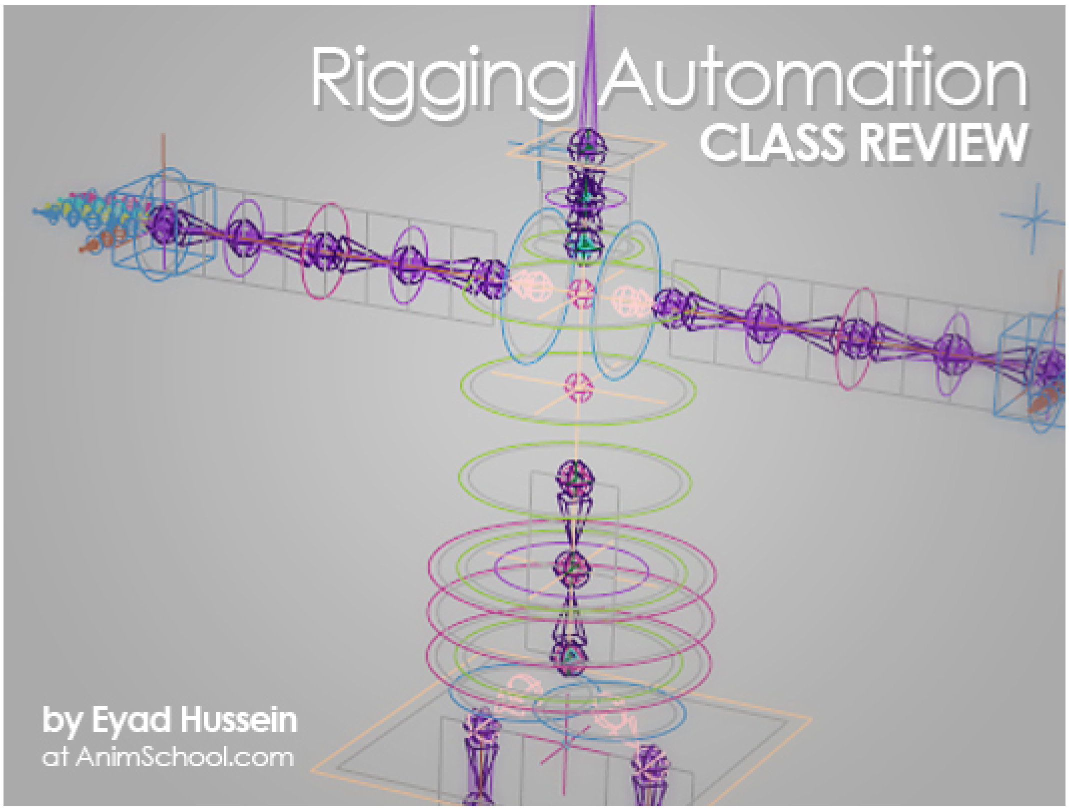
Eyad Hussein

May 24, 2014 - Comments Off on Rigging Automation

Rigging Automation



This article is a review about the **Rigging Automation and Tools Development Class** that I took at **AnimSchool.com** back in Fall 2013, it's my personal experience and I would like to share it with the students who are looking to enroll in this class. So what you will read in this article is not meant to be a tutorial, it's just a review of what you will expect at AnimSchool's Automation Class, and I will include some hints I've learnt within it.

First off, I'm coming from artistic background, so I don't have any programming experience from a university or even from a studio! Five years ago, I've learnt MEL scripting by myself; which was like a nightmare for me, I just used it to run several Maya functions that makes my work process a little bit faster! After that, I've did knew somehow that Python is much easier to learn than MEL, so I start searching for good tutorials around the net and I found that Chad Vernon; the guy who made the cvShapeInverter script!, has made some basic Python tutorials called Python Scripting for Maya Artists, I used them as my starting point in learning Python.

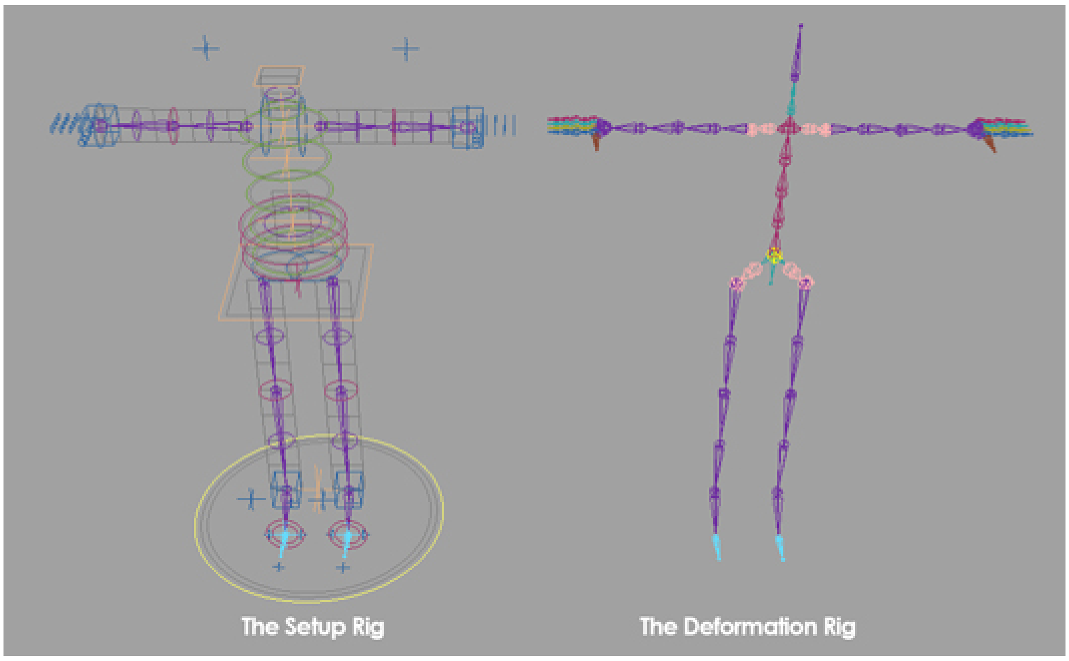
Then I decided to join AnimSchool's Automation class, and I was too worry that this class will be just a waste of time, but the fact that this class completely made a big change in my career as a character technical director!

Who's Walter Yoder?

Well, Walter is the Rigging Automation instructor at AnimSchool.com, he is working as a Character Technical Director at Walt Disney Animation Studios, and before that he was at Digital Domain in Florida, he worked on Frozen (2013), Big Hero 6 (2014), and Iron Man 3 (2013). I'm so lucky that I was able to meet him in person at Disney's open house in November 2013. If I want to describe Walter in few words, he is very smart guy, creative, talented and wonderful person, without him I would never reach to this level in Python programming, preparing rig packages and creating libraries.

Rigging Automation Class

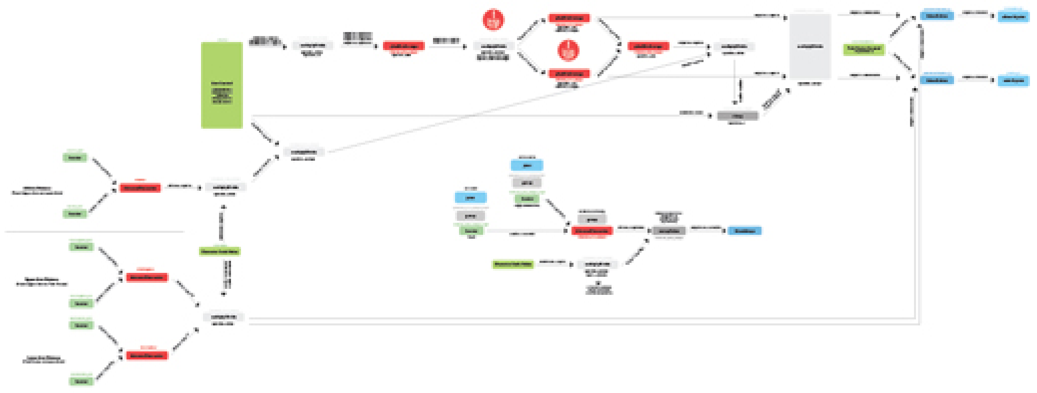
Before I start talking about the class itself, I would like to ask a question, why we should learn the Rigging Automation? In any professional rig, the rig is divided into two sections, the Setup Rig (which is mostly technical) and the Deformation Rig (which is mostly artistic), both are extremely important, but because the setup rig is mostly technical, we don't have to do it manually each time we do a rig, because it has thousands of connections and nodes. We can just run a rig automation that builds this setup rig for us! And then, we can spend all the time working on the deformation rig, which is mostly artistic, and it's finally what people will see when we render the shot, it's how the character skin will behave during the animation.



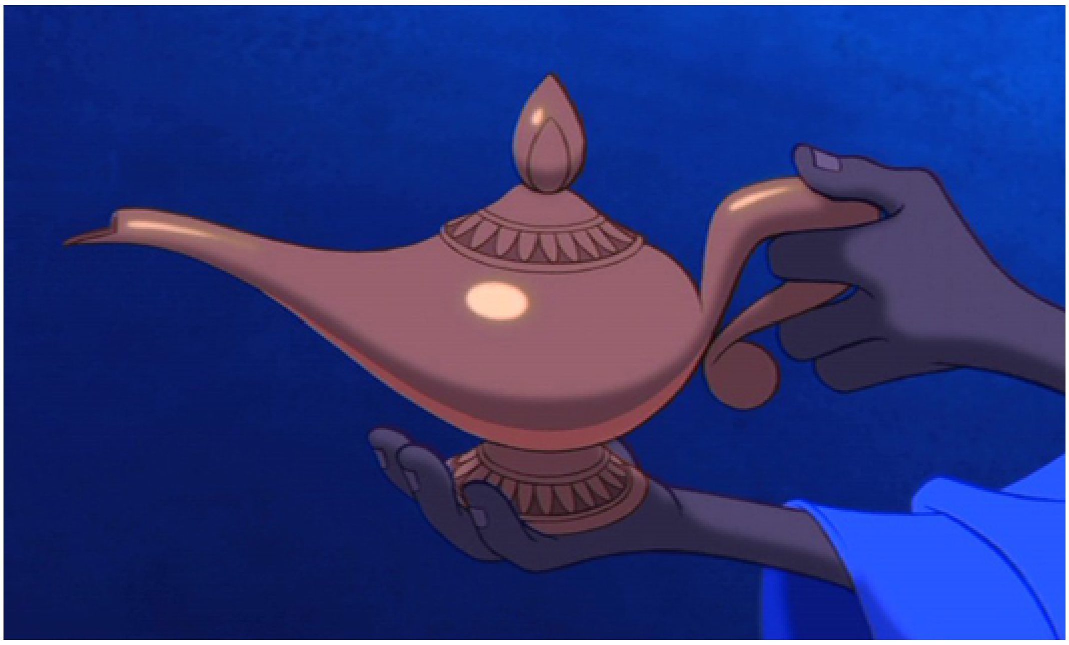
This class will teach you how to make an automation that will make your rigging process much faster and more efficient. The class is about prepare the automation package, write libraries, inherit classes and solve problems in creative way. At the end of the class, you will have a character ik/fk limb automation.

Plan your Code!

Before you start writing any code or building any library, you have to put a "good" plan, yes a good plan! Never ever start without knowing your goal and how you can reach it! What I usually do, I spend like a day or two in preparing the code plan, I start by planing on papers, then I move the plan to the Illustrator, you can use Power Point, or Google Docs, or whatever is good, I just like the Illustrator because its easy, I can do my own layouts. Try to write down everything goes in your mind, know exactly what are your inputs and what are your outputs, know your final result, know how you will modify your code in the future, write down your problems, the things that you need to learn, write everything.

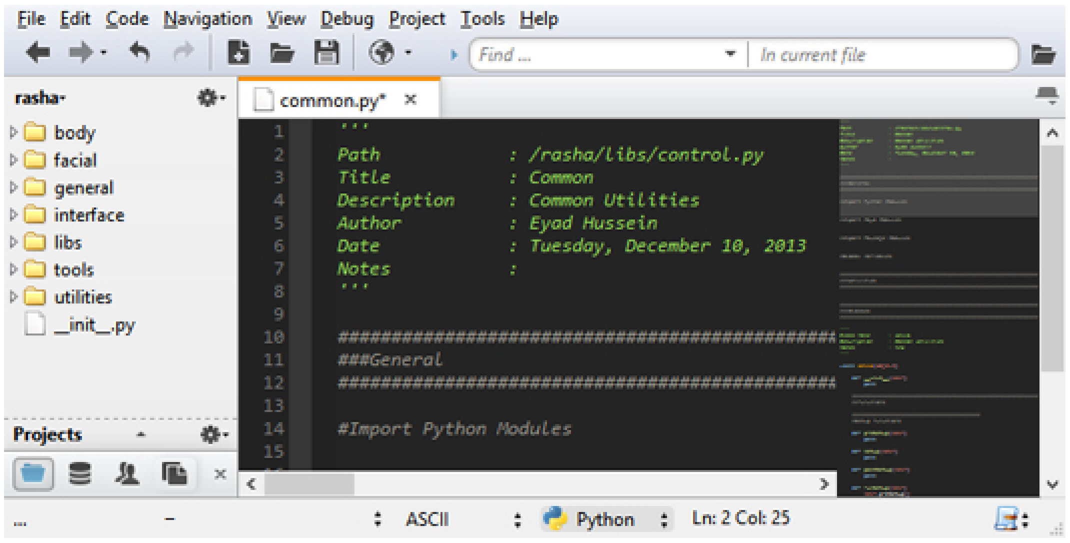


Many of people who start learning programing start thinking about the interface and how to make everything look nice and elegant, but for me the most important part is what inside your library and how it works. So, "Do not be fooled by its commonplace appearance. Like so many things, it is not what is outside, but what is inside that counts."
Aladdin (1992)



IDE?

The first time I ever heard about the IDEs was in Walter's class! Yes, that's very true!! IDE means Integrated Development Environment, in another word; it's the software that you will use to build your automation package. You can use Maya Script Editor to write codes, I used it for about five years and I still use it to write and test the small scripts; it's not a problem at all! But when it comes to preparing a package, the IDE makes your life much easier. You can easily track your code, create new files and folders, do search and replace for variables, it's basically an easy way to edit your codes. Walter recommended us to use Komodo IDE, it's very easy to install, use and learn. You can use Komodo Edit, it's free and work totally fine, you can create your package, modify it and so.

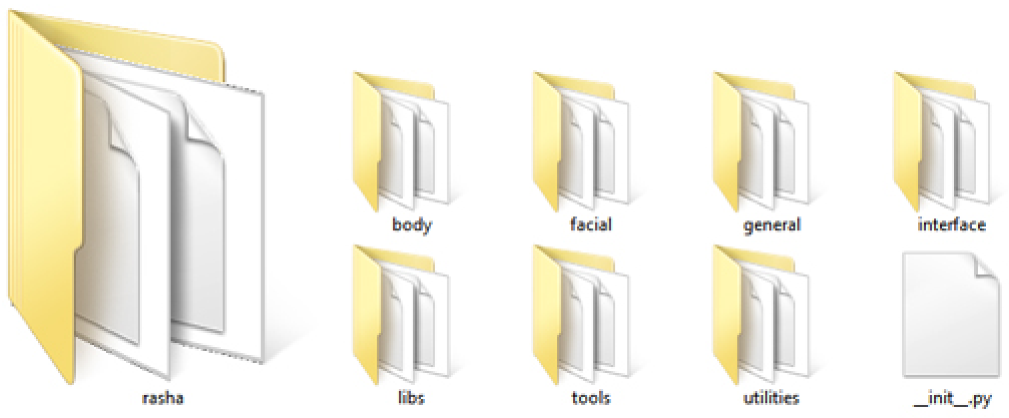


```
'''
Path            : /rasha/libs/control.py
Title           : Common
Description     : Common Utilities
Author          : Eyad Hussein
Date            : Tuesday, December 10, 2013
Notes           :
'''


##############################################################
###General
##############################################################

#Import Python Modules
```

The Package Folders

The idea of the package folders is to organize your libraries, scripts, files, etc… The first thing you have to think about is the main package folder, always try to name it something very unique, so you don't get confused when you start calling the libraries later on, something that you will never see inside your code, you will use this name just to import the Python files. My first mistake was naming this folder something like "bodyRig"! Later on, when I tried to change the name, I changed it manually inside the codes, because when I did the auto search and replace, it changed some variables that have the same name "bodyRig", and it destroyed some parts of my package. So try to name your package something from the solar system and the universe, like jupiter, neptune, sirius, lacaille, cygni, etc!

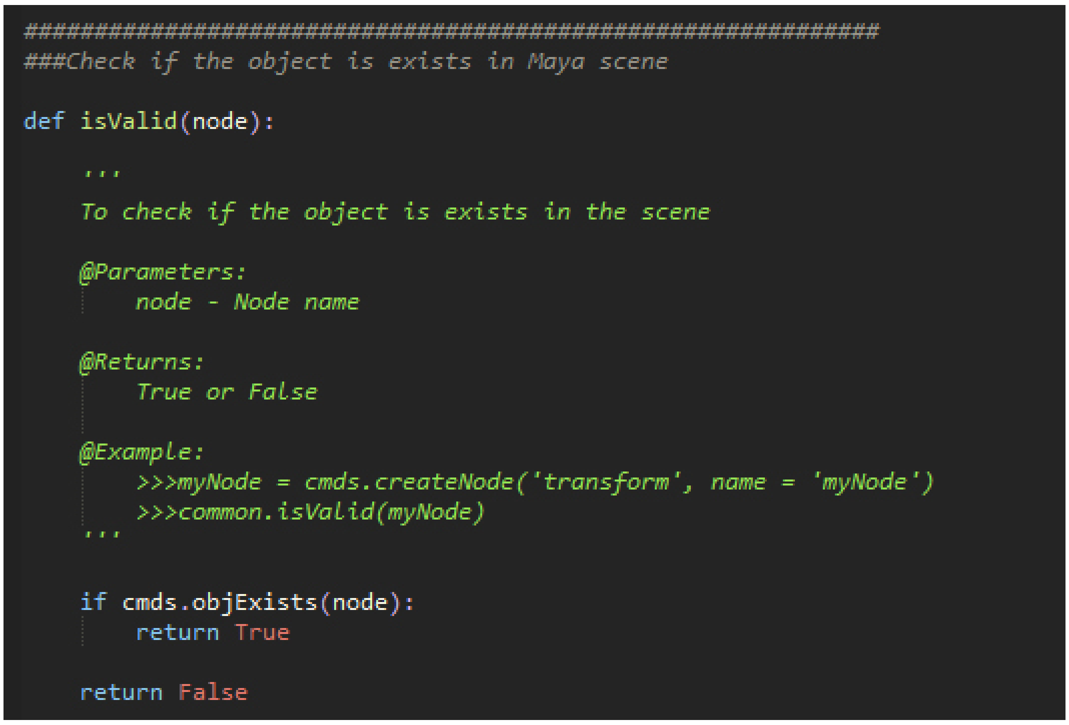
What goes inside this main folder is your preference, I like to have the following folders in my package:
general: Has the common scripts, the names, and the parameters
libs: Has all the automation libraries, from the control lib, to the joint lib, etc
tools: All the small scripts that you will use, like reset values
utilities: Has the pose readers, ribbons, orient, correctives, etc
interface: Has everything related the to the automation interface
body: All the body parts, like arm.py, spine.py
facial: All the facial parts, like eye.py, ear.py



Hint: The **__init__.py** is usually an empty file, is used to mark directories on disk as a Python package directories. If you remove the __init__.py file, Python will no longer look for sub modules inside that directory.

Understand the Python Environment

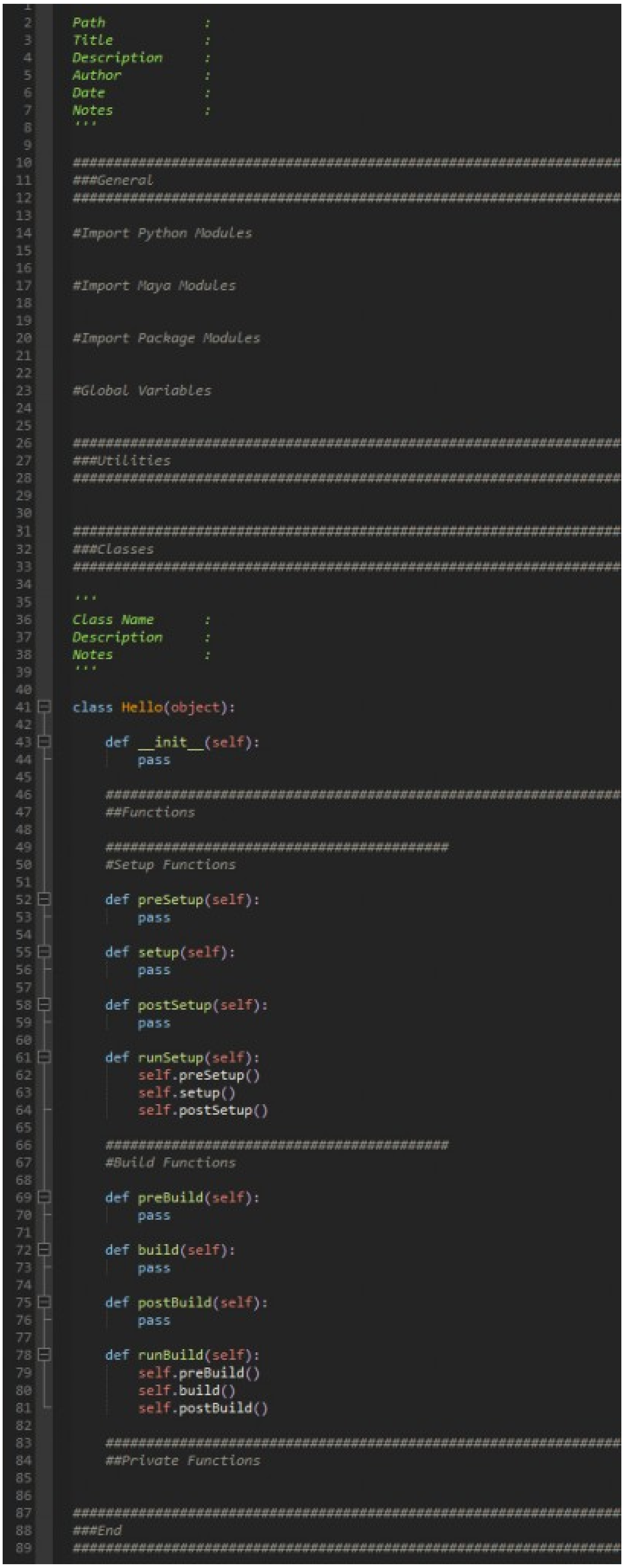
One of the most important things that you need to understand is the Python environment, how to write functions and classes, how to call libraries, how to inherit classes, understanding the general structure of the code will make your life much easier, and your coding experience more enjoyable… Always try to make your codes the smallest as possible, so you can track them. And always write comments, so you can understand what you have wrote when you back to your code, even if the code looks so simple, always write comments! Check the below example, the real code is just 4 lines!



```
##############################################################
###Check if the object is exists in Maya scene

def isValid(node):
    '''
    To check if the object is exists in the scene

    @Parameters:
        node - Node name

    @Returns:
        True or False

    @Example:
        >>>myNode = cmds.createNode('transform', name = 'myNode')
        >>>common.isValid(myNode)
    '''

    if cmds.objExists(node):
        return True

    return False
```

Code Template

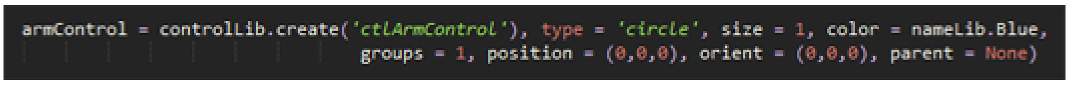
One of the most wonderful things that I've learnt in this class is preparing a template for my codes. In the past, I just used to open Maya and start coding without even thinking to sort things or clean my code. After this class, I developed a template based on what we took in the class, and here is my template below. The templates are very important specially when you start inherit classes, the functions inside the classes should have the same naming so the inheritance work good!



```python
Path            :
Title           :
Description     :
Author          :
Date            :
Notes           :
'''

###################################################################
###General
###################################################################

#Import Python Modules


#Import Maya Modules


#Import Package Modules


#Global Variables


###################################################################
###Utilities
###################################################################


###################################################################
###Classes
###################################################################

'''
Class Name      :
Description     :
Notes           :
'''

class Hello(object):

    def __init__(self):
        pass

    ###############################################################
    ##Functions

    ##########################################
    #Setup Functions

    def preSetup(self):
        pass

    def setup(self):
        pass

    def postSetup(self):
        pass

    def runSetup(self):
        self.preSetup()
        self.setup()
        self.postSetup()

    ##########################################
    #Build Functions

    def preBuild(self):
        pass

    def build(self):
        pass

    def postBuild(self):
        pass

    def runBuild(self):
        self.preBuild()
        self.build()
        self.postBuild()

    ###############################################################
    ##Private Functions


###################################################################
###End
###################################################################
```

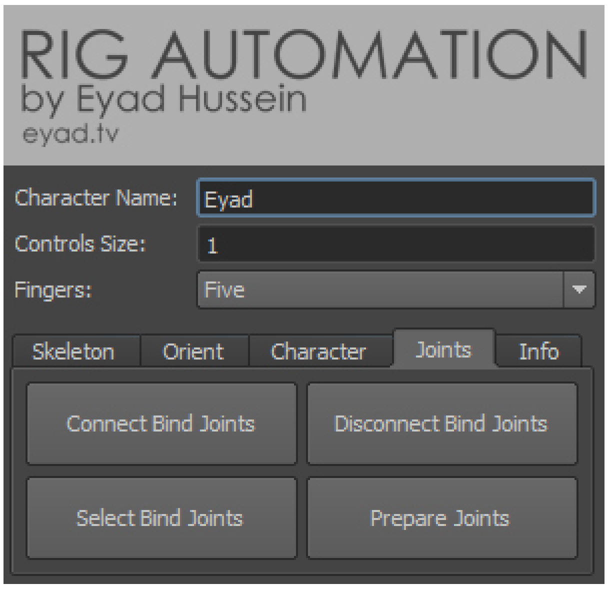
Libraries

What are the libraries? And why we write them? The library is collection of implementations of behavior, when you create a big package, you need to make calls, so you don’t write these lines over and over again inside the package. In another word, it's the re-use of the code. Let's say you have a code that creates arm, and a code that creates leg, and both of them have controls (or stretch behavior, ribbons, or whatever similar)! So you dont have to write the create control commands in both of the codes; instead you can call the Control Library and create the controls! In the library you specify several variables and functions, and you call these functions by giving the library these variables, for instance, if you want to create a control for the arm, check the below example:



```
armControl = controlLib.create('ctlArmControl'), type = 'circle', size = 1, color = nameLib.Blue,
                                    groups = 1, position = (0,0,0), orient = (0,0,0), parent = None)
```

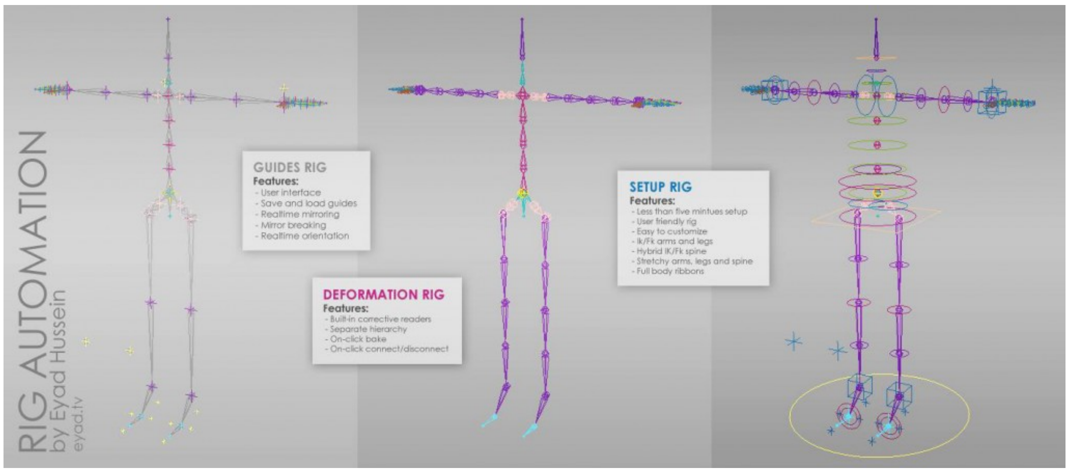
The Body Automation



RIG AUTOMATION
by Eyad Hussein
eyad.tv

Character Name: Eyad
Controls Size: 1
Fingers: Five

After I done from Walter's class, I decided to write a full body automation based on the package that we created in the class. So this is my third body rig automation, the first one was 4 years ago, I remember that all was MEL script, the second one was before I joined AnimSchool and it meant to be a game character automation. But the problem with these two rig automations that both of them built to be like certain of functions with no libraries at all, it was so hard for me to edit each part of the rig, all the code was in one page like 16000 lines of code (without comments)!!



References

The main two references that should be always open while you are writing a script:

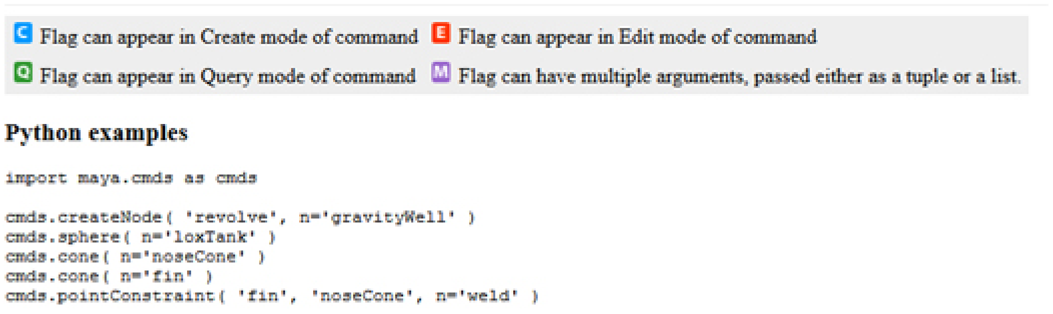
Maya Python Command Reference: (I used to work on Maya 2012, but now I'm Maya 2014 user!)

http://download.autodesk.com/global/docs/maya2014/en_us/CommandsPython/index.html

Python 2.x Documentation: (Maya uses Python version 2.7.3)

http://docs.python.org/2/

Trick: When you open any Maya command in the command reference book, just scroll down and under the flags, you will find an example code, copy it and start from there!



C Flag can appear in Create mode of command E Flag can appear in Edit mode of command

Q Flag can appear in Query mode of command M Flag can have multiple arguments, passed either as a tuple or a list.

Python examples

```
import maya.cmds as cmds

cmds.createNode( 'revolve', n='gravityWell' )
cmds.sphere( n='loxTank' )
cmds.cone( n='noseCone' )
cmds.cone( n='fin' )
cmds.pointConstraint( 'fin', 'noseCone', n='weld' )
```

Special Thanks

I'd like to thank Walter Yoder for his great efforts in this class, my classmate Andrew Chan, and AnimSchool Family.

Links

AnimSchool: http://www.animschool.com/

Walter Yoder: http://www.walteryoder.com/

Chad Vernon: http://www.chadvernon.com/

Komodo IDE: http://www.komodoide.com

Published by: eyadtv in Articles

FACEBOOK TWITTER GOOGLE+

Comments are closed.