Ikfk 스위치 추가필기

1. 베이스(Root) 컨트롤러 생성 함수 필기 ※def base\_Ctrl(self):

        cmds.addAttr(shortName= 'Switch', longName='Switch',attributeType="float",

                        hasMaxValue=True,maxValue=1,hasMinValue=True,minValue=0,keyable=True)

        cmds.addAttr(shortName= 'Stretch', longName='Stretch',attributeType="float",

                        hasMaxValue=True,maxValue=1,hasMinValue=True,minValue=0,keyable=True)

어트리뷰트 추가하기. 스위치와 스트레치를 추가했음.

사용한 옵션들은 밑에 있다.

1) 딱 봐도 이름 정하는 옵션: shortName= 'Switch', longName='Switch'

2) 최댓값을 가지느냐: hasMaxValue= True (트루 말고 1로 써도 되지만 난 늘 전부 치지...)

3) 그럼 그 최댓값이 얼마냐: maxValue= 1

4) 최솟값을 가지느냐: hasMinValue= True

5) 그럼 그 최솟값이 얼마냐: minValue= 0

6) 어트리뷰트창에 뜨도록 함. (키를 잡을 수 있게 함): keyable=True

7) 스트레치 옵션도 똑같이 세팅ㄱㄱ.

-

2. ik조인트 생성 함수 필기 ※def ik\_function()

cmds.select(cl=1)

조인트를 생성할 때에 페런츠 되지 말라고 초기화 하는 구문(=[clear=boolean]=허공클릭하기)

    for int ,B in enumerate(Df\_JntGrp):

int는 숫자(당연히 정수)를 지정함.

enumerate함수로 for문 돌리기 설명 <https://www.daleseo.com/python-enumerate/>

        Name = "Arm{}IK\_Jnt"

        Number = int +1

        new\_Jnt =cmds.joint(n = Name.format(Number))

파이썬은 숫자를 셀 때 0부터 세니까 int에 +1씩 해준 것을 number로 변수 지정 해준다. 그 넘버를 Arm{}IK\_Jnt의 { }에 넣는다. Format은 { }을 사용하면 같이 써줘야 하는 단어다.

-

import pprint

A = [x for x in range(99)]

pprint.pprint(A)

print (A)

+) 막간을 이용하여 추가 필기

pprint라는 모듈을 임포트하면 pprint.pprint( )이런 구문을 쓸 수 있게 되는데 print된 단어들이 가로로 길게 늘어져 스크롤을 사용하던 것을 세로로 정렬해준다. 솔직히 난 그냥 영어면 다 보기 힘든 듯.

-

3. 컨트롤러 생성 함수 필기 ※def add\_Ctrl(self)

        ik\_Ctrl\_1= cmds.circle(normal=(1,0,0), n = 'Arm1Ik\_Ctrl')

        cmds.matchTransform(ik\_Ctrl\_1, 'Arm1IK\_Jnt')

        IK\_Ctrl\_Grp1 =cmds.group(empty=True, n= 'Arm1Ik\_Ctrl\_G')

        cmds.matchTransform(IK\_Ctrl\_Grp1,ik\_Ctrl\_1)

        cmds.parent(ik\_Ctrl\_1,IK\_Ctrl\_Grp1)

일단은 ik. X축을 바라보고있는 원(=서클)을 만든다. 그리고 그 원을 Arm1IK\_Jnt에 매치트랜스폼 한다.

그리고 그룹을 만들고 거기에 순서대로 매치, 페런츠 하면서 컨트롤러들의 값을 초기화 시킨다.

        for int ,D in enumerate(FkJntList):

                Name = "Arm{}FK\_Ctrl".format

                Number = int +1

                new\_Fk\_Ctrl =cmds.curve(degree=1, point=[(0,1,-1),(0,-1,-1),(0,-1,1),(0,1,1),(0,1,-1)], n = Name(Number))

                cmds.matchTransform(new\_Fk\_Ctrl, D)

                FK\_Ctrl\_Grp =cmds.group(empty=True, n= new\_Fk\_Ctrl+'\_G')

                cmds.matchTransform(FK\_Ctrl\_Grp,new\_Fk\_Ctrl)

                cmds.parent(new\_Fk\_Ctrl,FK\_Ctrl\_Grp)

그리고 fk. Fk는 별거 없다. 전체 그룹을 지은 다음에 순서를 매기고, 만든 네모 컨트롤러를 매치 시킨다. 그리고 그룹을 만들어 페런츠.

별거 많네 ㅋㅋ

-

4. 컨트롤러 기능추가 함수 필기 ※def add\_Ctrl\_function(self):

        #ikConstraint

        cmds.ikHandle(n='Arm\_Ikhandle', startJoint='Arm1IK\_Jnt', endEffector='Arm3IK\_Jnt', solver='ikRPsolver')

        cmds.parent('Arm\_Ikhandle','Arm2Ik\_Ctrl')

        cmds.parentConstraint('Arm1Ik\_Ctrl','Arm1IK\_Jnt')

ikhandle만들고서는 Arm2Ik\_Ctrl에 집어넣음. 분량 끝.

        #fkConstraint

        cmds.parentConstraint('Arm1FK\_Ctrl','Arm1FK\_Jnt')

        cmds.parentConstraint('Arm2FK\_Ctrl','Arm2FK\_Jnt')

        cmds.parentConstraint('Arm3FK\_Ctrl','Arm3FK\_Jnt')

fk컨트롤러들을 조인트에 컨스트레인 시킴.

        #parentgroup

        cmds.parent('Arm2Ik\_Ctrl\_G','Arm1Ik\_Ctrl')

        cmds.parent('Arm3FK\_Ctrl\_G','Arm2FK\_Ctrl')

        cmds.parent('Arm2FK\_Ctrl\_G','Arm1FK\_Ctrl')

        cmds.parent('Arm1FK\_Ctrl\_G','Arm1Ik\_Ctrl\_G','Root\_Ctrl')

오브젝트들을 계층구조 맞춰서 정리한다. 그러면 딴 기능(스위치+스트래치) 을 넣을 차례다.

5. ikfk스위치 기능추가 함수 필기 ※def Switch\_function():

        Df\_JntGrp = cmds.ls('Df\*\_Jnt', type='joint')

        Fk\_JntGrp = cmds.ls("Arm\*FK\_Jnt", type='joint')

        Ik\_JntGrp = cmds.ls("Arm\*IK\_Jnt", type='joint')

        for Df\_Jnt, Fk\_Jnt in zip(Df\_JntGrp, Fk\_JntGrp):

            cmds.parentConstraint(Fk\_Jnt, Df\_Jnt)

        for Df\_Jnt, Ik\_Jnt in zip(Df\_JntGrp,Ik\_JntGrp):

            cmds.parentConstraint(Ik\_Jnt, Df\_Jnt)

Df\_Jnt에 ik와 fk조인트들을 둘 다 컨스트레인 한다.

zip함수로 for문 돌리기 설명 <https://www.daleseo.com/python-zip/>

        cmds.createNode('blendColors', n='ikfk\_Switch\_Bs')

        cmds.setAttr('ikfk\_Switch\_Bs.color1R', 0)

        cmds.setAttr('ikfk\_Switch\_Bs.color1G', 1)

        cmds.setAttr('ikfk\_Switch\_Bs.color2R', 1)

        cmds.setAttr('ikfk\_Switch\_Bs.color2B', 0)

그리고 블렌드 컬러 생성. 생성하자마자 뜰 기본 어트리 뷰트도 만져놓는다.

        cmds.connectAttr('Root\_Ctrl.Switch','ikfk\_Switch\_Bs.blender')

        cmds.connectAttr('ikfk\_Switch\_Bs.outputR', 'Df1\_Jnt\_parentConstraint1.Arm1FK\_JntW0')

        cmds.connectAttr('ikfk\_Switch\_Bs.outputG', 'Df1\_Jnt\_parentConstraint1.Arm1IK\_JntW1')

        cmds.connectAttr('ikfk\_Switch\_Bs.outputR', 'Df2\_Jnt\_parentConstraint1.Arm2FK\_JntW0')

        cmds.connectAttr('ikfk\_Switch\_Bs.outputG', 'Df2\_Jnt\_parentConstraint1.Arm2IK\_JntW1')

        cmds.connectAttr('ikfk\_Switch\_Bs.outputR', 'Df3\_Jnt\_parentConstraint1.Arm3FK\_JntW0')

        cmds.connectAttr('ikfk\_Switch\_Bs.outputG', 'Df3\_Jnt\_parentConstraint1.Arm3IK\_JntW1')

그리고 대망의 연결쓰. Df\_Jnt에 연결된 세개의 컨스트레인들을 전부 블렌드 컬러 노드에 연결시킨다. 나는 이번에 IK는 R에, FK는 G에 연결했다.

        cmds.connectAttr('ikfk\_Switch\_Bs.outputR', 'Arm1FK\_Ctrl\_G.visibility')

        cmds.connectAttr('ikfk\_Switch\_Bs.outputG', 'Arm1Ik\_Ctrl\_G.visibility')

그 참에 컨트롤러들도 블렌드 컬러에 연결시켜서 같이 스위치 될 수 있도록 한다.

-

6. 스트레치 기능추가 함수 필기 ※def stretch\_function():

        StartObj= 'Df1\_Jnt'

        EndObj= 'Df3\_Jnt'

위치를 지정함

        E = cmds.spaceLocator(n='Stretch\_start\_Loc')

        F = cmds.spaceLocator(n='Stretch\_end\_Loc')

        cmds.matchTransform(E, 'Arm1Ik\_Ctrl')

        cmds.matchTransform(F, 'Arm2Ik\_Ctrl')

        cmds.parent(E,'Arm1Ik\_Ctrl')

        cmds.parent(F,'Arm2Ik\_Ctrl')

        cmds.createNode('distanceDimShape', n='Arm\_DT')

        cmds.rename( 'distanceDimension1', 'ArmDT' )

        cmds.connectAttr('Stretch\_start\_LocShape.worldPosition','Arm\_DT.startPoint')

        cmds.connectAttr('Stretch\_end\_LocShape.worldPosition','Arm\_DT.endPoint')

        cmds.spaceLocator(n='ScaleDefault')

'Stretch\_start\_Loc','Stretch\_end\_Loc'라는 이름의 로케이터 두 개를 생성하고 변수 E와 F에 할당한다. 다음엔 컨트롤러에 매치, 그 밑에 집어넣어서 실시간 거리측정하도록 함. +스케일 디폴트 만들어 놓는다.

        S\_Pos = cmds.xform(StartObj , q =1, t =1 ,ws =1)

        E\_Pos = cmds.xform(EndObj, q=1, t=1, ws=1)

        Arm\_distance = DT = ((S\_Pos[0] - E\_Pos[0])\*\*2 + (S\_Pos[1] - E\_Pos[1])\*\*2 + (S\_Pos[2] - E\_Pos[2])\*\*2)\*\*0.5

위의 변수 처리된 로케이터(E,F)로 길이를 구한다. (이 길이는 빌드 때 마다 달라진다. =Df\*\_Jnt의 위치들에 영향을 받는다.)

+이 식은 유클리드or피타고라스 거리 구하기을 응용한 것임. <https://banjjak1.tistory.com/9>

        cmds.createNode('multiplyDivide',n='ScDefault\_Md')

        cmds.setAttr('ScDefault\_Md.input1X', Arm\_distance )

        cmds.setAttr('ScDefault\_Md.input1Y', Arm\_distance )

        cmds.connectAttr('ScaleDefault.scaleX','ScDefault\_Md.input2X')

위에서 구한 거리를 input에 집어 넣음. 이건 실시간으로 바뀌는 값은 아님.

        cmds.createNode('multiplyDivide',n='Distance\_Md')

        cmds.setAttr('Distance\_Md.operation', 2)

        cmds.connectAttr('Arm\_DT.distance', 'Distance\_Md.input1X')

        cmds.connectAttr('ScDefault\_Md.outputX', 'Distance\_Md.input2X')

실시간으로 바뀌는 값은 세 번째 줄의 ‘Arm\_DT.distance’ . 'Distance\_Md.input1X'에 연결한다.

        cmds.createNode('blendColors', n='Stretch\_Bc')

        cmds.setAttr('Stretch\_Bc.color2R', 1)

        cmds.setAttr('Stretch\_Bc.color2B', 0)

        cmds.connectAttr('Root\_Ctrl.Stretch','Stretch\_Bc.blender')

        cmds.connectAttr('Distance\_Md.outputX','Stretch\_Bc.color1R')

        cmds.createNode('condition',n='Stretch\_Cd')

        cmds.setAttr('Stretch\_Cd.operation', 3)

        cmds.connectAttr('Arm\_DT.distance','Stretch\_Cd.firstTerm')

        cmds.connectAttr('Distance\_Md.outputX','Stretch\_Cd.secondTerm')

        cmds.connectAttr('Stretch\_Bc.outputR','Stretch\_Cd.colorIfTrue.colorIfTrueR')

        cmds.connectAttr('Stretch\_Cd.outColor.outColorR','Arm1IK\_Jnt.scale.scaleX')

        cmds.connectAttr('Stretch\_Cd.outColor.outColorR','Arm2IK\_Jnt.scale.scaleX')

그리고 또 뭐 여러가지 옵션들을 연결. 핵심 내용은 원래 가지고 있던 길이보다 길어지면 그 길어진 값만큼 곱해진다는 것.

7. 그룹 정리 함수 필기    ※def cleanUpTheCode(self):

이건 뭐 정리 할 것도 딱히 없다.

        cmds.parentConstraint('Root\_Ctrl','ScaleDefault')

스케일 디폴트를 루트 컨트롤러에 집어넣는다.

        cmds.group(empty=True, n= 'Jnt\_Grp')

        cmds.group(empty=True, n= 'Sys\_Grp')

빈 그룹노드를 만든다.

        Sys\_Grp\_Obj= ('ArmDT','ScaleDefault')

        Jnt\_Grp\_Obj= ('Arm1FK\_Jnt','Arm1IK\_Jnt','Df1\_Jnt')

변수들에 오브젝트들을 한번에 때려 넣어 할당한다.

        cmds.parent(Jnt\_Grp\_Obj,'Jnt\_Grp')

        cmds.parent(Sys\_Grp\_Obj,'Sys\_Grp')

        cmds.parent('Jnt\_Grp','Sys\_Grp','Root\_Ctrl\_G')

그리고 아까 만들었던 그룹과 루트 그룹에 넣어주면서 정리.

-

8. 클래스 필기

class Arm\_build():

    def \_\_init\_\_(self):

        self.base\_Ctrl()

        self.ik\_BaseJnt()

        self.fk\_BaseJnt()

        self.add\_Ctrl()

        self.add\_Ctrl\_function()

        self.Switch\_function()

        self.stretch\_function()

        self.cleanUpTheCode()

그냥 이런 변수가 있다고 늘어놓는다. 그리고 이 밑은 전부 함수 나열.

특별한 일이 없다면 인수 self를 모든 함수에 집어넣어준다. (self라는 이름이 아니어도 됨.)

+특별한 일들의 예: 람다식을 사용하거나 추가로 인수를 더 넣어줘야 할 때 (아무튼 그런 일이 가끔 생김... ㅇㅇ.)

Arm\_build()

실행~