "Cd", -1, lvt2, {0, 0, 1});
```

5.50 Houdini VEX：组

VEX 打组

@group_name

npointsgroup() // 返回组中有多少个点 nprimitivesgroup() nverticesgroup()

```
if (@P.x > 0) {
    @group_up = 1;
}
```

组函数

setpointgroup()

inpointgroup()

expandgroup() 以数组的形式返回组中元素

5.51 Houdini VEX：头文件

5.52 Houdini VEX：插值函数

```
float  fit(float value, float omin, float omax, float nmin, float nmax)

<vector> fit(<vector>value, <vector>omin, <vector>omax, <vector>nmin, <vector>nmax)

float  fit01(float value, float nmin, float nmax)

<vector> fit01(<vector>value, <vector>nmin, <vector>nmax)

float  fit10(float value, float nmin, float nmax)
```

(下页继续)

(续上页)

```
<vector> fit10(<vector>value, <vector>nmin, <vector>nmax)

float  fit11(float value, float nmin, float nmax)

<vector> fit11(<vector>value, <vector>nmin, <vector>nmax)
```

5.53 Houdini VEX：数学函数

函数名	描述
abs()	返回参数的绝对值
cos()	cos 函数
ceil()	向下求整，返回大于或等于参数的最小整数
cross()	叉乘
dot()	点乘
floor()	向上求整，返回小于或等于参数的最大整数
frac()	返回参数的小数部分
length()	返回矢量的长度
normalize()	矢量规格化
pow()	次方
sin()	sin 函数
sqrt()	开平方
tan()	tan 函数

5.54 Houdini VEX：测量函数

- vector relbbox(<geometry>geometry, vector position)
- vector relbbox(<geometry>geometry, string primgroup, vector position)
- vector relbbox(vector position)

distance()

```
@Cd = relbbox(geoself(), @P);
```

- vector getbbox_size(<geometry>geometry)
- vector getbbox_size(<geometry>geometry, string primgroup)

```
vector size = getbbox_size(geoself());
printf("%g", size);
```

5.55 Houdini VEX：噪波函数

- float rand(float seed)
- vector2 rand(float seed)
- vector rand(float seed)
- vector4 rand(float seed)
- ...

```
@Cd = rand(@ptnum);
@Cd = rand(@primnum);
```

5.56 Houdini VEX：点云函数

nearpoint()	# 查找离我最近的点，如果自己也在几何体中，则返回自己
nearpoints()	# 按范围查找最近的点，如果自己也在几何体中，数组中第一个元素是自己
neighbour()	# 查找
neighbours()	# 查找某一个点临近点

新建 Atttribute Wrangle 创建随机点。

```
for (int i = 0; i < 1000; i++) {
    vector randPos = rand(i);
    // randPos.y = 0;
    addpoint(0, randPos);
}
```

查找最近的两个点去连线。

```
int nearpt = nearpoints(0, @P, 1, 2)[1];

if (@ptnum > nearpt) {
    addprim(0, "polyline", nearpt, @ptnum);
}
```

查找附近多个点进行连线。

```
int nearpts[] = nearpoints(0, @P, 1, 10)[1:];

foreach (int nearpt; nearpts) {
    if (@ptnum > nearpt) {
        addprim(0, "polyline", nearpt, @ptnum);
    }
}
```

最简单的颜色通过小球来控制颜色传递。

```
int nearpt = nearpoint(1, @P, 0.5);

if(nearpt != -1) {
    @Cd = {1, 0, 0};
}
```

查找临近点。

```
int nbs[] = neighbours(0, @ptnum);

foreach (int nb; nbs) {
    @Cd = v@Cd2;
    i@infected = 1;
}
```

For Loop 与 Solver 区别，For Loop 循环不会继承前一帧的效果，Solver 会继承上一帧的效果。

pcopen()	
pcnumfound()	
pciterate()	
pcimport()	
pcclose()	

```
int handle = pcopen(0, "P", @P, 0.05, 10);
i@num = pcnumfound(handle);
```

5.57 Houdini VEX：代码片段

https://github.com/kiryha/Houdini/wiki/vex-snippets

5.58 Houdini VEX：字符串函数

5.59 Houdini VEX：结构体

```
struct box {
    vector size = {1, 1, 1};
    vector center = {0, 0, 0};
    float uscale = 1;

    void create() {
        vector pos[] = {
            {0.5, 0.5, 0.5},
            {0.5, 0.5, 0.5},
            {0.5, 0.5, 0.5},
            {0.5, 0.5, 0.5},
            {0.5, 0.5, 0.5},
            {0.5, 0.5, 0.5},
            {0.5, 0.5, 0.5},
            {0.5, 0.5, 0.5}
        };
        foreach(vector pt; pos) {
            addpoint(0, pt);
        }
    }
}
```

```
#include <box.h>
box mybox;
printf("%g\n", mybox.size);
mybox->create();
```

5.60 Houdini VEX：基础语法

- VEX VS Python

VEX 语言是 Houdini 中 VOP 系统从编程角度的表达方式，而 VOP 系统是 VEX 的可视化编程节点操作方式。

VEX 语法规则和 C 语言接近，和 Python 有很大区别，下面列出一些和 Python 语法上的不同点，以便区分是 VEX 脚本还是 Python 脚本。

- VEX 行结尾使用分号;
- VEX 代码块使用花括号 {}
- VEX 声明变量定义变量类型
- VEX 声明函数定义函数返回值类型
- VEX 注释使用正斜杠，单行//，多行/*multi-code*/

Houdini 中可以在 Attribute Wrangle 节点（节点网络中 TAB 键输入 aw，后面简称 aw 节点）来编写 VEX 代码，这个节点很神奇，官网文档有说明它有一种内建的上下文处理机制，参数 Run Over 切换五种模式会得到不同的结果。

- Detail(only once)
- Primitives
- Points
- Vertices
- Numbers

默认参数 Points 意味代码将在每个点循环执行，虽然你代码并没有写循环，只要有任何处理点的操作即会循环。

创建个 box，连接一个 aw 节点，写入如下代码：

```
printf("hello vex\n");
printf("%d\n", @ptnum);

// hello vex
// 0
// 1
// 2
// 3
// 4
// 5
// 6
// 7
```

- 变量以及变量声明

```
vector <variableName>;
```

- 属性 VS 变量

属性实际也是变量，唯一的区别是属性是定义在几何体上的变量而已，用来处理几何体数据，可以传递给下游节点。

声明属性的时候只要在变量前加 @ 符号即可，当然会有一些小技巧。

属性有内置属性和自定义属性之分。

- 内置属性

Houdini 中有很多内置属性，也就是几何体有一些约定俗成的属性，这些属性不需要声明类型就可以使用，Houdini 能自动识别它们的数据类型，没有特别说明都假定为浮点型。

内置属性都可以在 VOP 系统中找到对应的参数入口。

内置类型	内置属性
int	@id, @ptnum, @primnum, @vtxnum, @numpt, @numprim, @numvtx, @group_*, @resx, @resy, @resz
float	@pscale, @Time, @Frame, @density, @age, @life, @opacity
string	@name
vector	@P, @Cd, @N, @force, @rest, @up, @uv, @v
vector4	@orient

更多内容可参考文档：http://www.sidefx.com/docs/houdini/model/attributes.html

- 自定义属性

自定义属性在声明的时候 @ 符号前加属性的数据类型，比如 v@ 表示向量，f@ 表示浮点数，i@ 表示整数等，后面引用属性的时候就可以像内置属性一样不用加具体的数据类型，如果没有声明变量的类型，则默认以浮点型作为类型，自定义属性最好是声明具体的类型。

- v@vectorAttributeName;
- f@floatAttributeName;
- i@intAttributeName;

属性一般都有创建、设置、获取三种行为，比如给当前几何体添加法线属性（法线属性 @N 是一种内置属性）。

```
@N = {0, 1, 0};
@N = set(0, 1, 0);
@N = @P;
```

或者自定义属性

```
v@pos = @P;
f@y_pos = @P.y;
```

变量声明 vector <variableName> 和自定义属性 v@attriName 区别在于变量只会在当前代码中发挥作用，而属性会传递给几何体，作为数据流的一部分可以传递给下游节点，是使用变量还是属性的原则就是数据是

否要给到下游，如果不需要就使用变量，因为几何体上的属性都是需要占用硬件资源的。

- 数据类型

变量声明	自定义属性	案例
int	i@	1, 2, 3
float	f@	3.14, 9.8
vector2	u@	{0, 0}, {0.1, 0.2}
vector	v@	{0, 0, 0}
vector4	p@	{0, 0, 0, 0}
array	i/f/s[]@	{1, 2, 3, 4, 5, 6, 7, 8}
matrix2	2@	{{0, 1}, {2, 3}}
matrix3	3@	{{1, 0, 0}, {0, 1, 0}, {0, 0, 1}}
matrix	4@	{{1, 0, 0, 1}, {0, 1, 0, 1}, {0, 0, 1, 1}, {0, 0, 1, 1}}
string	s@	“hello world”

更多内容可参考文档：http://www.sidefx.com/docs/houdini/vex/lang.html

- 数组

数组在 VEX 中是一种极其重要的容器，不管是向量还是四元素都离不开数组的组织数据。

- 字符串
- 切片

切片很容易理解，和 Python 中列表切片概念是一样的，通过元素的 index 来获取区间。

- 结构体
- 点操作符
  - .x 或.u 指向 vector2 变量或属性的第一个元素。
  - .x 或.r 指向 vector 和 vector4 变量或属性的第一个元素。
  - .y 或.v 指向 vector2 变量或属性的第二个元素。
  - .y 或.g 指向 vector 和 vector4 变量或属性的第二个元素。
  - .z 或.b 指向 vector 和 vector4 变量或属性的第三个元素。
  - .w 或.a 指向 vector4 变量或属性的第四个元素.

如果是矩阵，则

- .xx 指向 [0][0] 元素。
- .zz 指向 [2][2] 元素。
- .ax 指向 [3][0] 元素。

参考文档:

- http://www.sidefx.com/docs/houdini/vex/index.html
- http://www.sidefx.com/docs/houdini/vex/snippets.html
- http://www.tokeru.com/cgwiki/?title=HoudiniVex
- https://github.com/jtomori/vex_tutorial

5.61 Houdini VEX：贴图函数

colormap()	# 将贴图颜色信息映射到点

```
@Cd = colormap("texture path",  @uv);
```

5.62 Houdini VEX：变量

变量

数据类型

int float vector2 vector vector4 array struct matrix2 matrix3 matrix string bsdf

变量命名

26 个字母大小写，阿拉伯数字以及下划线，不能以数字开头，不能使用保留字

```
int i = 100;
float pi = 3.14;
// vector 分量都是浮点型数据
vector2 v1 = {1.0, 2.0};
vector v2 = {1.0, 2.0, 3.0};
vector4 v3 = {1, 2, 3, 4};
// matrix 中括号可要可不要，最好是便于阅读
matrix2 m1 = {{0, 1}, {1, 0}};
matrix2 m2 = {0, 1, 2, 3};
matrix3 m3 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
matrix m4 = {{1, 2, 3, 0}, {4, 5, 6, 0}, {7, 8, 9, 0}, {10, 11, 12, 0}};
string s = "Andy";
```

数组类型是有序的同类型数据的组合。

可以声明空数组.. code-block:: python

int a1[]; int a2[] = {}; int a3[] = array(); printf(“%gn” , a1); printf(“%gn” , a2); printf(“%gn” , a3);

{} 和 array() 区别，array() 中可以是变量，花括号中只能是数值。

5.63 Houdini VEX：视图属性

```
@gl_wireframe = true;
@gl_lit = true;
```

5.64 Houdini VEX：可视化编程

方案一：通过 PolyExtrude 节点可视化数据。

- 创建一个 Geometry。
- 创建一个 Grid，Rows->100，Columns->100。
- 创建一个 Attribute Wrangle。
- 创建一个 Attribute Promote，Original Name->zscale，New Class->Primitive，Promotion Method->Maximum。
- 创建一个 PolyExtrude，Divide Into->Individual Elements，Distance->1，Extrusion->Output Back，Local Control->Distance Scale。
- Attribute Wrangle 节点中写入如下的代码可得到可视化结果。

```
f@zscale = @P.x;
f@zscale = pow(@P.x, 2);
f@zscale = sin(@P.x);
f@zscale = floor(@P.x);
f@zscale = frac(@P.x);
f@zscale = abs(@P.x);
f@zscale = abs(sin(@P.x));
f@zscale = floor(sin(@P.x));
f@zscale = pow(frac(@P.x), 2);
```

```
float temp = noise(@P);

if (temp > 0.5) {
    f@zscale = 1;
}
```

方案二：通过点颜色可视化数据。

```
```

- 创建一个 Geometry。
- 创建一个 Grid，Rows->100，Columns->100。
- 创建一个 Attribute Wrangle。

```
@Cd = {0, 0, 0};
float temp = noise(@P);

if (temp > 0.5) {
    @Cd.r = 1;
}
```

- 创建一个 Geometry。
- 创建一个 Box。
- 创建一个。
- 创建一个 Attribute Wrangle。

方案三：通过点位置可视化数据。

- 创建一个 Geometry。
- 创建一个 Line，Origin->-2.5, 0, 0，Direction->1, 0, 0，Length->5，Points->1000。
- 创建一个 Attribute Wrangle。

```
@P.y = pow(@P.x);
```

5.65 Houdini VEX：体积函数

- vector volumegradient(<geometry>geometry, int primnum, vector pos) // 体积梯度
- vector volumegradient(<geometry>geometry, string volumename, vector pos) // 体积梯度
- float volumesample(<geometry>geometry, int primnum, vector pos) // 体积采样
- float volumesample(<geometry>geometry, string volumename, vector pos) // 体积采样

```
float sample = volumesample(1, 0, @P);

if (sample < 0) {
```

(下页继续)

(续上页)

```
    removepoint(0, @ptnum);
}
```

CHAPTER 6

Maya

Autodesk Maya 是美国 Autodesk 公司出品的三维动画软件，应用对象是专业的影视广告，角色动画，电影特技等。

Contents:

6.1 Maya AOV 分层工具

AOV（Arbitrary Output Variables），可以通过 Arnold Python API 添加修改删除 AOV。

```
import mtoa

print(mtoa.__file__)
print(type(mtoa))
print(dir(mtoa))

# ['__builtins__', '__doc__', '__file__', '__name__', '__package__', '__path__', 'aovs',
↪'api', 'callbacks', 'cmds', 'convertShaders', 'core', 'denoise', 'hooks', 'licensing',
↪'lightFilters', 'lightManager', 'makeTx', 'melUtils', 'renderToTexture', 'txManager',
↪'ui', 'utils']

from mtoa import aovs
```

(下页继续)

(续上页)

```
aovName = "beauty"
newAov = aovs.AOVInterface()
print(dir(newAov))

# ['__class__', '__delattr__', '__dict__', '__doc__', '__format__', '__getattribute__',
↪'__hash__', '__init__', '__module__', '__new__', '__reduce__', '__reduce_ex__', '__
↪repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_
↪aovAttr', '_node', '_removeAOVNode', 'addAOV', 'getAOVNode', 'getAOVNodes', 'getAOVs',
↪'nextAvailableAttr', 'node', 'removeAOV', 'removeAOVs', 'renameAOVs']

newAov.addAOV(aovName)
print(help(newAov.addAOV))

Help on method addAOV in module mtoa.aovs:

addAOV(self, aovName, aovType=None, aovShader=None) method of mtoa.aovs.AOVInterface␣
↪instance
    add an AOV to the active list for this AOV node

    returns the created AOV node

None
```

addAOV 方法有三个形参，两个缺省参数，所以至少传入 aovName，aovType 和 aovShader 可以缺省。

获取与移除 AOV

```
from mtoa import aovs

newAov = aovs.AOVInterface()
allAovs = newAov.getAOVs()
print(allAovs)
newAov.removeAOVs(allAovs)
```

6.2 Maya 命令行脚本

作为 TD 掌握命令行后台处理问题是及其重要的一项技能，大部分可执行程序都提供命令行执行参数，以便让计算机完成一些自动化工作。

- mayapy

- mayabatch
- maya
- render
- kick（arnold）

通过-h 标签查看命令行各参数作用以及后台命令工作原理。

```
"C:\Program Files\Autodesk\Maya2016\bin\mayapy.exe" -h
"C:\Program Files\Autodesk\Maya2018\bin\maya.exe" -h
"C:\Program Files\Autodesk\Maya2018\bin\mayabatch.exe" -h
"C:\Program Files\Autodesk\Maya2018\bin\Render.exe" -h
```

mayapy 后面可以接.py 文件运行，然后通过 sys.argv 来获取额外参数，比如写一个 testMayapy.py，内容如下

```
import sys
print(sys.argv)
```

模块文件后面跟的额外参数会以字符串的形式通过 sys.argv 打包成列表，这样我们可以传入一个 Maya 文件来分析贴图数据，效果是这样的，在命令行窗口检测 Maya 文件贴图数据是否丢失工具，打开命令行命令，输入

```
"C:\Program Files\Autodesk\Maya2016\bin\mayapy.exe"␣
↪D:\centralizeTools\maya\env\2018\scripts\checkTools\relationship_check.py␣
↪Z:\YYDTENN\Production\Department\LGT\EP01\sc006\YY_CG_sc006_lgt_color_v001_01.ma
```

在使用 mayapy.exe 的时候，一定要将 maya.standalone 初始化。

```
import maya.standalone

maya.standalone.initialize()

# -----------------
# main function
# -----------------

maya.standalone.uninitialize()
```

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

"""
```

（下页继续）

(续上页)

```
This standalone maya script can be check out reference & texture relationship
"""

import os
import sys
import logging
import maya.standalone
import maya.cmds as cmds
maya.standalone.initialize()

# logging.basicConfig(level=logging.WARNING, format="\nandy%(asctime)s - %(levelname)s -
↪%(message)s\n")
logging.basicConfig(format="%(asctime)s %(levelname)-8s %(message)s",
                    level=logging.WARNING, datefmt="%Y-%m-%d %H:%M:%S")


def main(check_maya_file):
    """
    """
    if not os.path.isfile(check_maya_file):
        logging.warning("Failed because Maya file was not found! - %s" % check_maya_file)

    logging.info("START CHECK - %s" % check_maya_file)
    cmds.file(check_maya_file, open=True, pmt=False)
    logging.info("# Maya Version : %s" % cmds.about(v=True))
    # logging.info("# Arnold Version : %s" % cmds.pluginInfo("mtoa", q=True, v=True))

    # Check reference relationship
    for ref in cmds.file(q=True, r=True):
        refNode = cmds.referenceQuery(ref, referenceNode=True)
        logging.info("Check reference : %s" % refNode)
        # referenced_file = cmds.referenceQuery(reference, f=True, wcn=True)
        logging.info("Check reference path - %s" % ref)

        if not os.path.isfile(ref):
            logging.error("[!] The reference was not exists! : %s" % ref)

        if not cmds.referenceQuery(refNode, isLoaded=True):
            logging.error("[!] The reference was not loaded! : %s" % refNode)
```

(下页继续)

(续上页)

```
        logging.info("> Check OK .")

    # Try to get "PROJECTWORKS" environment variable
    projectworks = os.environ.get("PROJECTWORKS", 0)

    if not projectworks:
        logging.error("The environment variable - \"PROJECTWORKS\" is undefined! Please␣
↪to check out Maya.env file.")

    # Check texture file
    for texture in cmds.ls(type="file"):
        logging.info("Check Textures - %s" % texture)
        tex_name = cmds.getAttr(texture + ".fileTextureName")

        if not tex_name:
            logging.warning("The %s is no texture." % texture)
            continue

        logging.info("Check Texture Path - %s" % tex_name)

        if tex_name.startswith("$"):
            replace_name = tex_name.replace("$PROJECTWORKS", projectworks)
            tex_name = replace_name
        if not os.path.isfile(tex_name):
            logging.warning("The texture was not found! - %s" % tex_name)


if __name__ == "__main__":
    main(sys.argv[1])
```

参考文档:

- http://help.autodesk.com/view/MAYAUL/2020/ENU/

6.3 Maya 检查拓扑结构

Maya Animation 菜单中 Deform>Blend Shape 工具的原理就是依据两个物体的拓扑结构相同。

拓扑结构相同可以决定缓存融合，BlendShape 变形方面的制作流程。

- 点数相同

- 面数相同
- 点与面的关系

```
# get model points
import pymel.core as pm
import maya.OpenMaya as om

def getGeoPoints(geo):
    geoNode = pm.PyNode(geo)
    iter = om.MItMeshVertex(geoNode.__apiobject__())

    while not iter.isDone():
        print(iter.index())
        iter.next()

if __name__ == "__main__":
    sels = pm.ls(sl=1)
    getGeoPoints(sels[0])
```

```
# set string io
from StringIO import StringIO
import pymel.core as pm
import maya.OpenMaya as om

def getGeoPoints(geo):

    pointsInfo = StringIO()
    geoNode = pm.PyNode(geo)
    iter = om.MItMeshVertex(geoNode.__apiobject__())

    while not iter.isDone():
        pointsInfo.write(str(iter.index()))
        pointsInfo.write("\n")
        # print(iter.index())
        iter.next()

    print(pointsInfo.getvalue())

if __name__ == "__main__":
    sels = pm.ls(sl=1)
    getGeoPoints(sels[0])
```

```
# get face id
from StringIO import StringIO
import pymel.core as pm
import maya.OpenMaya as om

def getGeoPoints(geo):

    pointsInfo = StringIO()
    geoNode = pm.PyNode(geo)
    faceID = om.MIntArray()
    iter = om.MItMeshVertex(geoNode.__apiobject__())

    while not iter.isDone():
        iter.getConnectedFaces(faceID)
        pointsInfo.write(str(iter.index()))
        pointsInfo.write(" ")
        pointsInfo.write(" ".join([str(i) for i in faceID]))
        pointsInfo.write("\n")
        # print(iter.index())
        iter.next()

    print(pointsInfo.getvalue())

if __name__ == "__main__":
    sels = pm.ls(sl=1)
    getGeoPoints(sels[0])
```

```
# check md5
import md5
from StringIO import StringIO
import pymel.core as pm
import maya.OpenMaya as om

def getGeoPoints(geo):

    pointsInfo = StringIO()
    geoNode = pm.PyNode(geo)
    faceID = om.MIntArray()
    iter = om.MItMeshVertex(geoNode.__apiobject__())
```

(下页继续)

(续上页)

```
    while not iter.isDone():
        iter.getConnectedFaces(faceID)
        pointsInfo.write(str(iter.index()))
        pointsInfo.write(" ")
        pointsInfo.write(" ".join([str(i) for i in faceID]))
        pointsInfo.write("\n")
        # print(iter.index())
        iter.next()

    return md5.new(pointsInfo.getvalue()).hexdigest()

if __name__ == "__main__":
    sels = pm.ls(sl=1)
    print(getGeoPoints(sels[0]))
```

6.4 Maya 自定义 mel 命令

6.5 Maya 自定义菜单

maya mod 环境变量管理 MAYA_MODULE_PATH 扩展开发层级结构 Python 模块加载的机制 PYTHONPATH 在 mod 中的配置

```
DO-VFX.mod
+ DO-VFX any \\server\manager\centralizeTools\maya\env\2018
```

扩展开发层级结构.scripts 存放代码的地方，mel 文件（userSetup.mel），py 文件以包的形式管理

比如下面创建 userSetup.mel 文件来加载自定义菜单

```
if (`menu -exists "menuDO"`){
    deleteUI -menu "menuDO";
}

menu -label "DO-VFX" -parent $gMainWindow -tearOff true "menuDO";
menuItem -divider true -dividerLabel "General";
menuItem -label "Choose Shot" -parent "menuDO" -command "print(1);";
menuItem -label "Upload RV" -parent "menuDO" -command "print(1);";
menuItem -label "Submit To Deadline" -parent "menuDO" -command "python(\"from submitJob␣
↪import submitter;reload(submitter);submitter.main()\")";
```

(下页继续)

(续上页)

```
menuItem -divider true -dividerLabel "Step";
menuItem -label "LAY" -parent "menuDO" -subMenu true -tearOff true "menuLay";
menuItem -label "Bake Camera" -parent "menuLay" -command "BakeCamera;";

menuItem -label "MOD" -parent "menuDO" -subMenu true -tearOff true "menuMod";
menuItem -label "Batch Transfer UV" -parent "menuMod" -command "BatchTransferUV;";
menuItem -label "Remove Duplicate Object Names" -parent "menuMod" -command
↪"RemoveDuplicateObjectNames;";
menuItem -label "Find Triangle Face" -parent "menuMod" -command "FindTriangleFace;";
menuItem -label "Remove All Namespaces" -parent "menuMod" -command "RemoveAllNamespaces;
↪";
menuItem -label "Rock Generator" -parent "menuMod" -command "RockGenerator;";

menuItem -label "ANM" -parent "menuDO" -subMenu true -tearOff true "menuAnm";
menuItem -label "Anim Playblast Tool" -parent "menuAnm" -command "AnimPlayblastTool;";
menuItem -label "Camera Shaker" -parent "menuAnm" -command "cameraShaker;";
menuItem -label "Anim Smooth Keys" -parent "menuAnm" -command "oaSmoothKeys;";

menuItem -label "LGT" -parent "menuDO" -subMenu true -tearOff true "menuLgt";
menuItem -label "Reference LookDev" -parent "menuLgt" -command "python(\"from lookDev␣
↪import referenceFile;reload(referenceFile);referenceFile.referenceLookdevFile()\")";
menuItem -label "Create Rivet Locator" -parent "menuLgt" -command "rivetEX;";
menuItem -label "Check Scene" -parent "menuLgt" -command "python(\"from checkScene␣
↪import ZCheck_JBY;reload(ZCheck_JBY);ZCheck_JBY.CheckTheScene_UI();ZCheck_JBY.show()\")
↪";
menuItem -label "File Texture Manager" -parent "menuLgt" -command "FileTextureManager;";
menuItem -label "Arnold AOV Creator" -parent "menuLgt" -command "tjh_Arnold_AOV_creator;
↪";
menuItem -label "Lost Textures Finder" -parent "menuLgt" -command "tjh_lost_textures_
↪finder_Main;";
menuItem -label "Vray Attributes Assigner" -parent "menuLgt" -command "tjh_vray_
↪attributes_assigner_Main;";
menuItem -label "Plant On Surface with Follicles" -parent "menuLgt" -command "tjh_
↪plantOnSurface_withFollicles_Main;";
menuItem -label "AOVs Arnold Masker" -parent "menuLgt" -command "AOVsArnoldMasker;";
menuItem -label "Arnold ID" -parent "menuLgt" -command "ArnoldID;";
menuItem -label "Arnold Masks" -parent "menuLgt" -command "ArnoldMasks;";
menuItem -label "ASS Proxy Switch" -parent "menuLgt" -command "ASSProxySwitch;";
menuItem -label "Batch View Render" -parent "menuLgt" -command "python(\"from␣
↪batchViewRender import mainWin;reload(mainWin);mainWin.mayaRenderWindowRender();
↪mainWin.checkSettings()\")";
```

(下页继续)

(续上页)

```
menuItem -label "Arnold Render Check" -parent "menuLgt" -command "python(\"from␣
↪renderCheck import linearWorkflowCheck;reload(linearWorkflowCheck);linearWorkflowCheck.
↪maya_ui()\")";
```

认识 PYTHONPATH 环境变量的作用

添加 PYTHONPATH 的几种方案 1、添加到系统环境变量或者用户环境变量 2、配置 Maya.env 文件 3、配置 mod 文件 4、配置启动文件去加载环境变量 5、抑或在使用的时候用 sys.path 添加

认识 sys.path 的作用 sys.path.append() sys.path.insert()

py 文件以包的形式存放在 scripts 文件夹中

6.6 Maya 自定义 mll 插件

6.7 Maya 自定义插件

代码所存在的问题类名大驼峰的写法代码不符合 PEP8 规范代码单引号双引号统一代码使用 Git 管理测试上线 varName != None 改成 not varName docstring 和注释需要规范

如何编写一个简单的.py 插件工具如何在启动的时候 Maya 扩展开发的层级结构

插件环境变量部署

```
# C:\Program Files\Autodesk\Maya2017\modules\hprender.mod
+ hpRender any C:/Users/huweiguo/Documents/.RENDER_ROOT/MayaPlugin/Setup/2017
PATH +:= bin
```

PATH +:= bin 意味着将路径 C:/Users/huweiguo/Documents/.RENDER_ROOT/MayaPlugin/Setup/2017/bin 路径添加到 PATH 环境变量中，这里的加 bin 文件夹并没有任何作用，可以删除

扩展开发层级结构

plug-ins 中添加需要加载的插件 scripts 中写菜单工具执行的模块代码

Maya 环境变量 MAYA_PLUG_IN_PATH 插件文件位置 MAYA_MODULE_PATH mod 文件位置 MAYA_SCRIPT_PATH 脚本文件位置 XBMLANGPATH 图标文件位置

Maya.env 配置的案例

```
MTOAROOT = \\server\manager\thirdParty\maya\mtoa\3.1.2.1\2018

REDSHIFT_COREDATAPATH = \\server\manager\thirdParty\maya\Redshift\2.5.48

QAULOTH_PATH = \\server\manager\thirdParty\maya\Qualoth
```

(下页继续)

(续上页)

```
REDSHIFT_PLUG_IN_PATH = %REDSHIFT_COREDATAPATH%\Plugins\Maya\2018\nt-x86-64
REDSHIFT_SCRIPT_PATH = %REDSHIFT_COREDATAPATH%\Plugins\Maya\Common\scripts
REDSHIFT_XBMLANGPATH = %REDSHIFT_COREDATAPATH%\Plugins\Maya\Common\icons
REDSHIFT_RENDER_DESC_PATH = %REDSHIFT_COREDATAPATH%\Plugins\Maya\Common\rendererDesc
REDSHIFT_CUSTOM_TEMPLATE_PATH = %REDSHIFT_COREDATAPATH
↪%\Plugins\Maya\Common\scripts\NETemplates
REDSHIFT_MAYAEXTENSIONSPATH = %REDSHIFT_PLUG_IN_PATH%\extensions
REDSHIFT_PROCEDURALSPATH = %REDSHIFT_COREDATAPATH%\Procedurals

MAYA_MODULE_PATH = %MAYA_MODULE_PATH%;%MTOAROOT%
PATH = %PATH%;%MTOAROOT%\bin;%REDSHIFT_COREDATAPATH%\bin
MAYA_RENDER_DESC_PATH = %MAYA_RENDER_DESC_PATH%;%MTOAROOT%;%REDSHIFT_RENDER_DESC_PATH%
PYTHONPATH = %PYTHONPATH%;%MTOAROOT%\scripts;%REDSHIFT_SCRIPT_PATH%
MAYA_PLUG_IN_PATH = %REDSHIFT_PLUG_IN_PATH%;%QAULOTH_PATH%\bin
MAYA_SCRIPT_PATH = %REDSHIFT_SCRIPT_PATH%;%QAULOTH_PATH%\script
XBMLANGPATH = %REDSHIFT_XBMLANGPATH%
MAYA_CUSTOM_TEMPLATE_PATH = %REDSHIFT_CUSTOM_TEMPLATE_PATH%

MAYA_DISABLE_CLIC_IPM = 1
```

来看下如何写个 py 的插件到插件管理器吧，分两步注册插件 C:UsershuweiguoDocuments.RENDER_ROOTMayaPluginSetup
ins hpRender.py

如何创建自定义菜单

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

"""
Custom menu plug-in
"""

import os
import sys

import maya.cmds as cmds
import maya.mel as mel
import maya.OpenMayaMPx as omm
```

(下页继续)

(续上页)

```
def initializePlugin(mobject):
"""
Initialize the script plug-in
"""
mPlugin = omm.MFnPlugin(mobject, "tdUtils", "1.0", "any")

if cmds.menu("menuTD", exists=1):
    cmds.deleteUI("menuTD", menu=1)

gMainWindow = mel.eval("global string $gMainWindow;$temp = $gMainWindow")
cmds.menu("menuTD", label="TD Tools", parent=gMainWindow,
            tearOff=1, allowOptionBoxes=1)
cmds.menuItem(dividerLabel="General", divider=True)
cmds.menuItem(label="Submit To Deadline", parent="menuTD",
                command="print(100)")
cmds.menuItem(dividerLabel="Step", divider=True)
cmds.menuItem("menuRIG", label="RIG", parent="menuTD",
                subMenu=True, tearOff=True)
cmds.menuItem(label="Select Skin Joint", parent="menuRIG",
                command="print(100)")


def uninitializePlugin(mobject):
"""
Uninitialize the script plug-in
"""
mPlugin = omm.MFnPlugin(mobject)

if cmds.menu("menuTD", exists=1):
    cmds.deleteUI("menuTD", menu=1)
```

sys.path 模块导入机制 import hpMayaClient reload(hpMayaClient) hpMayaClient.client().main()

main 函数的核心 checkList 检测客户端是否安装检测 C 盘是否有读写权限检测文件是否保存检测每一层渲染层的渲染帧数范围，这块没必要用正则表达式，可以直接获取 cmds.getAttr(“defaultRenderGlobals.startFrame”) 分析渲染层的函数如何获取当前场景中用到的所有插件如何获取当前场景中用到的所有资产文件 cmds.file(query=True, list=True, withoutCopyNumber=True) ver.ini 检测 Maya 版本信息检测插件版本 C:UsershuweiguoDocuments.RENDER_ROOTMayaPluginmayaPluginItems_2017.ini 检测每一层的渲染设置 UI cmds.window 窗口如果存在删除控件 (button, text, intFieldGrp, checkBox, textFieldGrp, radioButtonGrp) 布局

PyQt 重写界面 PySide & PyQt4 PySide2 & PyQt5 PyQt4 和 PyQt5 uic pyside-uic

```
# Maya Dialog
import sys
import maya.OpenMayaUI as omui
from PySide2 import QtCore
from PySide2 import QtWidgets
from shiboken2 import wrapInstance

path = "D:/2019/centralizeTools/houdini/scripts/python"

path in sys.path or sys.path.insert(0, path)
from houQt import mainDialog


def _get_maya_main_window():
    pointer = omui.MQtUtil.mainWindow()
    return wrapInstance(long(pointer), QtWidgets.QWidget)


class WindowA(QtWidgets.QDialog, mainDialog.Ui_Dialog):
    def __init__(self, parent=None):
        super(WindowA, self).__init__(parent)
        self.setupUi(self)


def show():
    dialog = WindowA(_get_maya_main_window())
    dialog.show()

if __name__ == "__main__":
    show()
```

```
# Maya Main Window
import sys
import maya.OpenMayaUI as omui
from PySide2 import QtCore
from PySide2 import QtWidgets
from shiboken2 import wrapInstance

path = "D:/2019/centralizeTools/houdini/scripts/python"
```

(下页继续)

(续上页)

```
path in sys.path or sys.path.insert(0, path)
from houQt import mainWin


def _get_maya_main_window():
    pointer = omui.MQtUtil.mainWindow()
    return wrapInstance(long(pointer), QtWidgets.QWidget)


class WindowB(QtWidgets.QMainWindow, mainWin.Ui_MainWindow):
    def __init__(self, parent=None):
        super(WindowB, self).__init__(parent)
        self.setupUi(self)


def show():
    dialog = WindowB(_get_maya_main_window())
    dialog.show()

if __name__ == "__main__":
    show()

if cmds.window(WINDOW_NAME, exists=True, q=True):
    cmds.deleteUI(WINDOW_NAME)
```

6.8 Maya 自定义 py 插件

6.9 Maya 自定义工具架

Maya 中自定义工具架工具还是挺有意思的，这篇文字介绍自定义工具架工具的方方面面。

首先有几种已知的方案可以添加工具架工具，首先我们应该要创建一个新的工具架，Maya 预留了一个空的 Custom 工具架给我们使用。

为了可以存储我们自己的工具建议还是自己创建一个工具架，点击工具架一栏左边的齿轮菜单，点击 New Shelf，输入 shelf name 点击 OK，这样我们就创建了一个新的工具架。

这个工具架的文件会存储在我的文档下面的位置。

```
C:\Users\{USERNAME}\Documents\maya\{VERSION}\prefs\shelves
```

Maya 工具架有个毛病是不能通过环境变量中心化管理，Maya 启动的时候只认我的文档这个位置的 shelf 文件加载。

在工具架上添加工具有下面几种已知的方法：

- 可以通过鼠标中键将已经存在的工具架工具拖拽到当前的工具架。
- 可以通过鼠标中键将左侧 ToolBox 中的工具拖拽到当前的工具架。
- 可以将任何一个主菜单中的工具通过 CTRL+SHIFT+ 单击的方式添加到当前的工具架。
- 可以选择脚本编辑器中的代码，通过鼠标中键拖拽到当前工具架。
- 可以通过 Shelf Editor 面板自定义添加工具到我们的当前工具架。

正常来说 Shelf Editor 是万能的自定义工具架工具的方法，不管上面是以哪种方式添加的工具，实际都可以通过 Shelf Editor 来添加与修改。

关于 Shelf Editor 面板有几点需要说明：

- Shelves: 添加或者删除某个工具架某个工具，设置工具图标，工具提示等等。
- Command: 单击工具执行的脚本，可以是 MEL，也可以是 Python。
- Double Click Command: 双击工具执行的脚本，可以是 MEL，也可以是 Python。
- Popup Menu Items: 可以给工具添加一些右键菜单，执行不同的脚本。

6.10 Maya 默认打开方式

管理员打开注册表

计算机 HKEY_CLASSES_ROOTMayaBinaryFileshellopencommand

计算机 HKEY_CLASSES_ROOTMayaAsciiFileshellopencommand

6.11 Maya 开发环境

.mod bat & vbs 模块导入流程自定义菜单导入自定义工具架导入

https://www.youtube.com/watch?v=18bMHq43K3U&t=2489s

如何制定一个 module

环境变量管理

.mod 文件扩展工具层级结构

我的文档/modules 我的文档/scripts 我的文档/plug-ins

DO-VFX.mod

```
+ DO-VFX any \\server\manager\centralizeTools\maya\env\2018
```

icons modules plug-ins scripts shelves Maya.env

.bat 批处理

```
from pprint import pprint
import pymel.core as pm

pprint(dir(pm))
pprint(pm.language.Env.envVars.items())
```

分析资产 XGen Reference Texture Fur Proxy Cache Alembic Assembly Yeti Qualoth …分析插件 Arnold Redshift 分析渲染层分析渲染设置

用户界面正则表达式上下文管理器文件操作字符编码读写配置文件 XMLGenerator json

如何来调试局部代码?

```
# optionVar 设置首选项变量
import maya.cmds as cmds
cmds.optionVar(intValue=("defaultTriangles", 100))
cmds.optionVar(query="defaultTriangles")

# file 文件操作
cmds.file(query=True, sceneName=True, shortName=True)

# getAttr & setAttr 属性读写操作
cmds.getAttr("defaultRenderGlobals.modifyExtension")

# listConnections & listRelatives 节点关联操作
cmds.listConnections("renderLayerManager.renderLayerId")

# 渲染层分析
def nzList(buf):
    if buf == None:
        return []
    return buf

def nzAnalyzeMayaGetAttr(layer, attr):
    attr = nzGetOverrideAttr(layer, attr)
    print(attr)
```

(下页继续)

(续上页)

```
    return cmds.getAttr(attr, expandEnvironmentVariables = True)

def nzGetOverrideAttr(layer, attr):
    if layer == cmds.editRenderLayerGlobals(query = True, currentRenderLayer = True):
        return attr
    layers = cmds.listConnections('renderLayerManager.renderLayerId[0]')
    layers.insert(0, layer)
    buf = cmds.listConnections(attr, source = False, plugs = True, connections = True)
    for layer in layers:
        print(layer)
        print(buf)
        for i in range(1, len(nzList(buf)), 2):
            if re.search('^' + layer + r'\.adjustments\[\d+]\.plug$', buf[i]) != None:
                # print ('w:' + str(buf[i]) + re.sub(r'\.plug$', '.value', buf[i]))
                return re.sub(r'\.plug$', '.value', buf[i])
    return attr

renderLayers = cmds.listConnections('renderLayerManager.renderLayerId')

print(renderLayers)

for layer in renderLayers:
    renderer = nzAnalyzeMayaGetAttr(layer, 'defaultRenderGlobals.currentRenderer')
    print(renderer)


# error 日志提醒
mom.MGlobal.displayError("File not existed!")
cmds.error("File not existed!")
```

```
# codecs
import codecs
def readFileContent(filePath):
    with codecs.open(filePath, "r", "utf-8") as f:
        lines = [line.strip() for line in f if line]
    return lines
```

6.12 Maya 环境变量

高级系统设置 > 环境变量

系统变量用户变量

命令行窗口如何运行 Maya? Path 如何修改 Maya 语言选项? MAYA_UI_LANGUAGE

putenv & getenv

Maya.env

如何阅读帮助文档?

MAYA_PLUG_IN_PATH MAYA_MODULE_PATH MAYA_SCRIPT_PATH XBMLANGPATH

- http://help.autodesk.com/view/MAYAUL/2018/CHS/?guid=GUID-925EB3B5-1839-45ED-AA2E-3184E3A45AC7

6.13 Maya 配置 Execute script nodes

在菜单 File>Open 打开文件的选项设置中有一项 Execute script nodes，默认是勾选状态，它的意思是打开文件的时候执行文件中 ScriptNode 节点中的脚本，这也是普天同庆恶意脚本传播的原理，所以在清理完 C 盘相关文件之后，只要简单配置关闭这个选项就可以防止这个恶意脚本的传播。

我们找到文件 C:UsershuweiguoDocumentsmaya2018prefsuserPrefs.mel，搜索 fileExecuteSN，将 1 改为 0 保存即可。

```
-iv "fileExecuteSN" 0
```

6.14 Maya 前台渲染脚本解决方案

一般来说前台渲染调用 RenderIntoNewWindow 或者 renderWindowRenderCamera 命令即可，批量渲染写个 for 循环问题不大，下面是 Python 与 MEL 两种写法。

```
# Python
import maya.cmds as cmds
import maya.mel as mel

for i in range(1, 241):
    cmds.currentTime(i)
    mel.eval("renderWindowRenderCamera redoPreviousRender renderView persp;")
```

```
// MEL
for (int $i = 1; $i <= 240; $i++) {
    evalDeferred("currentTime $i;");
    evalDeferred("RenderIntoNewWindow;");
}
```

但是遇到 Redshift 渲染器的时候，上面的 for 循环会报渲染占用的错误，解决方案可以在 Render Settings 中写 Post render frame MEL，将下面的任何一句 MEL 脚本写到此设置中，在点击 Render View 窗口渲染的时候会自动渲染下一帧，渲染的序列在工程路径/images/tmp 中，简单实现前台批量渲染的功能，结束渲染需要按 ESC 中断。

```
// MEL
int $frame = `currentTime -q`; evalDeferred("currentTime ($frame + 1);"); evalDeferred(
↪"RenderIntoNewWindow;");
int $frame = `currentTime -q`; evalDeferred("currentTime ($frame + 1);"); evalDeferred(
↪"renderWindowRenderCamera redoPreviousRender renderView persp;");
```

6.15 Maya 插件：Yeti

6.16 Maya 帮助文档

Maya 帮助文档中开发的部分需要理解的是 MEL Command 以及 Python Command 文档部分的用法，包括案例，包括两种命令的切换。

Maya 中编程语言分类大概有如下几种：

- MEL 类：命令、函数、表达式
- Python 类：cmds、pymel、Maya Python API 1.0、Maya Python API 2.0
- C++ 类：C++ API（不阐述）
- Script Editor

脚本编辑器可以通过菜单 Windows>General Editors>Script Editor 来打开，或者右下角脚本编辑器快捷按钮打开。

脚本编辑器中可以编写 MEL 和 Python 两种脚本语言，可以通过鼠标右键热盒菜单创建与删除面板。

- MEL Command

MEL 命令基本语法：命令 -标签 -标签 -标签操作对象;

从脚本编辑器的历史记录中提取 MEL 命令方法。

help/whatIs 方法

```
whatIs scale;
whatIs polyCube;
whatIs polySmooth;
whatIs polyCleanupArgList;
```

MEL 命令的 CQEM 模式

- C 直接使用新设置某个参数数值 move -x 1;
- Q 可以获取对象的当前参数前面加 q, 后面不加值 xform -q -bb;
- E 可以编辑对象的现有参数前面加 e, 后面加新值 joint -e -p 0 0 0;
- M 可以在一次命令下使用多次直接重复调用参数 curve -p 0 0 0 -p 0 1 0 -p…;

```
ls
listRelatives
listConnections
listAttr
getAttr
setAttr
```

- MEL Function

很多 MEL 函数是不在帮助文档中的, 存在两种 MEL 函数, 一种是 global function, 另一种是 helper function, helper 函数大多是 mel 文件中调用，与 global 函数作用域不同。

- Python
  - 封装 MEL 的伪 Python 代码 maya.cmds
  - 第三方更 Pythonic 的封装 PyMEL
  - Maya Python 1.0（封装 C++ API，只有 C++ 文档）
  - Maya Python 2.0（封装 C++ API，只有 C++ 文档）

cmds 是官方出的一种简单封装 MEL 指令伪 Python 代码，没有按 Python 数据类型统一性来封装，MEL 与 cmds 切换起来相对简单，写 cmds 实际就是在写 MEL 指令，文档中右上角可以通过 MEL version 或者 Python version 随意切换，MEL 中的 flag 在 Python 中就是 args。

pymel 是第三方按 Python 统一性来封装的 Python API，但是导入模块相对需要花一部分时间。

```
import maya.cmds as cmds
import maya.mel as mel
import pymel.core as pm
import maya.api.OpenMaya as om
```

MEL 是一种强类型语言

```
string $sels[] = `ls -sl`;
print($sels);

string $sels[] = `ls -sl -l`;
int $count = 1;

for ($sel in $sels) {
    print((string)$count + " : " + $sel + "\n");
    $count++;
}
```

```
import maya.cmds as cmds

sels = cmds.ls(sl=True, l=True)

for i, sel in enumerate(sels):
    print(str(i) + " : " + sel + "\n")
```

```
import maya.OpenMaya as om

selslist = om.MSelectionList()
om.MGlobal.getActiveSelection(selsList)

for idx in range(0,  selsList.length()):
    dagPath = om.MDagPath()
    sels.getDagPath(idx, dagPath)
    print(dagPath.fullPathName())
    print(dagPath.partialPathName())
    print("%d : long-name : %s" % (idx, dagPath.fullPathName()))
```

第三方包装 MEL 的 pythonic 的包

```
import pymel.core as pm

sels = pm.ls()

for sel in sels:
    print(sel.longName())
    print(sel.shortName())
    print("------------------")
```

(下页继续)

(续上页)

```
    print(sel.tx.get())
    print(sel.ty.get())
    print(sel.tz.get())
    sel.tx.set(0)
    sel.ty.set(0)
    sel.tz.set(0)
```

```
import maya.OpenMaya as om

selsList = om.MSelectionList()
om.MGlobal.getActiveSelectionList(selsList)

for idx in range(0, selsList.length()):
    mobject = om.MObject()
    selsList.getDependnode(idx, mobject)
    print(mobject.apiTypeStr())

    if mobject.apiType() == om.MFn.kTransform:
        print("This is a transform!")
    elif mobject.apiType() == om.MFn.kMesh:
        print("This is a mesh!")
    else:
        pass

    fnDependNode = om.MFnDependencyNode(mobject)
    print(fnDependnode.name())
```

参考文档:

- https://vfxplatform.com/
- https://www.youtube.com/watch?v=GiWkXufclTY
- https://knowledge.autodesk.com/support/maya/getting-started/caas/simplecontent/content/maya-documentation.html
- http://help.autodesk.com/view/MAYAUL/2020/CHS/
- http://help.autodesk.com/view/MAYAUL/2020/CHS/?guid=__PyMel_index_html
- http://help.autodesk.com/view/MAYAUL/2020/CHS/?guid=Maya_SDK_MERGED_cpp_ref_index_html

6.17 Maya 线性工作流工具

核心思想数据类型，对应的特性，属性与方法 listAttr getAttr setAttr 检测插件 & 加载插件 pymel pythonic 参数设置的两种方案实例化对象模块的导入导出 sys.path 设置 AOV

管中窥豹：延伸阅读 cmds VS pymel pymel 第三方文档 Maya 中有哪些扩展接口 MEL command & function import maya.cmds as cmds import maya.mel as mel import pymel.core as pm Python API 1.0 Python API 2.0 C++ API

检测插件 & 加载插件

```
import pymel.core as pm

if "mtoa" not in pm.pluginInfo(query=True, listPlugins=True):
    try:
        pm.loadPlugin("mtoa")
    except:
        pm.error("Fail to Load Arnold Render!!")
    pm.PyNode("defaultRenderGlobals").currentRender.set("arnold")
```

https://help.autodesk.com/cloudhelp/2018/JPN/Maya-Tech-Docs/PyMel/index.html

```
from pprint import pprint
import pymel.core as pm

# 返回所有节点
pprint(pm.ls())
# 实例化节点
renderGlobals = pm.PyNode("defaultRenderGlobals")
attrs = renderGlobals.listAttr()
pprint(attrs)

print(type(renderGlobals.currnetRenderer))
print(dir(renderGlobals.currnetRenderer))
renderGlobals.setAttr("currentRendererer", "arnold")
renderGlobals.currentRenderer.set("arnold")
pm.setAttr("defaultRenderGlobals.currentRenderer", "arnold")
pm.PyNode("defaultRenderGlobals.currentRenderer").set("arnold")

aiOptions = pm.PyNode("defaultArnoldRenderOptions")

# defaultRenderGlobals
```

(下页继续)

(续上页)

```
# defaultArnoldRenderOptions
# defaultArnoldFilter
# defaultArnoldDriver

pm.PyNode("defaultRenderGlobals.animation").set(True)

import maya.cmds as cmds
# cmEnabled
# renderingSpaceName
# viewTransformName
# defaultInputSpaceName
# colorManagePots
cmds.colorManagementPrefs(e=True, parm=value)


# 软件版本的兼容性
import pymel.core as pm

print(pm.about(version=True))

import sys
path = "D:/"
path in sys.path or sys.path.insert(0, path)
print(sys.path)

from renderTools import linearWorkflowCheck
reload(linearWorkflowCheck)
linearWorkflowCheck.maya_ui()
```

```
from pprint import pprint
import maya.cmds as cmds

pprint(cmds.file(query = True, list = True, withoutCopyNumber = True))
```

6.18 Maya 材质导入导出工具

```
# 获取所有 shading group
cmds.ls(type="shadingEngine")
```

(下页继续)

(续上页)

```
# 获取选择物体的 shape
cmds.listRelatives(geo, children=True, path=True)
cmds.listConnections(shp, destination=True, t="shadingEngine")
cmds.file(file_path, options="v=0;", typ="mayaAscii", pr=True, es=True)
cmds.sets(sg, q=True)
# reference
cmds.file(file_path, r=True, ns=name_space)
cmds.sets(filter_items, e=True, forceElement=sg)
```

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import os
import json

import maya.cmds as cmds


def get_all_sg_nodes():
    """
    获取所有的 sg 类型的节点
    """
    sg_nodes = cmds.ls(typ="shadingEngine")
    return sg_nodes


def get_sel_sg_nodes():
    """
    获取所选择物体链接的 sg 节点
    """
    sg_nodes = list()
    selected_geos = cmds.ls(sl=True)
    for geo in selected_geos:
        shapes = cmds.listRelatives(geo, children=True, path=True) or list()
        for shp in shapes:
            sg_node = cmds.listConnections(shp, destination=True, t="shadingEngine")
            sg_nodes.extend(sg_node)

    sg_nodes = [sg for i, sg in enumerate(sg_nodes) if sg not in sg_nodes[:i]]
```

(下页继续)

(续上页)

```
    return sg_nodes


def export_sg_nodes(sg_nodes, file_path):
    """
    导出 sg 节点到 ma 文件
    """
    if len(sg_nodes) == 0:
        return False

    cmds.select(sg_nodes, r=True, ne=True)
    cmds.file(file_path, options="v=0;", typ="mayaAscii", pr=True, es=True)
    return True


def export_all_sg_nodes(file_path):
    """
    导出所有 sg 节点
    """
    return export_sg_nodes(get_all_sg_nodes(), file_path)


def export_sel_sg_nodes(file_path):
    """
    导出选择相关的 sg 节点
    """
    return export_sg_nodes(get_sel_sg_nodes(), file_path)


def get_sg_members(sg_nodes=list()):
    """
    提取物体名字，判断不是 transform 类型，是 shape 的，就获取 shape 的父物体加到筛选列表里
    """
    data = dict()
    for sg in sg_nodes:
        members = cmds.sets(sg, q=True) or list()
        filter_members = list()
        for m in members:
            obj = m.split(".")
            if cmds.nodeType(obj[0]) != "transform":
```

(下页继续)

(续上页)

```
                obj[0] = cmds.listRelatives(obj[0], p=True)[0]
            filter_members.append(".".join(obj))

        data[sg] = filter_members

    return data


def get_all_sg_members():
    """
    """
    return get_sg_members(get_all_sg_nodes())


def get_sel_sg_members():
    """
    """
    return get_sg_members(get_sel_sg_nodes())


def export_sg_members(data, file_path):
    """
    """
    with open(file_path, "w") as f:
        json.dump(data, f, indent=4)

    return True


def export_all_sg_members(file_path):
    """
    """
    return export_sg_members(get_all_sg_members(), file_path)


def export_sel_sg_members(file_path):
    """
    """
    return export_sg_members(get_sel_sg_members(), file_path)
```

(下页继续)

(续上页)

```
def reference_shader_file(file_path):
    """
    """
    # - 如果文件在场景的 reference 列表里，直接返回文件的 namespace
    file_path = file_path.replace("\\", "/")
    ref_files = cmds.file(query=True, reference=True)
    if file_path in ref_files:
        return cmds.file(file_path, query=True, namespace=True)

    # - 如果文件没有 reference 过，就新 re 一份
    name_space = os.path.splitext(os.path.basename(file_path))[0]
    cmds.file(file_path, r=True, ns=name_space)
    return name_space


def assign_data_to_all(data_path, sg_namespace=None, geo_namespace=None):
    """
    """
    data = dict()
    with open(data_path, "r") as f:
        data = json.load(f)

    for sg, geos in data.items():
        # - 先过滤 sg 节点场景里存在不存在
        if sg_namespace:
            sg = "{0}:{1}".format(sg_namespace, sg)
        if not cmds.objExists(sg):
            continue

        # - 过滤 sg 对应的物体是否都存在
        filter_items = list()
        for g in geos:
            g = "{0}:{1}".format(geo_namespace, g)
            if cmds.objExists(g.split(".")[0]):
                filter_items.append(g)

        # - 连接 sg 节点与物体
        try:
            cmds.sets(filter_items, e=True, forceElement=sg)
```

(下页继续)

(续上页)

```
        except:
            pass

# export_sel_sg_nodes("D:/andy/lanzu.ma")
# export_sel_sg_members("D:/andy/lanzu.json")

assign_data_to_all("D:/andy/lanzu.json", "lanzu")
```

6.19 Maya MEL 基础

- Maya 脚本编辑器使用方法
- Maya 帮助文档查询方法
- Maya 中 MEL 命令转 Python 命令方法
- Maya 中 Python 中执行 MEL 命令

Script Editor 脚本编辑器，提取历史命令。

快捷键 Ctrl+Enter

单行注释

多行注释

MEL 语言的一些标识，$，分号，花括号，变量类型声明

MEL 与 Python 脚本的互转，如何在 MEL 脚本中执行 Python，Python 脚本中执行 MEL?

6.20 Maya MEL 常用命令

ls 命令

```
import maya.cmds as cmds

cmds.ls()

cmds.ls(sl=1)

cmds.ls(typ=["mesh", "lambert"], l=1)

cmds.ls(et="transform")
```

(下页继续)

(续上页)

```
cmds.ls(et="lambert")

cmds.ls(ext="transform")

cmds.ls("blendShape*")

cmds.ls("blendShape?Set")

cmds.ls("UUID")
```

rename 命令

```
import maya.cmds as cmds

cmds.rename(oldName, newName)
```

节点的层级关系

listConnections & listRelatives

```
import maya.cmds as cmds

cmds.listRelatives(object, p=1)
cmds.listRelatives(object, c=1)
cmds.listRelatives(object, ad=1)
cmds.listRelatives(object, ad=1, typ="joint", f=1)
```

parent & group

```
import maya.cmds as cmds

cmds.parent(childObject, parentObject)

cmds.parent(childObject, w=1)
cmds.parent(cmds.ls(sl=1), n="newGrp")
```

世界坐标 & 物体坐标 (相对与绝对)

move & rotate & scale & xform

```
import maya.cmds as cmds

```

(下页继续)

(续上页)

```
cmds.move(0, 0, 0, object, r=1)

cmds.rotate(0, 30, 0, object, a=1)

cmds.scale(1, 1, 1, object, r=1)

cmds.scale(1, 1, 1, object, a=1)

cmds.xform(object, q=1, t=1)
cmds.xform(object, t=(0, 0, 0))

cmds.xform(object, q=1, ro=1)
cmds.xform(object, ro=(0, 0, 0))

cmds.xform(object, q=1, t=1, ws=1)
cmds.xform(object, t=(0, 0, 0), ws=1)

cmds.xform(object, q=1, ro=1, ws=1)
cmds.xform(object, ro=(0, 0, 0), ws=1)
```

创建节点

```
import maya.cmds as cmds

cmds.polySphere()
cmds.circle()
cmds.curve()
cmds.joint(p=(0, 0, 0))
cmds.createNode("joint")
```

获取节点类型与属性

```
import maya.cmds as cmds

cmds.nodeType(object)
cmds.listAttr(object)
cmds.listAttr(object, k=1)
cmds.listAttr(object, ud=1)
cmds.getAttr()
cmds.setAttr()
cmds.setAttr(attribute, value, typ="string")
```

属性连接与断开

Windows>General Editors>Connection Editor

Windows>Node Editor

Windows>General Editors>Hypergraph: Hierarchy

Windows>General Editors>Hypergraph: Connections

查看节点技术文档

```
import maya.cmds as cmds

cmds.connnectAttr(attr1, attr2)
cmds.disconnectAttr(attr1, attr2)

cmds.connnectAttr(attr1, attr2, f=1)
cmds.connnectAttr("pCubeShape1.outMesh", "pSphereShape1.inMesh", f=1)
```

获取节点的连接

```
import maya.cmds as cmds

# 上游节点
cmds.listConnections(object, s=1, d=0)
# 下游节点
cmds.listConnections(object, s=0, d=1)
# 节点属性
cmds.listConnections(object, s=0, d=1, p=1)
```

6.21 Maya 打包工具

Maya 外链文件、缓存以及代理

- nParticle 缓存 (不包括已经过时的经典粒子)
- nHair 缓存
- nCloth 缓存
- Fluid 缓存
- Geo 缓存
- MASH 缓存
- Texture 贴图 (包括 UDIM-Mari)

- Alembic 缓存
- XGEN 描述及贴图
- Reference Ma 及 Mb 文件
- Arnold_Proxy 代理 (以及代理内的贴图)
- Vray_Proxy 代理
- Redshift_Proxy 代理
- Bifrost 缓存
- Audio 声音文件

通过插件加载情况来针对性检测外链文件

6.22 Maya 如何获取拍屏面板参数？

optionVar 命令可以获取一些 Maya 选项设置的参数

```
import maya.cmds as cmds

for elem in cmds.optionVar(l=True):

    if "playblast" in elem.lower():
        print(elem)
        print(cmds.optionVar(q=elem))
```

通过上面的代码可以将与拍屏相关的参数筛选出来。

```
PlayblastCmdAvi

PlayblastCmdFormatAvi

PlayblastCmdFormatQuicktime

PlayblastCmdQuicktime

optionBoxDimensionsPlayblast
[546L, 350L]
playblastClearCache
1
playblastCompression
H.264
```

(下页继续)

(续上页)

```
playblastDisplaySizeSource
1
playblastEndTime
10.0
playblastFile
playblast
playblastFormat
qt
playblastHeight
256
playblastMultiCamera
0
playblastOffscreen
0
playblastPadding
4
playblastQuality
70
playblastSaveToFile
0
playblastScale
0.5
playblastShowOrnaments
1
playblastStartTime
1.0
playblastUseSequenceTime
0
playblastUseStartEnd
0
playblastViewerOn
1
playblastWidth
256
```

参数与具体 option 的对照表

View:	playblastViewerOn
Show ornaments:	playblastShowOrnaments
Render offscreen:	playblastOffscreen
Multi-Camera Output:	playblastMultiCamera
Format:	playblastFormat
Encoding:	playblastCompression
Quality:	playblastQuality
Display size:	playblastWidth playblastHeight
Scale:	playblastScale
Frame padding:	playblastPadding
Remove temporary files:	playblastClearCache
Save to file:	playblastSaveToFile
Movie file:	playblastFile

6.23 Maya 拍屏工具

```
import uuid
import datetime
import maya.cmds as cmds

def playblast(path, cam, start, end, view=False):
    """
    """
    imagePath = os.path.expanduser("~/playblast")

    if not os.path.isdir(imagePath):
        os.makedirs(imagePath)

    imagePrefix = os.path.join(imagePath, "20200903_ABCDEF")
    print(imagePrefix)
    cmds.playblast(filename=imagePrefix,
                   format="image",
                   compression="tga",
                   width=1280,
                   height=720,
                   startTime=start,
                   endTime=end,
                   clearCache=1,
```

(下页继续)

(续上页)

```
                    viewer=1,
                    showOrnaments=1,
                    fp=4,
                    percent=100,
                    quality=100)

if __name__ == "__main__":
    playblast("", "persp", 1, 10)
```

6.24 Maya 插件：Arnold

6.25 Maya 插件：p+

6.26 Maya 插件：Qualoth

6.27 Maya 插件：Redshift

6.28 Maya 插件：Shave

6.29 Maya 插件：Yeti

6.30 Maya 代理文件贴图获取方案

获取 Redshift 渲染器 rs 代理文件所有贴图路径代码:

```
import re
import codecs
from pprint import pprint

import maya.cmds as cmds

def readFileCode_bag2(path):

    con = ""

    if os.path.exists(path):
```

(下页继续)

(续上页)

```
        Arg = codecs.open(path, "r")
        con = Arg.read()
        Arg.close()

    return con

getAllRs = []
getAllNodes = cmds.ls(type="RedshiftProxyMesh")

tm = re.compile("(\\x00\\x00\\x00[A-Za-z]:/.*?\\x00)+")
tm2 = re.compile("(\\x00\\x00\\x00//[0-9].*?\\x00)+")

for node in getAllNodes:
    getPath = cmds.getAttr(node + ".fileName")

    if getPath and getPath not in getAllRs:
        getAllRs.append(getPath)

for oneProxy in getAllRs:
    getAllPath = []
    print(oneProxy)
    getCon = readFileCode_bag2(oneProxy)
    getResult = tm.findall(getCon)
    getResult2 = tm2.findall(getCon)

    for oneString in getResult:
        if oneString and "\\" not in oneString and "_map_auto" not in oneString and "/"␣
↪in oneString:
            getPathMap = oneString.split("\x00\x00\x00")[1][:-1]
            if getPathMap and getPathMap not in getAllPath:
                getAllPath.append(getPathMap)

    for oneString in getResult2:
        if oneString and "\\" not in oneString and "_map_auto" not in oneString and "/"␣
↪in oneString:
            getPathMap = oneString.split("\x00\x00\x00")[1][:-1]
            if getPathMap and getPathMap not in getAllPath:
                getAllPath.append(getPathMap)

    pprint(getAllPath)
```

获取 Arnold 渲染器 ass 代理文件所有贴图路径代码:

```
from pprint import pprint
import maya.cmds as cmds

import arnold as ar


# 分析 ass 代理文件贴图路径
getAllAss = []
getAllNodes = cmds.ls(type="aiStandIn")

for node in getAllNodes:
    getPath = cmds.getAttr(node + ".dso")

    if getPath and getPath not in getAllAss:
        getAllAss.append(getPath)

for ass in getAllAss:
    getAllPath = []
    print(ass)
    ar.AiBegin()
    ar.AiMsgSetConsoleFlags(ar.AI_LOG_ALL)
    ar.AiASSLoad(ass, ar.AI_NODE_ALL)
    iterator = ar.AiUniverseGetNodeIterator(ar.AI_NODE_ALL)

    while not ar.AiNodeIteratorFinished(iterator):
        node = ar.AiNodeIteratorGetNext(iterator)

        if ar.AiNodeIs(node, "MayaFile") or ar.AiNodeIs(node, "image"):
            getPath = ar.AiNodeGetStr(node, "filename")

            if getPath and getPath not in getAllPath:
                getAllPath.append(getPath)

    ar.AiNodeIteratorDestroy(iterator)
    ar.AiEnd()
    pprint(getAllPath)
```

6.31 Maya PyMEL

什么是 pymel?

pymel 与 mel 以及 cmds 区别?

```
import pymel.core as pm
import maya.cmds as cmds
```

- 返回类型不同

cmds 只返回字符串

pymel 返回实例对象，可以用 type，dir 以及 help 内置函数

```
# 通过 pm.PyNode() 将节点实例化

resNode = pm.PyNode("defaultResolution")
# 列出节点所有属性
resNode.listAttr()
resNode.width.set(720)
resNode.width.get()
pm.selected()

Node Editor

node.inputs()

# 获取的是实例化对象，有属性和方法
pm.sceneName()
pm.env.minTime
pm.env.maxTime
pm.listCameras()

import pymel.core as pm

for k, v in pm.language.Env.envVars.items():
    print(k, v)


fileName = pm.sceneName().splitext()[0].basename()
startFrame = pm.env.minTime
endFrame = pm.env.maxTime
```

(下页继续)

(续上页)

```
cameraName = "%s_f%03d_f%03d" % (fileName, startFrame, endFrame)
cameraList = [cam for cam in pm.listCameras(p = True)]
selPanel = pm.getPanel(withFocus = True)
selCam = pm.modelPanel(selPanel, q=True, camera=True)
cam = pm.PyNode(selCam)
cam.rename(cameraName)

node = pm.PyNode("defaultRenderGlobals")
pm.listAttr(node)
node.imageFilePrefix.set("<Scene>/<RenderLayer>")
node.animation.set(1)
node.startFrame.set(startFrame)
node.endFrame.set(endFrame)
pm.setAttr("defaultResolution.width", 1280)
pm.setAttr("defaultResolution.height", 720)
```

https://help.autodesk.com/cloudhelp/2017/ENU/Maya-Tech-Docs/PyMel/

6.32 Maya Redshift 修改缓存路径

Maya Redshift 插件默认缓存路径 %LOCALAPPDATA%\Redshift\Cache，如果使用默认缓存路径，很快 C 盘空间就会爆掉，此处的 LOCALAPPDATA 是环境变量路径，可以通过命令行窗口输入下面指令得到路径。

```
C:\Users\huweiguo>set LOCALAPPDATA
LOCALAPPDATA=C:\Users\huweiguo\AppData\Local

C:\Users\huweiguo>
```

在 Maya 渲染面板 Render Settings 中切换 Redshift 渲染选项，找到 System>Global Preferences>Cache Folder，可以修改缓存文件夹路径。

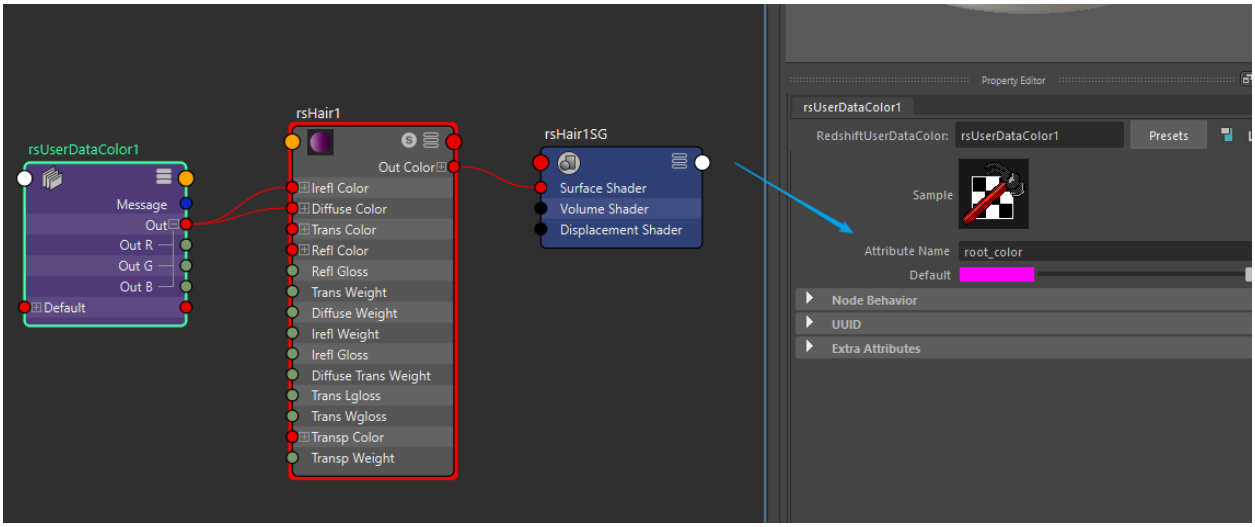
6.33 Maya Redshift 批量转贴图 rstexbin

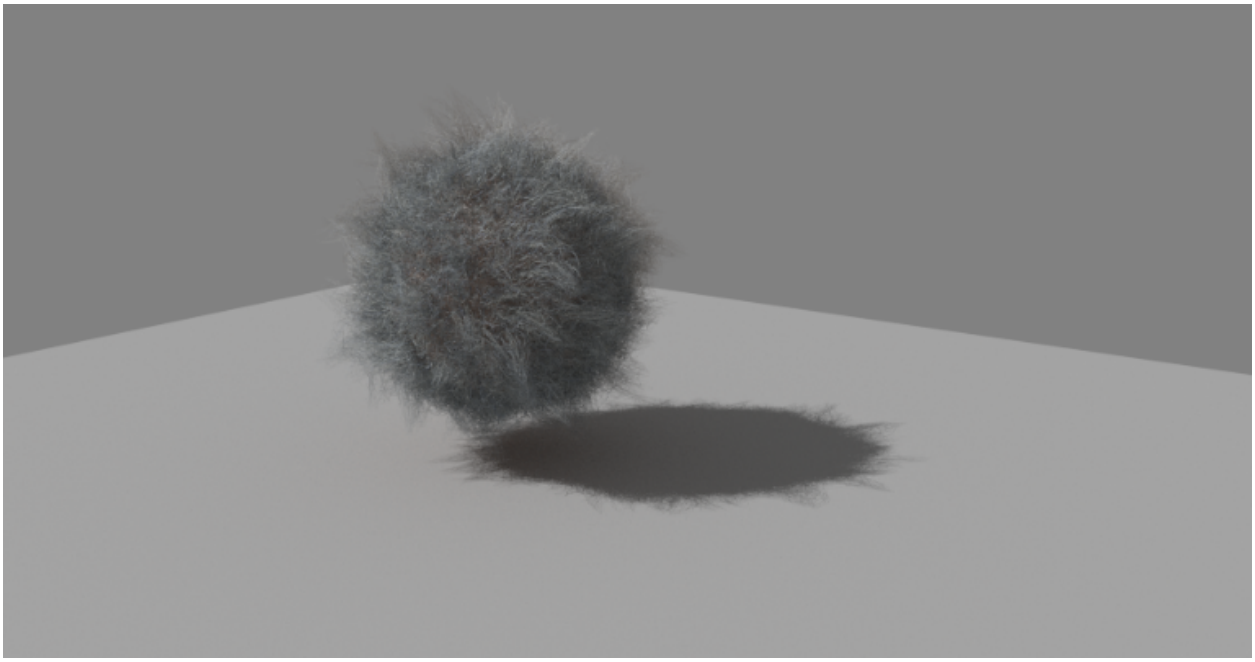
在 Redshift 安装路径 C:\ProgramData\Redshift\bin 下找到 redshiftTextureProcessor.exe，比如 D:\Texture 路径下有一批.jpg 贴图需要转 rstexbin，可以打开命令行窗口，写入下面的指令批量转换，有时候能解决贴图渲染卡住的问题。

```
C:\ProgramData\Redshift\bin\redshiftTextureProcessor.exe D:\Texture\*.jpg
```

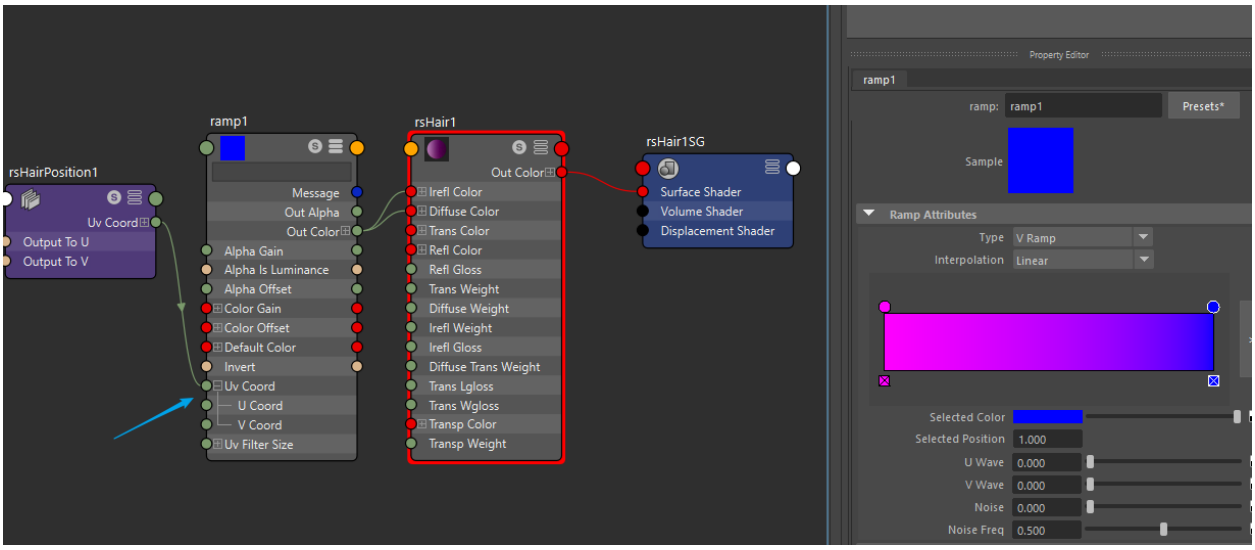
6.34 Maya Redshift 渲染 Xgen 毛发流程

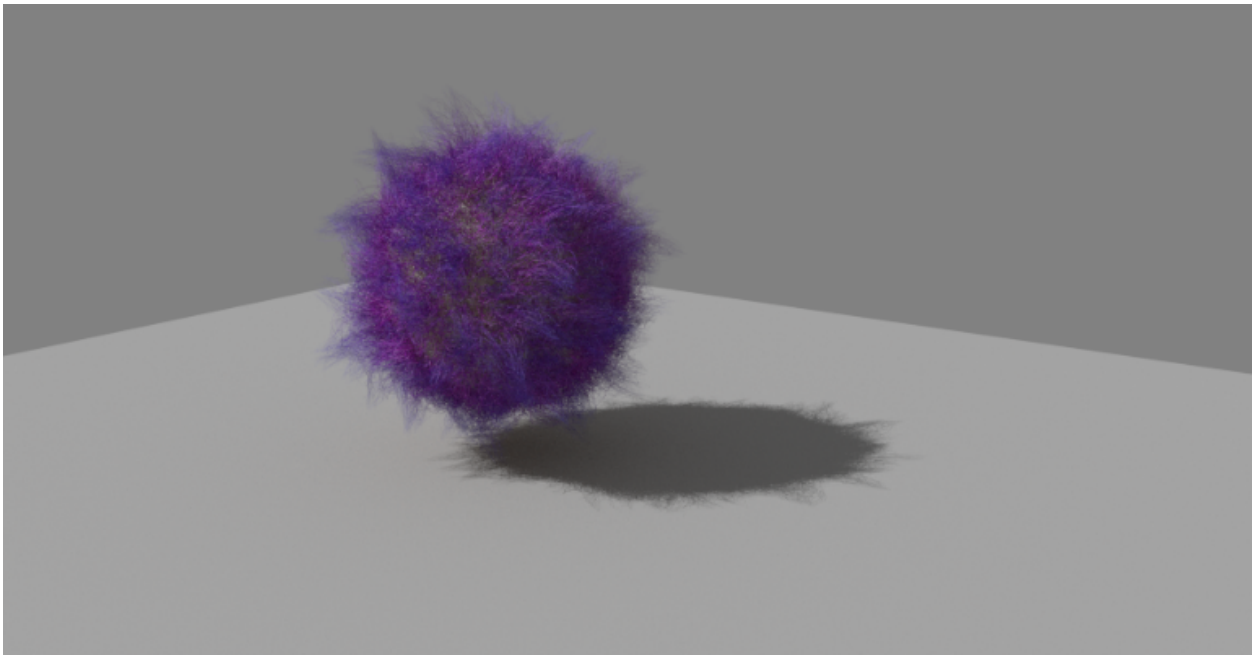
- 创建个球和地面，选择球 Import Preset From Library，选择 Sheep 预设
- 创建 Redshift Physical Light，修改 Light Type>Directional，Intensity Multiplier>1
- 创建 Redshift Dome Light，降低 Tint
- 创建 RedshiftHair 材质，选择 description 给上材质
- 修改 Xgen 面板 Renderer 为 Redshift
- 修改 Render Settings>GI>Brute Force，采样 32-128
- 创建 RedshiftUserDataColor 节点连接到 rsHair 节点





- 实现颜色过度，创建 RedshiftHairPosition 和 ramp[Texture] 节点连接





6.35 Maya 重命名工具

ls document 讲解

```
import maya.cmds as cmds

def renameTool(pattern, targets):
    if not targets:
        cmds.warning("Please select at least one object")
```

(下页继续)

(续上页)

```
        return None
    for target in targets:
        if not cmds.objExists(target):
            cmds.warning("%s is not exists" % target)
            continue
        print(target)
        cmds.rename(target, pattern)

rename("item", mc.ls(sl=True))
```

```
import maya.cmds as cmds

NUM_MATCH_SYMBOL = "#"

def renameTool(pattern, targets, start=1):
    """
    Naming function
        Arguments :
        pattern - str The naming pattern
        targets - list The need to renaming objects
        start - int The start number (default is 1)
    """
    newNameHead = ""
    newNameTail = ""
    # If start is negative, set it to zero
    if start < 0:
        start = 0
    # Find first sharp symbol position
    # If not found
    position = pattern.find(NUM_MATCH_SYMBOL)
    numberOfSharp = pattern.count(NUM_MATCH_SYMBOL)
    if position < 0:
        print("No \"#\" found, the number will suffix")
        newNameHead = pattern
        numberOfSharp = 1
    else:
        newNameHead = pattern.split(NUM_MATCH_SYMBOL)[0]
        newNameTail = "".join(pattern.split(NUM_MATCH_SYMBOL)[1:])
    counter = start
```

(下页继续)

(续上页)

```
    for target in targets:
        if not cmds.objExists(target):
            continue
        newName = newNameHead
        newName += str(counter).zfill(numberOfSharp)
        newName += newNameTail
        cmds.rename(target, newName)
        counter += 1
    return True

renameTool("item_###_grp", cmds.ls(sl=True), 10)
```

如何写成 package 使用

python 中执行 mel mel 中执行 python

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

try:
    from PySide import QtGui
    from PySide import QtCore
except ImportError:
    from PySide2 import QtWidgets as QtGui


class Example(QtGui.QDialog):
    def __init__(self, parent=None):
        super(Example, self).__init__(parent)
        self.initUI()

    def initUI(self):
        self.setGeometry(600, 300, 500, 500)
        self.setWindowTitle("Example")

if __name__ == "__main__":
    ex = Example()
    ex.show()
```

6.36 Maya 自定义工具架工具

load shelf

自定义工具架自定义工具架工具

1. 鼠标中键拖拽代码 2.Ctrl+Shift+ 鼠标左键点击菜单

Shelf Editor

Shelves Command Double Click Command Popup Menu Items

loadNewShelf

import pymel.core as pm

if not pm.shelfLayout(shelfName, q=True, ex=True)

6.37 Maya 用户界面

Maya 中自定义用户界面有很多种方案，MEL 可以写用户界面，cmds 也可以写用户界面，推荐使用内置的 PySide2 来写用户界面。

```
import maya.cmds as cmds

win = "myFirstWindow"

if cmds.window(win, q=1, exists=1):
    cmds.deleteUI(win, wnd=1)

if cmds.windowPref(win, q=1, exists=1):
    cmds.windowPref(win, r=1)

myWindow = cmds.window(win, w=400, h=300, t="My First Window Options:")

cmds.columnLayout()

for i in range(5):
    cmds.button(l="Button%d" % i, c="print(%d)" % i)

cmds.showWindow(myWindow)
```

6.38 Maya XGen Linux 资产切换 Win 版本

- notepad++ 文本修改.xgen 文件中的路径
- DG 模式
- Arnold 要先加载，mtoa 的版本要对应
- 修改 Xg Version 路径

CHAPTER 7

Nuke

NUKE 是由位于英国伦敦的 The Foundry 公司开发和销售的一款“基于节点”的数字影视合成软件，广泛应用于电影和电视的后期合成。NUKE 当前支持的操作系统为：Microsoft Windows，Mac OS X，以及 Linux。

Contents:

7.1 Nuke 执行后台命令脚本方案

两种后台命令脚本的模式

打开命令行可以通过标签-h 或者–help 查询帮助

```
"C:\Program Files\Nuke10.5v1\Nuke10.5.exe" -h
```

基本语法

./Nuke (–nukex) <switches> <script> <argv>

<switches> 标签 flag <script> .nk 文件的路径 <argv> 传入的参数可以中.nk 文件中以 [arg n] 来使用

简单的案例 “C:/Program Files/Nuke10.5v1/Nuke10.5.exe” “D:/test.nk” “C:/Program Files/Nuke10.5v1/Nuke10.5.exe” –nukex “D:/test.nk” 直接跟.nk 文件，表示打开这个合成文件

“C:/Program Files/Nuke10.5v1/Nuke10.5.exe” -x -F 1-80 -X Write1 -m 8 –cont “D:/test.nk” 标签-x 将.nk 文件输出标签-F 起始帧-结束帧标签-X 输出节点名称标签-m 设置线程数，默认占用电脑所有线程标签–cont 渲染出错继续

命令行输出多个 Write，创建一个 txt 文件，将扩展名改为 bat，内容写入 “C:/Program Files/Nuke10.5v1/Nuke10.5.exe” -x -F 1-80 -X Write1 -m 8 –cont “D:/test.nk” “C:/Program Files/Nuke10.5v1/Nuke10.5.exe” -x -F 1-80 -X Write2 -m 8 –cont “D:/test.nk” “C:/Program Files/Nuke10.5v1/Nuke10.5.exe” -x -F 1-80 -X Write3 -m 8 –cont “D:/test.nk”

写个渲染完自动关机的 bat，加 shutdown 关机命令 “C:/Program Files/Nuke10.5v1/Nuke10.5.exe” -x -F 1-80 -X Write1 -m 8 –cont “D:/test.nk” “C:/Program Files/Nuke10.5v1/Nuke10.5.exe” -x -F 1-80 -X Write2 -m 8 –cont “D:/test.nk” “C:/Program Files/Nuke10.5v1/Nuke10.5.exe” -x -F 1-80 -X Write3 -m 8 –cont “D:/test.nk” shutdown /f /s

-F range 的三种写法：A、A-B、A-BxC

标签-t 后面可以跟.py 文件，案例 “C:/Program Files/Nuke10.5v1/Nuke10.5.exe” -t “W:/houdini/scripts/python/slateMov/nkSlate.py”

另一种方案是通过“C:/Program Files/Nuke10.5v1/python.exe”“W:/houdini/scripts/python/slateMov/nkSlate.py”

.py 文件一般有两种写法，一种是通过代码生成或读取.nk 模板文件输出，另一种是不生成 nk 文件直接输出

分块合并的原理一般来说是几张图通过 Read 节点读入，然后连接不同的 transform 节点（注意 transform 参数需要按具体几张图来设置），最后通过 merge 节点将分块的图缝合拼接，再接 Write 节点输出一整张图（注意 nuke 不认中文或特殊字符的文件路径）

比如一张 2500x1250 的图被分 16 块，(156x1250x15 张 +160x1250x1 张) 渲染出了 16 张图，在 Nuke 中先创建一个 2500 宽 x1250 高的 constant 然后将 16 张图依次和 constant 背景 merge 中一起输出

算好每个 transform 的水平位置的平移值

subprocess 模块进程通信 subprocess.Popen 类初始化 ___init___(self, args, bufsize=0, executable=None, stdin=None, stdout=None, stderr=None, preexec_fn=None, close_fds=False, shell=False, cwd=None, env=None, universal_newlines=False, startupinfo=None, creationflags=0) 创建并返回一个子进程，并在这个进程中执行指定的程序。 实例化 Popen 可以通过许多参数详细定制子进程的环境，但是只有一个参数是必须的，即位置参数 args ，下面也会详细介绍剩余的具名参数。args：要执行的命令或可执行文件的路径。一个由字符串组成的序列（通常是列表），列表的第一个元素是可执行程序的路径，剩下的是传给这个程序的参数，如果没有要传给这个程序的参数，args 参数可以仅仅是一个字符串。

案例

```
import subprocess
# Start Houdini
subprocess.Popen("C:/Program Files/Side Effects Software/Houdini 16.5.378/bin/houdinifx.
↪exe")
# Start Nuke
subprocess.Popen("C:/Program Files/Nuke10.5v1/Nuke10.5.exe")
# Start NukeX
subprocess.Popen("'C:/Program Files/Nuke10.5v1/Nuke10.5.exe' -nukex")
# Start Nukex
```

(下页继续)

(续上页)

```
subprocess.Popen(["C:/Program Files/Nuke10.5v1/Nuke10.5.exe", "-nukex"])
# Nuke Write
subprocess.Popen(["C:/Program Files/Nuke10.5v1/Nuke10.5.exe", "-nukex -x -F 1", "A:/
↪working/101011621/b34dfc17-6395-4150-b310-7761074d26b6/Bin/tmp/7712/tmp370147.tga.nk",
↪"D:/2019/aaa/test.exr"])
# Python Write
subprocess.Popen(["C:/Program Files/Nuke10.5v1/Nuke10.5.exe", "-nukex", "-t", "D:/2019/
↪aaa/test.py", "A:/working/101011621/b34dfc17-6395-4150-b310-7761074d26b6/Bin/tmp/7712/
↪Tile", "D:/2019/aaa/test.exr"])
# subprocess.Popen(["C:/Program Files/Nuke10.5v1/python.exe", pyModule, nkPath, movPath,␣
↪fileStepName, reviewPath, USER, ext])

# nuke.scriptOpen(nk)
```

```
# test.py

import os
import re
import sys
import nuke

# print(sys.argv)

def abortSelected():
    return [i['selected'].setValue(False) for i in nuke.allNodes()]

abortSelected()
mergeNode = nuke.createNode("Merge2", inpanel=False)
abortSelected()
writeNode = nuke.createNode("Write", inpanel=False)
writeNode.setInput(0, mergeNode)
writeNode["file"].setValue(sys.argv[2])
abortSelected()
consNode = nuke.createNode("Constant", inpanel=False)
nuke.addFormat("%s %s %s" % (2500, 1250, "newRes"))
consNode["format"].setValue("newRes")

mergeNode.setInput(0, consNode)
```

(下页继续)

(续上页)

```
def sort_item(item):
    return int(re.search(r"(?<=__1x)\d+", item).group())

tempWidth = 0
for elem in sorted(os.listdir(sys.argv[1]), key=sort_item):
    # print(elem)
    i = sort_item(elem)
    print(i)
    res_check = re.search(r'(?<=__)\d+x\d+_[Ww]\d+[Hh]\d+(?=__.)', elem)
    cur_Width = re.search(r'(?<=_[Ww])\d+(?=[Hh])', res_check.group())
    cur_Height = re.search(r'(?<=[Hh])\d+$', res_check.group())
    # print(res_check.group())
    # print(cur_Width.group())
    # print(cur_Height.group())

    readNode = nuke.nodes.Read()
    readNode["file"].fromUserText(sys.argv[1] + "/" + elem)
    abortSelected()
    tranNode = nuke.createNode("Transform", inpanel=False)
    # print(tempWidth)
    tranNode.knob("translate").setValue((tempWidth, 0))

    tranNode.setInput(0, readNode)
    print(tranNode.name())
    mergeNode.setInput(i+2, tranNode)

    tempWidth += int(cur_Width.group())

# nuke.scriptSave("D:/2019/aaa/test.nk")
nuke.execute(writeNode, 1, 1, 1)
```

7.2 Nuke Python Callback 机制

修改创建节点的默认参数。

```
import nuke

nuke.knobDefault("Blur.size", "10")
```

7.3 Nuke 中心化配置插件

插件是否可以中心化，取决于插件是否按扩展开发的层级结构来处理，如果按照规范来开发的插件，可以通过修改 C:\Users\{用户名}\.nuke\init.py 文件来中心化插件。

```
import nuke

nuke.pluginAddPath("插件路径")
```

举个例子，可以从网络上下载一个插件

https://github.com/Psyop/Cryptomatte

下载完将其解压到 D 盘，可以通过下面代码将插件部署到 Nuke。

```
import nuke

nuke.pluginAddPath("D:/Cryptomatte")
```

7.4 Nuke 创建 Gizmo 流程

- Ctrl + G # 打组
- Ctrl + Alt + G # 解组

.gizmo 文件想在 Nuke 中可以使用，需要创建菜单来创建 gizmo，这样才可以按 tab 键找到节点。

7.5 Nuke Python 创建 Read 节点小技巧

用过 Nuke 的童鞋都知道我们可以拖拽 mov 或者文件夹素材到 Nuke 的 Node Graph 中，它可以自动分类这些素材形成几个 Read 节点，并自动识别素材的帧数范围以及 Metadata 关于素材的所有信息。

那么在编写工具的时候如果遇到需要使用代码来创建 Read 节点并读取素材该怎么办呢？正常思维能想到的就是先创建 Read 节点，然后设置节点上的 file 参数，尝试过之后你会发现得不到你想要的结果。正确的方式是通过节点参数实例的 fromUserText 方法来实现。

我们先执行下面的代码看下 fromUserText 方法的定义

```
import nuke
readNode = nuke.nodes.Read()
print(help(readNode["file"].fromUserText))
```

执行完之后会得到下面的帮助文档

```
# Result: Help on built-in function fromUserText:

fromUserText(...)
    self.fromUserText(s) -> None.
    Assign string to knob, parses frame range off the end and opens file to get set the␣
↪format.
    @param s: String to assign.
    @return: None.

None
```

意味着我需要传输素材的一个路径进去，比如一个 mov，可以这样创建 Read 节点

```
import nuke

movPath = r"D:\tests_nuke_read\XX_XXX_v001.mov"
readNode = nuke.nodes.Read()
readNode["file"].fromUserText(movPath)
```

你会发现和拖拽进来的素材丝毫不差，那么如果是 exr 序列呢？可以尝试将.mov 改成 ####.exr 或者%04d.exr，这样都得不到正确的结果，我们想获取到正确的序列，需要通过 nuke.getFileNameList 方法来解析一个文件夹中有哪些序列，如果一个文件夹中存在三套序列，它将返回一个三个元素组成的列表，类似下面这样

```
import nuke

exrPath = r"D:\tests_nuke_read\render"
print(nuke.getFileNameList(exrPath))
# Result: ['BillowySmoke1.####.exr 700-1300', 'finalRender_Smoke_Right_bg.####.exr 1-200
↪', 'qiangjiaohuo.####.exr 100-700']
```

其中每个元素的结尾都标明了对应序列的帧数范围，这样我们去创建 Read 节点就好办了，可以一次性创建三个 Read 节点

```
import nuke

exrPath = r"D:\tests_nuke_read\render"
seqs = nuke.getFileNameList(exrPath)

for seq in seqs:
    seqPath = "%s/%s" % (exrPath, seq)
    readNode = nuke.nodes.Read()
```

(下页继续)

(续上页)

```
    readNode["file"].fromUserText(seqPath)
```

完美

7.6 Nuke 自定义菜单

在 Nuke 中自定义菜单相对比较简单，都是通过 Python 来实现的，打开 Windows>Script Editor 查看 nuke.menu 函数的帮助文档。

- “Nuke” 获取主菜单的实例对象
- “Pane” 获取面板菜单的实例对象
- “Nodes” 获取工具栏菜单的实例对象
- “Properties” 获取参数菜单的实例对象
- “Animation” 获取动画曲线参数菜单的实例对象
- “Viewer” 获取视图菜单的实例对象
- “Node Graph” 获取节点网络菜单的实例对象
- “Axis” 获取 Axis 节点上 File 或 Snap 菜单的实例对象

可以通过 nuke.menu(“Nuke”) 来获取菜单实例，再次查看实例对象的属性与方法。

```
import nuke

mainMenu = nuke.menu("Nuke")
print(dir(mainMenu))

# Result: ['__class__', '__delattr__', '__doc__', '__format__', '__getattribute__', '__
↪hash__', '__init__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__
↪', '__sizeof__', '__str__', '__subclasshook__', 'action', 'addAction', 'addCommand',
↪'addMenu', 'addSeparator', 'clearMenu', 'findItem', 'icon', 'invoke', 'items', 'menu',
↪'name', 'removeItem', 'script', 'setEnabled', 'setIcon', 'setScript', 'setShortcut',
↪'setVisible', 'shortcut', 'updateMenuItems']
```

其中 addMenu 方法可以在此菜单对象上添加一个菜单，返回此菜单的实例对象，addCommand 可以添加一个执行命令的菜单。

- Menu class
- MenuItem class

```
import nuke

mainMenu = nuke.menu("Nuke")

toolMenu = mainMenu.addMenu("DO-VFX")
toolMenu.addCommand("Choose Task", "print(100)", shortcut="t")
```

shortcut 参数是设置菜单执行的快捷方式，如果想加上 Ctrl，Alt 或者 Shift，可以通过下面两种方案。

```
import nuke

mainMenu = nuke.menu("Nuke")

toolMenu = mainMenu.addMenu("DO-VFX")
toolMenu.addCommand("Choose Task", "print(100)", shortcut="ctrl+t")
toolMenu.addCommand("Choose Task", "print(100)", shortcut="^t")
toolMenu.addCommand("Choose Task", "print(100)", shortcut="alt+t")
toolMenu.addCommand("Choose Task", "print(100)", shortcut="#t")
toolMenu.addCommand("Choose Task", "print(100)", shortcut="shift+t")
toolMenu.addCommand("Choose Task", "print(100)", shortcut="+t")
```

nuke.menu(“Nuke”) 是主菜单的实例对象，如果想创建左侧工具栏工具，可以通过 nuke.menu(“Nodes”) 或者 nuke.toolbar(“Nodes”) 来获取实例对象。

```
import nuke

toolbar = nuke.toolbar("Nodes")
doToolbar = toolbar.addMenu("DO-VFX", "设置图标路径")
doToolbar.addCommand("Choose Task", "print(100)")
```

可以通过 Python 来执行某一个菜单。

```
import nuke

mainMenu = nuke.menu("Nuke")
mainMenu.findItem("Edit/Clone").invoke()
```

修改创建节点的默认参数。

```
import nuke

nuke.knobDefault("Blur.size", "10")
```

7.7 Nuke 自定义节点参数

Nuke 官方给了一些节点参数控件，可以通过简单添加来自定义节点参数。

- Floating Point Slider
- 2d Position Knob
- 3d Position Knob
- Width/Height Knob
- Bounding Box Knob
- Size Knob
- UV Coordinate Knob
- Integer Knob
- RGB Color Knob
- RGBA Color Knob
- Check Box
- TCL Script Button
- Python Script Button
- Python Custom
- Pulldown Choice
- Cascading Pulldown Choice
- Command Menu
- Text input Knob
- Filename
- Tab
- Group
- Text
- Divider Line
- Obsolete Knob

这些参数控件都非常简单，使用起来也很简单，只要创建过一次应该都可以上手掌握，但有几个细节值得注意:

- 我们往往会先创建一个 Tab，将自定义创建的参数归类放在一起，默认直接创建 knob 会在 User 选项中，也可以后期修改 User 的命名。

- Hide 可以隐藏参数控件。
- Python Custom 会在显示节点参数的时候初始化代码，用于事件触发。
- Start new line 可以让控件在不换行，比如并行两个 Button。
- 参数关联。

创建一个 NoOp，右键参数菜单 Manage User Knobs…>Add>Python Script Button…

Python Script Button 中添加可执行代码。

通过 Python 脚本添加节点参数

```
import nuke

tkTest = nuke.Text_Knob("test", "Text", "This is Text Knob Test")
node = nuke.createNode("Write")
node.addKnob(tkTest)
```

```
nuke.PyScript_Knob()
nuke.Tab_Knob()
```

参考文档:

- https://www.foundry.com/products/nuke/developers

7.8 Nuke 自定义工具集

7.9 Nuke 守护线程自动保存文件

7.10 Nuke 依据版本来加载不同插件

我们知道 Nuke 通常是通过此文件 `C:\Users\{USERNAME}\.nuke\init.py` 来添加插件路径。

```
import nuke

nuke.pluginAddPath("插件 1 路径")
nuke.pluginAddPath("插件 2 路径")
```

这里有个问题是不同的 Nuke 版本走的都是这个 init.py 文件来初始化插件工具，那么遇到不同版本中的插件该怎么办呢?

比如有个插件有两个版本，分别支持 NukeX 10.5v1 和 NukeX 11.2v3，实际解决方案很简单，在加载前做个版本判断即可。

首先我们得知道通过什么来获取 Nuke 版本

```
import nuke

for s in dir(nuke):

    if s.isupper():
        print(s, eval("nuke.%s" % s))

# ('ADD_VIEWS', 0)
# ('AFTER_CONST', 21)
# ('AFTER_LINEAR', 22)
# ('ALL', 1)
# ('ALWAYS_SAVE', 1048576)
# ('BEFORE_CONST', 19)
# ('BEFORE_LINEAR', 20)
# ('BREAK', 18)
# ('CATMULL_ROM', 3)
# ('CONSTANT', 1)
# ('CUBIC', 4)
# ('DISABLED', 128)
# ('DONT_CREATE_VIEWS', 2)
# ('DONT_SAVE_TO_NODEPRESET', 0)
# ('DO_NOT_WRITE', 512)
# ('ENDLINE', 8192)
# ('EXE_PATH', 'C:/Program Files/Nuke10.5v1/Nuke10.5.exe')
# ('EXPAND_TO_WIDTH', 0)
# ('EXPRESSIONS', 1)
# ('FLOAT', 5)
# ('FONT', 4)
# ('GEO', 16)
# ('GUI', True)
# ('HIDDEN_INPUTS', 4)
# ('HORIZONTAL', 17)
# ('IMAGE', 1)
# ('INPUTS', 2)
# ('INT16', 3)
# ('INT8', 2)
# ('INTERACTIVE', True)
# ('INVALIDHINT', -1)
# ('INVISIBLE', 1024)
```

(下页继续)

(续上页)

```
# ('KNOB_CHANGED_RECURSIVE', 134217728)
# ('LINEAR', 2)
# ('LOG', 4)
# ('MATCH_CLASS', 0)
# ('MATCH_COLOR', 2)
# ('MATCH_LABEL', 1)
# ('MONITOR', 0)
# ('NODIR', 2)
# ('NO_ANIMATION', 256)
# ('NO_CHECKMARKS', 1)
# ('NO_MULTIVIEW', 1073741824)
# ('NO_POSTAGESTAMPS', False)
# ('NO_UNDO', 524288)
# ('NUKE_VERSION_DATE', 'Dec  6 2016')
# ('NUKE_VERSION_MAJOR', 10)
# ('NUKE_VERSION_MINOR', 5)
# ('NUKE_VERSION_PHASE', '')
# ('NUKE_VERSION_PHASENUMBER', 0)
# ('NUKE_VERSION_RELEASE', 1)
# ('NUKE_VERSION_STRING', '10.5v1')
# ('NUM_CPUS', 48)
# ('NUM_INTERPOLATIONS', 5)
# ('PLUGIN_EXT', 'dll')
# ('PREPEND', 8)
# ('PROFILE_ENGINE', 3)
# ('PROFILE_REQUEST', 2)
# ('PROFILE_STORE', 0)
# ('PROFILE_VALIDATE', 1)
# ('PYTHON', 32)
# ('READ_ONLY', 268435456)
# ('REPLACE', 1)
# ('REPLACE_VIEWS', 1)
# ('SAVE_MENU', 33554432)
# ('SCRIPT', 2)
# ('SMOOTH', 0)
# ('STARTLINE', 4096)
# ('STRIP_CASCADE_PREFIX', 4)
# ('TABBEGINCLOSEDGROUP', 2)
# ('TABBEGINGROUP', 1)
# ('TABENDGROUP', -1)
```

(下页继续)

(续上页)

```
# ('TABKNOB', 0)
# ('THREADS', 48)
# ('TO_SCRIPT', 1)
# ('TO_VALUE', 2)
# ('USER_SET_SLOPE', 16)
# ('VIEWER', 1)
# ('VIEW_NAMES', 'input/view_names')
# ('WRITE_ALL', 8)
# ('WRITE_NON_DEFAULT_ONLY', 16)
# ('WRITE_USER_KNOB_DEFS', 4)
```

从代码获得的结果可以使用 NUKE_VERSION_ 相关属性来获取当前 NUKE 版本。

```
# ('NUKE_VERSION_DATE', 'Dec  6 2016')
# ('NUKE_VERSION_MAJOR', 10)
# ('NUKE_VERSION_MINOR', 5)
# ('NUKE_VERSION_PHASE', '')
# ('NUKE_VERSION_PHASENUMBER', 0)
# ('NUKE_VERSION_RELEASE', 1)
# ('NUKE_VERSION_STRING', '10.5v1')
```

能获取 Nuke 当前使用的版本，init.py 中加载插件代码就好处理了，不需要太精确就使用 nuke.NUKE_VERSION_MAJOR，需要精确一些就用 nuke.NUKE_VERSION_STRING。

```
import nuke

ver = nuke.NUKE_VERSION_MAJOR

if ver == 10:
    nuke.pluginAddPath("NukeX 10 插件 1 路径")
    nuke.pluginAddPath("NukeX 10 插件 2 路径")
elif ver == 11:
    nuke.pluginAddPath("NukeX 11 插件 1 路径")
    nuke.pluginAddPath("NukeX 11 插件 2 路径")
else:
    pass
```

```
import nuke

ver = nuke.NUKE_VERSION_STRING
```

(下页继续)

(续上页)

```
if ver == "10.5v1":
    nuke.pluginAddPath("NukeX 10.5v1 插件 1 路径")
    nuke.pluginAddPath("NukeX 10.5v1 插件 2 路径")
elif ver == "11.2v3":
    nuke.pluginAddPath("NukeX 11.2v3 插件 1 路径")
    nuke.pluginAddPath("NukeX 11.2v3 插件 2 路径")
else:
    pass
```

7.11 Nuke 开发环境搭建

VS Code

Auto Complete

NUKE_INTERACTIIVE=1

7.12 Nuke 扩展开发插件的层级结构

扩展开发的层级结构大概如下：

```
package1
    __init__.py
    module1.py
package2
    __init__.py
    module2.py
gizmo
    xxx1.gizmo
    xxx2.gizmo
icons
    xxx1.png
    xxx2.svg
xxx1.gizmo
xxx1.png
module1.py
module2.py
```

(下页继续)

(续上页)

```
init.py
menu.py
```

他们围绕着一个方法来控制路径查询机制，可以修改 C:\Users\{用户名}\.nuke\init.py 文件。

```
import nuke

nuke.pluginAddPath("插件路径")
```

menu.py 文件中一般写自定义菜单的代码，这样在 Nuke 启动的时候会自动加载菜单，而菜单中的执行的代码一般都是此文件夹下的模块或包的文件以及图标或 Gizmo 文件。

init.py 文件主要用来处理路径问题，如果不想 gizmo 和 icons 中文件通过 gizmo/xxx1.gizmo 来处理的话，可以在 init.py 写入如下脚本。

```
import nuke

nuke.pluginAddPath("gizmo")
nuke.pluginAddPath("icons")
```

7.13 Nuke 开发者文档

- HieroPythonDevGuide
- NDKDevGuide
- NDKReference
- NukePythonAPI
- NukePythonDevGuide

参考文档:

- Nuke Python Developer's Guide: https://learn.foundry.com/nuke/developers/latest/pythondevguide/
- Nuke Python API: https://learn.foundry.com/nuke/developers/latest/pythonreference/
- Nuke Pedia: http://www.nukepedia.com/
- Andrea Geremia: http://www.andreageremia.it/tutorial.html

7.14 Nuke 主目录

```
C:\Users\{USERNAME}\.nuke
```

- init.py
- menu.py

7.15 Nuke 模块：nuke

- 一般方法

```
import nuke

print(nuke)
print(type(nuke))
print(nuke.__file__)
print(dir(nuke))
# Result: <module 'nuke' from 'C:/Program Files/Nuke10.5v1/plugins\nuke\__init__.pyc'>
<type 'module'>
C:/Program Files/Nuke10.5v1/plugins\nuke\__init__.pyc
['AColor_Knob', 'ADD_VIEWS', 'AFTER_CONST', 'AFTER_LINEAR', 'ALL', 'ALWAYS_SAVE',
↪'AnimationCurve', 'AnimationKey', 'Array_Knob', 'Axis_Knob', 'BBox_Knob', 'BEFORE_CONST
↪', 'BEFORE_LINEAR', 'BREAK', 'BackdropNode', 'BeginTabGroup_Knob', 'Bitmask_Knob',
↪'Boolean_Knob', 'Box', 'Box3_Knob', 'CATMULL_ROM', 'CONSTANT', 'CUBIC', 'CancelledError
↪', 'CascadingEnumeration_Knob', 'ChannelMask_Knob', 'Channel_Knob', 'ColorChip_Knob',
↪'Color_Knob', 'ColorspaceLookupError', 'DISABLED', 'DONT_CREATE_VIEWS', 'DONT_SAVE_TO_
↪NODEPRESET', 'DO_NOT_WRITE', 'Disable_Knob', 'Double_Knob', 'ENDLINE', 'EXE_PATH',
↪'EXPAND_TO_WIDTH', 'EXPRESSIONS', 'EditableEnumeration_Knob', 'EndTabGroup_Knob',
↪'Enumeration_Knob', 'EvalString_Knob', 'Eyedropper_Knob', 'FLOAT', 'FONT', 'File_Knob',
↪ 'FnPySingleton', 'Font_Knob', 'Format', 'Format_Knob', 'FrameRange', 'FrameRanges',
↪'FreeType_Knob', 'GEO', 'GUI', 'GeoSelect_Knob', 'Gizmo', 'GlobalsEnvironment', 'Group
↪', 'HIDDEN_INPUTS', 'HORIZONTAL', 'Hash', 'Help_Knob', 'Histogram_Knob', 'IArray_Knob',
↪ 'IMAGE', 'INPUTS', 'INT16', 'INT8', 'INTERACTIVE', 'INVALIDHINT', 'INVISIBLE', 'Info',
↪ 'Int_Knob', 'KNOB_CHANGED_RECURSIVE', 'Keyer_Knob', 'Knob', 'KnobType', 'LINEAR', 'LOG
↪', 'Layer', 'Link_Knob', 'LinkableKnobInfo', 'LookupCurves_Knob', 'Lut', 'MATCH_CLASS',
↪ 'MATCH_COLOR', 'MATCH_LABEL', 'MONITOR', 'Menu', 'MenuBar', 'MenuItem', 'MultiView_
↪Knob', 'Multiline_Eval_String_Knob', 'NODIR', 'NO_ANIMATION', 'NO_CHECKMARKS', 'NO_
↪MULTIVIEW', 'NO_POSTAGESTAMPS', 'NO_UNDO', 'NUKE_VERSION_DATE', 'NUKE_VERSION_MAJOR',
↪'NUKE_VERSION_MINOR', 'NUKE_VERSION_PHASE', 'NUKE_VERSION_PHASENUMBER', 'NUKE_VERSION_
↪RELEASE', 'NUKE_VERSION_STRING', 'NUM_CPUS', 'NUM_INTERPOLATIONS', 'Node',
↪'NodeConstructor', 'Nodes', 'Obsolete_Knob', 'OneView_Knob', 'OutputContext', 'PLUGIN_
↪EXT', 'PREPEND', 'PROFILE_ENGINE', 'PROFILE_REQUEST', 'PROFILE_STORE', 'PROFILE_
↪VALIDATE', 'PYTHON', 'Panel', 'PanelNode', 'Password_Knob', 'Precomp', 'ProgressTask',
↪'Pulldown_Knob', 'PyCustom_Knob', 'PyScript_Knob', 'PythonCustomKnob', 'PythonKnob',
↪'READ_ONLY', 'REPLACE', 'REPLACE_VIEWS', 'Radio_Knob', 'Range_Knob', 'Root',
```

(下页继续)

(续上页)

```
# 获取全局属性
print(nuke.root())
nuke.zoom()
# 返回所有选中节点实例的列表
print(nuke.selectedNodes())
# 返回名字为 Write1 的节点实例
print(nuke.toNode("Write1"))
# 返回当前节点实例
print(nuke.thisNode())
# 创建节点实例两种方案，NodeClass 是节点类型，可以通过在具体节点上按 i 键或者通过 Class() 方法查看节点类型
print(nuke.createNode("NodeClass"))
print(nuke.nodes.NodeClass())
# 返回某一类型所有节点实例列表
print(nuke.allNodes("NodeClass"))
# 返回所有插件路径
print(nuke.pluginPath())
```

- Callback 函数

```
nuke.addAfterRender()
nuke.addOnUserCreate()
nuke.addKnobChange()
```

- 节点实例对象属性与方法

nuke 模块中有很多函数可以返回节点实例对象，下面是一些常用的函数。

```
nuke.selectedNodes()
nuke.toNode()
nuke.thisNode()
nuke.createNode()
nuke.nodes.Read()
```

有了节点实例对象就可以获取关于节点数据类型属性以及方法。

[‘Class’ , ‘___class___’ , ‘___delattr___’ , ‘___doc___’ , ‘___format___’ , ‘___getattribute___’ , ‘___getitem___’ , ‘___hash___’ , ‘___init___’ , ‘___len___’ , ‘___new___’ , ‘___reduce___’ , ‘___reduce_ex___’ , ‘___repr___’ , ‘___setattr___’ , ‘___sizeof___’ , ‘___str___’ , ‘___subclasshook___’ , ‘addKnob’ , ‘allKnobs’ , ‘autoplace’ , ‘bbox’ , ‘canSetInput’ , ‘channels’ , ‘clones’ , ‘connectInput’ , ‘deepSample’ , ‘deepSampleCount’ , ‘dependencies’ , ‘dependent’ , ‘error’ , ‘fileDependencies’ , ‘firstFrame’ , ‘forceUpdateLocalization’ , ‘forceValidate’ , ‘format’ , ‘frameRange’ , ‘fullName’ , ‘getNumKnobs’

, 'hasError', 'height', 'help', 'hideControlPanel', 'input', 'inputs', 'isLocalizationOutdated', 'isLocalized', 'isSelected', 'knob', 'knobs', 'lastFrame', 'linkableKnobs', 'localizationProgress', 'maxInputs', 'maxOutputs', 'maximumInputs', 'maximumOutputs', 'metadata', 'minInputs', 'minimumInputs', 'name', 'numKnobs', 'opHashes', 'optionalInput', 'performanceInfo', 'pixelAspect', 'proxy', 'readKnobs', 'redraw', 'removeKnob', 'resetKnobsToDefault', 'rootNode', 'running', 'sample', 'screenHeight', 'screenWidth', 'selectOnly', 'setInput', 'setName', 'setSelected', 'setTab', 'setXYpos', 'setXpos', 'setYpos', 'showControlPanel', 'showInfo', 'shown', 'treeHasError', 'upstreamFrameRange', 'width', 'writeKnobs', 'xpos', 'ypos']

```
# 返回节点类型
node.Class()
# 节点在节点网络中的位置相关方法
setXYpos, setXpos, setYpos, xpos, ypos
# 添加自定义参数
addKnob
```

- 参数实例对象属性与方法

获取节点参数实例对象有下面两种常用的方法。

```
node["<parm_name>"]
node.knob("<parm_name>")
nuke.thisKnob()
```

有了节点参数实例对象就可以获取关于参数数据类型属性以及方法。

['Class', '__class__', '__delattr__', '__doc__', '__format__', '__getattribute__', '__hash__', '__init__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'clearAnimated', 'clearFlag', 'critical', 'debug', 'enabled', 'error', 'evaluate', 'fromScript', 'fromUserText', 'fullyQualifiedName', 'getDerivative', 'getEvaluatedValue', 'getFlag', 'getIntegral', 'getKeyIndex', 'getKeyList', 'getKeyTime', 'getNthDerivative', 'getNumKeys', 'getText', 'getValue', 'getValueAt', 'hasExpression', 'isAnimated', 'isKey', 'isKeyAt', 'label', 'name', 'node', 'removeKey', 'removeKeyAt', 'setAnimated', 'setEnabled', 'setExpression', 'setFlag', 'setLabel', 'setName', 'setText', 'setTooltip', 'setValue', 'setValueAt', 'setVisible', 'splitView', 'toScript', 'tooltip', 'unsplitView', 'value', 'visible', 'warning']

```
# 返回参数类型
parm.Class()
# 读取写入操作相关方法
getValue, getValueAt, setValue, setValueAt, value
```

参考文档:

- https://www.foundry.com/products/nuke/developers

7.16 Nuke 模块：nukescripts

```
import nukescripts

print(nukescripts)
print(type(nukescripts))
print(nukescripts.__file__)
print(dir(nukescripts))
# Result: <module 'nukescripts' from 'C:/Program Files/Nuke10.5v1/plugins\nukescripts\__
↪init__.pyc'>
<type 'module'>
C:/Program Files/Nuke10.5v1/plugins\nukescripts\__init__.pyc
['ClassInfo', 'CreateNodePresetsPanel', 'CreateToolsetsPanel', 'DialogState',
↪'ExecuteDialog', 'FlipbookApplication', 'FlipbookDialog', 'FlipbookFactory',
↪'FlipbookLUTPathRegistry', 'FrameRangePanel', 'InputHighlighter', 'LicensedDeclarative
↪', 'LicensedMultimedia', 'LicensedOpenVG', 'LineNumberArea', 'LinkToTrackPanel', 'MSG',
↪ 'MetaFunction', 'NukeProfiler', 'POINT', 'PrecompOptions', 'PrecompOptionsDialog',
↪'PresetsDeletePanel', 'PresetsLoadPanel', 'Property', 'PySide', 'PythonPanel',
↪'PythonPanelKnobChanged', 'QAbstractAnimation', 'QAbstractButton',
↪'QAbstractEventDispatcher', 'QAbstractFileEngine', 'QAbstractFileEngineHandler',
↪'QAbstractFileEngineIterator', 'QAbstractGraphicsShapeItem', 'QAbstractItemDelegate',
↪'QAbstractItemModel', 'QAbstractItemView', 'QAbstractListModel',
↪'QAbstractPageSetupDialog', 'QAbstractPrintDialog', 'QAbstractProxyModel',
↪'QAbstractScrollArea', 'QAbstractSlider', 'QAbstractSpinBox', 'QAbstractState',
↪'QAbstractTableModel', 'QAbstractTextDocumentLayout', 'QAbstractTransition',
↪'QAccessibleEvent', 'QAction', 'QActionEvent', 'QActionGroup', 'QAnimationGroup',
↪'QApplication', 'QBasicTimer', 'QBitArray', 'QBitmap', 'QBoxLayout', 'QBrush', 'QBuffer
↪', 'QButtonGroup', 'QByteArray', 'QByteArrayMatcher', 'QCDEStyle', 'QCalendarWidget',
↪'QCheckBox', 'QChildEvent', 'QCleanlooksStyle', 'QClipboard', 'QClipboardEvent',
↪'QCloseEvent', 'QColor', 'QColorDialog', 'QColumnView', 'QComboBox',
↪'QCommandLinkButton', 'QCommonStyle', 'QCompleter', 'QConicalGradient',
↪'QContextMenuEvent', 'QCoreApplication', 'QCryptographicHash', 'QCursor', 'QDataStream
↪', 'QDataWidgetMapper', 'QDate', 'QDateEdit', 'QDateTime', 'QDateTimeEdit',
↪'QDesktopServices', 'QDesktopWidget', 'QDial', 'QDialog', 'QDialogButtonBox', 'QDir',
↪'QDirIterator', 'QDirModel', 'QDockWidget', 'QDoubleSpinBox', 'QDoubleValidator',
↪'QDrag', 'QDragEnterEvent', 'QDragLeaveEvent', 'QDragMoveEvent', 'QDropEvent',
↪'QDynamicPropertyChangeEvent', 'QEasingCurve', 'QElapsedTimer', 'QErrorMessage',
↪'QEvent', 'QEventLoop', 'QEventTransition', 'QFSFileEngine', 'QFactoryInterface',
↪'QFile', 'QFileDialog', 'QFileIconProvider', 'QFileInfo', 'QFileOpenEvent',
↪'QFileSystemModel', 'QFileSystemWatcher', 'QFinalState', 'QFocusEvent', 'QFocusFrame',
↪'QFont', 'QFontComboBox', 'QFontDatabase', 'QFontDialog', 'QFontInfo', 'QFontMetrics',
↪'QFontMetricsF', 'QFormLayout', 'QFrame', 'QGenericArgument', 'QGenericReturnArgument',
↪'QGesture', 'QGestureEvent', 'QGestureRecognizer', 'QGradient', 'QGraphicsAnchor',
↪'QGraphicsAnchorLayout', 'QGraphicsBlurEffect', 'QGraphicsColorizeEffect',
↪'QGraphicsDropShadowEffect', 'QGraphicsEffect', 'QGraphicsEllipseItem',
↪'QGraphicsGridLayout', 'QGraphicsItem', 'QGraphicsItemAnimation', 'QGraphicsItemGroup',
```

(下页继续)

(续上页)

7.17 Nuke 插件：Peregrine Labs Bokeh

和 Yeti 同一家出品的 Nuke 景深插件

- http://documentation.peregrinelabs.com/bokeh/

7.18 Nuke 插件：Crytomatte

https://github.com/Psyop/Cryptomatte

7.19 Nuke 插件：DFT

Digital Film Tools(DFT)

7.20 Nuke 插件：FilmConvert

FilmConvert Pro

7.21 Nuke 插件：Reel Smart Motion Blur

运动模块插件

7.22 Nuke 偏好配置

- 自定义布局

调整好自己喜欢的窗口布局之后，点击菜单 Workspace>Save Workspace…，给布局起个名字比如叫 blablabla。

```
.nuke/Workspaces/Nuke/blablabla.xml
```

保存完成之后点击菜单 Edit>Preferences>Behaviors>Startup>startup workspace>blablabla。

- 脚本编辑器

Edit>Preferences>Panels>clear input window on successful script execution 选项去勾，在执行完代码，代码就不会丢失。

- 临时文件

Edit>Preferences>Performance>Caching>temp directory 修改临时文件路径

- 兼容路径

Edit>Preferences>General>Path Substitutions，支持 OSX、Windows、Linux，这里配置了路径之后，节点上文件路径可以简写自动匹配。

7.23 Nuke Python 命令行传参的两种方案

后台调用 Nuke 的 Python 执行环境有三种方案，打开命令行窗口，下面的三句脚本都是可以打开 Nuke 的 Python 执行环境。

```
"C:\Program Files\Nuke10.5v1\python.exe" blablabla.py arg1 arg2 arg3
"C:\Program Files\Nuke10.5v1\Nuke10.5.exe" -t blablabla.py arg1 arg2 arg3
"C:\Program Files\Nuke10.5v1\Nuke10.5.exe" -x blablabla.py arg1 arg2 arg3
```

命令行窗口给 py 文件传参有两种方案，一种最普通的方式是通过 sys.argv 来获取参数，但是在三种执行环境中都有一些问题，第一种执行环境下 blablabla.py 文件中 import nuke 和 sys.argv 得有先后顺序。

```
import sys
print(sys.argv)

import nuke
print(nuke.__file__)
```

上面的代码是可以正常传递命令行参数的，但下面这种情况就不行，你会得到空列表。

```
import sys
import nuke

print(sys.argv)
print(nuke.__file__)
```

第二种和第三种执行环境下不存在这样的问题，但是对于数字的参数，它会遇到警告，这种参数通过 sys.argv 存储不了。比如：

```
"C:\Program Files\Nuke10.5v1\Nuke10.5.exe" -t blablabla.py 101 200
```

WARNING: The command line argument '101 200' will be used as a Frame Range argument and will not be forward to the python sys.argv. To define a frame range argument use the -F

option.

此时可以通过 nuke.rawArgs 来获取就是 OK 的。

7.24 Nuke Python 导入 nk 文件的几种方式

基于流程的话经常要通过代码来处理 nk 文件导入导出的问题，下面总结一些用法，不能以偏概全。

```
import nuke

path = "D:/blablabla.nk"

# 清空当前场景
nuke.scriptClear()
# 打开 nk 文件，如果当前场景为空，会直接打开此 nk 文件，如果场景不为空，则会另开 Nuke 软件打开，功能同菜单 File>Open Comp...
nuke.scriptOpen(path)
# 在当前场景中导入 nk 文件中所有节点，不管当前场景是否为空，导入成功导入的所有节点会是选择状态，功能同菜单 File>Insert Comp Nodes...
nuke.nodePaste(path)
# 在当前场景中导入 nk 文件中所有节点，不管当前场景是否为空，导入成功没有节点是被选中状态。
nuke.scriptReadFile(path)
# 同上
nuke.scriptReadText(path)
# 同上
nuke.scriptSource(path)
# 将当前场景文件存储到 path 设置的路径，功能同菜单 File>Save Comp
nuke.scriptSave(path)
# 功能同菜单 File>Save Comp As...
nuke.scriptSaveAs()
nuke.scriptSaveAndClear()
# 功能同菜单 File>Close Comp
nuke.scriptClose()
# 功能同菜单 File>Quit
nuke.scriptExit()
# 新建 Nuke，功能同菜单 File>New Comp...
nuke.scriptNew()
```

7.25 Nuke Python Panel

```
print(type(nuke.Panel))
print(dir(nuke.Panel))
# <type 'type'>
# ['__class__', '__delattr__', '__doc__', '__format__', '__getattribute__', '__hash__',
↪'__init__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__
↪sizeof__', '__str__', '__subclasshook__', 'addBooleanCheckBox', 'addButton',
↪'addClipnameSearch', 'addEnumerationPulldown', 'addExpressionInput', 'addFilenameSearch
↪', 'addMultilineTextInput', 'addNotepad', 'addPasswordInput', 'addRGBColorChip',
↪'addScriptCommand', 'addSingleLineInput', 'addTextFontPulldown', 'clear', 'execute',
↪'setTitle', 'setWidth', 'show', 'title', 'value', 'width']
```

7.26 Nuke TCL

```
[file dirname [value root.name]] # 当前 nuke 文件所在的文件夹
[join [lrange [split [file dirname [value root.name]] /] 0 end-1] /] # 当前 nuke 文件所在
文件夹的上一个层级
```

参考文档:

- http://www.jeangjenq.com/tcl-python-snippets/

CHAPTER 8

Ftrack

Ftrack 提供 CG 相关项目管理和 Pipeline 工具，可帮助 CG 工作室跟踪，安排，审查，协作和管理其数字资产。

Contents:

8.1 Ftrack Python API

创建用户

```
newUser = session.create("User", {"username": "huweiguo@do-vfx.com"})

session.commit()
```

创建任务

```
shot = session.query("Shot where name is 'f_CTS049'").first()

newTask = session.create("Task", {"name": "fx_autosphere", "parent": shot})

session.commit()
```

CHAPTER 9

Shotgun

Shotgun 提供 CG 相关项目管理和 Pipeline 工具，可帮助 CG 工作室跟踪，安排，审查，协作和管理其数字资产。

Contents:

9.1 Shotgun API 访问方案

- 通过用户名访问

管理员用户登陆找到 People entity，记住两个字段值 Login 和 Password。

```
sg = shotgun_api3.Shotgun("https://do-vfx.shotgunstudio.com",
                          login="字段 Login 值",
                          password="字段 Password 值")
sg.find("Shot", filters=[["sg_status_list", "is", "ip"]], fields=["code", "sg_status_list
↪"])
```

- 通过 Scripts 访问

管理员用户登陆找到 Scripts entity，通过 Add Script 添加一条 Script，记住两个字段值 Script Name 和 Application Key。

```
sg = shotgun_api3.Shotgun("https://do-vfx.shotgunstudio.com",
                          login="字段 Script Name 值",
```

(下页继续)

(续上页)

```
                        password="字段 Application Key 值")
sg.find("Shot", filters=[["sg_status_list", "is", "ip"]], fields=["code", "sg_status_list
↪"])
```

如果是写工具来访问 Shotgun 数据，推荐使用 Scripts 的方法来访问。

参考文档:

- https://developer.shotgunsoftware.com/
- https://developer.shotgunsoftware.com/python-api/

9.2 Shotgun Tickets

Shotgun Tickets 是为了项目辅助任务提供填写 timelog 的表格，以便更准确地统计项目所花费的人力成本，比如制片的项目管理，TD 的流程搭建等等，它不涉及具体镜头和资产的制作时间。

Shotgun 创建的项目默认状态是禁用 Tickets 的，直接创建 Ticket 会遇到下面的错误信息。

Sorry, you cannot create Tickets in XYJ, because they are not available in the project. To make them available, please configure the project's tracking settings and try this again. To learn more about project tracking settings, visit our help center.

启用具体项目的 Tickets 页面很简单，打开需要启用的项目的任何页面，比如 Shots，找到右上角 Project Actions>Tracking Settings，在 HIDDEN 菜单中找到 Ticket，找到右上角 Hide Ticket 将其显示，然后点击 DONE 即可。

9.3 Shotgun Events 开发流程

9.4 Shotgun RV 审片室工作流程

CHAPTER 10

CGTeamwork

CGTeamwork 是一款国产 CG 流程软件，非常符合国内外包流程，提供网盘功能

Contents:

CHAPTER 11

Deadline

Deadline 是用于 Windows，Linux 和基于 macOS 的渲染农场的无忧管理和计算管理工具包。它支持开箱即用的 80 多种 DCC 应用程序。它为各种规模的渲染农场和计算集群提供了灵活性和广泛的管理选项，并允许用户自由地轻松访问内部部署或基于云的资源的任意组合以满足其渲染和处理需求。

Contents:

11.1 Deadline 渲染农场配置软件版本

配置软件版本需要配置两个地方，首先如果你是使用默认菜单提交方式，想添加软件版本可以通过修改下面的模块文件

```
DeadlineRepository10\scripts\Submission\HoudiniSubmission.py
```

此时我们只是能提交我们配置的软件版本，但任务肯定会报错，因为我们还需要配置插件，也就是后台会去调用什么软件版本去跑任务。这里需要去明确配置。

```
DeadlineRepository10\plugins\Houdini\Houdini.param
```

11.2 Deadline 渲染农场渲染 XGen 问题

11.3 Deadline 渲染农场事件插件开发

- Deadline Event Plugins
- OnJobStarted
- OnSlaveIdle
- 深入 Deadline API
- docs.thinkboxsoftware.com

关联任务

Deadline 自定义 Event 插件

- 路径：DeadlineRepository10customevents
- 掌握调试技巧
- RepositoryUtils
- OnJobStarted

View Job Reports

OnJobStarted.param

```
[State]
Type=Enum
Items=Global Enabled;Opt-In;Disabled
Category=Options
CategoryOrder=0
CategoryIndex=0
Label=State
Default=Disabled
Description=How this event plug-in should respond to events. If Global, all jobs and␣
↪slaves will trigger the events for this plugin. If Opt-In, jobs and slaves can choose␣
↪to trigger the events for this plugin. If Disabled, no events are triggered for this␣
↪plugin.
```

OnJobStarted.py

```
########################################################################
# Imports
########################################################################
```

(下页继续)

(续上页)

```
import math
from System import *

from Deadline.Events import *
from Deadline.Scripting import *


########################################################################################################
↪#########
# This is the function called by Deadline to get an instance of the Draft event listener.
########################################################################################################
↪#########
def GetDeadlineEventListener():
    return OverrideJobPriorityListener()


def CleanupDeadlineEventListener(eventListener):
    eventListener.Cleanup()


######################################################################
# The event listener class.
######################################################################
class OverrideJobPriorityListener(DeadlineEventListener):
    def __init__(self):
        self.OnJobStartedCallback += self.OnJobStarted

    def Cleanup(self):
        del self.OnJobStartedCallback

    def OnJobStarted(self, job):

        # Exit this event plugin as soon as possible if not a Houdini job
        if job.JobPlugin != "Houdini":
            return

        job.Priority = 80

        RepositoryUtils.SaveJob(job)
```

(下页继续)

(续上页)

```
        jobs = RepositoryUtils.GetJobsInState(["Active", "Pending"])
        slaveSettings = RepositoryUtils.GetSlaveSettingsList(True)
        slaves = [x for x in slaveSettings if x.SlaveEnabled]

        if jobs and slaves:
            userNum = len(set(x.UserName for x in jobs if x.JobTaskCount != 2))

            self.LogInfo("User Num: %d" % userNum)
            self.LogInfo("Machine Num: %d" % len(slaves))

            if not userNum:
                return

            num = int(math.ceil(float(len(slaves)) / float(userNum)))

            if userNum == 1:

                for x in jobs:

                    if x.MachineLimit != 0:
                        self.LogInfo(x.Name)
                        RepositoryUtils.SetMachineLimitMaximum(x.ID, 0)

            else:

                for x in jobs:

                    if x.MachineLimit != num:
                        self.LogInfo(x.Name)
                        RepositoryUtils.SetMachineLimitMaximum(x.ID, num)
```

- OnSlaveIdle

OnSlaveIdle.param

```
[State]
Type=Enum
Items=Global Enabled;Opt-In;Disabled
Category=Options
CategoryOrder=0
CategoryIndex=0
```

(下页继续)

(续上页)

```
Label=State
Default=Disabled
Description=How this event plug-in should respond to events. If Global, all jobs and␣
↪slaves will trigger the events for this plugin. If Opt-In, jobs and slaves can choose␣
↪to trigger the events for this plugin. If Disabled, no events are triggered for this␣
↪plugin.c
```

OnSlaveIdle.py

```
########################################################################
# Imports
########################################################################
import math
from System import *

from Deadline.Events import *
from Deadline.Scripting import *


########################################################################################################
↪#########
# This is the function called by Deadline to get an instance of the Draft event listener.
########################################################################################################
↪#########
def GetDeadlineEventListener():
    return OverrideMachineLimitListener()


def CleanupDeadlineEventListener(eventListener):
    eventListener.Cleanup()


########################################################################
# The event listener class.
########################################################################
class OverrideMachineLimitListener(DeadlineEventListener):
    def __init__(self):
        self.OnSlaveIdleCallback += self.OnSlaveIdle

    def Cleanup(self):
```

(下页继续)

(续上页)

```
        del self.OnSlaveIdleCallback

    def OnSlaveIdle(self, string):
        jobs = RepositoryUtils.GetJobsInState(["Active", "Pending"])
        # slaveInfos = RepositoryUtils.GetSlaveInfos(True)
        # slaves = [x for x in slaveInfos if x.SlaveState in ["Idle", "Rendering"]]
        # self.LogInfo("%d" % len(slaves))
        # enableSlaves = [x for x in slaveInfos if x.SlaveIsActive in ["Enable"]]
        # self.LogInfo("%d" % len(enableSlaves))
        # for slaveInfo in slaveInfos:
        #     self.LogInfo(slaveInfo.SlaveStatus)
        # INFO: AWSInfo,CPUUsage,CPUs,CompareTo,CompletedTasks,ConfigName,
↪CurrentJobGroup,CurrentJobId,CurrentJobName,CurrentJobPool,CurrentJobPriority,
↪CurrentJobUser,CurrentPlugin,CurrentTaskIds,CurrentTaskNames,CurrentTaskProgresses,
↪CurrentTaskRenderTimes,CurrentTaskStatus,DiskReads,DiskSpace,DiskSpaceString,
↪DiskWrites,Equals,ExtraElements,FailedTasks,Finalize,FreeMemory,FreeMode,GetHashCode,
↪GetType,GroupsString,HostName,ID,IPAddress,IsAWSPortalInstance,IsLicensePermanent,
↪IsLicensedByUsage,IsRunningAsService,LastReadRepoTime,LastReadTime,LastWriteTime,
↪LicenseDaysLeftToExpiry,LicenseLastErrorMessage,LicenseServer,LimitGroupStubs,
↪ListeningPort,MACAddress,MachineArchitecture,MachineCPUUsage,MachineCPUs,
↪MachineDiskSpace,MachineFreeMemory,MachineIPAddress,MachineMACAddress,MachineMemory,
↪MachineOperatingSystem,MachineProcessorSpeed,MachineRealName,MachineUserName,
↪MachineVideoCard,MemberwiseClone,Memory,NetworkReceived,NetworkSent,OSShortName,
↪OnLastTaskComplete,Overloads,PoolsString,ProcessorArchitecture,ProcessorSpeed,
↪ReferenceEquals,Region,RenderingTime,SLAVE_NAME,SlaveCompletedTasks,SlaveConfigName,
↪SlaveCurrentJobGroup,SlaveCurrentJobId,SlaveCurrentJobName,SlaveCurrentJobPool,
↪SlaveCurrentJobPriority,SlaveCurrentJobUserName,SlaveCurrentPlugin,SlaveCurrentTaskIds,
↪SlaveCurrentTaskNames,SlaveCurrentTaskProgresses,SlaveCurrentTaskRenderTimes,
↪SlaveCurrentTaskStatus,SlaveFailedTasks,SlaveFreeMode,SlaveIsActive,
↪SlaveIsLicensePermanent,SlaveIsLicensedByUsage,SlaveLastLicenseErrorMessage,
↪SlaveLastMessage,SlaveLicenseDaysLeftToExpiry,SlaveLicenseServer,SlaveLimitGroupStubs,
↪SlaveMessage,SlaveName,SlaveOnLastTaskComplete,SlaveRegion,SlaveRenderingTime,
↪SlaveRunningTime,SlaveSettingsGroups,SlaveSettingsPools,SlaveState,SlaveStatus,
↪StateDateTime,SwapUsage,ToString,UpTimeSeconds,UpdateDateTime,UserName,Version,
↪VideoCard,__call__,__class__,__cmp__,__delattr__,__delitem__,__doc__,__format__,__
↪getattribute__,__getitem__,__hash__,__init__,__iter__,__module__,__new__,__overloads__,
↪__reduce__,__reduce_ex__,__repr__,__setattr__,__setitem__,__sizeof__,__str__,__
↪subclasshook__,get_AWSInfo,get_CPUUsage,get_CPUs,get_CompletedTasks,get_ConfigName,get_
↪CurrentJobGroup,get_CurrentJobId,get_CurrentJobName,get_CurrentJobPool,get_
↪CurrentJobPriority,get_CurrentJobUser,get_CurrentPlugin,get_CurrentTaskIds,get_
↪CurrentTaskNames,get_CurrentTaskProgresses,get_CurrentTaskRenderTimes,get_
↪CurrentTaskStatus,get_DiskReads,get_DiskSpace,get_DiskSpaceString,get_DiskWrites,get_
↪ExtraElements,get_FailedTasks,get_FreeMemory,get_FreeMode,get_GroupsString,get_
↪HostName,get_ID,get_IPAddress,get_IsAWSPortalInstance,get_IsLicensePermanent,get_
↪IsLicensedByUsage,get_IsRunningAsService,get_LastReadRepoTime,get_LastReadTime,get_
```

(下页继续)

(续上页)

```
        # self.LogInfo(",".join(dir(slaveInfos[0])))
        # INFO: Clone,Comment,ConcurrentTasksLimit,CpuAffinity,Description,
↪EnableIdleCpuThreshold,EnableIdleProcessCheck,EnableIdleRamMBThreshold,
↪EnableIdleRamPercentThreshold,EnableIdleUserCheck,Enabled,Equals,EventOptInArray,
↪EventOptIns,EventOptInsStr,ExtraElements,ExtraInfo0,ExtraInfo1,ExtraInfo2,ExtraInfo3,
↪ExtraInfo4,ExtraInfo5,ExtraInfo6,ExtraInfo7,ExtraInfo8,ExtraInfo9,ExtraInfoDictionary,
↪Finalize,FinishTaskWhenStoppingIfNotIdle,GetHashCode,GetSlaveDataPath,
↪GetSlaveExtraInfoKeyValue,GetSlaveExtraInfoKeyValueWithDefault,GetSlaveExtraInfoKeys,
↪GetSlaveExtreInfoKeys,GetSlavePluginPath,GetType,GpuAffinity,GroupMappingID,Groups,
↪HostMachineIPAddressOverride,ID,IdleCpuThreshold,IdleMinutes,IdleProcessNames,
↪IdleRamMBThreshold,IdleRamPercentThreshold,IdleUserNames,IsCloudSlave,LastReadRepoTime,
↪LastReadTime,LastWriteTime,ListeningPort,MacAddressOverride,MemberwiseClone,
↪NormalizedRenderTimeMultiplier,NormalizedTimeoutMultiplier,
↪OnlyStopSlaveIfStartedByIdleDetection,Overloads,OverrideCpuAffinity,
↪OverrideGpuAffinity,OverrideListeningPort,OverrideSlaveScheduling,Pools,
↪ReferenceEquals,Region,SLAVENAME,SchedulingMode,SetSlaveGroups,
↪SetSlaveIdleProcessNames,SetSlaveIdleUserNames,SetSlavePools,SetSlaveUserJobsModeNames,
↪SlaveComment,SlaveConcurrentTasksLimit,SlaveCpuAffinity,SlaveDescription,
↪SlaveEnableIdleCpuThreshold,SlaveEnableIdleProcessCheck,SlaveEnableIdleRamMBThreshold,
↪SlaveEnableIdleRamPercentThreshold,SlaveEnableIdleUserCheck,SlaveEnabled,
↪SlaveExtraInfo0,SlaveExtraInfo1,SlaveExtraInfo2,SlaveExtraInfo3,SlaveExtraInfo4,
↪SlaveExtraInfo5,SlaveExtraInfo6,SlaveExtraInfo7,SlaveExtraInfo8,SlaveExtraInfo9,
↪SlaveExtraInfoDictionary,SlaveFinishTaskWhenStoppingIfNotIdle,SlaveGpuAffinity,
↪SlaveGroups,SlaveHostMachineIPAddressOverride,SlaveIdleCpuThreshold,SlaveIdleMinutes,
↪SlaveIdleProcessNames,SlaveIdleRamMBThreshold,SlaveIdleRamPercentThreshold,
↪SlaveIdleUserNames,SlaveListeningPort,SlaveMacAddressOverride,SlaveName,
↪SlaveNormalizedRenderTimeMultiplier,SlaveNormalizedTimeoutMultiplier,
↪SlaveOnlyStopSlaveIfStartedByIdleDetection,SlaveOverrideCpuAffinity,
↪SlaveOverrideGpuAffinity,SlaveOverrideListeningPort,SlaveOverrideSlaveScheduling,
↪SlavePools,SlaveSchedulingMode,SlaveStartSlaveIfIdle,SlaveStopSlaveIfNotIdle,
↪SlaveUserJobsModeNames,StartSlaveIfIdle,StopSlaveIfNotIdle,ToString,UseTmpDataPath,
↪UserJobsModeNames,__call__,__class__,__cmp__,__delattr__,__delitem__,__doc__,__format__
↪,__getattribute__,__getitem__,__hash__,__init__,__iter__,__module__,__new__,__
↪overloads__,__reduce__,__reduce_ex__,__repr__,__setattr__,__setitem__,__sizeof__,__str_
↪_,__subclasshook__,get_Comment,get_ConcurrentTasksLimit,get_CpuAffinity,get_
↪Description,get_EnableIdleCpuThreshold,get_EnableIdleProcessCheck,get_
↪EnableIdleRamMBThreshold,get_EnableIdleRamPercentThreshold,get_EnableIdleUserCheck,get_
↪Enabled,get_EventOptInArray,get_EventOptIns,get_EventOptInsStr,get_ExtraElements,get_
↪ExtraInfo0,get_ExtraInfo1,get_ExtraInfo2,get_ExtraInfo3,get_ExtraInfo4,get_ExtraInfo5,
↪get_ExtraInfo6,get_ExtraInfo7,get_ExtraInfo8,get_ExtraInfo9,get_ExtraInfoDictionary,
↪get_FinishTaskWhenStoppingIfNotIdle,get_GpuAffinity,get_GroupMappingID,get_Groups,get_
↪HostMachineIPAddressOverride,get_ID,get_IdleCpuThreshold,get_IdleMinutes,get_
↪IdleProcessNames,get_IdleRamMBThreshold,get_IdleRamPercentThreshold,get_IdleUserNames,
↪get_IsCloudSlave,get_LastReadRepoTime,get_LastReadTime,get_LastWriteTime,get_
↪ListeningPort,get_MacAddressOverride,get_NormalizedRenderTimeMultiplier,get_
```

(下页继续)

(续上页)

```
        slaveSettings = RepositoryUtils.GetSlaveSettingsList(True)
        # self.LogInfo(",".join(dir(slaveSettings[0])))
        slaves = [x for x in slaveSettings if x.SlaveEnabled]
        # self.LogInfo("%d" % len(slaves))
        # self.LogInfo(",".join(dir(jobs[0])))
        # for job in jobs:
        #     self.LogInfo(job.Name)
        #     self.LogInfo("%d" % job.JobTaskCount)
        # INFO: AWSPortalAssets,AuxiliarySubmissionFileNames,BadSlaves,BatchName,
↪ChunkSize,Comment,CompletedChunks,CompletedDateTime,CompletedDateTimeString,
↪CompletedFrames,ConcurrentTasks,CustomEventPluginDirectory,CustomPluginDirectory,
↪DeleteJobEnvironmentKey,DeleteJobExtraInfoKey,Department,DisabledScheduleTime,
↪EmailNotification,EnableAutoTimeout,EnableFrameTimeouts,EnableTimeoutsForScriptTasks,
↪EnvironmentDictionary,Equals,ErrorReports,EventOptIns,ExtraElements,ExtraInfo0,
↪ExtraInfo1,ExtraInfo2,ExtraInfo3,ExtraInfo4,ExtraInfo5,ExtraInfo6,ExtraInfo7,
↪ExtraInfo8,ExtraInfo9,ExtraInfoDictionary,ExtraInfoKeyValues,FailedChunks,
↪FailureDetectionJobErrors,FailureDetectionTaskErrors,Finalize,FirstFrame,
↪FrameDependencyOffsetEnd,FrameDependencyOffsetStart,Frames,FramesList,FridayStartTime,
↪FridayStopTime,GetEstimatedRemainingTime,GetFullOutputFileNamesForTask,
↪GetFullOutputTileFileNamesForTask,GetHashCode,GetJobEnvironmentKeyValue,
↪GetJobEnvironmentKeys,GetJobExtraInfoKeyValue,GetJobExtraInfoKeyValueWithDefault,
↪GetJobExtraInfoKeys,GetJobInfoKeyValue,GetJobInfoKeys,GetJobPluginInfoKeyValue,
↪GetJobPluginInfoKeys,GetOutputFileNamesForTask,GetOutputTileFileNamesForTask,GetType,
↪GetUniqueHash,GetUniqueJobName,Group,HasDependencies,ID,IgnoreBadJobDetection,
↪InitialCompletedTaskIds,InitialUncompletedTaskIds,InitializePluginTimeout,
↪InitializePluginTimeoutSeconds,InterruptibleFlag,InterruptiblePercentage,IsBeingPurged,
↪IsCorrupted,IsFrameDependent,IsSubmitted,JobAuxiliarySubmissionFileNames,JobBatchName,
↪JobCleanupDays,JobComment,JobCompletedDateTime,JobCompletedTasks,JobConcurrentTasks,
↪JobCustomEventPluginDirectory,JobCustomPluginDirectory,JobDepartment,JobDependencies,
↪JobDependencyIDs,JobDependencyPercentage,JobDependencyPercentageValue,
↪JobEmailNotification,JobEnableAutoTimeout,JobEnableFrameTimeouts,
↪JobEnableTimeoutsForScriptTasks,JobExtraInfo0,JobExtraInfo1,JobExtraInfo2,
↪JobExtraInfo3,JobExtraInfo4,JobExtraInfo5,JobExtraInfo6,JobExtraInfo7,JobExtraInfo8,
↪JobExtraInfo9,JobFailedTasks,JobFailureDetectionJobErrors,
↪JobFailureDetectionTaskErrors,JobForceReloadPlugin,JobFrameDependencyOffsetEnd,
↪JobFrameDependencyOffsetStart,JobFrames,JobFramesList,JobFramesPerTask,
↪JobFridayStartTime,JobFridayStopTime,JobGroup,JobId,JobIgnoreBadSlaveDetection,
↪JobInfoFromJob,JobInitializePluginTimeoutSeconds,JobInterruptible,
↪JobInterruptiblePercentage,JobIsFrameDependent,JobLimitGroups,
↪JobLimitTasksToNumberOfCpus,JobListedSlaves,JobMachineLimit,JobMachineLimitProgress,
↪JobMaintenanceJob,JobMaintenanceJobEndFrame,JobMaintenanceJobStartFrame,
↪JobMinRenderTimeSeconds,JobMondayStartTime,JobMondayStopTime,JobName,
↪JobNotificationEmails,JobNotificationNote,JobNotificationTargets,JobOnComplete,
↪JobOnTaskTimeout,JobOutputDirectories,JobOutputFileNames,JobOutputTileFileNames,
↪JobOverrideAutoJobCleanup,JobOverrideJobCleanup,JobOverrideJobCleanupDays,
```

(下页继续)

(续上页)

```
        if jobs and slaves:
            userNum = len(set(job.UserName for job in jobs if job.JobTaskCount != 2))
            self.LogInfo("User Num: %d" % userNum)
            self.LogInfo("Machine Num: %d" % len(slaves))

            if not userNum:
                return

            num = int(math.ceil(float(len(slaves)) / float(userNum)))

            if userNum == 1:

                for job in jobs:

                    if job.MachineLimit != 0:
                        self.LogInfo(job.Name)
                        RepositoryUtils.SetMachineLimitMaximum(job.ID, 0)

            else:

                for job in jobs:

                    if job.MachineLimit != num:
                        self.LogInfo(job.Name)
                        RepositoryUtils.SetMachineLimitMaximum(job.ID, num)
```

11.4 Deadline 渲染农场 Houdini 提交工具

- 案例：编写 Houdini、Nuke 提交工具
- Deadline Python API
- 其它 CG 软件的提交代码
- 其它农场提交代码

```
import sys
sys.path.insert(0, r"Y:\Program\DeadlineRepository10\api\python")
from Deadline.DeadlineConnect import DeadlineCon as Connect
con = Connect("farm.do-vfx.com", 8082)
```

(下页继续)

(续上页)

```
job = con.Jobs.GetJob("5bfce04f4096fe78c0f6d640")

print(job)
import Deadline.DeadlineConnect as Connect

Deadline = Connect.DeadlineCon('farm.do-vfx.com', 8082)
print(Deadline)
JobInfo = {
    "Name": "Submitted via Python",
    "UserName": "huweiguo",
    "Frames": "1-24",
    "Plugin": "Houdini"
    }

PluginInfo = {
    "OutputDriver": "/obj/ropnet1/mantra1",
    "SceneFile": "Y:/shotTest/untitled.hip",
    "Version": "17.0"
    }

try:
    newJob = Deadline.Jobs.SubmitJob(JobInfo, PluginInfo)
    print newJob
except:
    print "Sorry, Web Service is currently down!"
```

Houdini 提交工具

- Python API 提交任务
- PySide2 构建 UI 界面

Nuke 提交工具

- Python API 提交任务
- PySide2 构建 UI 界面

11.5 Deadline 渲染农场安装服务启动错误

这是一个典型的常见安装错误 `Error starting service Deadline10DatabaseService`，遇见过几次，问题都是出在杀毒软件上，可以将杀毒软件暂时退出再进行安装，安装完再启动杀毒软件，我这边用到了火绒

杀毒，关掉就好了。

11.6 Deadline 渲染农场 MongoDB 离线安装

在安装 DeadlineRepository 的时候，会遇到在线安装 MongoDB 数据库服务器的需求，如果在线下载安装往往非常非常慢，无法忍受。

在 Deadline 官方安装文档中有提到可以先下载 MongoDB 数据库服务器离线安装包，再来安装，这样速度是极快的。

Deadline 10.1 版本所要求的 MongoDB 版本在 3.2.12~3.6.16 之间。

上 MongoDB 官网 https://www.mongodb.com/download-center 下载，找到 Server 选项，选择配置，我这里拿 Windows 10 操作系统做的测试，稳定性考虑建议使用服务器操作系统，将 Deadline 搭建在服务器上，比如 Windows Server 2012 R2 Standard，下面的配置是测试配置按实际环境修改。

- Version: 3.4.24(previous release)
- OS: Windows x64
- Package: ZIP

我这里下下来是这样一个安装包 mongodb-win32-x86_64-2008plus-ssl-3.4.24.zip，在安装 DeadlineRepository 选择这个离线安装包安装就可以了。

参考文档:

- https://docs.thinkboxsoftware.com/products/deadline/10.1/1_User%20Manual/manual/install-db-repo.html

11.7 Deadline 渲染农场 Maya 提交工具

11.8 Deadline 渲染农场 Nuke 提交工具

11.9 Deadline 渲染农场池和组的概念

Groups 很容易理解，渲染节点的硬件和软件配置不同可以设置不同的组以便适应不同任务渲染。比如现在有一批 Houdini 18 的文件需要渲染，你只能在已经安装过 Houdini 18 的机器上才能正确渲染，别的机器渲染肯定会报错，此时为了排除不正确的机器就可以将机器编组，提交任务的时候将任务丢到组中，这样就不会调度没有组的机器。

池可以说是有优先级顺序的组，相当于把农场几分天下，有时候有这样一些需求就是灯光任务优先级永远是最高的，或者分配一部分机器给某一个项目优先使用，此时就要用到池的概念，一个任务进农场调度机器首先会检查它属于哪个池，如果没有设置池的属性，渲染优先级都是最低的，设置了池，任务就会到这个池中

然后按正常的优先级，提交日期和组去调度池中的机器。如果池中没有了任务，机器才会调度给没有设置池的任务，即使他们的优先级设置了 100 也没用，池中优先级是最优先的。

11.10 Deadline 渲染农场权限管理

咱们来解决一个农场的一个问题，如何有效防止制作上传任务修改自己任务优先级?

想从根本上解决这个问题，我们得规范提交流程，各大 DCC 软件中提交工具是必须写的，因为从 Deadline Monitor 面板或者 DCC 软件中集成的菜单提交是没法禁用这个优先级的修改的（我是没找到，只找到个优先级最大值的设置，这个没啥用），任何人提交的时候都可以将任务优先级修改为 100（默认最高优先级是 100）。

不光光提交任务的时候可以事先修改这个优先级，在提交完成之后 Deadline 默认配置是个人可以修改自己的任务优先级，那如何杜绝这个问题。

首先我们得配置一个 Super User 的密码，Tools>Configure Repository Options>User Security。

配置完我们配置用户组，分管理员 Administrator 和普通用户 Custom。需要授予管理员权限的用户添加到 Administrator，一般用户权限添加到 Custom。

这样处理之后管理员用户不用每次都要输入密码切换 Super User Mode，普通用户可以随时通过管理员密码切换到 Super User。

我们这里将禁止修改任务优先级权限，要做三件事，第一件事禁用提交按钮，比如去掉 Maya 提交。

通过菜单 Tools>Configure Script Menus>Edit Selection，将 Maya 提交菜单 Disabled 掉，自然我们软件里的 Deadline 提供的提交任务插件也是不能部署的。

第二件事禁用修改已经提交的任务的优先级，这个在我们的 Tools>Manage User Group…进入 Custom>Job Properties 将 Job Priority 选项 Disabled。

第三件事自己通过 Deadline Python API 写提交工具，提升百倍提交任务效率，避免手动提交的错误以及文件规范性，数海提交农场任务都是秒传没毛病，从创业两年以来提交总任务数 59021 个，如果以手动提交一个任务 30 秒的速度（往往是不止的）话，需要花费一个人 491.8 小时的工作量，这里不包括手动提交的设置问题，所以非常建议大家自己写自己的提交工具，不超过百行的 Python 代码。

11.11 Deadline 渲染农场提交农场接口

使用 Deadline Python API 提交任务需要开启 deadlinewebservice.exe 服务。

```
>>> import sys
>>>
>>>
>>> path = r"Y:\Program\DeadlineRepository10\api\python"
>>>
```

(下页继续)

(续上页)

```
>>> path in sys.path or sys.path.insert(0, path)
>>> import Deadline.DeadlineConnect as Connect
>>> con = Connect.DeadlineCon("192.168.0.111", 8082)
>>> con
<Deadline.DeadlineConnect.DeadlineCon instance at 0x00E90B48>
>>> type(con)
<type 'instance'>
>>>
>>>
>>>
>>> dir(con)
['AuthenticationModeEnabled', 'Balancer', 'EnableAuthentication', 'Groups', 'JobReports',
↪ 'Jobs', 'LimitGroups', 'MappedPaths', 'MaximumPriority', 'Plugins', 'Pools', 'Pulse',
↪'Repository', 'SetAuthenticationCredentials', 'Slaves', 'SlavesRenderingJob',
↪'TaskReports', 'Tasks', 'Users', '__doc__', '__init__', '__module__',
↪'connectionProperties']
>>>
>>>
>>> con.Groups
<Deadline.Groups.Groups instance at 0x00E90DA0>
>>> dir(con.Groups)
['AddGroup', 'AddGroups', 'DeleteGroup', 'DeleteGroups', 'GetGroupNames', 'PurgeGroups',
↪'__doc__', '__init__', '__module__', 'connectionProperties']
>>> con.Groups.GetGroupNames()
[u'none', u'nuke', u'arnold', u'cache', u'gpu', u'maya2016', u'largem', u'core48', u'pdg
↪', u'maya2017', u'rs3', u'maya2018']
>>>
>>> job = con.Jobs.GetJob("5f16b24ab3a4a507c0359b71")
>>> type(job)
<type 'dict'>
>>> from pprint import pprint
>>> pprint(job)
>>> job["Props"]["Pri"] = 97
>>> pprint(job)
>>> con.Jobs.SaveJob(job)
'Success'
>>>
```

11.12 Deadline 渲染农场服务器安装部署

文档采用服务器集中式部署方案，将 DeadlineRepository 与 DeadlineClient 全部部署在服务器，从而客户端无需安装任何东西即可使用的方案。

部署环境如下:

- 服务器：Windows Server 2012 R2 Standard
- 客户端：Window 10 企业版
- Deadline 版本：Thinkbox Deadline v10.0.20.2 Win

远程控制，需要开启 deadlinepulse.exe

Deadline 如果使用服务器部署存在三个问题需要解决，一个是安装的路径得是共享路径，在 D 盘你是没法在另一台电脑上开启客户端软件连接数据库的。客户端电脑有时候打不开 Deadline 的原因可能是需要安装.net 的库，一般安装完 Maya 就可以解决这个问题。还有会遇到 deadline.ini 文件找不到的问题，可以直接拷贝 ThinkboxDeadline10binThinkboxDeadline10deadline.ini 文件到下面的路径中。

```
C:\ProgramData\Thinkbox\Deadline10
```

CHAPTER 12

Linux

Linux，全称 GNU/Linux，是一套免费使用和自由传播的类 Unix 操作系统，是一个基于 POSIX 和 Unix 的多用户、多任务、支持多线程和多 CPU 的操作系统。伴随着互联网的发展，Linux 得到了来自全世界软件爱好者、组织、公司的支持。它除了在服务器方面保持着强劲的发展势头以外，在个人电脑、嵌入式系统上都有着长足的进步。使用者不仅可以直观地获取该操作系统的实现机制，而且可以根据自身的需要来修改完善 Linux，使其最大化地适应用户的需要。

Linux 不仅系统性能稳定，而且是开源软件。其核心防火墙组件性能高效、配置简单，保证了系统的安全。在很多企业网络中，为了追求速度和安全，Linux 不仅仅是被网络运维人员当作服务器使用，Linux 既可以当作服务器，又可以当作网络防火墙是 Linux 的一大亮点。

Linux 具有开放源码、没有版权、技术社区用户多等特点，开放源码使得用户可以自由裁剪，灵活性高，功能强大，成本低。尤其系统中内嵌网络协议栈，经过适当的配置就可实现路由器的功能。这些特点使得 Linux 成为开发路由交换设备的理想开发平台。

Contents:

12.1 Linux 常用指令大全

防火墙相关指令:

```
systemctl status firewalld
systemctl start firewalld
systemctl stop firewalld
```

(下页继续)

(续上页)

```
systemctl disable firewalld
systemctl enable firewalld
```

参考文档:

- https://www.bilibili.com/video/BV1nW411L7xm
- https://www.bilibili.com/video/BV1DW411G7VB

CHAPTER 13

SQL

SQL（Structured Query Language）结构化查询语言是一种特定目的的编程语言，用于管理关系数据库管理系统（RDBMS）。

Contents:

13.1 SQL 查询按字段打组

GROUP BY 是个很有意思的操作。

```
SELECT new.code, path_cache, version_number, task__id, published_file_type__id, new.
↪created_by__id, new.created_at, humanuser.name, publishedfiletype.code AS file_type,␣
↪content FROM (SELECT code, path_cache, max.version_number, max.task__id, published_
↪file_type__id, created_by__id, created_at, name FROM (SELECT task__id, MAX(version_
↪number) AS version_number FROM publishedfile WHERE project__id=%d AND entity__id=%d␣
↪AND _active='t' GROUP BY task__id, published_file_type__id) max JOIN publishedfile ON␣
↪max.task__id=publishedfile.task__id AND max.version_number=publishedfile.version_
↪number AND _active='t') new, publishedfiletype, humanuser, task WHERE new.published_
↪file_type__id=publishedfiletype.id AND new.created_by__id=humanuser.id AND new.task__
↪id=task.id
```

13.2 SQL 查询按数字排序

这里要特别注意如果字段是字符串类型的数字，需要转换 int 类型再做排序，不然会出问题。

```
SELECT version_number FROM publishedfile WHERE task__id = 11086 and _active = 't' ORDER␣
↪BY CAST(version_number AS int)
```

CHAPTER 14

Devops

DevOps（Development 和 Operations 的组合詞）是一种重视「软件开发人员（Dev）」和「IT 运维技术人员（Ops）」之间沟通合作的文化、运动或慣例。透过自动化「软件交付」和「架构变更」的流程，来使得构建、测试、发布软件能够更加地快捷、频繁和可靠。

Contents:

14.1 FreeNAS

FreeNAS 是一套基于 FreeBSD 操作系统核心的开放源代码的网络存储设备（NAS）服务器系统，支持众多服务，用户访问权限管理，提供网页设置接口。

Contents:

14.1.1 FreeNAS：回收站无权访问刷新 ACL 控制列表

FreeNAS 配置的回收站.recycle 文件夹时间久了不知道为什么权限就消失了，无法访问别人删除的文件

此时进 FreeNAS 网页端，找到共享 >Windows 共享（SMB）> 编辑访问控制列表 > 递归应用权限 > 将权限应用于子数据集 > 保存就行

14.1.2 FreeNAS：硬件配置以及系统安装

FreeNAS 是一套基于 FreeBSD 操作系统核心的开放源代码的网络存储设备（NAS）服务器系统，支持众多服务，用户访问权限管理，提供网页设置接口。

硬件配置：

- R730XD E5 2678v3*2 32G*4 8T*12 H330 阵列卡 x710 四口万兆网卡

因为没有配置 SSD 固态盘作为系统盘，FreeNAS 使用软 RAID，所以不通过 BIOS 来配置硬盘的 RAID，我们先做系统，为了将操作系统和存储数据分开，准备两个 U 盘，如果你有专门固态盘作为系统盘的话只需要一个 U 盘，将系统做到 SSD 中即可，这里我选择将系统做到另一个 U 盘中。这里我将 FreeNAS 做到 32G 的 U 盘中。

随便找台 win 电脑，将 FreeNAS 镜像烧录到其中一个 U 盘。需要下载两个东西。

- win32diskimager-1.0.0-install.exe
- FreeNAS-11.3-RELEASE.iso

一旦烧录了 FreeNAS 操作系统，U 盘将不在盘符区显示，如果需要重新烧录，你会发现 U 盘无法格式化，推荐你们使用 DiskGenius 工具格式化分区，非常好用。

开机按 F2 配置 BIOS 启动（去掉 UEFI 如果设置过），重启按 F11 选择 U 盘启动。

- [ ] 测试 U 盘恢复 FreeNAS 系统
- [ ] FreeNAS 域环境配置
- [ ] 测试 FreeNAS 读写速度
- [ ] FreeNAS 软 RAID
- [√] FreeNAS 共享文件
- [ ] FreeNAS 文件系统怎么用
- [ ] FreeNAS 配置备份，如何一键恢复
- [√] FreeNAS 如何固定 IP
- [√] FreeNAS 权限管理
- [ ] FreeNAS 软链接是否可用

FreeNAS 设置静态 IP，在 Web 端选择 Network>Interfaces，找到对应的网卡，比如 bge1，DHCP 自动分配去勾选，设置 IP Address。

FreeNAS 创建用户以及用户权限。

参考文档

- https://www.getnas.com/freenas-installation/
- https://blog.csdn.net/sanwe3333/article/details/104831493
- http://www.freenas.com.cn/
- https://www.getnas.com/freenas-static-ip/
- https://www.cnblogs.com/hjc4025/p/7079364.html

14.2 Others

DevOps（Development 和 Operations 的组合詞）是一种重视「软件开发人员（Dev）」和「IT 运维技术人员（Ops）」之间沟通合作的文化、运动或惯例。透过自动化「软件交付」和「架构变更」的流程，来使得构建、测试、发布软件能够更加地快捷、频繁和可靠。

Contents:

14.2.1 局域网子网掩码

一般家用没有交换机，只有个路由器，此时路由器充当数据交换的作用，关闭路由器 DHCP 服务，手动设置 IP，只要在同一个 IP 段，数据就可以互相访问

企业有交换机的情况下，实际是不需要路由器就可以连局域网的

正常 192.168.x.0~192.168.x.255 这样是一个 IP 段，默认子网掩码是 255.255.255.0，比如 192.168.8.0~192.168.8.255

上面的 IP 段除掉头尾只有 254 个 IP 可供使用，想要扩展更多的 IP 在同一个局域网可以通过子网掩码，比如设置成 255.255.254.0

这样 192.168.8.0~192.168.8.255 和 192.168.9.0~192.168.9.255 这样是在同一个 IP 段，IP 翻倍变成 512 个

机器如果是固定 IP 得自己配置子网掩码，如果是通过路由器 DHCP 服务自动获取 IP，可以将路由器子网掩码设置成 255.255.254.0

freenas 可在网络 > 接口 > 编辑钟设置 IP 地址为 192.168.0.123/23，23 代表这 255.255.254.0

群晖在控制面板 > 网络 > 网络界面 > 编辑 > 子网掩码

14.2.2 C 盘空间清理几个方案

- SpaceSniffer

SpaceSniffer 工具非常好用，可以很方便查看具体文件夹以及文件大小分布情况，从而可以知道哪些文件占用了磁盘空间。

- %tmp%

Windows 系统中通过 `Win+R` 键打开运行面板，输入 `%tmp%` 确定可以打开 C 盘产生的临时文件，文件夹 `C:\Users\{USERNAME}\AppData\Local\Temp` 中的文件都可以删除，提示占用勾选全部跳过即可。

- Redshift

Redshift 贴图缓存文件路径默认是在 C 盘 `$LOCALAPPDATA\Cache`，如果不更改而使用，会逐渐占用很多 C 盘空间，建议设置在 D 盘路径或者定期清理，路径在 `C:\Users\{USERNAME}\AppData\Local\Redshift\Cache`

- hiberfil.sys

hiberfil.sys 是 Windows 系统休眠文件，往往占用很大的磁盘空间，如果你会重装系统就直接通过下面的命令关闭休眠功能，这样可以删除 hiberfil.sys 文件。

```
powercfg -h off
```

重新启用休眠功能可以使用下面的指令。

```
powercfg -h on
```

14.2.3 电脑配置一些坑

[×] CPU（认准核心数和线程数以及算力赫兹），AMD 的算力比 Intel 要好很多，渲染不然，个人使用建议 AMD 3950x 往上，如果单纯 CPU 渲染，可以淘二手志强双路的洋垃圾，也是相当不错的。

[√] 显卡（认准显存大小，8G 往上）1070，2070 往上（如果有 GPU 渲染，可以上 2080Ti 往上）。

[×] 主板（要考虑是否板载显卡，以及集成显卡和独立显卡切换问题，还有是否支持温度监控以及网卡唤醒功能，要考虑内存卡槽以及使用什么级别的内存，比如 DDR3 还是 DDR4 要注意一下，再有就是大板还是小板，用哪个实际无所谓，要注意能插大显卡，如果想插多张显卡也要考虑一下）

[×] 主机箱（考虑散热的情况下尽量小而轻一些，搬家太费劲）

[×] 内存（16G 暂时够用，要注意看下内存品牌以及等级，这个看你主板的内存槽个数，个数少建议单根内存要大些，比如 32G 一根，四根就是 128G，不要搞太多根，很多时候主板内存插槽不是每个都能使用）

[×] 网卡（正常都是千兆网卡，如果有带宽传输需求，可以考虑万兆网卡）

[×] 声卡（主板不带板载声卡，需要另外安装，可以让商家赠送）

[√] 硬盘（搞块 1T SSD 固态硬盘足够使用）

[×] 键盘鼠标（让商家赠送吧，实际没必要讲究，有钱玩玩机械硬盘，个人感觉又贵又不好用）

[×] 显示器（选 dell 吧，普通选 1080 分辨率，23.8 寸够用，高级一点选 2K 分辨率，25 寸尚可，比如 U2518D，显示器的坑比较多，比如检查有没有坏点，便宜显示器色差比较大，这个很影响渲染图效果，所以还是要选贵点的显示器做灯光渲染合成会比较好，连接线用 mDP 或者 HDMI，如果配置双屏注意和显卡的配合，显卡上的插槽和显示器的插槽要对应上，贵的一般都有很多插槽，如果插槽搞的不对，只能自己买转接线才能玩）

[×] 散热器（风冷，水冷，散热效果，噪音）

14.2.4 Windows 自定义快捷方式

Nuke 打开频繁遇到下面这个问题。

Frame Server failed to bind TCP port 5559 and found existing Frame Server processes. Would you like Nuke to attempt to kill these processes?

软件启动一般可以带一些参数起到不同启动的效果，往往我们需要自定义一些软件启动的快捷方式。默认 NukeX 的启动目标是这样的。

```
"C:\Program Files\Nuke12.1v2\Nuke12.1.exe" --nukex
```

可以通过右键新建 > 快捷方式，写入启动内容。

```
"C:\Program Files\Nuke11.2v3\Nuke11.2.exe" --nukex --disable-nuke-frameserver
```

比如启动 Maya 关闭 Console 窗口，可以这样干。

```
"C:\Program Files\Autodesk\Maya2018\bin\maya.exe" -hideConsole
```

14.2.5 Davinci 专业剪辑调色软件

达芬奇是一款非常出色的剪辑调色软件。

14.2.6 Confluence 文档服务器部署

14.2.7 格式工厂

14.2.8 Gitlab 代码仓库服务器部署

14.2.9 Jira 项目跟踪服务器部署

14.2.10 JupyterLab 在线代码开发环境搭建

14.2.11 KMS 服务器搭建

14.2.12 Rocket.Chat 免费聊天工具部署

```
from rocketchat_API.rocketchat import RocketChat

rocket = RocketChat("Robot", "<password>", server_url="http://192.168.0.244:3000")
rocket.chat_post_message("@Andy test!", channel="general")
```

14.2.13 Seafile 文件同步服务器部署

14.2.14 SecureCRT

14.2.15 Squid 代理服务器搭建

VMware 部署一台 CentOS7 Linux 虚拟机。

```
# 安装
yum install squid -y
yum install httpd-tools -y
# 服务启动
systemctl start squid.service
# 服务停止
systemctl stop squid.service
# 配置开机自启动
systemctl enable squid.service
# 服务重启
systemctl restart squid.service
# 关闭防火墙
systemctl stop firewalld.service
# 禁用防火墙
systemctl disable firewalld.service
```

14.2.16 U 盘重装 Win10 操作系统

所谓硬件如果没什么问题，所有软件问题都不是问题，重装系统一时爽，一直重装一直爽，系统装的好，备胎当到老。

重装系统和硬件以及要安装的系统版本都有很大的关系，通俗讲就是每台电脑的系统重装可能都会在细节上有不同之处，也会遇到完全不同的问题。

但是套路大体相同，逐一排查，多问度娘，你遇到的坑总有人已经填过，多来几次就是了，摸清自己电脑脾气，写个安装文档，再次重装的时候就会顺很多。

文章只讲一种我常用的 U 盘安装纯净版操作系统的方法，实际重装系统方案在网络上天花乱坠，随个人喜好，我写的不一定就是最好的，只是个人比较喜欢的。

- 下载 Windows 操作系统网站：https://msdn.itellyou.cn/。
- 安装 UltraISO，选择继续试用。
- 打开下载好的系统镜像文件.iso。
- 选择菜单启动 > 写入硬盘映像，正常来说插入 U 盘会自动识别 U 盘，先格式化，后写入。

- 正常来说此 U 盘插到电脑上在 BIOS 中选择从 U 盘启动即可重装系统了。
- 如果需要使用 UEFI 可以选择便捷启动 > 写入新的驱动器引导扇区 >Windows 10/8.1/8/7/Vista 即可。

14.2.17 Markdown 工具

https://typora.io/

14.2.18 虚幻引擎

14.2.19 硬盘检测工具

- ATTO Disk
- AS SSD Benchmark
- CrystalDiskMark
- HD Tune Pro

推荐 ATTO Disk 测试硬盘读写速度。

14.2.20 磁盘分区工具

DiskGenius 非常好用的磁盘分区格式化工具。

参考文档

- https://www.diskgenius.cn/download.php

14.2.21 FFmpeg 批量转码工具

```
echo off
M:
cd M:\thirdParty\tools\ffmpeg\bin

set OUTPUT_PATH=%1\build

if not exist %OUTPUT_PATH% (
    md %OUTPUT_PATH%
)
```

(下页继续)

(续上页)

```
for %%i in (%1\*.mov) do ffmpeg -i %%i -vcodec h264 -pix_fmt yuv420p -y %%~dpibuild\%%~
↪ni.mov
for %%i in (%1\*.mov) do echo %%~dpibuild\%%~ni.mov
echo "Convert H.264 Successed!"
echo "Output Path--->" %OUTPUT_PATH%
pause
```

14.2.22 FileLocator Pro 全盘检索工具

非常好用的全盘索引检索工具，提供文件内容检索，效率非常高。

14.2.23 软件安装：Clarisse

- 选择 isotropix_clarisse_ifx_4.0_win64.exe，右键以管理员身份运行。
- 一直 Next 傻瓜式安装，遇到 python 安装可以先不安装（如果安装过 python27 的话）。
- 修改配置文件 C:Program FilesIsotropixClarisse iFX 3.0Clarisseclarisse.env

```
PYTHONHOME=C:\Python27
```

14.2.24 软件安装：Katana

- 选择 Katana3.5v2-win-x86-release-64.exe，右键以管理员身份运行。
- 一直 Next 傻瓜式安装即可。

14.2.25 软件安装：Marvelous Designer

- 选择 Setup.exe，右键以管理员身份运行。
- 一直 Next 傻瓜式安装即可。

14.2.26 软件安装：UE4

通过 Epic 安装 UE4 一般限于网速比较困难，可以网络上下载离线安装包，解压完会有四个文件夹。

```
Engine
FeaturePacks
Samples
Templates
```

直接找下面的.exe 文件打开运行即可。

```
Engine\Binaries\Win64\UE4Editor.exe
```

14.2.27 软件安装：ZBrush

- 选择 ZBrush_2019_Installer.exe，右键以管理员身份运行。
- 一直 Next 傻瓜式安装即可。

14.2.28 花生壳内网穿透技术解析

内网穿透的必备条件是有搭建好的本地服务器，比如 Confluence，Jira，RocketChat，Seafile 等等，一般搭建好服务器都可以通过本地的 IP 地址加端口号来访问服务器，但是如果想外网也能访问内部服务器，这里就要使用到内网穿透技术，一般如果有固定的公网 IP，直接在路由器上做好设置也没有什么问题，但是如果网络是公用的，那么是没有固定公网 IP 分配的，可以打电话给营业商咨询。如果实在没有公网 IP，那么使用类似花生壳这样的软件来做动态 IP 解析也是可以的。

在花生壳申请域名，不知道这个能不能用自己购买的域名，然后添加使用 HTTP 协议的内网穿透配置即可，花生壳需要实名认证，否则功能会被禁用。

```
https://www.oray.com/
```

14.2.29 mklink 符号链接技术

14.2.30 命令行禁用网卡再启用操作

有时候域网络首次无法识别，需要禁用网卡再启用才能识别，但很多时候需要远程操作就比较麻烦，网络被禁用就无法远程，所以写两条指令自动禁用再启用比较 OK。

管理员身份运行 cmd，然后将下面三行 shell 脚本粘贴到命名行窗口执行即可，注意网卡名称“以太网”需要改成对应的名称，pause 是必须的，不然无法执行第二条指令。

```
netsh interface set interface "以太网" disabled
netsh interface set interface "以太网" enabled
pause
```

14.2.31 WPS Office、Foxmail 办公软件推荐

Office 推荐 WPS 个人版

邮箱推荐 Foxmail，选择手动配置

- 接收服务器类型：POP3
- 邮件账号：huweiguo@do-vfx.com
- POP 服务器：pop3.mxhichina.com 端口：110
- SMTP 服务器：smtp.mxhichina.com 端口：25
- 设置最小化时，在任务栏显示
- 关闭主窗口时，退出程序

内网禁网机制，添加钉邮 IP

```
route print

route delete 0.0.0.0/0

:: email
route -p add 106.11.252.0/24 192.168.0.1
route -p add 140.205.106.0/24 192.168.0.1
route -p add 140.205.2.0/24 192.168.0.1
route -p add 140.205.23.0/24 192.168.0.1
route -p add 140.205.28.0/24 192.168.0.1
route -p add 140.205.77.0/24 192.168.0.1
route -p add 140.205.80.0/24 192.168.0.1
route -p add 140.205.97.0/24 192.168.0.1
route -p add 42.120.149.0/24 192.168.0.1
route -p add 42.120.151.0/24 192.168.0.1
route -p add 42.120.158.0/24 192.168.0.1
route -p add 42.120.214.0/24 192.168.0.1
route -p add 42.120.219.0/24 192.168.0.1
route -p add 42.120.226.0/24 192.168.0.1
route -p add 42.120.230.0/24 192.168.0.1
route -p add 42.156.140.0/24 192.168.0.1
route -p add 42.156.141.0/24 192.168.0.1
route -p add 42.156.235.0/24 192.168.0.1
route -p add 198.11.189.0/24 192.168.0.1
route -p add 205.204.101.0/24 192.168.0.1
route -p add 47.89.74.0/24 192.168.0.1
route -p add 47.89.80.0/24 192.168.0.1
route -p add 47.88.68.0/24 192.168.0.1
```

14.2.32 OneDrive 协同办公流程

OneDrive 网页端被墙，客户端可以正常登陆和使用。

14.2.33 Win10 调出照片查看器

写一个注册表文件.reg，将下面的文本拷贝到其中保存之后运行合并即可。

```
Windows Registry Editor Version 5.00

 ; Change Extension's File Type

 [HKEY_CURRENT_USER\Software\Classes\.jpg]

 @="PhotoViewer.FileAssoc.Tiff"

 ; Change Extension's File Type

 [HKEY_CURRENT_USER\Software\Classes\.jpeg]

 @="PhotoViewer.FileAssoc.Tiff"

 ; Change Extension's File Type

 [HKEY_CURRENT_USER\Software\Classes\.gif]

 @="PhotoViewer.FileAssoc.Tiff"

 ; Change Extension's File Type

 [HKEY_CURRENT_USER\Software\Classes\.png]

 @="PhotoViewer.FileAssoc.Tiff"

 ; Change Extension's File Type

 [HKEY_CURRENT_USER\Software\Classes\.bmp]

 @="PhotoViewer.FileAssoc.Tiff"

 ; Change Extension's File Type
```

(下页继续)

(续上页)

```
[HKEY_CURRENT_USER\Software\Classes\.tiff]

@="PhotoViewer.FileAssoc.Tiff"

; Change Extension's File Type

[HKEY_CURRENT_USER\Software\Classes\.ico]

@="PhotoViewer.FileAssoc.Tiff"
```

14.2.34 pip 命令使用手册

pip 下载源决定了安装第三方模块的速度与稳定性，如果下载源在国外服务器可能会比较缓慢，可以切换到阿里云上安装。

- pip install <pkg>
- pip install <pkg> -i https://mirrors.aliyun.com/pypi/simple

14.2.35 ReadTheDocs+Sphinx+Github 搭建托管文档环境

1. 安装 Python

测试的操作系统版本 Windows 10 1809，Python 版本 3.7.2 ，你可以选用 Python 更高的版本。上官网 https://www.python.org/downloads/ 下载。Python 3 在安装的时候记得自定义安装路径选择 `C:\Python37` ，默认路径是在当前用户文件夹，后面使用起来不太方便。

2. 配置环境变量 Path

这里为了下文命令能使用，需要将 `C:\Python37` 和 `C:\Python37\Scripts` 两个路径加入到用户变量或者系统变量 Path 中。注意如果有安装 Python 其它版本注意 Path 的先后问题，确保下文所调用的命令是文件夹 `C:\Python37\Scripts` 中的应用，如果确实搞不定这个，直接用绝对路径执行亦可。

3. 安装 Sphinx

通过下面的 pip 命令安装 Sphinx。

```
pip install sphinx
pip install sphinx-autobuild
pip install sphinx_rtd_theme
pip install recommonmark
pip install sphinx-markdown-tables
```

如果使用国外镜像源安装非常慢并且经常报错失败，可以像下面命令添加阿里云的镜像安装。

```
pip install sphinx -i https://mirrors.aliyun.com/pypi/simple
pip install sphinx-autobuild -i https://mirrors.aliyun.com/pypi/simple
pip install sphinx_rtd_theme -i https://mirrors.aliyun.com/pypi/simple
pip install recommonmark -i https://mirrors.aliyun.com/pypi/simple
pip install sphinx-markdown-tables -i https://mirrors.aliyun.com/pypi/simple
```

4. 初始化项目配置

在计算机中任何位置创建个文件夹，比如 CGTDCourse。打开命令行窗口，将当前路径切换到此文件夹，执行命令。

```
sphinx-quickstart
```

写上项目名，作者名，版本号，其它默认，如果顺利完成则会产生两个文件夹和两个文件。找到 `source\conf.py` 配置文件，添加下面两行配置，将主页配置修改，否则后面 ReadTheDocs 构建文档时会遇到 contents.rst 错误。

```
# The master toctree document.
master_doc = 'index'
```

修改下面两行以改变主题风格。

```
# html_theme = 'alabaster'
html_theme = 'sphinx_rtd_theme'
```

5. 创建案例文件

主要以 rst 为格式的文件，创建个 readme.rst，里面通过 rst 语法格式书写作者信息，在 index.rst 中配置 readme。具体内容可以研究文末参考文档。

6. 清空编译项目

此时在命令行窗口继续执行下面的命令。

```
make clean
make html
```

构建过程中如果报错，需要检查环境变量的设置以及配置或者文档写的语法不正确，到此本地文档就完成了，可以打开 build 中 index.html 查看，后面更改文件之后可以通过这两条命令迭代更新文档。

7. 托管 Github

Github 上创建一个项目 CGTDCourse，将项目所有文件上传 Github 管理。

8. ReadTheDocs 配置自动编译

登陆 ReadTheDocs 官网，import 这个 Github 项目之后构建，构建成功阅读文档即可，之后在本地写的文章只要通过 Git 提交到代码仓库，ReadTheDocs 会自动构建成在线文档。

参考文档

《Sphinx+github+ReadtheDocs 书写笔记》

《Python Cookbook 3rd Edition Documentation》

14.2.36 几款录屏软件对比

录屏软件有操作系统平台之分，Win 操作系统相对多一些。

Bandicam 录屏在 Window 系统使用整体还可以，很遗憾没有 mac 版本。

Snagit 录屏偶尔会录完保存的时候会卡崩，丢了文件，很头疼，有 Win 版本也有 mac 版本。

Camstia 录制出来的视频需要二次输出 mp4，只想录制视频比较蛋疼，有 win 和 mac 版本。

N 卡自带 GeForce 录屏功能，得有 N 卡才行，有时候快捷键调不出来。

OBS 跨平台免费软件，我用我的 mac pro 录制掉帧非常严重，不知道原因，很多人推荐，弃了，估计是吃硬件。

mac 自带 Quick Time 录屏，Shift+Command+5，默认设置录制的视频比较大，比较蛋疼。

Screenflow 一款 mac 录屏，也是需要二次输出 mp4，比较蛋疼。

格式工厂也可以录屏，只有 win 版本，没有 mac 版本。

WPS 也可以录屏，mac pro 一直卡死，还是算了。

Captura 没去试过，只有 win 版本，有人推荐。

screenbits 没有试过。

14.2.37 Redshift 导致 Houdini 渲崩的可能原因

软件版本：

- Houdini 18.0.416
- Redshift 3.0.20

测试的是一张英伟达 GTX 1060 显卡，实际我有尝试别的显卡，比如 GTX 1050 Ti、GTX 1070 Ti、GTX 1080 Ti、GTX TITAN X、RTX 2080 Super 都出现了相同的问题，Redshift 点击渲染导致 Houdini 直接崩掉，没有任何商量的余地。尝试了几种解决方案都没用，最后不得不重装系统得以解决。

但是自己电脑实在不想重装系统，太多东西要重新配置，所以还是纠结其原因，Houdini Crash 完可以从 Redshift 安装包中找到日志。

```
C:\ProgramData\Redshift\Log
```

在 latest log 中你可以找到这样的错误信息。

```
================================================================================================================
ASSERT FAILED

File OptiXDenoiser7.cpp
Line 137

Failed to init Optix 7.0
================================================================================================================
```

从错误信息上可以怀疑是显卡驱动可能存在问题，实际我的显卡驱动已经是相对比较新的版本了，尝试到英伟达官网下载驱动。

https://www.nvidia.cn/Download/index.aspx?lang=cn

注意下载类型，我之前一直使用 Studio 驱动程序（SD），重新下载最新版本 Game Ready 驱动程序（GRD）安装完渲染没有什么问题，如果你们也遇到类似问题，可以尝试更新自己的显卡驱动。

14.2.38 Win10 远程桌面出现身份验证错误

Win10 操作系统在使用 mstsc 远程桌面连接的时候偶尔会遇到

出现身份验证错误。要求的函数不受支持远程计算机：这可能是由于 CredSSP 加密 Oracle 修正。

遇到这个问题的时候，在遇到问题的计算机上 Win+R 打开 “运行”，输入 gpedit.msc 打开本地组策略编辑器。

找到计算机配置 > 管理模板 > 系统 > 凭据分配 > 加密 Oracle 修正。

选择启用并选择易受攻击即可。

14.2.39 移除 CD 驱动器

首先你得下载一个软件 UltraISO 安装，这个软件也是做 U 盘系统会用到的一款软件。

打开软件选择继续试用，找到菜单选项 > 配置 > 虚拟光驱 > 设备数量 > 无，确认。

14.2.40 RenderBus 客户端配置 Sock5 代理

参考下面两篇文章，配置 Sock5 代理即可解析文件提交任务。

https://www.renderbus.com/support-new/inssub https://www.renderbus.com/support-new/network-requirements

14.2.41 路由器配置固定 IP

路由器 WAN 口局域网的 IP 是可以设置的，LAN 口的局域网 IP 也是可以设置的，一般路由器的 IP 要么是 192.168.0.1，要么是 192.168.1.1。具体如何看这个 IP 非常简单，将路由器接猫，找个笔记本接路由器 LAN 口，一般路由器默认会是 DHCP 自动给笔记本分配 IP，这个时候你就可以看到路由器的网关，这个网关就是路由器的局域网 IP，可以通过此 IP 访问路由器的设置页面。比如一款红米路由器，它的网关 IP 就是 192.168.31.1。

固定 IP 我一般是采取关掉路由器的 DHCP 服务，通过设置每台电脑的 IP 去固定每台电脑的 IP，自然域环境中需要设置域的 DNS。

现在有个需求需要实现，就是办公室，会议室以及机房之间距离太远，老板要求会议室和办公室要覆盖一个他专用的无线网，可以访问内网，可以移动办公，专线无线网只能他可以使用。现在手头上有两个路由器，如何搭建一个无线网而无需切换呢?

本来想搞无线扩展卡或者无线 AP 的东西，想想还要购买硬件，而且无线的效果不一定好，准备好两个路由器，分主路由 A 和副路由 B，首先将路由 A 接到猫，修改网关 IP 为 192.168.0.1，设置无线网络名比如 DO-VFX，开启 DHCP 服务（默认是开启的），然后将路由 B 也接到猫，设置网关 IP 为 192.168.0.2，注意是和路由 A 在同一个 IP 段，设置无线网络名也为 DO-VFX，关闭 DHCP 服务（切记），路由 A 留在机房，路由 B 丢到办公室，注意下面的接法，将路由 A 的一个 LAN 口和路由 B 的一个 LAN 口相连，这点非常重要，然后你在办公室以及会议室就都可以访问 DO-VFX 无线网络啦。

14.2.42 reStructuredText(rst) 语法说明文档

- 注释

```
.. comment
```

- 代码块

```
.. code-block:: python

    code
```

- 标题

```
=========
一级标题
=========


---------
二级标题
---------
```

- 列表

```
* 符号列表 1
* 符号列表 2
```

- 表格

```
==========  =========
第一列        第二列
==========  =========
```

参考文档

《Quick reStructured Text》

《reStructuredText(rst) 快速入门语法说明》

14.2.43 Shell 技术

cd == change directory

pwd == print working directory

想在当前文件夹路径下快捷打开命令行可以在资源管理器直接输入 cmd 即可

14.2.44 中小型企业好用的软件推荐

- 火绒安全
- Rocket.Chat
- 网易邮箱大师
- WPS
- 钉钉
- 阿里云邮箱
- Snipaste
- Deadline
- Shotgun
- 腾讯会议
- Splashtop
- 7-Zip

- XMind
- Navicat（访问数据库表）
- VS Code（开源跨平台）
- Notepad++
- Seafile
- 拖把更名器
- QuickTime
- Chrome（谷歌翻译）
- Firefox
- Git
- TortoiseGit
- Connector
- VMware
- CentOS
- KMS
- Squid
- Bandicam（Win 录屏，Mac 录屏使用 QuickTime 自带录屏）
- IObit
- 格式工厂（可以录屏，可以加字幕）
- RDCMan（局域网远程桌面）

14.2.45 Windows AD 域服务器搭建

14.2.46 路由跟踪指令 traceroute

- Linux

```
nslookup do-vfx.shotgunstudio.com
traceroute do-vfx.shotgunstudio.com
nslookup sg-media-tokyo.s3-accelerate.amazonaws.com
traceroute sg-media-tokyo.s3-accelerate.amazonaws.com
```

- Windows

```
nslookup do-vfx.shotgunstudio.com
tracert do-vfx.shotgunstudio.com
nslookup sg-media-tokyo.s3-accelerate.amazonaws.com
tracert sg-media-tokyo.s3-accelerate.amazonaws.com
```

14.2.47 TrueNAS：系统安装

这里演示下载与安装 TrueNAS CORE（TrueNAS SCALE 还相对不完善）

https://www.truenas.com/download-truenas-core/

使用 Win32DiskImager 写 U 盘系统，然后从 U 盘启动即可

14.2.48 VMware 虚拟机 Edge 浏览器右键字体模糊

在浏览器中找到设置 > 系统和性能 > 使用硬件加速 (如可用)，将此选项禁用即可

14.2.49 维基百科镜像

国内上维基百科步步维艰，可以通过维基百科镜像访问。

参考文档

- http://www.zgc261.com/wikipedia.html
- https://www.wanweibaike.com/

14.2.50 Win Server 2012 文件共享权限管理

正常来说我们将一个分区共享出去并配置 Everyone 读写权限，这是个一劳永逸的方案，任何用户都能访问共享，但是经常我们有具体文件夹具体权限分配的问题，比如一个分区中有个档案文件夹需要关闭所有人访问权限，只授予某些人的单独权限，可以在档案文件夹上右键选择共享 > 停止共享，这样设置之后就没有用户能访问这个文件夹了，然后在右键 > 属性 > 安全添加某一个用户的权限，这样就可以将这个文件夹权限做了特定用户权限，而无须每个文件夹去配置权限。

14.2.51 Youtube 视频下载

下载可以尝试下面的网址，因为政策一直在变，也要跟着变化

https://loader.to/en55/1080p-video-downloader.html

下载字幕

https://downsub.com/

下载中英混合字幕（SRT 中文在上，以中文时间轴为准）

最新版格式工厂混流加音频与字幕，字幕加粗即可

CHAPTER 15

Clarisse

如今的 VFX 与 CG 项目需要比以往更加精细和复杂的环境。Clarisse iFX 为用户提供能满足这一需求的工具。艺术家能生成精美和丰富的数字化环境，达到前所未有的交互水平，同时处理数不胜数的多边形。布局、场景构建、布景和摄影机投影都为了实现全面的灵活性而打造；由实时几何体更新提供动力的非线性工作流程意味着艺术家能随时随地工作，与他人的合作也更多。

Contents:

15.1 Clarisse Python 开发界面

GUI 依然是通过 Python 代码来实现，同样的道理也有两种方案，第一种方案是用 Clarisse 封装的 GUI 库，以 Gui 开头的 Class，它封装的比较全面，可以一用，另一种方案就是用 Python 中 PyQt 模块。

```
window = ix.api.GuiWindow(ix.application, 0, 0, 640, 480)
window.show()
while window.is_shown(): ix.application.check_for_events()
```

```
class CustomButton(ix.api.GuiPushButton):
    def __init__(self, parent, x, y, w, h, label):
        ix.api.GuiPushButton.__init__(self, parent, x, y, w, h, label)
        self.connect(self, "EVT_ID_PUSH_BUTTON_CLICK", self.on_click)

    def on_click(self, sender, evtid):
```

(下页继续)

(续上页)

```
        name = list.get_selected_item_name()

        if name == "Cube":
            ix.cmds.CreateObject("box", "GeometryBox", "Global", "project://scene")
            print("Created a cube success!!!")
        elif name == "Sphere":
            ix.cmds.CreateObject("sphere", "GeometrySphere", "Global", "project://scene")
            print("Created a sphere success!!!")

if __name__ == "__main__":
    app_x = ix.application.get_event_window().get_position()[0]
    app_y = ix.application.get_event_window().get_position()[1]
    app_w = ix.application.get_event_window().get_width()
    app_h = ix.application.get_event_window().get_height()

    window_x = 150
    window_y = 30

    window = ix.api.GuiWindow(ix.application,
                              app_x + (app_w - window_x) / 2,
                              app_y + (app_h - window_y) / 2,
                              window_x,
                              window_y,
                              "Create Object")

    list = ix.api.GuiListButton(window, 0, 0, 100, 30)
    list.add_item("Cube")
    list.add_item("Sphere")
    btn = CustomButton(window, 100, 0, 50, 30, "OK")

    window.show()

    while window.is_shown():
        ix.application.check_for_events()
```

```
for method in dir(ix.api):
    if "EVT_" in method:
        print(method)
```

15.2 Clarisse Python 开发环境

在 Clarisse 测试 Python 代码需要打开 Script Editor 和 Log 两个面板，Log 面板中将会获取一些反馈结果。

或者选择菜单 Layout>Presets>Scripting Workshop 即可。

```
print(type(ix))
# <type 'module'>
print(dir(ix))
# ['ApplicationSelection', '__builtins__', '__doc__', '__file__', '__name__', '__package_
↪_', '_get_of_object', 'add_attribute', 'api', 'application', 'begin_command_batch',
↪'check_need_save', 'cmds', 'create_context', 'create_generic_object', 'create_object',
↪'delete_item', 'disable_command_history', 'disable_echo_command', 'enable_command_
↪history', 'enable_echo_command', 'end_command_batch', 'export_context_as_project',
↪'export_geometries', 'export_geometry', 'export_render_archive', 'get_current_context',
↪ 'get_current_frame', 'get_item', 'import_geometries', 'import_geometry', 'import_image
↪', 'import_images', 'import_map_file', 'import_map_files', 'import_project', 'import_
↪scene', 'import_volume', 'import_volumes', 'inspect', 'is_context_exists', 'is_gui_
↪application', 'is_interactive_application', 'is_process_application', 'item_exists',
↪'ix', 'load_project', 'log_error', 'log_info', 'log_warning', 'make_absolute_of_path',
↪'os', 'reference_export_context', 'reference_file', 'reference_make_local', 'render_
↪image', 'save_bmp', 'save_exr16', 'save_exr32', 'save_image', 'save_jpg', 'save_png16',
↪ 'save_png8', 'save_project', 'save_tga', 'save_tif16', 'save_tif32', 'save_tif8',
↪'selection', 'set_current_context', 'set_current_frame']
print(dir(ix.selection))
# ['__doc__', '__getitem__', '__module__', 'add', 'deselect_all', 'get', 'get_contexts',
↪'get_count', 'get_objects', 'is_empty', 'select', 'select_all']
print(ix.selection.get_count())
# 获取当前选择物体的个数
for i in range(ix.selection.get_count()):
    print(ix.selection[i])
```

Clarisse API 文档在 help>Contents>Reference>Scripting/API 中，因为是从 C++ API 封装而来，大部分还是要看 C++ 文档来写对应的 Python 代码。

```
for i in range(ix.selection.get_count()):
    sel = ix.selection[i]

    if sel.is_kindof("GeometryBox"):
        sel.attrs.scale[0] = 10
        sel.attrs.scale[1] = 10
        sel.attrs.scale[2] = 10
```

(下页继续)

(续上页)

```
    else:
        print(ix.selection[i])
```

```
ix.cmds.CreateObject("box", "GeometryBox", "Global", "project://scene")
```

CHAPTER 16

Katana

The Foundry Katana 是一款相当优秀的专业化 3D 渲染软件，The Foundry Katana 最新版功能强劲，为用户提供了专业性的灯光照明增强功能，为您带来高效的灯光照明效果，The Foundry Katana 软件便捷好用，能够满足当今最苛刻的视觉效果项目的需求。

Contents:

16.1 Katana 基础操作

默认视窗分为五大块: 渲染窗口、大纲窗口、三维视窗、节点编辑窗口和属性面板。

三维视窗的操作和 Maya 相同，非常简单，结合 Alt+ 左中右键可以调整视窗，按 F 居中显示场景物体。

选中相应的物体可以通过 WER 键来移动、旋转、缩放。

可以自定义视窗布局，通过 Layouts>Save Current Layout 保存自定义视窗。

在 Node Graph 窗口可以通过 Tab 来创建节点（也可以通过 New 菜单来创建节点）。

节点的左侧方格是在 Scene Graph 中显示，节点右侧方格是显示节点的属性面板，对应快捷键 V 和 E。

修改节点名称可以通过属性面板修改或者选中节点按 N 键修改。

快捷键 G 可以将选中节点打组 Group，Alt+G 可以将选中节点打组 GroupStack，可以使用 Ctrl+ 鼠标中键或者 Ctrl+Enter 可以进组修改。

Node Graph>Edit>Fit Backdrop Node to Selected Nodes 给节点归类添加 Backdrop。

Viewer(Hydra)>Viewer>Monitor Layer: 视窗渲染

Viewer(Hydra)>Viewer>Monitor Layer Options>Display While Manipulating: 视窗实时监视渲染

Viewer(Hydra)>Viewer>Monitor Layer Options>Ignore Alpha: 视窗透明渲染

Scene Graph>Live Render Updates: 勾选需要实时渲染的选项。

常用一些节点解释

节点类型	节点描述
PrimitiveCreate	模型节点
Merge	合并节点
Dot	点节点
HierarchyCopy	复制节点
Transform3D	移动节点
CameraCreate	相机节点
Material	材质节点
MaterialAssign	材质赋予节点
GafferThree	灯光组节点
RenderSettings	渲染设置节点
DISettings	采样设置节点
Render	渲染节点

XGen 毛发进 Katana 步骤

- [ ] 如何导入显示?
- [ ] 默认渲染器如何渲染?
- [ ] Katana 中心化部署流程?
- [ ] Arnold 如何渲染，配置与渲染?

文字 Katana 部署 Arnold 文字 Katana 导入 XGen 流程文字 Katana 中心化部署

Katana 中隐藏模型渲染?

Katana 渲染图保存对比?

Katana 背景色怎么设置?

Katana 帧数范围在哪设置?

参考文档:

- https://rmanwiki.pixar.com/display/RFK/XGen+in+Katana
- https://docs.arnoldrenderer.com/display/A5KTN/Installation

16.2 Katana 回调事件处理机制

回调和事件在很多软件都是存在一种处理机制，回调是添加到 Katana 环境中的一段 Python 代码，当 Katana 中发生各种事件（例如创建节点或加载脚本）时，它会自动运行。事件是由于用户操作或其他来源（例如单击鼠标或按下键）而发生的操作。事件处理程序是用于处理事件的例程，允许程序员编写将在事件发生时执行的代码。

```
print(dir(Callbacks))
print(dir(Callbacks.Type))
# ['Type', '__class__', '__delattr__', '__dict__', '__doc__', '__format__', '__
↪getattribute__', '__hash__', '__init__', '__module__', '__new__', '__reduce__', '__
↪reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__',
↪'__weakref__', 'addCallback', 'delCallback']
# ['__doc__', '__module__', 'exportAsset', 'finalize3DNodeChanges', 'onGafferLightCreated
↪', 'onGafferMasterMaterialCreated', 'onGafferRigCreated', 'onGafferShaderSelected',
↪'onLiveRenderCommand', 'onLiveRenderUpdate', 'onLookFileMaterialActiveSet',
↪'onLookFileMaterialSet', 'onMasterMaterialAssigned', 'onNewScene', 'onNodeCreate',
↪'onNodeDelete', 'onRenderSetup', 'onSceneAboutToLoad', 'onSceneLoad', 'onSceneSave',
↪'onShutdown', 'onStartup', 'onStartupComplete', 'onTabCreated', 'postLiveGroupPublish',
↪ 'postLookFileBake', 'preLiveGroupPublish', 'preLookFileBake']
```

```
def sayHello(**kwargs):
    for i in kwargs.keys():
        print(i)

Callbacks.addCallback(Callbacks.Type.onSceneLoad, sayHello)
```

16.3 Katana 配置本地离线帮助文档

非常有用的小功能，默认 Katana 菜单 Help 中的帮助文档都是链接到在线文档，虽然在线文档可能是访问最新版本文档，但很多时候限于网速原因以及大多时候不能上网的局限，需要配置本地离线帮助文档，查看当前版本的帮助文档。

找到主菜单 Edit>Perferences>documentation 将 source 从 Remote(via learn.foundry.com) 配置成 Local 即可，此时查看 Online Help、Developer Guide 以及 API Reference 都是本地离线帮助文档了，玩 API 特别方便吧。

NukeX 本地离线帮助文档会遇到一点问题，虽然 NukeX 菜单 Edit>Preferences…>Behaviors>Documentation 中也有设置 documentation source 为 local 的选项，但是设置和不设置好像是一样的，我们只能魔改本地 index.html 文件了。以 Nuke 12.1v2 为例，找到 C:Program FilesNuke12.1v2Documentationindex.html 文件，用记事本打开。

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
<title>
    Nuke 12.1.2
</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link href="img/visionmongers.css" rel="stylesheet" type="text/css">
</head>
<body>
<h1>
    <img src="img/nuke.png" alt="[Nuke Logo]">
</h1>
<blockquote>
    <p>
    Nuke is an Academy Award winning compositor.
    It has been used to create extraordinary images on scores of
    feature films including <em>Avatar</em>, <em>Harry Potter And The Deathly Hallows</
↪em>,
    <em>Tron: Legacy</em>, <em>Super 8</em> and <em>The Curious Case Of Benjamin Button</
↪em>
    as well as countless commercials and music videos.
    The following
    links take you to Nuke user documentation in various forms,
    here you'll find answers to your questions, whether you're
    a beginner, an experienced Nuke user or a Nuke developer.
    </p>

    <h3>Links</h3>

    <p>
    <b>
        <a href="https://www.foundry.com/products/nuke/online-help">Nuke User Guide␣
↪Tutorial Resources</a>
    </b> - A link to the Tutorial resources used in the Nuke User Guide.
    (These can be located on our website on the Support > User guides page.).
    </p>
    <p>
    <b>
        <a href="http://learn.foundry.com/nuke/12.1/">Foundry Online Help</a>
```

(下页继续)

(续上页)

```
    </b> - The Foundry online help for Nuke.
    </p>
    <p>
    <b>
        <a href="http://docs.thefoundry.co.uk/nuke/12.1/pythondevguide/">Nuke Python␣
↪Developers Guide</a>
    </b> - HTML documentation for Nuke Python developers.
    </p>
    <p>
    <b>
        <a href="http://docs.thefoundry.co.uk/hiero/12.1/hieropythondevguide/">Hiero␣
↪Python Developers Guide</a>
    </b> - HTML documentation for Hiero Python developers.
    </p>
    <p>
    <b>
        <a href="http://docs.thefoundry.co.uk/nuke/12.1/pythonreference/">Python␣
↪Scripting Reference</a>
    </b> - An HTML reference for all of Nuke's Python methods and types.
    </p>
    <p>
    <b><a href="Tcl/index.html">TCL Scripting</a></b> - An HTML resource for TCL␣
↪scripting.
    </p>
    <p>
    <b>
        <a href="Tcl/group__tcl__expressions.html">Knob Math Expressions</a>
    </b> - An HTML list of functions you can use in Nuke expressions.
    </p>
    <p>
    <b>
        <a href="NDKExamples/index.html">C++ Plug-in Development</a>
    </b> - A link to Nuke Developer Kit (NDK) resources.
    </p>
    <p>
    <b>
        <a href="http://docs.thefoundry.co.uk/nuke/12.1/BlinkKernels/">Guide to Writing␣
↪Blink Kernels</a>
    </b> - An HTML guide for users of the BlinkScript node in Nuke.
    </p>
```

(下页继续)

(续上页)

```
</blockquote>
<br>
<hr>
    <table width="100%"  border="0" cellspacing="0" cellpadding="0">
    <tr>
        <td>
        <span class="legal">
            &copy;2020 The Foundry Visionmongers, Ltd.  All Rights Reserved.
        </span></td>
        <td>
        <div align="right">
            <span class="legal">
            <a href="http://www.foundry.com" target="_blank">www.foundry.com</a>
            </span>
        </div>
        </td>
    </tr>
    </table>
</body>
</html>
```

直接修改链接的网址路径，像下面这样。

```
<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
<title>
    Nuke 12.1.2
</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link href="img/visionmongers.css" rel="stylesheet" type="text/css">
</head>
<body>
<h1>
    <img src="img/nuke.png" alt="[Nuke Logo]">
</h1>
<blockquote>
    <p>
    Nuke is an Academy Award winning compositor.
    It has been used to create extraordinary images on scores of
```

(下页继续)

(续上页)

```
    feature films including <em>Avatar</em>, <em>Harry Potter And The Deathly Hallows</
↪em>,
    <em>Tron: Legacy</em>, <em>Super 8</em> and <em>The Curious Case Of Benjamin Button</
↪em>
    as well as countless commercials and music videos.
    The following
    links take you to Nuke user documentation in various forms,
    here you'll find answers to your questions, whether you're
    a beginner, an experienced Nuke user or a Nuke developer.
    </p>

    <h3>Links</h3>

    <p>
    <b>
        <a href="https://www.foundry.com/products/nuke/online-help">Nuke User Guide␣
↪Tutorial Resources</a>
    </b> - A link to the Tutorial resources used in the Nuke User Guide.
    (These can be located on our website on the Support > User guides page.).
    </p>
    <p>
    <b>
        <a href="html/content/learn_nuke.html">Foundry Online Help</a>
    </b> - The Foundry online help for Nuke.
    </p>
    <p>
    <b>
        <a href="PythonDevGuide/Nuke/index.html">Nuke Python Developers Guide</a>
    </b> - HTML documentation for Nuke Python developers.
    </p>
    <p>
    <b>
        <a href="PythonDevGuide/Hiero/index.html">Hiero Python Developers Guide</a>
    </b> - HTML documentation for Hiero Python developers.
    </p>
    <p>
    <b>
        <a href="PythonReference/index.html">Python Scripting Reference</a>
    </b> - An HTML reference for all of Nuke's Python methods and types.
    </p>
```

(下页继续)

(续上页)

```
    <p>
    <b><a href="Tcl/index.html">TCL Scripting</a></b> - An HTML resource for TCL␣
↪scripting.
    </p>
    <p>
    <b>
        <a href="Tcl/group__tcl__expressions.html">Knob Math Expressions</a>
    </b> - An HTML list of functions you can use in Nuke expressions.
    </p>
    <p>
    <b>
        <a href="NDKExamples/index.html">C++ Plug-in Development</a>
    </b> - A link to Nuke Developer Kit (NDK) resources.
    </p>
    <p>
    <b>
        <a href="Blink/index.html">Guide to Writing Blink Kernels</a>
    </b> - An HTML guide for users of the BlinkScript node in Nuke.
    </p>
</blockquote>
<br>
<hr>
    <table width="100%"  border="0" cellspacing="0" cellpadding="0">
    <tr>
        <td>
        <span class="legal">
            &copy;2020 The Foundry Visionmongers, Ltd.  All Rights Reserved.
        </span></td>
        <td>
        <div align="right">
            <span class="legal">
            <a href="http://www.foundry.com" target="_blank">www.foundry.com</a>
            </span>
        </div>
        </td>
    </tr>
    </table>
</body>
</html>
```

重启 NukeX，选择菜单 Help>Documentation，就可以直接查看本地的离线帮助文档了，美滋滋，美中不足

的是还有一部分链接是在线链接，但大部分都是 OK 的，也无关紧要。

16.4 Katana 自定义节点菜单

```
from Katana import UI4, NodegraphAPI
from UI4.FormMaster.NodeActionDelegate import (BaseNodeActionDelegate,␣
↪RegisterActionDelegate)
from UI4.Manifest import QtCore, QtGui, QtWidgets


class MyNodeActionDelegate(BaseNodeActionDelegate.BaseNodeActionDelegate):
    class _SelectNodeAction(QtWidgets.QAction):

        def __init__(self, parent, node):
            QtWidgets.QAction.__init__(self, 'Select "%s"' % node.getName(), parent)

            self.__node = node
            if node:
                self.triggered.connect(self.__triggered)
            self.setEnabled(self.__node is not None
                            and not self.__node.isLocked(True))

        def __triggered(self, checked):
            NodegraphAPI.SetNodeSelected(self.__node, True)

    class _DeselectNodeAction(QtWidgets.QAction):

        def __init__(self, parent, node):
            QtWidgets.QAction.__init__(self, 'Deselect "%s"' % node.getName(), parent)

            self.__node = node
            if node:
                self.triggered.connect(self.__triggered)
            self.setEnabled(self.__node is not None
                            and not self.__node.isLocked(True))

        def __triggered(self, checked):
            NodegraphAPI.SetNodeSelected(self.__node, False)

    def addToContextMenu(self, menu, node):
```

(下页继续)

(续上页)

```
        menu.addAction(self._SelectNodeAction(menu, node))
        menu.addAction(self._DeselectNodeAction(menu, node))

    def addToWrenchMenu(self, menu, node, hints=None):
        pass


RegisterActionDelegate("CameraCreate", MyNodeActionDelegate())
```

- https://support.foundry.com/hc/en-us/articles/208838305-Q100108-Adding-custom-menu-items-to-the-context-menu

16.5 Katana 自定义插件层级结构

参考示例 C:Program FilesKatana3.5v2pluginsResourcesExamples 或者 $KATANA_ROOT/plugins/Resources/Examples

- Args
- AssetPlugins
- Gaffer*
- GenericAssign
- Importmatic*
- Libs
- Macros
- Ops
- Plugins
- Python: 配置 PYTHONPATH 或通过 Startup 添加 sys.path 或直接在插件中添加 sys.path。
- RenderBin
- Resolutions
- Shaders
- Shelves: 自定义主菜单工具架工具
- ShelvesNodeSpecific: 自定义节点参数面板工具架工具
- ShelvesScenegraph: 自定义 Scene Graph 面板工具架工具
- Startup: init.py 自启动执行脚本
- SuperTools*

- Tabs*
- UIPlugins
- ViewerManipulators
- https://learn.foundry.com/katana/Content/ug/installation_licensing/katana_resources.html

16.6 Katana 自定义工具架工具

工具架工具有几个地方可以定义 Shelf Actions，主菜单上的齿轮菜单、Scene Graph 中的齿轮菜单、节点参数面板中的齿轮菜单。

点击齿轮菜单 >Add>New Shelf>New Item，工具架工具会存储在下面路径。

C:Users<USERNAME>.katanaShelves

工具架的几个参数定义

```
"""
NAME: Set abcAsset Path
ICON: icon.png
KEYBOARD_SHORTCUT:
SCOPE:
Enter Description Here

"""
```

每个工具架工具代码中开头都有这么一段配置脚本。

NAME 即工具的名称，ICON 设置工具图标，KEYBOARD_SHORTCUT 设置工具的快捷方式。

图标文件在这里 C:Program FilesKatana3.5v2binpythonUI4ResourcesIcons

即 $KATANA_HOMEbinpythonUI4ResourcesIcons

SCOPE 限制特定节点类型，写描述。

我们可以在.katana/Shelves 创建文件夹 Custom_Shelves，然后创建一个 firstShelfTool.py 文件，写入如下内容。

```
"""
NAME: Float Selected
ICON: Icons\Scenegraph\camera128.png
KEYBOARD_SHORTCUT: T
SCOPE: none
Float Selected Nodes via Keyboard Shortcut
```

(下页继续)

(续上页)

```
"""

# Get list of selected nodes
nodeList = NodegraphAPI.GetAllSelectedNodes()

# Find Nodegraph tab and float nodes
nodegraphTab = UI4.App.Tabs.FindTopTab("Node Graph")

if nodegraphTab:
    nodegraphTab.floatNodes(nodeList)
```

- https://support.foundry.com/hc/en-us/articles/360001163884-Q100401-Creating-a-Shelf-Item-in-Katana
- https://support.foundry.com/hc/en-us/articles/360001235064
- https://learn.foundry.com/katana/content/tg/shelf_item_scripts/shelf_item_scripts.html?TocPath=Technical%20Information%7C_____5#User-def

16.7 Katana 加载模块文件的三种方案

最基本的方案就是在代码中将模块路径添加到 sys.path

另一种是写 bat 文件，设置 PYTHONPATH 环境变量

还有一种麻烦一点通过 Startup/init.py 文件配置 sys.path

16.8 Katana 开发文档

```
import Katana
print(type(Katana))
print(dir(Katana))
# <type 'module'>
# ['AssetAPI', 'AssetBrowser', 'AttrDump', 'BezierModule', 'CEL', 'CacheManager',
↪'Callbacks', 'CatalogAPI', 'CatalogManager', 'ChildProcess', 'ColorPaletteManager',
↪'ColorUtils', 'Configuration', 'Decorators', 'Documentation', 'DrawingModule',
↪'EnvUtils', 'ExpressionMath', 'FaceSelectionManager', 'FarmAPI', 'FarmManager',
↪'FileUtils', 'FnAttribute', 'FnGeolib', 'FnGeolibServices', 'FnKatImport', 'FormMaster
↪', 'GeoAPI', 'Hints', 'Imath', 'Initialize', 'KatanaFeatures', 'KatanaFile',
↪'KatanaPrefs', 'KatanaResources', 'LayeredMenuAPI', 'LensDistortUtils', 'LiveRenderAPI
↪', 'LogGLHandlers', 'LogGLHandlersOldLevel', 'LookFileBakeAPI', 'MachineInfo',
↪'Manifest', 'MediaCache', 'MediaCacheHandler', 'Naming', 'NodeDebugOutput',
↪'NodeGraphView', 'NodeMaster', 'NodegraphAPI', 'Nodes2DAPI', 'Nodes3DAPI',
↪'NonUIPluginManager', 'OCIO', 'OpenEXR', 'OpenGL', 'PluginSystemAPI', 'Plugins',
↪'PrefNames', 'PyFCurve', 'PyRerenderEventMapper', 'PyScenegraphAttr', 'PyXmlIO',
↪'QT4Browser', 'QT4Color', 'QT4FormWidgets', 'QT4GLLayerStack', 'QT4Panels', 'QT4Widgets
↪', 'QTFCurve', 'Qt', 'QtCore', 'QtDesigner', 'QtGui', 'QtMultimedia', 'QtNetwork',
```

(下页继续)

(续上页)

- KatanaFile：文件操作模块
- RenderManager：渲染输出模块

源代码：C:Program FilesKatana3.5v2binpython

```
print(KatanaFile.__file__)
print(NodegraphAPI.__file__)
print(RenderManager.__file__)

# C:\Program Files\Katana3.5v2/bin/python\PyUtilModule\KatanaFile.pyc
# C:\Program Files\Katana3.5v2/bin/python\NodegraphAPI\__init__.pyc
# C:\Program Files\Katana3.5v2/bin/python\PyUtilModule\RenderManager\__init__.pyc
```

```
renderNode = NodegraphAPI.GetNode("Render")
renderSettings = RenderManager.RenderingSettings()

for frame in range(1, 6):
    print("-" * 80)
    renderSettings.frame = frame
    RenderManager.StartRender("diskRender", node=renderNode, settings=renderSettings)
```

批量修改某一类型节点参数

```
import NodegraphAPI

print(dir(NodegraphAPI))

nodes = NodegraphAPI.GetAllNodesByType("Alembic_In")

for node in nodes:
    print(node)
    parm = node.getParameter("abcAsset")
    oldPath = parm.getValue(0)
    newPath = oldPath.replace("D:/cache/", "Z:/cache/")
    parm.setValue(newPath, 0)
```

创建节点，自动组装

```
rootNode = NodegraphAPI.GetRootNode()
```

(下页继续)

(续上页)

```
print(rootNode)

print(dir(rootNode))

primNode = NodegraphAPI.CreateNode("PrimitiveCreate", rootNode)

print(primNode)

mergeNode = NodegraphAPI.CreateNode("Merge", rootNode)

print(mergeNode)

print(dir(mergeNode))

print(help(mergeNode.addInputPort))

port = mergeNode.addInputPort("i11")

print(port)
print(dir(port))
print(dir(primNode))
port.connect(primNode.getOutputPorts()[0])


node = NodegraphAPI.GetNode("InChar")

print(node)

print(node.getBaseType())
print(node.getParameter("type").getValue(0))

print(mergeNode.getInputPorts())

import NodegraphAPI

node = NodegraphAPI.GetNode("Material_Stack")

print(type(node))
print(dir(node))
```

(下页继续)

(续上页)

```
node.getChildNodes()

material = NodegraphAPI.CreateNode("Material", NodegraphAPI.GetRootNode())

node.buildChildNode(material)

print(dir(material))

print(dir(NodegraphAPI))

material = NodegraphAPI.CreateNode("Material")
material.setParent(NodegraphAPI.GetRootNode())

print(material)

print(NodegraphAPI.GetAllNodes())
```

参考文档:

- https://learn.foundry.com/katana/3.2/dev-guide/py-modindex.html
- https://learn.foundry.com/katana/dev-guide/index.html

16.9 Katana 开发环境

Katana 中支持三种语言编程：Python，Lua 和 C++。Python 应用于工作流非常方便，Lua 用于 OpScript 节点。

Katana 中选择菜单 Tabs>Python 即可打开 Katana 脚本编辑器。

API 文档在 Help>API Reference>Python APIs。

```
print(KatanaFile.__file__)
print(type(KatanaFile))
print(dir(KatanaFile))
# C:\Program Files\Katana3.5v2/bin/python\PyUtilModule\KatanaFile.pyc
# <type 'module'>
# ['AllowFileOverwrite', 'AssetAPI', 'Callbacks', 'Configuration', 'CrashSave',
↪'CrashSaveDisable', 'CrashSaveEnable', 'CreateSceneAsset', 'Export', 'FileUtils',
↪'GetCrashActions', 'GetCrashTime', 'GetKatanaXMLElement', 'GetNextCrashFileName',
↪'GetViableCrashFiles', 'Import', 'IsFileDirty', 'IsFileSavedAsAsset', 'Load', 'MAX_
↪CRASH_FILES', 'New', 'NodegraphAPI', 'Nodes2DAPI', 'Nodes3DAPI', 'Paste',
↪'PopFileDirtyState', 'PostCreateSceneAsset', 'PushFileDirtyState',
↪'RegisterCrashHandler', 'Revert', 'Save', 'ScenegraphBookmarkManager', 'SetFileDirty',
↪'UndoEntries', 'Utils', 'WasFileLoadedFromCrashFile', 'WasFileVersionedOnLoad', '_
↪CleanupRenderNodes', '_ContainsModifiedLiveGroupNodes', '_CrashActions', '_CrashDisable
```

(下页继续)

(续上页)

参考文档:

- https://learn.foundry.com/katana/3.5/dev-guide/index.html
- https://support.foundry.com/hc/zh-cn/articles/360001288699-Q100443-Katana%E4%B8%AD%E7%9A%84%E8%84%9A%E6%9C%AC%E5%92%8C%E7%BC%96%E7%A8%8B

16.10 Katana XGen 毛发渲染流程

解决三个问题。

XGen 毛发 nHair 驱动解算流程?

在制作毛发资产的时候需要通过 Guides To Curves 转引导曲线，以引导曲线转 nHair 动力学做毛发解算。

Katana 渲染 XGen 毛发的流程?

这里涉及到环境部署，首先 Katana 的安装，比较傻瓜式，这里不细说，然后是 KTOA（Arnold for Katana）的安装，KTOA 可以部署到服务器（也就是共享文件夹中），也是傻瓜式部署，部署完安装路径下有一个 launchKtoA.bat 的启动项，我们需要稍加修改，才能渲染 XGen 毛发。内容中 MTOA_PATH 和 MAYA_PATH 需要配置，特别是 MTOA 和 KTOA 的版本搭配有一定的要求。

```
set "KATANA_HOME=C:\Program Files\Katana3.5v2"
set "KTOA_HOME=%cd%"
set "MTOA_PATH=\\server\manager\thirdParty\maya\mtoa\3.3.0.1\2017"
set "MAYA_PATH=C:\Program Files\Autodesk\Maya2017"
set DEFAULT_RENDERER=arnold
set "KATANA_TAGLINE=With KtoA 2.4.0.5 and Arnold 5.4.0.3"

set "path=%KTOA_HOME%\bin;%path%"
set "KATANA_RESOURCES=%KTOA_HOME%"
"%KATANA_HOME%\bin\katanaBin.exe"
```

修改完之后双击 bat 启动 Katana，创建 ArnoldXGen 节点导入.xgen 文件，创建 Material（给 ambient_occlusion 材质），创建 MaterialAssign 赋予材质，创建相机，RenderSettings 以及 Render 节点就可以渲染出 XGen 毛发啦。注意的是，XGen 毛发在 Katana 中是无法显示的，只能渲染。

毛发解算完迭代渲染流程?

解算环节输出引导曲线的 abc 缓存，在 Maya 中替换驱动毛发，然后通过 Export Patches for Batch Render 导出.xgen 文件即可。

关闭 nucleus>Enable，关闭 hairSystem>Use Nucleus Solver，改 static

Katana 本地批量渲染

https://www.aducg.com/2015/07/22/katana-local-batch-rendering-command/

/usr/local/Katana1.5v1/katana –batch –katana-file=/path/to/file/scene.katana –render-node=Render_Node_Name -t 1-20

如何渲染多帧

如何导入相机

如何设置参数

如何随机颜色

```
import NodegraphAPI
from Katana import KatanaFile
from Katana import RenderManager
def messageHandler( sequenceID, message ):
    print message

RenderNode = NodegraphAPI.GetNode('Render') # Getting Render node
renderSettings = RenderManager.RenderingSettings()
renderSettings.mode=RenderManager.RenderModes.DISK_RENDER
renderSettings.asynchRenderMessageCB=messageHandler
renderSettings.asynch=False
for frame in range(6, 100):
    print '-' * 80
    print '\nRendering Node "%s" frame %s...' % (RenderNode.getName(), frame)
    renderSettings.frame = frame
    RenderManager.StartRender('diskRender', node=RenderNode, settings=renderSettings)
```

遇到的坑

- [×] XGen 渲染运动模糊需要单独设置
- [×] XGen 解算的时候引导曲线（所有的）需要单独导出小数帧给会 XGen（-0.1，0.1），在 Katana 中只导入.xgen 文件，.xgen 文件记录了所有缓存路径信息。
- [×] XGen 生长面小数帧也不能有任何问题。
- [×] XGen 需要单独输出生长面的 abc 缓存（小数帧）
- [×] XGen 生长面和引导曲线的缓存不能脱离，小数帧也不能脱离，不然就会 ci 掉

CHAPTER 17

UE4

UE4 是 Unreal Engine 4 的缩写，中文译为 “虚幻引擎 4”，UE4 是一款代码开源、商业收费、学习免费的游戏引擎，支持 PC、手机、掌机等各种平台，能够充分发挥硬件的性能。UE4 虽然也有自己的脚本语言，但性能堪忧，所以在大型游戏开发中人们一般使用 C++。

Contents:

17.1 UE4 安装以及虚幻工程下载使用

首先上虚幻引擎官网下载并安装 Epic，需要注册登陆账号。

然后选择库，点击引擎版本后面加号，选择需要安装的 Unreal Engine 版本，点击安装即可。

虚幻商城中的工程是可以免费或者购买直接使用的。

17.2 UE4 启用 Python 执行环境以及可执行方法

17.3 UE4 蓝图分享网址

「蓝图」是虚幻引擎为开发者开发的一款脚本语言，它已经封装好大量的函数和带有可视化编程的特点，深受广大开发者的喜爱，不过这种方便带来的弊端是性能上的损耗，所以蓝图更适合开发中小型项目。

分享你的蓝图脚本可以使用下面的网址：

```
https://blueprintue.com/
http://blueprintue.cn/
```

临时分享代码的网址还有一个，也挺有意思。

```
https://paste.ubuntu.com/
```

CHAPTER 18

UE5

UE5 是 Unreal Engine 5 的缩写，中文译为 “虚幻引擎 5”，UE5 是一款代码开源、商业收费、学习免费的游戏引擎，支持 PC、手机、掌机等各种平台，能够充分发挥硬件的性能。UE5 虽然也有自己的脚本语言，但性能堪忧，所以在大型游戏开发中人们一般使用 C++。

Contents:

CHAPTER 19

Blender

Blender 是一款开源的三维软件

Contents:

CHAPTER 20

C4D

Cinema 4D 是一套由德国公司 Maxon Computer 开发的三维绘图软件，以其高的运算速度和强大的渲染外挂着称。Cinema 4D 应用广泛，在广告、电影、工业设计等方面都有出色的表现。

Contents:

20.1 C4D 部署 Redshift

C4D 中部署 Redshift 非常简单，在安装 Redshift 的时候如果没有安装 C4D 或者自动配置环境的选项没有勾选，则 C4D 不会自动加载 Redshift 插件。后期部署 Redshift 也非常简单。这里假设你已经安装好了 Redshift 以及 C4D R19，以管理员身份打开命令行窗口（切记以管理员身份运行）。

```
C:\Users\huweiguo>cd C:\ProgramData\Redshift\Plugins\C4D
C:\ProgramData\Redshift\Plugins\C4D>install_c4d.bat R19 "C:\Program Files\MAXON\Cinema␣
↪4D R19\plugins"
```

CHAPTER 21

后记

21.1 书籍

这里将自己阅读过的一些书和读后感推荐给大家，还有很多英文原版书籍，这里就不推荐了，本身看代码就很枯燥了，再看英文心累。

- 《流畅的 Python》：图灵出版的一本极好的 Python 进阶书，很多概念和理论另辟蹊径，讲解的非常好，建议有一定基础的同学多看几遍，不适合入门。
- 《深入理解 Python 特性》：
- 《Python 学习笔记》：作者雨痕自己写的笔记，很多心得写的不错，案例也很新颖，值得多看几遍。
- 《Python Cookbook》：图灵出版的一本 Python 进阶书，针对具体问题给出解决方案，没有具体的知识体系，如果能将书中的知识片段用于工作之中，代码质量将会发生质的飞跃，值得没事的时候翻阅看看并且用于实践。
- 《廖雪峰 Python 教程》：适合入门，很多浅显易懂的案例是极好的，基础不好可以多看几遍，这也是我当初开始写 Python 读的第一本书。
- 《像计算机科学家一样思考 Python》：图灵出版的一本书，教科书式的编排，内容比较繁琐，可以当一本 Python 字典在用到的时候查阅相关知识点，不建议从头看到尾，不过我倒是看过一本《像艺术家一样思考》的书，是一本画画方面的好书，但和 Python 无关，闲暇之余可以看看。
- 《Python 源码剖析》：作者陈儒，一本国产讲 Python 运行原理的书，剖析 Python 底层 C 的实现机制，适合喜欢寻根究底的你，要对 Python 有足够深的理解才能看懂，不推荐新手阅读，但是是 Python 进阶必读的一本书。

- 《Python 学习手册》：图灵出版的长达 800 页的一本书，教科书式的编排，内容很详细，详细到很🇫嗦，知识体系比较杂乱，可以当一本 Python 字典在用到的时候查阅相关知识点，不建议从头看到尾，针对知识点来阅读效果会很好。
- 《Python 技术手册》：图灵出版的长达 600 页的一本书，教科书式的编排，相比《Python 学习手册》，内容上更进阶一些，大段大段理论知识堆砌，比较枯燥乏味。
- 《Python 参考手册》：长达 500 页的一本书，教科书式的编排，内容详细允长，可以当一本 Python 字典在用到的时候查阅相关知识点，不建议从头看到尾，针对知识点来阅读效果会很好。
- 《Python 编程从入门到实践》：入门级，很多新手喜欢啃这本书，我倒是不建议，编程最快的学习方法就是在工作中得以实践，书中很多内容是用不上的，字字斟酌太耗精力，建议一目十行将它看完丢弃。
- 《Python 核心编程》：长达 900 页的一本书，教科书式的编排，内容详细允长，可以当一本 Python 字典在用到的时候查阅相关知识点，不建议从头看到尾，针对知识点来阅读效果会很好。
- 《Python 高级编程》：一本 Python 进阶书，内容大多是比较实用的，不过有点枯燥乏味，可以看看，不适合新手。
- 《编写高质量代码改善 Python 程序的 91 个建议》：一本 Python 进阶书，提升代码质量的 91 条建议，可以茶余饭后阅读，尝试使用更好的解决方案，相同的问题，你会发现代码可以写的更简洁。
- 《Effective Python 编写高质量 Python 代码的 59 个有效方法》：一本 Python 进阶书，提升代码质量的 59 条建议，可以茶余饭后阅读，尝试使用更好的解决方案，相同的问题，你会发现代码可以写的更简洁。
- 《可爱的 Python》：前面故事篇可能跟实际工作关联不大，没多大意思，但从环境篇，语法篇，模块篇，框架篇，友邻篇倒是讲解了很多有用的知识点，估计很多人在故事篇就放弃了，没有看完可以回头再翻翻。
- 《简明 Python 教程》：教科书式的编排，内容比较入门，没多大意思。
- 《Python 高性能编程》：图灵出版的一本 Python 进阶书，内容比较晦涩，探究了更多底层与框架的知识，内容主要目的是编写更高效的 Python 代码，适合寻根究底的你。
- 《21 天学通 Python》：21 天是个噱头，学不学通一门编程语言这和代码量成正比，代码写的越多，理解就会越深，解决方案自然也就更多，这本书内容繁杂允长，看它不如去啃《Python 学习手册》或者 B 站找套白嫖视频教程，学习效率会高很多。
- 《Python 基础教程》：图灵出版的一本 Python 基础书，教科书式的编排，内容没什么新颖的地方，适合入门看看。
- 《Python 语言及其应用》：图灵出版的一本 Python 基础书，教科书式的编排，内容没什么新颖的地方，适合入门看看。
- 《Python 灰帽子》：一本 Python 进阶书，书的内容是面向黑客与逆向工程师，因此很多内容不太容易理解，如果能看懂《Python 源码剖析》，再来看这本书会轻松一些，不过术业有专攻，书中描述的内容在工作中不一定能帮上忙，适合闲暇时阅读。
- 《父与子的编程之旅：与小卡特一起学 Python》：如果实在觉得智商不够用，那就看看这本写给中小学生的 Python 教材吧，如果看不懂就不应该再学习 Python 了，内容以讲故事和发问的方式讲述 Python，

非常有意思，适合对计算机以及编程一无所知的你。

- 《编程小白的第 1 本 Python 入门书》：比较入门级，作者的一些经验之谈，一些基础知识吧，没太多比较有用的内容。
- 《笨办法学 Python》：以习题和解答的方式让读者动手解决问题，入门级，内容比较简单。
- 《Python 算法教程》：算法是编程语言的灵魂，所以闲来无事多看看算法相关资料是有必要的，不推荐新手阅读。
- 《Python 精要参考》：一本 Python 基础书，教科书式的编排，适合入门看看。
- 《Python 3 程序开发指南》：长达 500 页的一本书，教科书式的编排，适合入门看看。

其它编程相关书籍

- 《程序员修炼之道：从小工到专家》：
- 《代码整洁之道》：
- 《代码大全（第 2 版）》：
- 《重构：改善既有代码的设计》：
- 《深入浅出设计模式》：
- 《人月神话》：
- 《程序员的职业素养》：
- 《修改代码的艺术》：
- 《设计模式：可复用面向对象软件的基础》：
- 《程序员面试金典》：
- 《软技能：代码之外的生存指南》：
- 《点石成金：访客至上的网页设计秘笈》：
- 《编码》：
- 《算法导论》：
- 《人件》：
- 《编程珠玑》：
- 《企业应用架构模式》：
- 《计算机程序的构造和解释》：
- 《计算机程序设计艺术》：
- 《领域驱动设计：软件核心复杂性应对之道》：
- 《编程人生：15 位软件先驱访谈录》：
- 《快速软件开发：有效控制与完成进度计划》：

- 《The Self-Taught Programmer》:
- 《算法》:
- 《持续交付：发布可靠软件的系统方法》:
- 《代码里的世界观：通往架构师之路》：没有很深的代码功力会看的云里雾里，可闲时一观。

Qt 相关书籍和资料网络上以 C++ 居多，对 PyQt 的学习有很大的借鉴意义。

- 《PyQt4 参考文档》
- 《Qt 学习之路》
- 《Qt Creator 快速入门》
- 《Qt 高级编程》
- 《Qt 中的 C++ 技术》
- 《Qt5 开发实践》
- 《Qt5.10 GUI 完全参考手册》
- 《C++ Qt5 范例开发大全》
- 《Qt 教程及软件 (非常适合初学者)》
- 《Qt 样式表》

21.2 网站

- https://github.com/dabeaz-course/practical-python
- https://github.com/PyQt5/PyQt
- http://vfxplatform.com/
- https://paste.ubuntu.com/
- https://www.youtube.com/
- https://www.bilibili.com/
- https://www.vimeo.com/

21.3 一些大佬的博客

- https://www.iteye.com/blog/user/schi

21.4 开源

- https://github.com/SmartPipeline