Body of “__init__.py”

```
##
##
##AUTO HDRI LIGHT GEN : Ver 1.0
##
##
##Created By : Yong Kim
##Date : Apr 2013
##Contact info : kim_yk@hotmail.com
##Web page : http://magicalpixel.blogspot.com/
##
##
##This script generates light rig automatically based on input HDRI.
##I designed this tool as a part of VFX production lighting pipeline.
##And this tool is more suited for VFX match lighting; method that rely more on HDRI.
##This tool only works for Maya Vray.
##For more information and how-to-use video, go to http://magicalpixel.blogspot.com/
##
##Special thanks to Henry Van Der Beek (Pipeline TD - Method Studios LA) and Brian Burke (Lighting Lead - Method Studios LA)
##
##

"""
this __init__.py is requird by python to use "AutoLight" directory as custom module
"""

from export import *
from light import *
from position import *
from roto import *
from wrapper import *
from preview import *
from render import *
from genPy import *
```

Body of “export.py”

```
##
##
##AUTO HDRI LIGHT GEN : Ver 1.0
##
##
##Created By : Yong Kim
##Date : Apr 2013
##Contact info : kim_yk@hotmail.com
##Web page : http://magicalpixel.blogspot.com/
##
##
##This script generates light rig automatically based on input HDRI.
##I designed this tool as a part of VFX production lighting pipeline.
##And this tool is more suited for VFX match lighting; method that rely more on HDRI.
##This tool only works for Maya Vray.
##For more information and how-to-use video, go to http://magicalpixel.blogspot.com/
##
##Special thanks to Henry Van Der Beek (Pipeline TD - Method Studios LA) and Brian Burke (Lighting Lead - Method Studios LA)
##
##

import os, nuke, re, AutoLight
import math as m

def exportPy():
        """
        This method will trigger "render.py" to render all of Write nodes
        """

        #grap infos from nuke node window
        posNode = nuke.toNode('HDRI_Light_Gen')
        smaWrtNode = nuke.toNode('Write Small HDRI')
        cleWrtNode = nuke.toNode ('Write Clean HDRI')
        jpgWrtNode = nuke.toNode('Write LDR Image')
        iconWrNode = nuke.toNode('Write Icon')
        #error message when user click button w/o setup
        posPath = posNode.knob('path').getValue()
        posInputs = posNode.inputs()
        if not posPath:
 nuke.message('Select Path To Export')
```

```
    elif not posInputs:
    nuke.message('Plug HDRI File')

    else:
    inputNode = posNode.input(0)
    AutoLight.render.renderImg(inputNode, posPath ,smaWrtNode, cleWrtNode, jpgWrtNode, iconWrNode)
    return genScript(posNode, smaWrtNode, cleWrtNode)


def genScript(posNode, smaWrtNode, cleWrtNode):
    """
    This method will trigger "genPy.py" to generate Python MEL
    """

    pknobs = posNode.knobs()
    pkeys = pknobs.keys()

    inputNode = posNode.input(0)
    inPath = posNode.knob('path').getValue()

    inFile = inputNode.knob('file').getValue()
    pathSplit = os.path.split(inFile)
    #strip only the name of input file
    fileName = os.path.splitext(pathSplit[1])
    #use the file name as name of directory
    outPath = inPath + fileName[0]
    #file path for the script
    filepathPy = (outPath +'/'+ fileName[0]+'_PHY.py')
    #use clean hdri file for dome light; use low res hdri as texture for each light
    domLitTex = cleWrtNode.knob('file').value()
    recLitTex = smaWrtNode.knob('file').value()
    #height and width data of input image
    imgHig = inputNode.height()
    imgWid = inputNode.width()
    ltPos = filter ((lambda pkeys: re.search(r'light_pos', pkeys)), pkeys)
    #Call "generatePy" method
    AutoLight.genPy.generatePy(filepathPy, domLitTex, recLitTex, imgHig, imgWid, ltPos, posNode)
```

Body of “genPy.py”

```
##
##
##AUTO HDRI LIGHT GEN : Ver 1.0
##
##
##Created By : Yong Kim
##Date : Apr 2013
##Contact info : kim_yk@hotmail.com
##Web page : http://magicalpixel.blogspot.com/
##
##
##This script generates light rig automatically based on input HDRI.
##I designed this tool as a part of VFX production lighting pipeline.
##And this tool is more suited for VFX match lighting; method that rely more on HDRI.
##This tool only works for Maya Vray.
##For more information and how-to-use video, go to http://magicalpixel.blogspot.com/
##
##Special thanks to Henry Van Der Beek (Pipeline TD - Method Studios LA) and Brian Burke (Lighting Lead - Method Studios LA)
##
##

import os, nuke, re
import math as m

def generatePy(filepathPy, domLitTex, recLitTex, imgHig, imgWid, ltPos, posNode):
    """
    Generate MEL Python for maya lighting
    Only works for Maya Vray
    """

    #nuke uses six spaces as Tab, but maya uses four spaces as tab, so I'm going to force it
    T = "    "
    #this is head and end part of lighting rig script
    #There will be intensity control called "Global_control" at the orgin of Maya viewport
    head = (
    'import maya.cmds as cmds \n'+
    'import maya.mel as mel \n' +
    'import math as m \n\n\n' +
```

```
'def nukeToMaya (distance): \n' +
T +'cmds.spaceLocator (p=[0,0,0]) \n'+
T +'cmds.rename ("locator1", "Global_control") \n' +
T +'cmds.addAttr ("|Global_control", longName = "global_intensity", attributeType = "double", defaultValue=1.0) \n' +
T +'cmds.setAttr ("|Global_control.global_intensity",keyable = True ) \n\n' +
T +'#annotate the name of main controler \n'+
T +'cmds.annotate( "Global_control", tx= "Gobal Controler", p=(7, 6, 5)) \n\n' +
T +'# Vray dome light for hdri \n' +
T +'mel.eval("shadingNode -asLight VRayLightDomeShape") \n'+
T +'cmds.rename ("VRayLightDome1", "%s") \n' %('HDRI_DOME_LIGHT') +
T +'cmds.setAttr ("HDRI_DOME_LIGHTShape.domeSpherical", 1) \n' +
T +'cmds.setAttr ("HDRI_DOME_LIGHTShape.useDomeTex", 1) \n\n' +
T +'#texture file for dome light \n' +
T +'hdri_file = mel.eval("shadingNode -asTexture file") \n' +
T +'cmds.rename (hdri_file, "hdri_texture_file") \n' +
T +'cmds.connectAttr ("hdri_texture_file.outColor", "HDRI_DOME_LIGHTShape.domeTex",f = True) \n' +
T +'cmds.setAttr ("hdri_texture_file.fileTextureName", "%s",type = "string") \n' % (domLitTex) +
T +'mel.eval( \'vrayCreatePlaceEnvTex(%s, 2, 0, 0)\') \n\n' % ('"HDRI_DOME_LIGHTShape.domeTex"'))

end = (
T +'cmds.select ("vray_*", r = True)\n' +
T +'cmds.select ("HDRI_DOME_LIGHT", add = True)\n' +
T +'cmds.select ("Global_control", add = True)\n' +
T +'cmds.select ("annotation1", add = True)\n' +
T +'cmds.group (name = "LIGHT_RIG") \n\n' )

#start writing out file
py_file = open(filepathPy,'w')
py_file.write(head)
#This section is prep section for conversion formula
#Which is modified "Parametric Sphere Formula"
#I want to give a big thank to Henry, Pipeline TD
#W/O his help I wouldn't be able to figure out this part

for i in ltPos:
#width, height, center x and y of each rectangle
width = (posNode.knob(i).value()[2]) - (posNode.knob(i).value()[0])
height = (posNode.knob(i).value()[3]) - (posNode.knob(i).value()[1])
imgCenX = width / 2.0 +(posNode.knob(i).value()[0])
imgCenY = height / 2.0 +(posNode.knob(i).value()[1])
tx = posNode.knob(i).value()[0]
ty = posNode.knob(i).value()[3]
bx = posNode.knob(i).value()[2]
```

```
        by = posNode.knob(i).value()[1]
        #this is for maya texture place2d node
        #I designed this tool to use small hdri as a texture for light, but crop out unwanted part to isolate
        repeatU = (1-(tx/imgWid))- (1-(bx/imgWid))
        repeatV = (ty/imgHig)- (by/imgHig)
        offsetU = 1-(bx/imgWid)
        offsetV = by/imgHig
        #This is part of conversion formula
        U = imgCenX/imgWid
        V = imgCenY/imgHig
        L = m.radians(360.0*(U))
        theta = m.radians(180.0*(V-0.5))
        #
        #x,y,z coordinates for each light
        #I'll place this part inside of light loop
        #This is like an main artery of my tool
        #
        #xcoor = ra*m.cos(theta)*m.cos(L)
        #ycoor = ra*m.sin(theta)
        #zcoor = ra*m.cos(theta)*m.sin(L)
        #

        #loop for each light
        body =(
        T + '#<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<LIGHT
BEGIN>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> \n' +
        T + '#create vray rec light and rename it \n'+
        T +'mel.eval("shadingNode -asLight VRayLightRectShape") \n'+
        T +'cmds.rename ("VRayLightRect1", "%s") \n' % ('vray_' + i) +
        T +'ltShape = cmds.pickWalk ("%s", d = "down") \n' % ('vray_' + i) +
        T +'cmds.expression (s = "%s.intensityMult = 1 * |Global_control.global_intensity", ae = 1, uc = all) \n \n'% ('vray_' + i)
+

        T + '#calculate size of each light \n'+
        T +'u = %f / %f * distance * 3.09 \n' %(width, imgWid) +
        T +'v = %f / %f / 2.0 * distance * 3.45 \n' %(height, imgHig) +
        T +'cmds.setAttr (ltShape[0] + ".uSize",u) \n' +
        T +'cmds.setAttr (ltShape[0] + ".vSize",v) \n \n'+
        T + '#2d to 3d convert equation \n'+
        T +'xcoor = distance * m.cos(%f) * m.cos(%f) \n'%(theta, L)+
        T +'ycoor = distance * m.sin(%f) \n'%(theta)+
        T +'zcoor = distance * m.cos(%f) * m.sin(%f) \n'%(theta, L)+
        T +'cmds.setAttr ("%s.translateX",xcoor) \n' % ('vray_' + i)+
```

```
        T +'cmds.setAttr ("%s.translateY",ycoor) \n' % ('vray_' + i)+
        T +'cmds.setAttr ("%s.translateZ",zcoor) \n \n' % ('vray_' + i)+

        T + '#create place2d and file node, rename both \n'+
        T +'light_tex = mel.eval("shadingNode -asTexture file") \n' +
        T +'light_tex_place2d = mel.eval("shadingNode -asUtility place2dTexture") \n' +
        T +'cmds.rename (light_tex, "light_tex_%s") \n'  % ('vray_' + i) +
        T +'cmds.rename (light_tex_place2d, "lt_tex_place_%s") \n \n' % ('vray_' + i)+

        T +'cmds.connectAttr ("lt_tex_place_%s.coverage", "light_tex_%s.coverage",f = True) \n'  % ('vray_' + i, 'vray_' + i)+
        T +'cmds.connectAttr ("lt_tex_place_%s.translateFrame", "light_tex_%s.translateFrame",f = True) \n' % ('vray_' + i,
'vray_' + i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.rotateFrame", "light_tex_%s.rotateFrame",f = True) \n' % ('vray_' + i, 'vray_' +
i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.mirrorU", "light_tex_%s.mirrorU",f = True) \n' % ('vray_' + i, 'vray_' + i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.mirrorV", "light_tex_%s.mirrorV",f = True) \n' % ('vray_' + i, 'vray_' + i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.stagger", "light_tex_%s.stagger",f = True) \n'% ('vray_' + i, 'vray_' + i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.wrapU", "light_tex_%s.wrapU",f = True) \n' % ('vray_' + i, 'vray_' + i)+
        T +'cmds.connectAttr ("lt_tex_place_%s.wrapV", "light_tex_%s.wrapV",f = True) \n'% ('vray_' + i, 'vray_' + i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.repeatUV", "light_tex_%s.repeatUV",f = True) \n' % ('vray_' + i, 'vray_' + i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.offset", "light_tex_%s.offset",f = True) \n' % ('vray_' + i, 'vray_' + i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.rotateUV", "light_tex_%s.rotateUV",f = True) \n' % ('vray_' + i, 'vray_' + i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.noiseUV", "light_tex_%s.noiseUV",f = True) \n' % ('vray_' + i, 'vray_' + i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.vertexUvOne", "light_tex_%s.vertexUvOne",f = True) \n' % ('vray_' + i, 'vray_'
+ i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.vertexUvTwo", "light_tex_%s.vertexUvTwo",f = True) \n' % ('vray_' + i, 'vray_'
+ i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.vertexUvThree", "light_tex_%s.vertexUvThree",f = True) \n' % ('vray_' + i,
'vray_' + i) +
        T +'cmds.connectAttr ("lt_tex_place_%s.vertexCameraOne", "light_tex_%s.vertexCameraOne",f = True) \n' % ('vray_'
+ i, 'vray_' + i) +
        T +'cmds.connectAttr ( "lt_tex_place_%s.outUV", "light_tex_%s.uv") \n' % ('vray_' + i, 'vray_' + i) +
        T +'cmds.connectAttr ( "lt_tex_place_%s.outUvFilterSize", "light_tex_%s.uvFilterSize") \n\n' % ('vray_' + i, 'vray_' + i)
+
        T + '#place 2d node tweak to use small hdri as texture \n' +
        T +'cmds.setAttr ("lt_tex_place_%s.repeatU", %f) \n' % ('vray_' + i, repeatU) +
        T +'cmds.setAttr ("lt_tex_place_%s.repeatV", %f) \n' % ('vray_' + i, repeatV) +
        T +'cmds.setAttr ("lt_tex_place_%s.offsetU", %f) \n' % ('vray_' + i, offsetU) +
        T +'cmds.setAttr ("lt_tex_place_%s.offsetV", %f) \n\n' % ('vray_' + i, offsetV) +
        T + '#connect texture into vray rec light \n' +
        T +'cmds.setAttr (ltShape[0] + ".useRectTex", True) \n' +
        T +'cmds.connectAttr ("light_tex_%s.outColor", ltShape[0] + ".rectTex",f = True) \n' % ('vray_' + i) +
        T +'cmds.setAttr ("light_tex_%s.fileTextureName", "%s",type = "string") \n \n' % ('vray_' + i, recLitTex) +
```

```
        T + '#aim constranint light to Global control \n' +
        T +'cmds.select ("Global_control",r=1) \n'+
        T +'cmds.select ("%s",tgl=1) \n' % ('vray_' + i)+
        T +'cmds.aimConstraint(aimVector=[0,0,-1],upVector=[0,1,0],worldUpType="vector",worldUpVector=[0,1,0]) \n'+
        T + '#<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<LIGHT
END>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> \n\n\n')

        py_file.write(body)
        py_file.write(end)
        py_file.close()
```

Body of “light.py”

```
##
##
##AUTO HDRI LIGHT GEN : Ver 1.0
##
##
##Created By : Yong Kim
##Date : Apr 2013
##Contact info : kim_yk@hotmail.com
##Web page : http://magicalpixel.blogspot.com/
##
##
##This script generates light rig automatically based on input HDRI.
##I designed this tool as a part of VFX production lighting pipeline.
##And this tool is more suited for VFX match lighting; method that rely more on HDRI.
##This tool only works for Maya Vray.
##For more information and how-to-use video, go to http://magicalpixel.blogspot.com/
##
##Special thanks to Henry Van Der Beek (Pipeline TD - Method Studios LA) and Brian Burke (Lighting Lead - Method Studios LA)
##
##

import nuke

#default number of light is 1
numOfLi = [1]
```

```
def addLight ():
	"""
	This method adds rectangle box (BBOX Knob) to group node
	And each box represents one Vray rectagle light
	"""
	posNode = nuke.toNode('HDRI_Light_Gen')
	pInput = posNode.input(0)

	if not pInput:
 nuke.message('Connect Input Image')
	else:
	defalBoxU = pInput.height()*0.2
	defalBoxV = pInput.width()*0.2

	length = len(numOfLi)

	if length == 1:
	numOfLi.append(2)
	bbox = nuke.BBox_Knob('light_pos_1', 'Light Position 1')
	posNode.addKnob(bbox)

	posNode.knob('light_pos_1').setT(defalBoxU)
	posNode.knob('light_pos_1').setR(defalBoxV)

	else:
	numOfLi.append(1+length)
	bbox = nuke.BBox_Knob('light_pos_%s' %str(length), 'Light Position %s' %str(length))
	posNode.addKnob(bbox)

	posNode.knob('light_pos_%s' %str(length)).setT(defalBoxU)
	posNode.knob('light_pos_%s' %str(length)).setR(defalBoxV)
```

Body of “position.py”

```
##
##
##AUTO HDRI LIGHT GEN : Ver 1.0
##
##
##Created By : Yong Kim
##Date : Apr 2013
```

```
##Contact info : kim_yk@hotmail.com
##Web page : http://magicalpixel.blogspot.com/
##
##
##This script generates light rig automatically based on input HDRI.
##I designed this tool as a part of VFX production lighting pipeline.
##And this tool is more suited for VFX match lighting; method that rely more on HDRI.
##This tool only works for Maya Vray.
##For more information and how-to-use video, go to http://magicalpixel.blogspot.com/
##
##Special thanks to Henry Van Der Beek (Pipeline TD - Method Studios LA) and Brian Burke (Lighting Lead - Method Studios LA)
##
##

import nuke, AutoLight, nukescripts

def createNodes():
	"""
	This onw creates comp trick script to remove unwanted light contribution area from HDRI
	"""
	rmaNode = nuke.nodes.Remove(channels = 'alpha')
	#hdri w/o light contribution only for env reflection purpose
	rpNode = nuke.nodes.RotoPaint(name='Light Mask', inputs = [rmaNode])
	rpNode.knob('output').setValue('rgba.alpha')
	#
	#rather than marker removoal to remove light soure, I'll use traditional compositing technique.
	#This method produce much faster and better than marker removal
	#clamp hdri to value in 0 to 1 and blur the image to clean up the patch
	#
	#mkNode = nuke.nodes.MarkerRemoval(name='Remove Light From HDRI', inputs = [rpNode],linearsearch = 250)
	clampAll = nuke.nodes.Clamp(inputs = [rpNode], name = 'Clamp To 1')
	gradeRGB = nuke.nodes.Grade(inputs = [clampAll], name = 'Multiply 0',  multiply = 0, maskChannelInput = 'rgba.alpha')
	blurRGB = nuke.nodes.Blur( inputs = [gradeRGB], name = 'Blur RGB', channels = 'rgb', size = 'width/10')
	blurAlpha = nuke.nodes.Blur( inputs = [blurRGB], name = 'Blur Alpha', channels = 'alpha', size = 'width/100')
	gradeAlpha = nuke.nodes.Grade(inputs = [blurAlpha], name = 'Whitepoint Alpha', channels = 'alpha', whitepoint = 0.62, white = 2.1)
	premultClean =  nuke.nodes.Premult(inputs = [gradeAlpha], name = 'Premult Clean')
	mergeOver =  nuke.nodes.Merge(inputs = [ rmaNode, premultClean], name = 'Clean HDRI')
	wriNode = nuke.nodes.Write(inputs = [mergeOver], name = 'Write Clean HDRI')

	#smaller size of original hdri for light texture
```

```
        mirrNode = nuke.nodes.Mirror(inputs = [rmaNode], name = 'Horizontal Flip', Horizontal = True)
        refNode = nuke.nodes.Reformat (name = 'Reformat HDRI', type = 'scale', scale = 0.3, inputs = [mirrNode])
        smWrNode = nuke.nodes.Write(inputs = [refNode], name = 'Write Small HDRI')

        #small jpg preview image for pyqt window; size 400 x 200
        refSmall = nuke.nodes.Reformat (name = 'Preview Size', type = 'to box', resize = 'distort', box_width = '400',
box_height = '200', inputs = [rmaNode])
        jpgWrNode = nuke.nodes.Write(inputs = [refSmall], name = 'Write LDR Image')

        #small mirror ball icon for QListWidget
        sphTran = nuke.nodes.SphericalTransform(inputs = [rmaNode], input = 'Lat Long map', output = 'Mirror Ball', format =
'square_256')
        refIcon = nuke.nodes.Reformat (inputs = [sphTran], name = 'Icon Size', type = 'to box', resize = 'distort', box_width =
'60', box_height = '60')
        iconWrNode = nuke.nodes.Write(inputs = [refIcon], name = 'Write Icon')

        #dummy switch nodes; to put everything into group node, it reqires one output
        switchOne = nuke.nodes.Switch(name = 'Dummy Switch 01', inputs = [wriNode, smWrNode])
        switchTwo = nuke.nodes.Switch(name = 'Dummy Switch 02', inputs = [switchOne, jpgWrNode])
        switchThree = nuke.nodes.Switch(name = 'Dummy Switch 03', inputs = [switchTwo, iconWrNode])

        #select connected nodes
        nukescripts.clear_selection_recursive()
        rmaNode.knob('selected').setValue(True)
        nuke.selectConnectedNodes()

        return groupNode()


def groupNode():
        """
        group node for light position and export button
        """

        grpNode = nuke.collapseToGroup()
        grpNode.setName('HDRI Light Gen')
        #add knobs and buttons to group node
        pyButton = nuke.PyScript_Knob('export', 'Export ".ibl" File')
        iblPreButton = nuke.PyScript_Knob('ibl', 'Open IBL Preview')
        addButton = nuke.PyScript_Knob('add', 'Add Light')
        file = nuke.File_Knob('path','Export File To...')
        for k in (addButton, pyButton, iblPreButton, file):
        grpNode.addKnob(k)
```

```
    grpNode.knob('export').setValue('AutoLight.roto.rotoShape()')
    grpNode.knob('add').setValue('AutoLight.light.addLight()')
    grpNode.knob('ibl').setValue ('AutoLight.preview.previewButton()')
```

Body of “preview.py”

```
##
##
##AUTO HDRI LIGHT GEN : Ver 1.0
##
##
##Created By : Yong Kim
##Date : Apr 2013
##Contact info : kim_yk@hotmail.com
##Web page : http://magicalpixel.blogspot.com/
##
##
##This script generates light rig automatically based on input HDRI.
##I designed this tool as a part of VFX production lighting pipeline.
##And this tool is more suited for VFX match lighting; method that rely more on HDRI.
##This tool only works for Maya Vray.
##For more information and how-to-use video, go to http://magicalpixel.blogspot.com/
##
##Special thanks to Henry Van Der Beek (Pipeline TD - Method Studios LA) and Brian Burke (Lighting Lead - Method Studios LA)
##
##

import nuke, AutoLight, os
def previewButton():
    """
    open "IBL_preview.py" file for GUI window
    This method will look for nuke plugin path
    So there is no need to type path information to access the file
    """
    envlist = nuke.pluginPath()
    for e in range(len(envlist)):
    if envlist[e][-5:] == ".nuke":
    cmd = (envlist[e] + "/IBL_List_Window/IBL_preview.py")
    #I'm using os.system insted of subprocess.call, bacasue it's much easier to access the pyqt
```

```
    os.system(cmd)
```

Body of “render.py”

```
##
##
##AUTO HDRI LIGHT GEN : Ver 1.0
##
##
##Created By : Yong Kim
##Date : Apr 2013
##Contact info : kim_yk@hotmail.com
##Web page : http://magicalpixel.blogspot.com/
##
##
##This script generates light rig automatically based on input HDRI.
##I designed this tool as a part of VFX production lighting pipeline.
##And this tool is more suited for VFX match lighting; method that rely more on HDRI.
##This tool only works for Maya Vray.
##For more information and how-to-use video, go to http://magicalpixel.blogspot.com/
##
##Special thanks to Henry Van Der Beek (Pipeline TD - Method Studios LA) and Brian Burke (Lighting Lead - Method Studios LA)
##
##

import nuke, os, shutil

def renderImg (inputNode, posPath, smaWrtNode, cleWrtNode, jpgWrtNode, iconWrNode ):
    """
    Auto naming method
    This one names the write nodes to correct name
    And write out at the end
    """

    inputImg = inputNode.knob('file').value()
    pathSplit = os.path.split(inputImg)
    #stript only name of input file
    fileName = os.path.splitext(pathSplit[1])
    #use inPath + fileName
```

```
    outPath = posPath + fileName[0]
    #first check the path
    if os.path.exists(outPath) == True:
    shutil.rmtree(outPath)
    os.mkdir(outPath)
    else :
    os.mkdir(outPath)

    #clean hdri file auto name; exr file
    clnOutput = (outPath +'/' + fileName[0] + "_CLN.exr")
    cleWrtNode.knob('file').setValue(clnOutput)
    cleWrtNode.knob('file_type').setValue('exr')
    cleWrtNode.knob('datatype').setValue('32 bit float')

    #small hdri file auto name; exr file
    smlOutput =(outPath +'/' + fileName[0] +  "_SML.exr")
    smaWrtNode.knob('file').setValue(smlOutput)
    smaWrtNode.knob('file_type').setValue('exr')
    smaWrtNode.knob('datatype').setValue('32 bit float')
    #LDR file auto name ; jpg file
    jpgOutput = (outPath +'/' + fileName[0] + "_PRE.jpg")
    jpgWrtNode.knob('file').setValue(jpgOutput)
    jpgWrtNode.knob('file_type').setValue('jpg')
    jpgWrtNode.knob('_jpeg_quality').setValue(1)
    #Icon file auto name;  mirror ball icon; jpg file
    iconOutput = (outPath +'/' + fileName[0] + "_ICO.jpg")
    iconWrNode.knob('file').setValue(iconOutput)
    iconWrNode.knob('file_type').setValue('jpg')
    iconWrNode.knob('_jpeg_quality').setValue(1)

    for w in (smaWrtNode, cleWrtNode, jpgWrtNode, iconWrNode):
    nuke.execute(w, 1, 1, 1)

    return generateIbl(outPath, jpgOutput, iconOutput)


def generateIbl(outPath, jpgOutput,iconOutput):
    """
    Create .ibl file that contains several path infomations
    This custom log file is to communicate with pyqt gui later
    """
    if os.path.isfile(outPath + '.ibl'):
    os.remove(outPath + '.ibl')
```

```
    return
    else:

    iblFile = open(outPath + '.ibl', 'w')
    whatToWrite =(
    'hdri_preview=%s\n' %(jpgOutput)+
    'hdri_path=%s\n' % (outPath+'/')+
    'icon=%s\n' % (iconOutput)+
    'maya_preview=')
    iblFile.write(whatToWrite)
    iblFile.close()
```

Body of “roto.py”

```
##
##
##AUTO HDRI LIGHT GEN : Ver 1.0
##
##
##Created By : Yong Kim
##Date : Apr 2013
##Contact info : kim_yk@hotmail.com
##Web page : http://magicalpixel.blogspot.com/
##
##
##This script generates light rig automatically based on input HDRI.
##I designed this tool as a part of VFX production lighting pipeline.
##And this tool is more suited for VFX match lighting; method that rely more on HDRI.
##This tool only works for Maya Vray.
##For more information and how-to-use video, go to http://magicalpixel.blogspot.com/
##
##Special thanks to Henry Van Der Beek (Pipeline TD - Method Studios LA) and Brian Burke (Lighting Lead - Method Studios LA)
##
##

import re
import nuke.rotopaint as rp
import nuke, AutoLight
```

```
def rotoShape ():
    """
    Get position data from "light_pos" knob and make rectangel box
    This box will be use as mask to remove illumination area from hdri
    """
    rpNode = nuke.toNode('Light Mask')
    posNode = nuke.toNode('HDRI_Light_Gen')
    pknobs = posNode.knobs()
    pkeys = pknobs.keys()

    ltPos = filter ((lambda pkeys: re.search(r'light_pos', pkeys)), pkeys)

    for i in ltPos:
    cKnob = rpNode['curves']
    shape = rp.Shape(cKnob)
    finalPos01 = [posNode.knob(i).value()[0],posNode.knob(i).value()[1]]
    finalPos02 = [posNode.knob(i).value()[0],posNode.knob(i).value()[3]]
    finalPos03 = [posNode.knob(i).value()[2],posNode.knob(i).value()[3]]
    finalPos04 = [posNode.knob(i).value()[2],posNode.knob(i).value()[1]]
    shape.append(finalPos01)
    shape.append(finalPos02)
    shape.append(finalPos03)
    shape.append(finalPos04)
    cKnob.rootLayer.append(shape)

    return AutoLight.export.exportPy()
```

Body of “wrapper.py”

```
##
##
##AUTO HDRI LIGHT GEN : Ver 1.0
##
##
##Created By : Yong Kim
##Date : Apr 2013
##Contact info : kim_yk@hotmail.com
##Web page : http://magicalpixel.blogspot.com/
##
##
```

```
##This script generates light rig automatically based on input HDRI.
##I designed this tool as a part of VFX production lighting pipeline.
##And this tool is more suited for VFX match lighting; method that rely more on HDRI.
##This tool only works for Maya Vray.
##For more information and how-to-use video, go to http://magicalpixel.blogspot.com/
##
##Special thanks to Henry Van Der Beek (Pipeline TD - Method Studios LA) and Brian Burke (Lighting Lead - Method Studios LA)
##
##

import  os, nuke, AutoLight

#default number of light is 1
numOfLi = [1]

def samplerWrapper():
	"""
	This method contains codes for nuke call back system
	gather position and light datas and feed into necessary spot
	Get x,y,r,t datas from bbox_knob and calculate size of light
	"""
	tnode = nuke.thisNode()
	tknob = nuke.thisKnob()
	if tnode.Class() == 'Group':
	if tnode.inputs() > 0:
	for e in range(0, len(numOfLi)):
		if tknob.name() == 'light_pos_%s' % str(e+1):
		#r-x = width, t-y = height, (r-x)/2+x = centerX, (t-y)/2+y = centerY
		width = (tknob.value()[2]) - (tknob.value()[0])
		height = (tknob.value()[3]) - (tknob.value()[1])
		xcor = width / 2 +(tknob.value()[0])
		ycor = height / 2 +(tknob.value()[1])

nuke.addKnobChanged(samplerWrapper, nodeClass = '*')
```