미리 작성 해 놓았던 Khn\_Ui를 pyside2를 이용하여 재수정함. 그 방법을 정리한 글.

수정하는 이유는 다른 팀원들과 같은 방식으로 스크립트를 짜면서 수정과 이동, 합침이 쉽게끔 하기 위함 + 익숙해지면 이 방법이 배는 더 빠르기 때문...

pyside를 공부한 흔적이라고 보는게...

import maya.mel as mel

import maya.OpenMayaUI as omui

import getpass

import maya.cmds as cmds

from PySide2.QtUiTools import \*

from PySide2 import QtCore

from PySide2 import QtWidgets

from PySide2 import QtUiTools

from PySide2 import QtGui

from shiboken2 import wrapInstance

일단은 모듈들을 불러온다. 거의 대부분은 maya와 pyside를 연결시키고 외부 파일을 가져오는것에 사용된다.

User= getpass.getuser()

version = cmds.about(version=True)

Path= "C:/Users/{}/Documents/maya/{}/scripts/GimHa".format(User,version)

이 Ui에 띄울 이미지나, 기타 링크들을 집어넣을 폴더의 경로를 설정해준다. 가져온다는 말이 더 맞을 듯. User와 version을 따로 빼준 이유는 컴퓨터를 바꿀 때마다 달라지는 환경변수이기 때문이다. 그때마다 변경하기 귀찮음.

def Gim\_window():

    Gim\_window\_ptr = omui.MQtUtil.mainWindow()

    return wrapInstance(int(Gim\_window\_ptr), QtWidgets.QWidget)

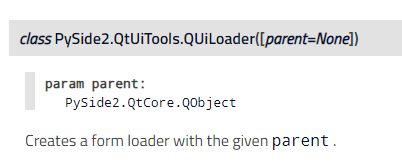
ui를 집어 넣기 전, 윈도우를 먼저 띄운다. 이름은 Gim\_Window 나중에 수정이 쉽도록 return하여 변수로 만들어 놓는다. 푸하하하하.

class DesignerUI(QtWidgets.QDialog):

    def \_\_init\_\_(self, parent=Gim\_window()):

        super(DesignerUI, self).\_\_init\_\_(parent)

UI 클래스를 생성해준다. 클래스의 컷 메서드는 \_\_init\_\_여야 한다. (문법임)

 <- 요것을 참고...

        self.setWindowTitle("KHN\_UI\_Test")

        self.resize(477,280)

        self.init\_ui()

        self.create\_connection()

윈도우 창에 대한 메서드들을 정의한다. 창 위에 띄울 이름은 KHN\_UI\_Test (솔직히 연습 아니고 내 사활을 건 ui지만 나중에 보면 연습일거라 그렇게 썼음) 처음 작동했을 때 크기를 477,280로 고정한다. 그리고 밑은 곧 나올 메서드들의 실행창.

    def init\_ui(self):

        f = QtCore.QFile(Path + "/pyside2\_DesignerTest\_ver.04.ui")

        #-pyside파일 경로설정

        f.open(QtCore.QFile.ReadOnly)

        loader = QtUiTools.QUiLoader()

        self.ui = loader.load(f, parentWidget=self)

        f.close()

        #-열었다가 닫는 이유는 무한루프를 끊기 위함.

실행되는 self.init\_ui(). 변수 화 시켰던 Path를 파일 앞에 붙여 경로를 설정한다. Pyside 파일을 불러오는 메서드다. (메서드는 클래스 내에 들어있는 함수다.) read전용으로 가져온다. 그리고 오픈했다가 다시 닫는 이유는 무한 루프를 멈추기 위함. ([참고 링크](https://onlytojay.medium.com/pyside2-1-%EA%B8%B0%EB%B3%B8%EB%8F%99%EC%9E%91%EC%9B%90%EB%A6%AC-72ea6572a65b))

    def create\_connection(self):

        self.ui.paintSkinWeightTool\_bt.clicked.connect(self.ArtPaintSkinWeightsTool)

        self.ui.deleteHistoryAndFreeze\_bt.clicked.connect(self.DeleteHisAndFreeze)

        self.ui.handJointClean\_bt.clicked.connect(self.LfHand\_Joint\_Cleaner)

        self.ui.oneclick\_L\_J\_bt.clicked.connect(self.oneClickMatchJ\_L)

        self.ui.oneclick\_C\_J\_bt.clicked.connect(self.oneClickMatchC\_L)

        self.ui.oneclick\_D\_G\_bt.clicked.connect(self.oneClickblend)

        self.ui.create\_Enter\_bt.clicked.connect(self.executeCreation)

        #-라디오 버튼 (https://pbj0812.tistory.com/339)

        self.ui.findingNodeEnter\_bt.clicked.connect(lambda: self.FindSet(self))

        #-텍스트 필드는 함수로 연결해줘야 함. 인수는 self

        #-버튼과 함수들을 연결

위에 실행되는 self.create\_connection(). f 변수 선언에서 보았던 ui파일 속 버튼들과 내 함수들을 연결시킨다. 인수는 self다. 그리고 가끔 그냥은 연결되지 않는 놈들이 있는데 그런 애들은 lamda함수로 연결한다. 람다의 매개변수 없이 표현식만 쓴다. 어떻게 가능한건지는 아직 모름. 걍 아는게 없음.

    #-함수시작

    def LfHand\_Joint\_Cleaner(\*args):

        Index = cmds.listRelatives('LfIndexRoot\_Jnt', ad=True, type='joint')

        Middle = cmds.listRelatives('LfMiddleRoot\_Jnt', ad=True, type='joint')

        Ring = cmds.listRelatives('LfRingRoot\_Jnt', ad=True, type='joint')

        Thumb = cmds.listRelatives('LfThumbRoot\_Jnt', ad=True, type='joint')

        Pinky = cmds.listRelatives('LfPinkyRoot\_Jnt', ad=True, type='joint')

        if Index:

            for IntheIndex in Index:

                cmds.setAttr(IntheIndex + ".jointOrientX", 0)

                cmds.setAttr(IntheIndex + ".jointOrientY", 0)

                cmds.setAttr(IntheIndex + ".jointOrientZ", 0)

        if Middle:

            for IntheMiddle in Middle:

                cmds.setAttr(IntheMiddle + ".jointOrientX", 0)

                cmds.setAttr(IntheMiddle + ".jointOrientY", 0)

                cmds.setAttr(IntheMiddle + ".jointOrientZ", 0)

        if Ring:

            for IntheRing in Ring:

                cmds.setAttr(IntheRing + ".jointOrientX", 0)

                cmds.setAttr(IntheRing + ".jointOrientY", 0)

                cmds.setAttr(IntheRing + ".jointOrientZ", 0)

        if Thumb:

            for IntheThumb in Thumb:

                cmds.setAttr(IntheThumb + ".jointOrientX", 0)

                cmds.setAttr(IntheThumb + ".jointOrientY", 0)

                cmds.setAttr(IntheThumb + ".jointOrientZ", 0)

        if Pinky:

            for InthePinky in Pinky:

                cmds.setAttr(InthePinky + ".jointOrientX", 0)

                cmds.setAttr(InthePinky + ".jointOrientY", 0)

                cmds.setAttr(InthePinky + ".jointOrientZ", 0)

        for IntheIndex in Index:

            cmds.setAttr(IntheIndex + ".rotateX", 0)

            cmds.setAttr(IntheIndex + ".rotateY", 0)

            cmds.setAttr(IntheIndex + ".rotateZ", 0)

        for IntheMiddle in Middle:

            cmds.setAttr(IntheMiddle + ".rotateX", 0)

            cmds.setAttr(IntheMiddle + ".rotateY", 0)

            cmds.setAttr(IntheMiddle + ".rotateZ", 0)

        for IntheRing in Ring:

            cmds.setAttr(IntheRing + ".rotateX", 0)

            cmds.setAttr(IntheRing + ".rotateY", 0)

            cmds.setAttr(IntheRing + ".rotateZ", 0)

        for IntheThumb in Thumb:

            cmds.setAttr(IntheThumb + ".rotateX", 0)

            cmds.setAttr(IntheThumb + ".rotateY", 0)

            cmds.setAttr(IntheThumb + ".rotateZ", 0)

        for InthePinky in Pinky:

            cmds.setAttr(InthePinky + ".rotateX", 0)

            cmds.setAttr(InthePinky + ".rotateY", 0)

            cmds.setAttr(InthePinky + ".rotateZ", 0)

    def ArtPaintSkinWeightsTool(\*args):

        cmds.ArtPaintSkinWeightsTool()

    def DeleteHisAndFreeze(\*args):

        cmds.select(cl=True)

        selA = cmds.ls('\*\_Geo', '\*\_geo', type='transform')

        for Geo in selA:

            cmds.delete(Geo, constructionHistory=True)

            cmds.makeIdentity(Geo, apply=True, t=1, r=1, s=1)

    def oneClickMatchJ\_L(\*args):

        selected = cmds.ls(selection=True)

        locList = []

        jntList = []

        for obj in selected:

            if '\_Jnt' in obj:

                jntList.append(obj)

            if '\_Loc' in obj:

                locList.append(obj)

        for loc in locList:

            loc\_n\_name = loc.split('\_')[-2]

            for jnt in jntList:

                jnt\_n\_name = jnt.split('\_')[-2]

                if jnt\_n\_name.isdigit() and loc\_n\_name.isdigit() and jnt\_n\_name == loc\_n\_name:

                    cmds.matchTransform(loc, jnt)

                    break

    def oneClickMatchC\_L(\*args):

        selected = cmds.ls(selection=True)

        ctrlList = []

        jntList = []

        for obj in selected:

            if '\_Jnt' in obj:

                jntList.append(obj)

            if '\_Ctrl' in obj:

                ctrlList.append(obj)

        for ctrl in ctrlList:

            ctrl\_n\_name = ctrl.split('\_')[-2]

            for jnt in jntList:

                jnt\_n\_name = jnt.split('\_')[-2]

                if jnt\_n\_name.isdigit() and ctrl\_n\_name.isdigit() and jnt\_n\_name == ctrl\_n\_name:

                    cmds.matchTransform(ctrl, jnt)

                    break

    def oneClickblend(\*args):

        selected = cmds.ls(selection=True)

        before = []

        after = []

        for obj in selected:

            if '\_Geo' in obj:

                before.append(obj)

            if '\_Dummy' in obj:

                after.append(obj)

        for Geo in before:

            Geo\_n\_name = Geo.split('\_')[-2]

            for Dum in after:

                Dum\_n\_name = Dum.split('\_')[-2]

                if Geo\_n\_name.isdigit() and Dum\_n\_name.isdigit() and Geo\_n\_name == Dum\_n\_name:

                    cmds.blendShape(Dum, Geo)

함수들을 쫘라락 늘어놓는다. 그리고 이 함수들과 버튼 (pyside에서 지정한 이름들이랑 같아야 함)들과 연결시킨다.

**문법= self.ui.(pyside에서 설정한 버튼이름).clicked.connect(self.(연결하고싶은 함수이름))**

밑에 설명할 함수 내부에 연결하는 것들(라디오 버튼, 텍스트 필드등등)은 따로 설명한다.

    def FindSet(self,Query):

        Query = self.ui.findingnode\_TextF\_txf.toPlainText()

                    #-필드에 쓰인 텍스트를 가져와야 하므로 toPlainText을 사용(https://wikidocs.net/35491)

        variable\_obj = []

        for obj in cmds.ls():

            if Query in obj:

                variable\_obj.append(obj)

        if variable\_obj:

            cmds.select(variable\_obj, replace=True)

        else:

            print("Not found")

이 함수가 함수 내부에 파이사이드를 연결시켜야 하는 놈들 중 하나다. 이 아이가 받는 인수에 self를 추가하고, 원래 가지고 있던 변수 Query에 **self.ui.findingnode\_TextF\_txf.toPlainText()**를 가져온다. 이 값은 텍스트 필드에 쓰인 글자를 가져온것이다. toPlainText() 요 놈이 그 역할을 한다. 그리고 람다를 사용해서 연결. 왜 이 놈만 따로 함수식으로 버튼 연결을 시켜줘야 하는지는 알 수 가 없다. 모르는 것 투성이다...

    def executeCreation(self):

        self.create\_Loc\_rbt = self.ui.create\_Loc\_rbt.isChecked()

        self.create\_Clu\_rbt = self.ui.create\_Clu\_rbt.isChecked()

        self.create\_Jnt\_rbt = self.ui.create\_Jnt\_rbt.isChecked()

        if self.create\_Loc\_rbt == True:

            cmds.spaceLocator()

        elif self.create\_Clu\_rbt == True:

            cmds.cluster()

        else:

            cmds.joint()

    #-함수 끝

아까 전 함수들보다 비교적 쉽다. 실행 버튼은 위에 이미 연결해 두었으니(외부로 빼도 상관없음) 라디오 버튼들을 변수화 하여 함수에 각각 연결하면 된다. isChecked()요놈이 그 기능을 해준다.

if \_\_name\_\_ == "\_\_main\_\_":

    try:

        desinger\_ui.close()

        desinger\_ui.deleteLater()

    except:

        pass

    desinger\_ui = DesignerUI()

    desinger\_ui.show()

#-클래스를 변수화 하여 열기

그리고 대망의 실행. window.show() app.exec\_()보다 쉽고 빠르고 작동한다. 작동이 중요한 포인트임 ㅋ... 잘된다. if구절은 대체 왜 쓰는지 알 수 없지만 문법이라고 생각하고 넘어가면 대충 70%는 납득됨. ㅋㅋ.

--

+) 색 설정 메모

-버튼의 색상을 변경할 때엔 스크립트 상에서 함수로 변경하는게 아니라 pyside 파일 내에서 변경한다. rgb색상표 보고 작성하지 않아도 색깔 찍어 넣을 수 있어서 편하다. 찍으면 자동으로 명령어도 같이 찍히긴 하지만 가끔 안 찍히니까 적어 놓음.

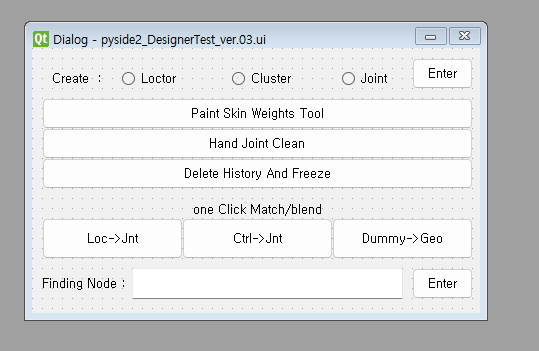
-background-color:rgb( rgb 색상표); =버튼 색 설정

-color:rgb( rgb 색상표); =글자 색 설정

-네번째 숫자값을 추가하면 불투명도(알파=a)를 추가로 가지게 됨. 예시: background-color:rgb(0, 200, 255,50); =오퍼시티 50인 파란색 버튼 생성 기재하지 않으면 자동으로 불투명이된다. (불투명을 나타내는 값은 255.), “#ff0000”이런 색상표로도 나타낼 수 있으나 나는 rgb색상으로 보는 것이 더 직관적이라 생각한다.

+) 레이아웃 메모

horizontalSpacer가 어디에 쓰는 것인지 의문이었는데 자동 Layout을 잡을 때 모양을 잡아주는 역할을 한다. 나는 이걸 못 써서 하나하나 레이아웃을 잡아준 것이고. 익숙하지 않으면 그게 확실히 예쁜 모양을 만들 수 있을 듯.



**+ 추가 수정(버튼과 기능 추가.)**

1. copy skin

2. POICF(loc)

3. Min Find, delete

4. tutle delete

5. key delete

6. sub lock, unlock

7. foli (x,y,z축 버튼/숫자로 크기 조절 버튼)

+) 1, 2, 3, 4, 5는 단순히 버튼에 함수만 연결했기 때문에 추가로 설명하지 않음.

6. sub lock, unlock

        self.ui.sub\_lock\_bt.clicked.connect(lambda: self.Sub\_lock(self))

        self.ui.sub\_unlock\_bt.clicked.connect(self.Sub\_unlock)

    def Sub\_lock(Check,self):

        pre\_lst = cmds.ls(type='RedshiftMeshParameters')

        lst = [x for x in pre\_lst if '\_sub' in x]

        if Check:

            lock = True

        else:

            lock = False

        for obj in lst:

            cmds.lockNode(obj, lock=lock)

    def Sub\_unlock(self):

        pre\_lst = cmds.ls(type='RedshiftMeshParameters')

        lst = [x for x in pre\_lst if '\_sub' in x]

        lock = False

        lst\_lock = [cmds.lockNode(x, l=lock) for x in lst]

+이 버튼들은 한 함수를 나눠 쓰지 않고 그냥 함수를 두 개 만들었다. 그게 더 편하다 판단해서. 스크립트 길이가 길어지 긴 했으나 실재로 더 편했음. ㅋ케켘ㅋㅋㅋ

7. Function\_JntFoli

    def Function\_JntFoli(self):

        sel = cmds.ls(sl=1 )

        self.foli\_x\_rbt = self.ui.foli\_x\_rbt.isChecked()

        self.foli\_y\_rbt = self.ui.foli\_y\_rbt.isChecked()

        self.foli\_z\_rbt = self.ui.foli\_z\_rbt.isChecked()

        self.foli\_spin\_rbt = self.ui.foli\_spin\_rbt.value()

        if self.foli\_x\_rbt == True:

            value = 'X'

        elif self.foli\_y\_rbt == True:

            value = 'Y'

        else:

            value = 'Z'

        A = c\_JntFoli(sel ,value, self.foli\_spin\_rbt)

    #class인 함수는 따로 빼놓고 위 처럼 def의 변수로 처리해서 넣어버리기

class c\_JntFoli():

    def \_\_init\_\_(self, lst\_Select, normal, size):

        self.lst\_Select = lst\_Select

        self.normal = normal

        self.size = size

        self.lst\_Foli = []

        self.lst\_PN = []

        if self.normal == 'X':

            nor\_lst = [1, 0, 0]

        elif self.normal == 'Y':

            nor\_lst = [0, 1, 0]

        else:

            nor\_lst = [0, 0, 1]

        for x in lst\_Select:

            PN = cmds.polyPlane(ax=nor\_lst, n='PN\_Foli\_' + x, sx=1, sy=1, w=self.size, h=self.size)

            pos\_t = cmds.xform(x, ws=1, t=1, q=1)

            pos\_r = cmds.xform(x, ws=1, ro=1, q=1)

            cmds.xform(PN[0], ws=1, t=pos\_t)

            cmds.xform(PN[0], ws=1, ro=pos\_r)

            Foli = cmds.createNode('follicle', n='Foli\_' + x + 'Shape')

            Tf = cmds.listRelatives(Foli, ap=1, type='transform')

            PN\_shp = cmds.listRelatives(PN[0], s=1, type='mesh')

            cmds.connectAttr(Foli + '.outTranslate', Tf[0] + '.translate', f=1)

            cmds.connectAttr(Foli + '.outRotate', Tf[0] + '.rotate', f=1)

            cmds.connectAttr(PN\_shp[0] + '.outMesh', Foli + '.inputMesh')

            cmds.connectAttr(PN\_shp[0] + '.worldMatrix[0]', Foli + '.inputWorldMatrix')

            cmds.setAttr(Foli + '.parameterU', 0.5)

            cmds.setAttr(Foli + '.parameterV', 0.5)

            grp = cmds.listRelatives(x, p=1, type='transform')

            prime = cmds.listRelatives(grp, p=1, type='transform')

            try:

                cmds.parentConstraint(Tf[0], grp, mo=1)

            except:

                pass

            self.lst\_PN.append(PN[0])

            self.lst\_Foli.append(Tf[0])

        g\_Foli = cmds.group(self.lst\_Foli, n='grp\_All\_Foli\_' + lst\_Select[0])

        g\_PN = cmds.group(self.lst\_PN, n='grp\_All\_PN\_' + lst\_Select[0])

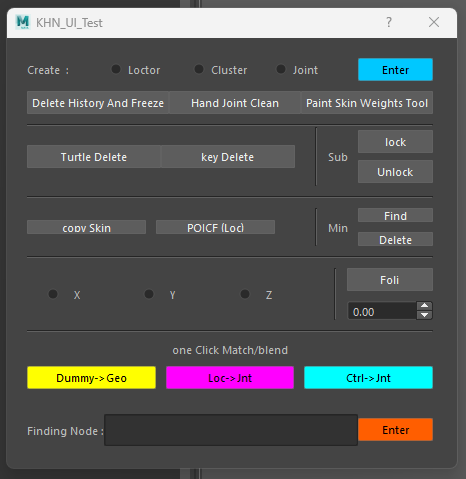
+ class c\_JntFoli():를 따로 뺀 뒤, 변수 화 시킨다. (변수 이름은 A<-뭘 해도 상관없다.)

A = c\_JntFoli(sel ,value, self.foli\_spin\_rbt)

따로 빼준 클래스는 아무것도 건들지 않아도 됨.

변수화 시킨 것을 함수에 적용시키고 작동방식, 버튼도 변수 화 함수에 적용한다.

밑은 변경된 UI... 어차피 내가 쓸 테니 설명은 필요 없겠지만 딱히 할 일도 없고, 나중 가서 이 스크립트를 보면 또 다 까먹을 게 뻔하기 때문에 그냥 쓴다.



첫번째 열: 로케이터, 클러스터, 조인트를 생성한다. 당연하지만 라디오 버튼을 체크한 뒤 파란색 엔터를 누를 것.

두번째 열: 왼쪽은 이름에\_geo들어간 오브젝트들의 히스토리 날리고 프리즈 시킴. 가운데는 우리 오토리깅의 손가락 조인트를 정렬하기 너무 귀찮아서 따로 만들어둔 버튼. 오른쪽은 페인트 스킨웨잇툴 꺼내기가 너무 귀찮아서 따로 만들었다.

세번째 열: 쓸모 없는 애들 삭제하는 버튼 두개, 오른쪽은 렌더시 sub먹어야 하는 오브젝트 셋들을 잠금 하거나 푸는 버튼이다.

네번째 열: 왼쪽은 첫번째로 누른 오브젝트를 뒤이어 누른 오브젝트들로 카피스킨 함. 가운데는 선택한 오브젝트들의 위치를 참고한 커브와 그 커브를 따라가는 로케이터를 생성함. 오른쪽은 오브젝트들 트랜스레잇값에 –e가 붙었는지를 확인하거나 삭제한다.

다섯번 째 열: X혹은Y,Z축을 향하는 폴리클을 지정된 값의 크기 만큼, 선택한 오브젝트들 위에 생성, 연결한다.

여섯 번 째 열: 버튼에 쓰여져 있는 대로 매치시킨다.

일곱 번 째 열: 텍스트 필드에 쓴 글자를 포함하는 오브젝트들을 찾는다. 빨간색 엔터를 누르면 찾을 수 있다. (이 함수의 문제점은 아웃라이너에 있는 것만 찾는 게 아닌, 이 씬 내의 모든 노드를 다 찾아서 선택한다는 점.)