"buildMe" - Tool
I wrote this Python tool for making any body parts, any full rigs.
It's include IK/FK 2/3 bones and spline any numbers bones IK/FK kinematics with squash, stretch.
+ Dynamics bones/body parts
+ Mocap module, for retargeting animations from any mocap rigs
+ Tool for change controls shape, scale
UI: windows and context menu, wrote by Classes.
Context menu for animators include most used commands:
Mirror, Switch IK/FK(save pose), Reset...

I: http://www.fancyart3d.com/character.files/buildMe.mp4
II: http://www.fancyart3d.com/character.files/buildMe_dragonfly.mp4

Code example:

```
##############################################################

class buildMeMainWindow(object):

    def __init__(self, mode, charName, suffCust, custScale, *win ):
        self.mode = mode
        if charName.find("_char")!=-1:
            self.charName = charName[:-5]
        else:
            self.charName = charName
        self.suffCust = suffCust
        self.custScale = custScale
        self.deleteMember = delUi
        global winBM
        self.txFilAll = []
        self.list_setUp_grp = []
        self.mocapDo = mocapDo

        self.deleteMember(self,"mainWindow", "mocapWindow")

        self.buildMeMW = cm.window( "mainWindow", title = fa3d(s=1), rtf=1)
        winBM = self.buildMeMW

        self.formMain = cm.formLayout()
        self.tabs = cm.tabLayout(innerMarginWidth=5, innerMarginHeight=5,
bs='none')
        cm.formLayout( self.formMain, e=1, attachForm=((self.tabs, 'top', 0), (self.tabs,
'left', 0), (self.tabs, 'bottom', 0), (self.tabs, 'right', 0)) )

        self.childRig = cm.columnLayout( )
        self.begin = cm.frameLayout( l='Begin', cll=1, cl=1, w=318 )
        self.form = cm.formLayout(numberOfDivisions=300)
```

```
        self.colLayBegin = cm.columnLayout(  )
        self.charNameTF = cm.textFieldGrp( label='Character:', text=self.charName,
cw2=[70,180] )
        cm.rowColumnLayout( numberOfColumns=2 )
        self.suffTF = cm.textFieldGrp( label='Suffix:', text=self.suffCust, cw2=[68,80] )
        self.scaleTF = cm.textFieldGrp( label='Scale:', text=self.custScale, cw2=[50,43] )
        cm.formLayout( self.form, e=1, attachForm=[(self.colLayBegin, 'top', 6),
(self.colLayBegin, 'left', 10), (self.colLayBegin, 'bottom', 0)], attachPosition=[(self.col-
LayBegin, 'right', 10, 300)])
        cm.setParent( '..' )
        cm.setParent( '..' )
        cm.setParent( '..' )

        self.make = cm.frameLayout( l='Make', w=318 )
        self.formMake = cm.formLayout(numberOfDivisions=300)
        self.colLayMake = cm.columnLayout(  )
        cm.formLayout( self.formMake, e=1, attachForm=[(self.colLayMake, 'top', 5),
(self.colLayMake, 'left', 10)], attachPosition=[(self.colLayMake, 'right', 10, 300)])
        cm.setParent( '..' )
        cm.setParent( '..' )
        cm.setParent( '..' )

        self.set = cm.frameLayout( label='Set', w=318 )
        cm.setParent( '..' )
        self.rowColumnTxFil = cm.rowColumnLayout( numberOfColumns=2 )
        cm.setParent( '..' )


        self.do = cm.frameLayout( l='Do', w=318 )
        self.formDo = cm.formLayout( numberOfDivisions=300 )
        self.columnLayoutDo = cm.columnLayout( columnAttach=('both', 0), rowSpa-
cing=10, columnWidth=315, h=63 )
        cm.button( label='Make It',     c= "exec( open( pathPy + 'make.py', 'r').read(),
locals().update( winBM.makeItBut() ) );winBM.GET_setUp_grp(nSetUpGrp)", w=320 )
        cm.button( label='Build It',    c= "winBM.buildBut();exec(  open( pathPy + 'ma-
ke.py', 'r').read())", w=320 )
        cm.formLayout( self.formDo, e=1, attachForm=[(self.columnLayoutDo, 'top',
11), (self.columnLayoutDo, 'bottom', 7)])
        cm.setParent( '..' )
        cm.setParent( '..' )
        cm.setParent( '..' )
        cm.setParent( '..' )

        self.show = cm.frameLayout( label='Show', cll=1, cl=1, w=318 )
        self.form4 = cm.formLayout( numberOfDivisions=300 )
        self.shButt = cm.rowColumnLayout( numberOfRows=1 , cs=[1, 9], rat =[1, 'both',
5])
        comStr = '"toggleObj( 0, "cm.headsUpDisplay( 'LopeButtonHUD', ex=1 )",
"vpMB.deleteMember('LopeButtonHUD', 'axisSliderHUD', 'shapeSliderHUD', 'scaleSli-
```

```
derHUD');vpMB.lopeHud()", "vpMB.deleteMember('LopeButtonHUD', 'axisSliderHUD',
'shapeSliderHUD', 'scaleSliderHUD')" )'"
        cm.button( label='Leap',  c= comStr, w= 72)
        comStr = '"toggleObj( 0, "cm.headsUpDisplay( 'scaleSliderHUD', ex=1 )",
"vpMB.deleteMember('LopeButtonHUD', 'axisSliderHUD', 'shapeSliderHUD', 'scaleSli-
derHUD');vpMB.scaleHud()", "vpMB.deleteMember('LopeButtonHUD', 'axisSlider-
HUD', 'shapeSliderHUD', 'scaleSliderHUD')" )'"
        cm.button( label='Shape', c= comStr, w= 72)
        comStr = "toggleObj( 0, 'menuMC.toggle', 'menuMC.shoPop()', 'me-
nuMC.cler()' )"
        cm.button( label='Menu', c= comStr, w= 72)
        cm.button( label='SetUp', c= partial( self.importPref ), w= 72)
        cm.formLayout( self.form4, e=1, attachForm=[(self.shButt, 'top', 7), (self.shButt,
'bottom', 7)] )
        cm.setParent( '..' )
        cm.setParent( '..' )
        cm.setParent( '..' )

        self.mocap = cm.frameLayout( label='Mocap', cll=1, cl=1, w=318 )
        self.form5 = cm.formLayout( numberOfDivisions=300 )
        self.mocButt = cm.rowColumnLayout( numberOfRows=1 )

        cm.rowColumnLayout( numberOfRows=1 , cs=[1, 9], rat =[1, 'both', 5])
        self.collectionMocap = cm.radioCollection()
        cm.radioButton( 'radioButtonIK', label='IK',              onc = partial( self.ikFkMo-
cap, 'rbIK' ), w=40 )
        cm.radioButton( 'radioButtonFK', label='FK', align='left', onc = partial( self.ikFk-
Mocap, 'rbFK' ), w=45 )
        cm.button( label='List',  c= partial(self.listBut), w= 90)
        cm.button( label='Bind', c= partial(self.bindBut), w= 90)
        cm.setParent( '..' )
        cm.formLayout( self.form5, e=1, attachForm=[ (self.mocButt, 'top', 7),
(self.mocButt, 'left', 16), (self.mocButt, 'bottom', 7) ] )
        cm.setParent( '..' )
        cm.setParent( '..' )
        cm.setParent( '..' )

        self.formClose = cm.formLayout( numberOfDivisions=300 )
        self.colLayClose = cm.columnLayout( columnAttach=('both', 0), column-
Width=315  )
        cm.button( label='End', c= partial( self.deleteMember, "mainWindow", "mo-
capWindow", 'LopeButtonHUD', 'axisSliderHUD', 'shapeSliderHUD', 'scaleSliderHUD' ),
w=320 )
        cm.formLayout( self.formClose, e=1, attachForm=[(self.colLayClose, 'top', 5),
(self.colLayClose, 'bottom', 7)] )

        self.collectionLayoutMake = cm.rowColumnLayout( numberOfColumns=3, co-
lumnWidth=[(1, 98), (2, 98), (3, 98) ] , p=self.colLayMake )
        self.collectionMake = cm.radioCollection()
```

```
        copy_modeType = list(modeType)
        copy_modeType.pop(modeType.index('build'))
        copy_modeType.pop(copy_modeType.index('work'))
        copy_modeType.pop(copy_modeType.index('piece'))
        for i in copy_modeType:
            self.rb = cm.radioButton( 'radButt%d' %modeType.index(i), l='%s' %i, onc=par-
tial(self.report, i ) )
        cm.radioCollection( self.collectionMake, e=1, sl='radButt%d' %modeType.in-
dex(mode) )

        cm.radioCollection( self.collectionMocap, e=1, sl='radioButtonIK' )

        self.childTools = cm.rowColumnLayout(numberOfColumns=2 , p= self.tabs )
        cm.button()
        cm.button()
        cm.button()
        cm.setParent( '..' )

        cm.tabLayout( self.tabs, e=1, tabLabel=((self.childRig, 'Rig'), ( self.childTools,
'Tools')) )

        cm.window( self.buildMeMW, e=1, w=320.0, h=105)
        cm.showWindow()


    def getAttrProc( self, *args):
        global centreMocap
        global sideMocap
        global mocapPrefix
        global mocapSuffix
        if args[2] == 'centreMocap':
            centreMocap[args[0]] = args[4]
            if cm.objExists(mocapPrefix + args[4] + mocapSuffix)==0:
                # centreMocap[args[0]] =''
                textStr = 'Not found: ' + args[4].replace( 'Not found: ', '' )
            elif len(cm.ls( '*' + args[0] + suff[1] )) == 0 and args[0] not in side:
                textStr = 'Not found: ' + args[0] + suff[1].replace( 'Not found: ', '' )
                centreMocap[args[0]] = args[4] + suff[1]
            else:
                textStr = args[4]
            cm.textFieldGrp( args[3], e=1, text = textStr )

        elif args[2] == 'sideMocap':
            sideMocap[args[0]] = args[4]
            if cm.objExists( mocapPrefix + sideMocap["l_"] + args[4] + mocapSuffix)==0
and args[4] not in side:
                # sideMocap[args[0]] =''
                textStr = 'Not found: ' + args[4].replace( 'Not found: ', '' )
            elif len(cm.ls( '*' + args[0] + suff[1] )) == 0 and args[0] not in side:
```

```
            textStr = 'Not found: ' + args[0] + suff[1].replace( 'Not found: ', '' )
            sideMocap[args[0]] = args[4] + suff[1]
        else:
            textStr = args[4]
        cm.textFieldGrp( args[3], e=1, text = textStr )

    elif args[2] == 'prefix':
        mocapPrefix = args[3]
    elif args[2] == 'suffix':
        mocapSuffix = args[3]
    elif args[2] == 'characterName':
        pass


  def mocapWindow( self ):
    global mocapPrefix
    global mocapSuffix
    global centreMocap
    global sideMocap

    global centreMocapFK
    global sideMocapFK
    global centreMocapIK
    global sideMocapIK

    global centreMocapFK_MB
    global sideMocapFK_MB
    global centreMocapIK_MB
    global sideMocapIK_MB

    self.deleteMember("mocapWindow")
    mocapMW = cm.window( "mocapWindow", title = "Mocap list", rtf=1)

    if cm.ls(rf=1)== []:
        cm.warning( "There's no a reference character in the scene.")
    elif len(cm.ls(rf=1)) > 1:
        cm.warning( "There's too much a reference in the scene: " + ',
'.join(cm.ls(rf=1)) )
    elif cm.referenceQuery( cm.ls(rf=1)[0], n=1 ) != None:

        ref = cm.referenceQuery( cm.ls(rf=1)[0], n=1 )
        for nam in ref:
            mocapPrefixNew =''
            if nam.find(':Hips')!=-1 and cm.objectType( nam, isType='joint' ):
                if cm.radioCollection( self.collectionMocap, q=1, sl=1 ) == 'radioButto-
nIK':
                    centreMocap = centreMocapIK_MB
                    sideMocap =   sideMocapIK_MB
                else:
```

```
                centreMocap = centreMocapFK_MB
                sideMocap =   sideMocapFK_MB

            namN = nam.split(':')
            for n in namN[:-1]:
                mocapPrefixNew += n +':'
            mocapPrefix = mocapPrefixNew
            mocapSuffix = ''
            break
        elif nam.find('hip')!=-1 and cm.objectType( nam, isType='joint' ):
            if cm.radioCollection( self.collectionMocap, q=1, sl=1 ) == 'radioButto-
nIK':
                centreMocap = centreMocapIK
                sideMocap =   sideMocapIK
            else:
                centreMocap = centreMocapFK
                sideMocap =   sideMocapFK
            sideMocap.update(fingersMocap)

            namN = nam.split(':')
            for n in namN[:-1]:
                mocapPrefixNew += n +':'
            if namN[-1].find( suff[33][1:] )==-1:
                mocapPrefix = mocapPrefixNew
            else:
                mocapPrefix = mocapPrefixNew + suff[33][1:]+'_'
            if len( namN[-1].split('_') )>1:
                mocapSuffix = '_'+str( namN[-1].split('_')[-1] )
            else:mocapSuffix = ''
            break

    prefTF = cm.textFieldGrp( 'prefTF', label= 'Prefix', text= mocapPrefix,
cw2=[130,180], cc= partial(  self.getAttrProc, '', mocapPrefix, 'prefix' ) )
    sufTF = cm.textFieldGrp( label= 'Suffix', text= mocapSuffix, cw2=[130,180],
cc= partial(  self.getAttrProc, '', mocapSuffix, 'suffix' ))

    cm.columnLayout(  )
    formCen = cm.formLayout(numberOfDivisions=300 )

    centreMocapFrame = cm.frameLayout( l='List of center joints' )
    for key, value in centreMocap.items():
        if cm.objExists(mocapPrefix + value + mocapSuffix)==0 and key not in si-
de:
            textStr = 'Not found: ' + value
            # centreMocap[key] =''
        elif len(cm.ls( '*' + key + suff[1] )) == 0 and key not in side:
            textStr = 'Not found: ' + key + suff[1]
            centreMocap[key] = key + suff[1]
        else:
```

```
                textStr = value
            charNameTF = cm.textFieldGrp( key + '_charNameTF', label=key, text=
textStr, cw2=[120,180], cc= partial(  self.getAttrProc, key, value, 'centreMocap', key
+ '_charNameTF' ))

        cm.formLayout( formCen, e=1, attachForm=[(centreMocapFrame, 'top', 5),
(centreMocapFrame, 'left', 10)], attachPosition=[(centreMocapFrame, 'right', 10,
300)])

        cm.setParent( '..' )
        cm.setParent( '..' )

        formSid = cm.formLayout(numberOfDivisions=300 )
        sideMocapFrame = cm.frameLayout( l='List of side joints' )
        for key, value in sideMocap.items():
            if cm.objExists( mocapPrefix + sideMocap["l_"] + value + mocapSuffix)==0
and key not in side:
                # sideMocap[key] ="
                textStr = 'Not found: ' + value
                # charNameTF = cm.textFieldGrp( label=key, text='Not found',
cw2=[120,180], cc= partial(  self.getAttrProc, key, value, 'sideMocap' ), ip=1, it= '    ')
            elif len(cm.ls( '*' + key + suff[1] )) == 0 and key not in side:
                textStr = 'Not found: ' + key + suff[1]
                centreMocap[key] = key + suff[1]
            else:
                textStr = value
            charNameTF = cm.textFieldGrp( key + '_charNameTF',label=key, text=
textStr, cw2=[120,180], cc= partial(  self.getAttrProc, key, value, 'sideMocap', key +
'_charNameTF' ) )

        cm.formLayout( formSid, e=1, attachForm=[(sideMocapFrame, 'top', 5), (si-
deMocapFrame, 'left', 10)], attachPosition=[(sideMocapFrame, 'right', 10, 300)])

        cm.button( label='Close', c= 'delUi("mocapWindow", "LopeButtonHUD", "axis-
SliderHUD", "shapeSliderHUD", "scaleSliderHUD" )' )

        cm.window( mocapMW, e=1, w=300.0, h=200)
        cm.showWindow()

        self.ikFkMocap()

    def ikFkMocap( self, *args ):

        if args==():
            if cm.radioCollection( self.collectionMocap, q=1, sl=1 ) == 'radioButtonIK':
                for f in cm.ls('*_flag_ctrl.ikFk'):cm.setAttr( f, 0 )
                for f in cm.ls('*_flag_ctrl.autoStretch'):cm.setAttr( f, 1 )
            else:
                for f in cm.ls('*_flag_ctrl.ikFk'):cm.setAttr( f, 1 )
```

```
            for f in cm.ls('*_flag_ctrl.autoStretch'):cm.setAttr( f, 0 )
        else:
            if args[0] == 'rbIK':pass
            else:pass
            if cm.window( "mocapWindow", ex=1 ):
                self.mocapWindow()

    def listBut( self, *args ):
        global mocapPrefix
        global mocapSuffix
        global centreMocapIK

        self.ikFkMocap()

        if cm.ls(rf=1)== []:
            if cm.objExists( mocapPrefix + centreMocapIK['body'] + mocapSuffix )==0
and cm.objExists( '*ground_ctrl.xn' ):
                makeStufMocap( )
                self.mocapWindow()
            else:
                cm.warning( "There's no a reference character in the scene.")
        elif len(cm.ls(rf=1)) > 1:
            cm.warning( "There's too much a reference in the scene: " + ',
'.join(cm.ls(rf=1)) )
        elif len(cm.ls(rf=1)) == 1:
            ref = cm.ls(rf=1)[0]
            try:
                cm.referenceQuery( cm.ls(rf=1)[0], n=1 )
            except:
                cm.warning( "Is this a reference character?: " + ref )
            else:
                if cm.referenceQuery( cm.ls(rf=1)[0], n=1 ) != None:
                    makeStufMocap( 'dictEdit' )
                    self.mocapWindow()
                else:
                    cm.warning( "There's a reference not load in the scene.")


    def bindBut( self, *args ):
        global centreMocap
        global sideMocap
        global mocapPrefix

        self.ikFkMocap()
        if cm.ls(rf=1)!= [] and cm.objExists( 'ground_ctrl' ) and cm.objExists( mocapPre-
fix + centreMocap['body'] + mocapSuffix ):
            cm.select('ground_ctrl')
            locHideAtr(workCtrSet(), 'dpS')
            cm.select(cl=1)
```

```
        if cm.objExists( suff[33][1:] ):
            cm.delete(suff[33][1:],'mocap_set')
        self.mocapDo( 'centreMocap', centreMocap, 1 )
        self.mocapDo( 'sideMocap', sideMocap, 1 )


    def report(self, buttonIndex, value ):
        global mode
        global charName
        global suffCust
        global custScale
        mode = buttonIndex
        charName = cm.textFieldGrp(self.charNameTF, q=1, tx =1)
        suffCust = cm.textFieldGrp(self.suffTF, q=1, tx =1)
        custScale = float(cm.textFieldGrp(self.scaleTF, q=1, tx =1))
        for tf in self.txFilAll:
            self.deleteMember( tf )
        self.txFilAll =[]
        if buttonIndex in custAtrPrefMode:
            txFild = custAtrPrefMode[buttonIndex]

            boolUi = [ ui for ui in txFild.items() if isinstance( ui[1], bool )]
            if boolUi !=[]:
                self.deleteMember( 'formboolUi' )
                self.formboolUi = cm.formLayout( 'formboolUi', numberOfDivisions=300,
parent = self.set )
                self.rowColumnLayoutBoolUi = cm.rowColumnLayout( numberOfCo-
lumns=3 )
                for key, value in boolUi:
                    self.addTxtFil = cm.checkBox( key, v=value, w= 98, h=20 )
                    self.txFilAll.append( self.addTxtFil )
                cm.formLayout( self.formboolUi, e=1, attachForm=[(self.rowColumn-
LayoutBoolUi, 'top', 3), (self.rowColumnLayoutBoolUi, 'left', 12), (self.rowColumn-
LayoutBoolUi, 'bottom', 0)] )

            for key, value in txFild.items():
                if isinstance( value, bool ) ==0:
                    self.addTxtFil = cm.textFieldGrp( l = key +':', text=value, cw2=[75,70],
vis=1, parent = self.rowColumnTxFil, h=22 )
                    self.txFilAll.append( self.addTxtFil )


    def makeItBut(self, *args):
        vpMB.makeButtVP('loadMake')
        global suffCust
        global custScale
        global mode
        mode = cm.radioButton( cm.radioCollection( self.collectionMake, q=1, sl=1 ),
q=1, l=1 )
```

```
        dicCurVar={}
        for tf in self.txFilAll:
            try:curSetLeb = str( cm.textFieldGrp(tf, q=1, l =1 )[:-1] )
            except:curSetLeb = str( cm.checkBox( tf, q=1, l =1 ) )
            try:
                curSetTex = int( cm.textFieldGrp(tf, q=1, tx =1 )  )
            except:
                try:curSetTex = str(cm.textFieldGrp(tf, q=1, tx =1 ))
                except:curSetTex = cm.checkBox( tf, q=1, v =1 )
            dicCurVar.update( { curSetLeb: curSetTex } )
            if mode == modeType[11] and curSetLeb =='controls' and curSetTex in [ 1, 2
]:
                mode = modeType[12]

        if len(cm.textFieldGrp(self.charNameTF, q=1, tx =1)) > 0:
            characterName( chNa = cm.textFieldGrp(self.charNameTF, q=1, tx =1) + '_'
+ stufName[0] )
        else:
            characterName( chNa = stufName[0] )
        suffCust = cm.textFieldGrp(self.suffTF, q=1, tx =1)
        custScale = float(cm.textFieldGrp(self.scaleTF, q=1, tx =1))
        return dicCurVar


    def buildBut(self,*args):
        allSkinObj = cm.ls( sl=1, ni=1, ap=1, dag=1, typ='mesh' )

        if len( cm.ls( '*'+ suff[4] + suff[2] )) !=0:
            global GlobScale
            listScale = [ cm.getAttr ( obj +'.scaleX' ) for obj in cm.ls( '*'+ suff[4] + suff[2] ) ]
            GlobScale = sum(listScale) / float(len(listScale))

        vpMB.makeButtVP()
        fa3d(0,'b')

        if cm.objExists( '*_setUp_grp.xn' )==0:
            return
        if 'nSetUpGrp' not in locals():
            nSetUpGrp = ''

        global mode
        # global charName
        global suffCust
        global custScale
        mode = 'build'
        # charName = cm.textFieldGrp(self.charNameTF, q=1, tx =1)
        suffCust = ''#cm.textFieldGrp(self.suffTF, q=1, tx =1)
        custScale = float(cm.textFieldGrp(self.scaleTF, q=1, tx =1))
```

```
    self.GET_setUp_grp(nSetUpGrp)
    self.extortPref()

    if cm.objExists( '*char.sb' )==0:
        chN = cm.ls('*char') [0]
        cm.addAttr( chN, ln='showButton', sn='sb', at= 'bool' )
        cm.setAttr( chN + '.sb', cb= 1 )
    vpMB.showButtScrJob( cm.ls('*char')[0] )
    cm.setAttr( cm.ls('*char')[0] + '.showButton', 1 )

    cm.select( allSkinObj )


def extortPref( self, obj = [] ):
    if obj == []:
        sel = cm.getAttr('load.setUpGro').split(';')
    else:
        sel = obj
    if  sel[0] != '':
        [ cm.setAttr( '%s.v'%con, 1 ) for con in sel ]
        slis =[]
        for s in sel:
            s = s.replace( "_setUp_grp", "_START_set" )
            slis.append(s)
        cm.select( sel )
        cm.select( slis, ne=1, add=1 )
        tmpMa = dumpProc( 'cod', 'tmp', '', '')
        tmpZp = dumpProc( 'cod', 'tmp', '', '')
        cm.file( tmpMa, es=1, de=0, typ='mayaAscii', f=1 )
        cm.delete()
        dumpProc( 'zip', 'in', tmpMa, 'fil', tmpZp, 'fil', 'expPref.ma' )
        dumpProc( 'cod', 'in', tmpZp, 'fil', 'load.setUp', 'atr' )
        tmpMa = dumpProc( 'cod', 'tmD', tmpMa, 'fil' )
        tmpZp = dumpProc( 'cod', 'tmD', tmpZp, 'fil' )


def importPref(self,*args):
    if cm.objExists( 'load.setUp' ) and cm.getAttr('load.setUp') not in [ '', None ]:
        sel = cm.getAttr('load.setUp')
        zFile = dumpProc( 'cod', 'out', 'load.setUp', 'atr' )
        zFile = dumpProc( 'cod', 'tmp', zFile, 'srt' )
        sPa = zFile.split( '\\' )[:-1]
        sPa = '\\'.join(sPa)
        dumpProc( 'zip', 'out', zFile, 'fil', sPa, 'fil' )
        cm.file( sPa + '/expPref.ma', i=1, iv=1, typ='mayaAscii', f=1 )
        dumpProc( 'cod', 'tmD', zFile, 'fil' )
        dumpProc( 'cod', 'tmD', sPa + '/expPref.ma', 'fil' )
        cm.setAttr( 'load.setUp', '', typ='string' )
```

```
        if cm.objExists( '*char.xn' ):
            charName = cm.getAttr( '*char.xn', asString=1 )
            if cm.objExists( '*char.showButton' ):
                cm.setAttr( charName + '.showButton', 1 )
    elif cm.objExists( 'load.setUp' ):
        self.GET_setUp_grp(nSetUpGrp)
        self.extortPref()

        if cm.objExists( '*char.xn' ):
            charName = cm.getAttr( '*char.xn', asString=1 )
            if cm.objExists( '*char.showButton'):
                cm.setAttr( charName + '.showButton', 0 )
    workingState(1)


def GET_setUp_grp(self, nSetUpGrp):
    for obj in cm.ls( '*_setUp_grp', type='transform' ):
        if obj not in self.list_setUp_grp:
            self.list_setUp_grp.append(obj)

    if nSetUpGrp not in self.list_setUp_grp:
        self.list_setUp_grp.append(nSetUpGrp)

    copylist_setUp_grp = self.list_setUp_grp[:]
    for obj in copylist_setUp_grp:
        if cm.objExists( obj )==0:
            self.list_setUp_grp.remove(obj)

    f = ';'.join( self.list_setUp_grp )
    cm.setAttr( 'load.setUpGro', f, type="string" )
```