#Heather Falconsong Howard
#Created Sept 4, 2011
#Requirements- Maya 2011

########## How to use the program: ##########
#	1) Place featherGenerator.ui into the Maya 2011 scripts directory.
#	2) Open WingBase.ma.
#	3) To run the rig program, copy and paste the FeatherGenerator.py text into the script editor
#		and run it; this will open up the graphic user interface (GUI) for the program.
#	4) In the GUI input how many feathers for each feather group you want (must be 3 or more) and
#		chose a side (right or left). If you are not familiar with feather groups, a wing diagram
#		has been provided.
#	5) Click Generate Feathers to generate the feathers.
#	6) Click Generate Motion System to generate the motion system. Motion system cannot be generated
#		before feathers.
#	7) Repeat steps 2-4 for the opposite side.
#	8) Close feather creator.
#	9) Copy and paste the AttachWingmeat.py into the script editor and run it. This will parent the
#		stand-in geometry of the wing and body to the control system.

########## What the program requires in order to create a wing: ##########
Feather geometry: L_PrimaryBase, L_SecondaryBase, L_PrimaryCovertsBase, L_SecondaryCovertsBase,
#		L_MedianCovertsBase, L_AlulaBase, R_PrimaryBase, R_SecondaryBase,
R_PrimaryCovertsBase,
#		R_SecondaryCovertsBase, R_MedianCovertsBase, R_AlulaBase, TertialBase.
#		-- These are a right and left of all feathers for each group, centered with the tip
#			of the quill at the origin and pointing down the positive Z axis. Only
the
#			tertial doesn't need a right and a left due to it being nearly symmetrical.
Locators: L_Wing_1, L_Wing_2, L_Wing_3, L_Wing_4, R_Wing_1, R_Wing_2, R_Wing_3, R_Wing_4.
#		-- These correspond to the shoulder joint, elbow joint, wrist joint, and the
#			tip of the hand (not tip of the feathers) respectively.
Controls: CONBASE, BoxConBase, ArrowConBase, MainConBase,MoveA MoveAllConBase, and SubConBase
#		with parented controls Wing_Flex_1 and Wing_Flex_2.
Blendshapes (OPTIONAL): L_Primary_BLND, L_Secondary_BLND, R_Primary_BLND, R_Secondary_BLND.
#		-- Blendshapes must be derived from the feather geometry (for example
#			L_Primary_BLND must be derived from L_PrimaryBase) and must
NOT have
#			transforms frozen. Freezing transforms breaks the blendshape, causing
#			it to shift position origin when the blendshape is keyed in.

########## Notes: ##########
- If the program has a problem, check the script editor for errors; if any of the needed parts are
missing (controls or geometry) or already created, the program should give feedback in the
#	script editor stating what is wrong.
- If you use your own geometry for the feathers, please note that they should be proportional to
#	each other. Giving the program a primary that is much smaller than a secondary will result in
#	primaries that are small in the wing. This also applies to controls.
- Names are specific, typos or mus-capitalizations will cause problems.
- Blendshapes are optional. If no blendshapes are given, then the program will simply not add
#	flexing to the feathers.
- The file contains some stand-in geometry for the wing. Since the thesis only deals with the
#	major feather groups and not the lesser secondary coverts, I added stand-in geometry for those
#	areas so that the wing was easier to visualize as a full wing. I added an extra script
#	(AttachWingmeat.py) that is not part of the thesis but is simply a time saver. This program
#	will take the stand-in geometry for the wing and parent it to the correct joints. I am not
#	looking for critique on this script.

########## Rig Controls Notes: ##########

```
# -Rig has two layers that can be toggled on and off, one for the stand-in Poly shapes, another
#	for the feathers.
# -Sub controls are hidden under main controls: Middle, Tip and End Feathers controls have variables
#	called Group Controls and Flex Controls. Turn these on to show the sub controls for that group.
#	Group controls control individual feather groups, Flex controls control flex of the
#	Remiges (flight feathers).
# -Shoulder control has a control for IK/FK switching. (This is a switch, not an IK FK Blend)
# -Wingfold control will ONLY work in FK mode

import maya.cmds as m
import math

#Main function to start the program.
def featherGenerator():

	#set the scripts directory so we can load our ui file
	scriptsDirectory = m.internalVar(userScriptDir=True)
	#load the ui file
	featherGeneratorWindow = m.loadUI(f=scriptsDirectory+'/FeatherGenerator.ui')
	#show the window
	m.showWindow (featherGeneratorWindow)

#Default number of feathers
numPrimaries = 10.0
numSecondaries = 13.0
numPrimaryCoverts = 10.0
numSecondaryCoverts = 13.0
numMedianCoverts = 13.0
numAlulas = 4
numTertials = 3

#Function to run at startup- It creates the group structure for the rig
def runAtStartup():
	#Create group hierarchy
	m.group( em=True, name = 'Controls' )
	m.group( em=True, name = 'Joints' )
	m.group( em=True, name = 'NoXForm' )
	m.group( em=True, name = 'FeatherGroups' )
	m.group( em=True, name = 'Wings' )
	m.group( em=True, name = 'XForm' )

	m.duplicate('MainConBase', n= 'MainWing_CON')
	m.showHidden( 'MainWing_CON' )
	m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

	m.parent( 'MainWing_CON',  'Wings' )
	m.parent('FeatherGroups',  'XForm' )
	m.parent('Joints',  'XForm' )
	m.parent('Controls',  'XForm' )
	m.parent('NoXForm',  'Wings' )
	m.parent('XForm',  'MainWing_CON' )

	m.select(cl=True)

	#Create a layer for the feathers
	m.createDisplayLayer( noRecurse=True, name='FeathersLayer' )

	m.select(cl=True)
```

```
#Function that generates primary feathers
def buildPrimaries():
	#Calculate distance between both points
	DistX = (Loc4Coord[0] - Loc3Coord[0])
	DistY = (Loc4Coord[1] - Loc3Coord[1])
	DistZ = (Loc4Coord[2] - Loc3Coord[2])

	#Divide by number of feathers to get the distance between each feather
	DistBetweenX = DistX/(numPrimaries - 1)
	DistBetweenZ = DistZ/(numPrimaries - 1)

	i = 1 #the step size for the function, so the function goes from 1->#	of feathers
	placementX = Loc3Coord[0]
	placementZ = Loc3Coord[2]

	numBaseFeathers= 10.0 #Number of feathers used to create the static function
	newStep = (numBaseFeathers/numPrimaries) #New step size based on the original number of feathers,
                                                 #Divide the base number
of feathers by the new amount to
                                                 #get the number to scale
by for the function
	input= 1 #start function input at one then incriment it by the new step on each iteration

	#Duplicate feathers and move them into position
	if (side == "R"):
		while (i < (numPrimaries+1)):
			#duplicate from the base feather
			m.duplicate( side + '_PrimaryBase', n= side + '_Primary_' + repr(i))
			m.select(side + '_Primary_' + repr(i))
			m.showHidden( side + '_Primary_' + repr(i) )
			#Add it to the feathers layer
			m.editDisplayLayerMembers( 'FeathersLayer', side + '_Primary_' + repr(i))
			#move and rotate it into the correct place
			m.move( placementX, 0, placementZ, ws=True )
			m.rotate( 0, 0, 5.0, r=True )
			placementX = placementX + DistBetweenX
			placementZ = placementZ + DistBetweenZ

			#Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
			#the function blows up without that much precision)
			rotationVar = (7.56 * input) + 25.0
			scaleVar = 0.688533 + ( 0.242191 * input) - (0.153958 * math.pow(input,2)) + (0.041241 *
math.pow(input,3)) - (0.00439272 * math.pow(input,4)) + (0.000156154 * math.pow(input,5))

			tempI = int(i) #Feather name needs an integer value so temporarally cast it to an int
			m.select(side + '_Primary_' + repr(tempI)) #Select the proper feather
			m.rotate( 0, rotationVar, 0, r=True ) #Rotate it
			m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

			input = input + newStep; #Incriment the step amount
			i= i+1 #incriment the feather number

	elif (side == "L"):
		while (i < (numPrimaries+1)):
			#duplicate from the base feather
			m.duplicate( side + '_PrimaryBase', n= side + '_Primary_' + repr(i))
			m.select(side + '_Primary_' + repr(i))
			m.showHidden( side + '_Primary_' + repr(i) )
			#Add it to the feathers layer
```

```
                        m.editDisplayLayerMembers( 'FeathersLayer', side + '_Primary_' + repr(i))
                        #move and rotate it into the correct place
                        m.move( placementX, 0, placementZ, ws=True )
                        m.rotate( 0, 0, -5.0, r=True )
                        placementX = placementX + DistBetweenX
                        placementZ = placementZ + DistBetweenZ

                        rotationVar = (7.56 * input) + 25.0
                        #See note in version 13
                        scaleVar = 0.688533 + ( 0.242191 * input) - (0.153958 * math.pow(input,2)) + (0.041241 *
math.pow(input,3)) - (0.00439272 * math.pow(input,4)) + (0.000156154 * math.pow(input,5))

                        tempI = int(i) #Feather name needs an integer value so temporarally cast it to an int
                        m.select(side + '_Primary_' + repr(tempI)) #Select the proper feather
                        m.rotate( 0, -rotationVar, 0, r=True ) #Rotate it
                        m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

                        input = input + newStep
                        i= i+1

        m.select(cl=True)

#Function that generates Secondary feathers
def buildSecondaries():
        #Calculate distance between both points
        DistX = (Loc3Coord[0] - Loc2Coord[0])
        DistY = (Loc3Coord[1] - Loc2Coord[1])
        DistZ = (Loc3Coord[2] - Loc2Coord[2])

        #Divide by number of feathers to get the distance between each feather
        DistBetweenX = DistX/(numSecondaries-1)
        DistBetweenZ = DistZ/(numSecondaries-1)

        i = 1 #the step size for the function, so the function goes from 1->#of feathers
        placementX = Loc3Coord[0]
        placementZ = Loc3Coord[2]

        numBaseFeathers= 13.0 #Number of feathers used to create the static function
        newStep = (numBaseFeathers/numSecondaries) #New step size based on the original number of feathers,
                                                                                #Divide the base number
of feathers by the new amount to
                                                                                #get the number to scale
by for the function
        input= 1 #start function input at one then incriment it by the new step on each iteration

        #Duplicate feathers and move them into position
        if (side == "R"):
                while (i < numSecondaries+1):
                        #duplicate from the base feather
                        m.duplicate( side + '_SecondaryBase', n= side + '_Secondary_' + repr(i))
                        m.select( side + '_Secondary_'+ repr(i))
                        m.showHidden( side + '_Secondary_' + repr(i) )
                        #Add it to the feathers layer
                        m.editDisplayLayerMembers( 'FeathersLayer', side + '_Secondary_' + repr(i))
                        #move and rotate it into the correct place
                        m.move( placementX, 0, placementZ, ws=True )
                        m.rotate( 0, 0, 5.0, r=True )
                        placementX = placementX - DistBetweenX
                        placementZ = placementZ - DistBetweenZ
```

```
                        #Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
                        #the function blows up without that much precision)
                        rotationVar = (0.1124 * math.pow(input,2)) - (5.5425 * input) + 28.121
                        scaleVar = -(0.0042 * math.pow(input,2)) + (0.03242 * input) + 0.9941

                        m.select(side + '_Secondary_' + repr(i)) #Select the proper feather
                        m.rotate( 0, rotationVar, 0, r=True ) #Rotate it
                        m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

                        input = input + newStep; #Incriment the step amount
                        i= i+1     #Incriment feathers number

        elif (side == "L"):
                while (i < numSecondaries+1):
                        #duplicate from the base feather
                        m.duplicate( side + '_SecondaryBase', n= side + '_Secondary_' + repr(i))
                        m.select( side + '_Secondary_'+ repr(i))
                        m.showHidden( side + '_Secondary_' + repr(i) )
                        #Add it to the feathers layer
                        m.editDisplayLayerMembers( 'FeathersLayer', side + '_Secondary_' + repr(i))
                        #move and rotate it into the correct place
                        m.move( placementX, 0, placementZ, ws=True )
                        m.rotate( 0, 0, -5.0, r=True )
                        placementX = placementX - DistBetweenX
                        placementZ = placementZ - DistBetweenZ

                        #Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
                        #the function blows up without that much precision)
                        rotationVar = (0.1124 * math.pow(input,2)) - (5.5425 * input) + 28.121
                        scaleVar = -(0.0042 * math.pow(input,2)) + (0.03242 * input) + 0.9941

                        m.select(side + '_Secondary_' + repr(i)) #Select the proper feather
                        m.rotate( 0, -rotationVar, 0, r=True ) #Rotate it
                        m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

                        input = input + newStep #Incriment the step amount
                        i= i+1 #incriment the feather number

        #Add in the tertial feathers
        #percentage of the length of bone for the distance of the elbow/tertials
        DistElbowX = DistX * 0.05
        DistElbowZ = DistZ * 0.05
        tempVar = Loc1Coord[0]

        #Divide by number of feathers to get the distance between each feather
        DistBetweenXTert = DistElbowX/numTertials
        DistBetweenZTert = DistElbowZ/numTertials

        j = int(numSecondaries + 1) #Cast it to an int

        placementTertX = Loc2Coord[0] - DistBetweenXTert
        placementTertZ = Loc2Coord[2] + DistBetweenZTert

        numBaseTertFeathers= 3.0 #Number of feathers used to create the static function
        newStep = (numBaseTertFeathers/numTertials) #New step size based on the original number of feathers,
                                                                                        #Divide the base
number of feathers by the new amount to
                                                                                        #get the number
```

```
to scale by for the function
	input2= 1 #start function input at one then incriment it by the new step on each iteration

	if (side == "R"):
		while ( j < (numSecondaries + numTertials +1) ):
			#duplicate from the base feather
			m.duplicate( side + '_SecondaryBase', n= side + '_Secondary_' + repr(j))
			m.select(side + '_Secondary_' + repr(j))
			m.showHidden( side + '_Secondary_' + repr(j))
			#Add it to the feathers layer
			m.editDisplayLayerMembers( 'FeathersLayer', side + '_Secondary_' + repr(j))
			#move and rotate it into the correct place
			m.move( placementTertX, 0, placementTertZ, ws=True )
			m.rotate( 0, 0, -5.0, r=True )
			placementTertX = placementTertX - DistBetweenXTert
			placementTertZ = placementTertZ + DistBetweenZTert

			#Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
			#the function blows up without that much precision)
			rotationVar = (3.7495 * input2) + 23.3233333
			scaleVar = -(0.0845 * input2) + 0.72233333

			m.select(side + '_Secondary_' + repr(j)) #Select the proper feather
			m.rotate( 0, -rotationVar, 0, r=True ) #Rotate it
			m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

			input2 = input2 + newStep		#Incriment the step amount
			j = j + 1 #incriment the feather number

	elif (side == "L"):
		while ( j < (numSecondaries + numTertials +1) ):
			#duplicate from the base feather
			m.duplicate( side + '_SecondaryBase', n= side + '_Secondary_' + repr(j))
			m.select(side + '_Secondary_' + repr(j))
			m.showHidden( side + '_Secondary_' + repr(j))
			#Add it to the feathers layer
			m.editDisplayLayerMembers( 'FeathersLayer', side + '_Secondary_' + repr(j))
			#move and rotate it into the correct place
			m.move( placementTertX, 0, placementTertZ, ws=True )
			m.rotate( 0, 0, -5.0, r=True )
			placementTertX = placementTertX - DistBetweenXTert
			placementTertZ = placementTertZ + DistBetweenZTert

			#Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
			#the function blows up without that much precision)
			rotationVar = (3.7495 * input2) + 23.3233333
			scaleVar = -(0.0845 * input2) + 0.72233333

			m.select(side + '_Secondary_' + repr(j)) #Select the proper feather
			m.rotate( 0, rotationVar, 0, r=True ) #Rotate it
			m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

			input2 = input2 + newStep		#Incriment the step amount
			j = j + 1 #incriment the feather number


	m.select(cl=True)

#Function that generates primary covert feathers
```

```
def buildPrimaryCoverts():
	#Calculate distance between both points
	DistX = (Loc4Coord[0] - Loc3Coord[0])
	DistY = (Loc4Coord[1] - Loc3Coord[1])
	DistZ = (Loc4Coord[2] - Loc3Coord[2])

	#Divide by number of feathers to get the distance between each feather
	DistBetweenX = DistX/(numPrimaryCoverts - 1)
	DistBetweenY = 0
	DistBetweenZ = DistZ/(numPrimaryCoverts - 1)

	i = 1 #the step size for the function, so the function goes from 1->#	of feathers
	placementX = Loc3Coord[0]
	placementZ = Loc3Coord[2]

	numBaseFeathers= 10.0 #Number of feathers I used to create the static function
	newStep = (numBaseFeathers/numPrimaryCoverts) #Divide the base number of feathers by the new amount to get the number to scale by for the function
	input= 1 #start function input at one then incriment it by the new step on each iteration for the function  in order to scale it based on the
				#new number of feathers in relation to the base number of feathers

	#Duplicate feathers and move them into position
	if (side == "R"):
		while (i < (numPrimaryCoverts+1)):
			#duplicate from the base feather
			m.duplicate( side + '_PrimaryCovertsBase', n= side + '_PrimaryCoverts_' + repr(i))
			m.select(side + '_PrimaryCoverts_' + repr(i))
			m.showHidden( side + '_PrimaryCoverts_' + repr(i) )
			#Add it to the feathers layer
			m.editDisplayLayerMembers( 'FeathersLayer', side + '_PrimaryCoverts_' + repr(i))
			#move and rotate it into the correct place
			m.move( placementX, 0.2, placementZ, ws=True )
			m.rotate( 0, 0, 5.0, r=True )
			placementX = placementX + DistBetweenX
			placementZ = placementZ + DistBetweenZ

			#Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
			#the function blows up without that much precision)
			rotationVar = (7.56 * input) + 25.0
			scaleVar = 0.688533 + ( 0.242191 * input) - (0.153958 * math.pow(input,2)) + (0.041241 * math.pow(input,3)) - (0.00439272 * math.pow(input,4)) + (0.000156154 * math.pow(input,5))
			scaleVar = (2.0/3.0) * scaleVar #Scale it down by 2/3 because the coverts are smaller than the secondaries

			tempI = int(i) #Feather name needs an integer value so temporarally cast it to an int
			m.select(side + '_PrimaryCoverts_' + repr(tempI)) #Select the proper feather
			m.rotate( 0, rotationVar, 0, r=True ) #Rotate it
			m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

			input = input + newStep #Incriment the step amount
			i= i+1 #incriment the feather number

	elif (side == "L"):
		while (i < (numPrimaryCoverts+1)):
			#duplicate from the base feather
			m.duplicate( side + '_PrimaryCovertsBase', n= side + '_PrimaryCoverts_' + repr(i))
			m.select(side + '_PrimaryCoverts_' + repr(i))
			m.showHidden( side + '_PrimaryCoverts_' + repr(i) )
```

```
                        #Add it to the feathers layer
                        m.editDisplayLayerMembers( 'FeathersLayer', side + '_PrimaryCoverts_' + repr(i))
                        #move and rotate it into the correct place
                        m.move( placementX, 0.2, placementZ, ws=True )
                        m.rotate( 0, 0, -5.0, r=True )
                        placementX = placementX + DistBetweenX
                        placementZ = placementZ + DistBetweenZ

                        #Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
                        #the function blows up without that much precision)
                        rotationVar = (7.56 * input) + 25.0
                        scaleVar = 0.688533 + ( 0.242191 * input) - (0.153958 * math.pow(input,2)) + (0.041241 *
math.pow(input,3)) - (0.00439272 * math.pow(input,4)) + (0.000156154 * math.pow(input,5))
                        scaleVar = (2.0/3.0) * scaleVar  #Scale it down by 2/3 because the coverts are smaller than the
secondaries

                        tempI = int(i) #Feather name needs an integer value so temporarally cast it to an int
                        m.select(side + '_PrimaryCoverts_' + repr(tempI)) #Select the proper feather
                        m.rotate( 0, -rotationVar, 0, r=True ) #Rotate it
                        m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

                        input = input + newStep #Incriment the step amount
                        i= i+1 #incriment the feather number

        m.select(cl=True)

#Function that generates Secondary covert feathers
def buildSecondaryCoverts():
        #Calculate distance between both points
        DistX = (Loc3Coord[0] - Loc2Coord[0])
        DistY = (Loc3Coord[1] - Loc2Coord[1])
        DistZ = (Loc3Coord[2] - Loc2Coord[2])

        #Divide by number of feathers to get the distance between each feather
        DistBetweenX = DistX/(numSecondaryCoverts-1)
        DistBetweenZ = DistZ/(numSecondaryCoverts-1)

        i = 1 #the step size for the function, so the function goes from 1->#  of feathers
        placementX = Loc3Coord[0]
        placementZ = Loc3Coord[2]

        numBaseFeathers= 13.0 #Number of feathers used to create the static function
        newStep = (numBaseFeathers/numSecondaryCoverts) #New step size based on the original number of feathers,
                                                                                                #Divide
the base number of feathers by the new amount to
                                                                                                #get the
number to scale by for the function
        input= 1 #start function input at one then incriment it by the new step on each iteration

        #Duplicate feathers and move them into position
        if (side == "R"):
                while (i < numSecondaryCoverts+1):
                        #duplicate from the base feather
                        m.duplicate( side + '_SecondaryCovertsBase', n= side + '_SecondaryCoverts_' + repr(i))
                        m.select( side + '_SecondaryCoverts_' + repr(i))
                        m.showHidden( side + '_SecondaryCoverts_' + repr(i) )
                        #Add it to the feathers layer
                        m.editDisplayLayerMembers( 'FeathersLayer', side + '_SecondaryCoverts_' + repr(i))
                        #move and rotate it into the correct place
```

```
                        m.move( placementX, 0.2, placementZ, ws=True )
                        m.rotate( -1.0, 0, 5.0, r=True )
                        placementX = placementX - DistBetweenX
                        placementZ = placementZ - DistBetweenZ

                        #Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
                        #the function blows up without that much precision)
                        rotationVar = (0.1124 * math.pow(input,2)) - (5.5425 * input) + 28.121
                        scaleVar = -(0.0042 * math.pow(input,2)) + (0.03242 * input) + 0.9941
                        scaleVar = (2.0/3.0) * scaleVar #Scale it down by 2/3 because the coverts are smaller than the
secondaries

                        m.select(side + '_SecondaryCoverts_' + repr(i)) #Select the proper feather
                        m.rotate( 0, rotationVar, 0, r=True ) #Rotate it
                        m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

                        input = input + newStep #Incriment the step amount
                        i= i+1 #incriment the feather number

        elif (side == "L"):
                while (i < numSecondaryCoverts+1):
                        #duplicate from the base feather
                        m.duplicate( side + '_SecondaryCovertsBase', n= side + '_SecondaryCoverts_' + repr(i))
                        m.select( side + '_SecondaryCoverts_'+ repr(i))
                        m.showHidden( side + '_SecondaryCoverts_' + repr(i) )
                        #Add it to the feathers layer
                        m.editDisplayLayerMembers( 'FeathersLayer', side + '_SecondaryCoverts_' + repr(i))
                        #move and rotate it into the correct place
                        m.move( placementX, 0.2, placementZ, ws=True )
                        m.rotate( -1.0, 0, -5.0, r=True )
                        placementX = placementX - DistBetweenX
                        placementZ = placementZ - DistBetweenZ

                        #Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
                        #the function blows up without that much precision)
                        rotationVar = (0.1124 * math.pow(input,2)) - (5.5425 * input) + 28.121
                        scaleVar = -(0.0042 * math.pow(input,2)) + (0.03242 * input) + 0.9941
                        scaleVar = (2.0/3.0) * scaleVar #Scale it down by 2/3 because the coverts are smaller than the
secondaries

                        m.select(side + '_SecondaryCoverts_' + repr(i)) #Select the proper feather
                        m.rotate( 0, -rotationVar, 0, r=True ) #Rotate it
                        m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

                        input = input + newStep #Incriment the step amount
                        i= i+1 #incriment the feather number

        m.select(cl=True)

#Function that generates median covert feathers
def buildMedianCoverts():
        #Find point 1's XYZ coord
        Loc1Coord = m.xform( side + '_Wing_2', t=True, q=True, ws=True)

        #Find point 2's XYZ coord
        Loc2Coord = m.xform( side + '_Wing_3', t=True, q=True, ws=True)

        #Calculate distance between both points
        DistX = (Loc2Coord[0] - Loc1Coord[0])
```

```
	DistY = (Loc2Coord[1] - Loc1Coord[1])
	DistZ = (Loc2Coord[2] - Loc1Coord[2])

	#Divide by number of feathers to get the distance between each feather
	DistBetweenX = DistX/(numMedianCoverts-1)
	DistBetweenZ = DistZ/(numMedianCoverts-1)

	i = 1 #the step size for the function, so the function goes from 1->#	of feathers
	placementX = Loc2Coord[0]
	placementZ = Loc2Coord[2]

	numBaseFeathers= 13.0 #Number of feathers used to create the static function
	newStep = (numBaseFeathers/numMedianCoverts) #New step size based on the original number of feathers,
                                                     #Divide the base
number of feathers by the new amount to
                                                     #get the number
to scale by for the function
	input= 1 #start function input at one then incriment it by the new step on each iteration

	#Duplicate feathers and move them into position
	if (side == "R"):
		while (i < numMedianCoverts+1):
			#duplicate from the base feather
			m.duplicate( side + '_MedianCovertsBase', n= side + '_MedianCoverts_' + repr(i))
			m.select( side + '_MedianCoverts_' + repr(i))
			m.showHidden( side + '_MedianCoverts_' + repr(i) )
			#Add it to the feathers layer
			m.editDisplayLayerMembers( 'FeathersLayer', side + '_MedianCoverts_' + repr(i))
			#move and rotate it into the correct place
			m.move( placementX, 0.4, placementZ - 0.4, ws=True )
			m.rotate( -2.0, 0, 5.0, r=True )
			placementX = placementX - DistBetweenX
			placementZ = placementZ - DistBetweenZ

			#Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
			#the function blows up without that much precision)
			rotationVar = -(0.0994 * math.pow(input,2)) - (4.194 * input) + 52.228
			scaleVar = -(0.0083 * math.pow(input,2)) + (0.1161 * input) + 0.4887

			m.select(side + '_MedianCoverts_' + repr(i)) #Select the proper feather
			m.rotate( 0, rotationVar, 0, r=True ) #Rotate it
			m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

			input = input + newStep #Incriment the step amount
			i= i+1 #incriment the feather number

	elif (side == "L"):
		while (i < numMedianCoverts+1):
			#duplicate from the base feather
			m.duplicate( side + '_MedianCovertsBase', n= side + '_MedianCoverts_' + repr(i))
			m.select( side + '_MedianCoverts_'+ repr(i))
			m.showHidden( side + '_MedianCoverts_' + repr(i) )
			#Add it to the feathers layer
			m.editDisplayLayerMembers( 'FeathersLayer', side + '_MedianCoverts_' + repr(i))
			#move and rotate it into the correct place
			m.move( placementX, 0.4, placementZ - 0.4, ws=True )
			m.rotate( -2.0, 0, -5.0, r=True )
			placementX = placementX - DistBetweenX
			placementZ = placementZ - DistBetweenZ
```

```
                        #Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
                        #the function blows up without that much precision)
                        rotationVar = -(0.0994 * math.pow(input,2)) - (4.194 * input) + 52.228
                        scaleVar = -(0.0083 * math.pow(input,2)) + (0.1161 * input) + 0.4887

                        m.select(side + '_MedianCoverts_' + repr(i)) #Select the proper feather
                        m.rotate( 0, -rotationVar, 0, r=True ) #Rotate it
                        m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

                        input = input + newStep #Incriment the step amount
                        i= i+1 #incriment the feather number

        m.select(cl=True)

#Function that generates Alula feathers
def buildAlulas():
        #Calculate distance between both points
        DistX = (Loc4Coord[0] - Loc3Coord[0])
        DistY = (Loc4Coord[1] - Loc3Coord[1])
        DistZ = (Loc4Coord[2] - Loc3Coord[2])

        #Distance for the alula is 1/3 the distance between the original Points
        DistX = (1.0/3.0) * DistX
        DistY = (1.0/3.0) * DistY
        DistZ = (1.0/3.0) * DistZ

        #Divide by number of feathers to get the distance between each feather
        DistBetweenX = DistX/(numAlulas - 1)
        DistBetweenZ = DistZ/(numAlulas - 1)

        i = 1 #the step size for the function, so the function goes from 1->#  of feathers
        placementX = Loc3Coord[0]
        placementZ = Loc3Coord[2]

        #Duplicate feathers and move them into position
        if (side == "R"):
                while (i < (numAlulas+1)):
                        #duplicate from the base feather
                        m.duplicate( side + '_AlulaBase', n= side + '_Alula_' + repr(i))
                        m.select(side + '_Alula_' + repr(i))
                        m.showHidden( side + '_Alula_' + repr(i) )
                        #Add it to the feathers layer
                        m.editDisplayLayerMembers( 'FeathersLayer', side + '_Alula_' + repr(i))
                        #move and rotate it into the correct place
                        m.move( placementX, 0.4, placementZ - 0.4, ws=True )
                        m.rotate( 0, 0, 5.0, r=True )
                        placementX = placementX + DistBetweenX
                        placementZ = placementZ + DistBetweenZ*2

                        #Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
                        #the function blows up without that much precision)
                        rotationVar = (6.9112 * i) + 74.102
                        scaleVar= (0.1159 * i) + 0.5365

                        tempI = int(i) #Feather name needs an integer value so temporarally cast it to an int
                        m.select(side + '_Alula_' + repr(tempI)) #Select the proper feather
                        m.rotate( 0, rotationVar, 0, r=True ) #Rotate it
                        m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it
```

```
                i= i+1

        elif (side == "L"):
                while (i < (numAlulas+1)):
                        #duplicate from the base feather
                        m.duplicate( side + '_AlulaBase', n= side + '_Alula_' + repr(i))
                        m.select(side + '_Alula_' + repr(i))
                        m.showHidden( side + '_Alula_' + repr(i) )
                        #Add it to the feathers layer
                        m.editDisplayLayerMembers( 'FeathersLayer', side + '_Alula_' + repr(i))
                        #move and rotate it into the correct place
                        m.move( placementX, 0.4, placementZ - 0.4, ws=True )
                        m.rotate( 0, 0, -5.0, r=True )
                        placementX = placementX + DistBetweenX
                        placementZ = placementZ + DistBetweenZ*2

                        #Rotation and scale functions (Yes, there needs to be that many significant digits in scale,
                        #the function blows up without that much precision)
                        rotationVar = (6.9112 * i) + 74.102
                        scaleVar= (0.1159 * i) + 0.5365

                        tempI = int(i) #Feather name needs an integer value so temporarally cast it to an int
                        m.select(side + '_Alula_' + repr(tempI)) #Select the proper feather
                        m.rotate( 0, -rotationVar, 0, r=True ) #Rotate it
                        m.scale( scaleVar, scaleVar, scaleVar, r=True) #Scale it

                        i= i+1

        m.select(cl=True)

#Create controls
def controlsSetup():

        #Create Shoulder Control and move to shoulder position
        m.duplicate('MoveAllConBase', n= side + '_Shoulder_CON')
        m.showHidden( side + '_Shoulder_CON' )
        m.group( em=True, name=side + '_Shoulder_Con_GRP' )
        m.parent( side + '_Shoulder_CON', side + '_Shoulder_Con_GRP' )
        m.select( side + '_Shoulder_Con_GRP')
        m.move( Loc1Coord[0], Loc1Coord[1], Loc1Coord[2], ws=True )
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        #Create Wrist Control and move to wrist position
        m.duplicate('BoxConBase', n= side + '_Wrist_CON')
        m.showHidden( side + '_Wrist_CON' )
        m.group( em=True, name=side + '_Wrist_Con_GRP' )
        m.parent( side + '_Wrist_CON', side + '_Wrist_Con_GRP' )
        m.select( side + '_Wrist_Con_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        #Create Pole Vecor Control and move into position behind the wing
        m.polyCone(n=side + '_Wing_Pole_CON', r=1, h=2)
        DistPole = abs(Loc4Coord[0]) + 10 #set pole control distance to be the distance of the size of the wing to
accomidate different size wings, feathers
        m.move(m.getAttr(side+'_Wing_2.translateX'), m.getAttr(side+'_Wing_2.translateY'),
m.getAttr(side+'_Wing_2.translateZ' ) + DistPole, ws=True)
        m.rotate( '90deg', 0, 0, r=True )
```

```
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        # FEATHER CONTROLS #
        ### Primaries ###
        #Create second Primaries control and move into position
        m.duplicate('ArrowConBase', n= side + '_Primary_Con_2')
        m.showHidden( side + '_Primary_Con_2' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_Primary_Con_2_GRP' )
        m.parent( side + '_Primary_Con_2', side + '_Primary_Con_2_GRP' )
        m.select( side + '_Primary_Con_2_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_Primary_1.rotateX'), m.getAttr(side + '_Primary_1.rotateY'), m.getAttr(side +
'_Primary_1.rotateZ'), r=True ) #Rotate it

        m.scale( 1.2, 1.2, 1.2, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_Primary_Con_2')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        tempPrimary= int(numPrimaries) #typecast to an integer

        #Create first Primaries control and move into position
        m.duplicate('ArrowConBase', n= side + '_Primary_Con_1')
        m.showHidden( side + '_Primary_Con_1' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_Primary_Con_1_GRP' )
        m.parent( side + '_Primary_Con_1', side + '_Primary_Con_1_GRP' )
        m.select( side + '_Primary_Con_1_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_Primary_' + repr(tempPrimary) + '.rotateX'),  m.getAttr(side + '_Primary_' +
repr(tempPrimary) + '.rotateY'), m.getAttr(side + '_Primary_' + repr(tempPrimary) + '.rotateZ'), r=True ) #Rotate it

        m.scale( 1.4, 1.4, 1.4, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_Primary_Con_1')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        #### Secondaries ###
        #Create second secondary control and move into position
        m.duplicate('ArrowConBase', n= side + '_Secondary_Con_2')
        m.showHidden( side + '_Secondary_Con_2' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_Secondary_Con_2_GRP' )
        m.parent( side + '_Secondary_Con_2', side + '_Secondary_Con_2_GRP' )
        m.select( side + '_Secondary_Con_2_GRP')
        m.showHidden( side + '_Secondary_Con_2' )
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_Secondary_1.rotateX'), m.getAttr(side + '_Secondary_1.rotateY'), m.getAttr(side +
'_Secondary_1.rotateZ'), r=True ) #Rotate it

        m.scale( 1.2, 1.2, 1.2, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_Secondary_Con_2')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        tempSecondary= int(numSecondaries) #typecast to an integer

        #Create first secondaries control and move into position
```

```
        m.duplicate('ArrowConBase', n= side + '_Secondary_Con_1')
        m.showHidden( side + '_Secondary_Con_1' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_Secondary_Con_1_GRP' )
        m.parent( side + '_Secondary_Con_1', side + '_Secondary_Con_1_GRP' )
        m.select( side + '_Secondary_Con_1_GRP')
        m.move( Loc2Coord[0], Loc2Coord[1], Loc2Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_Secondary_' + repr(tempSecondary) + '.rotateX'), m.getAttr(side + '_Secondary_' +
repr(tempSecondary) + '.rotateY'), m.getAttr(side + '_Secondary_' + repr(tempSecondary) + '.rotateZ'), r=True ) #Rotate it

        m.scale( 1, 1, 1, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_Secondary_Con_1')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)


        ### Secondary Coverts ###
        #Create second secondary covert control and move into position
        m.duplicate('ArrowConBase', n= side + '_SecondaryCoverts_Con_2')
        m.showHidden( side + '_SecondaryCoverts_Con_2' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_SecondaryCoverts_Con_2_GRP' )
        m.parent( side + '_SecondaryCoverts_Con_2', side + '_SecondaryCoverts_Con_2_GRP' )
        m.select( side + '_SecondaryCoverts_Con_2_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_SecondaryCoverts_1.rotateX'), m.getAttr(side + '_SecondaryCoverts_1.rotateY'),
m.getAttr(side + '_SecondaryCoverts_1.rotateZ'), r=True ) #Rotate it

        m.scale( 1, 1, 1, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_SecondaryCoverts_Con_2')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        tempSecondaryCov= int(numSecondaryCoverts) #typecast to an integer

        #Create first secondary coverts control and move into position
        m.duplicate('ArrowConBase', n= side + '_SecondaryCoverts_Con_1')
        m.showHidden( side + '_SecondaryCoverts_Con_1' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_SecondaryCoverts_Con_1_GRP' )
        m.parent( side + '_SecondaryCoverts_Con_1', side + '_SecondaryCoverts_Con_1_GRP' )
        m.select( side + '_SecondaryCoverts_Con_1_GRP')
        m.move( Loc2Coord[0], Loc2Coord[1], Loc2Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_SecondaryCoverts_' + repr(tempSecondaryCov) + '.rotateX'), m.getAttr(side +
'_SecondaryCoverts_' + repr(tempSecondaryCov) + '.rotateY'), m.getAttr(side + '_SecondaryCoverts_' +
repr(tempSecondaryCov) + '.rotateZ'), r=True ) #Rotate it

        m.scale( 0.9, 0.9, 0.9, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_SecondaryCoverts_Con_1')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        ### Median Coverts ###
        #Create second median coverts control and move into position
        m.duplicate('ArrowConBase', n= side + '_MedianCoverts_Con_2')
        m.showHidden( side + '_MedianCoverts_Con_2' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_MedianCoverts_Con_2_GRP' )
        m.parent( side + '_MedianCoverts_Con_2', side + '_MedianCoverts_Con_2_GRP' )
```

```
        m.select( side + '_MedianCoverts_Con_2_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_MedianCoverts_1.rotateX'), m.getAttr(side + '_MedianCoverts_1.rotateY'),
m.getAttr(side + '_MedianCoverts_1.rotateZ'), r=True ) #Rotate it

        m.scale( 1, 1, 1, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_MedianCoverts_Con_2')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        tempMedianCov= int(numMedianCoverts) #typecast to an integer

        #Create first median coverts control and move into position
        m.duplicate('ArrowConBase', n= side + '_MedianCoverts_Con_1')
        m.showHidden( side + '_MedianCoverts_Con_1' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_MedianCoverts_Con_1_GRP' )
        m.parent( side + '_MedianCoverts_Con_1', side + '_MedianCoverts_Con_1_GRP' )
        m.select( side + '_MedianCoverts_Con_1_GRP')
        m.move( Loc2Coord[0], Loc2Coord[1], Loc2Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_MedianCoverts_' + repr(tempMedianCov) + '.rotateX'), m.getAttr(side +
'_MedianCoverts_' + repr(tempMedianCov) + '.rotateY'), m.getAttr(side + '_MedianCoverts_' + repr(tempMedianCov) +
'.rotateZ'), r=True ) #Rotate it

        m.scale( 0.8, 0.8, 0.8, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_MedianCoverts_Con_1')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)


        ### Primary Coverts ##
        #Create second primary coverts control and move into position
        m.duplicate('ArrowConBase', n= side + '_PrimaryCoverts_Con_2')
        m.showHidden( side + '_PrimaryCoverts_Con_2' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_PrimaryCoverts_Con_2_GRP' )
        m.parent( side + '_PrimaryCoverts_Con_2', side + '_PrimaryCoverts_Con_2_GRP' )
        m.select( side + '_PrimaryCoverts_Con_2_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_PrimaryCoverts_1.rotateX'),  m.getAttr(side + '_PrimaryCoverts_1.rotateY'),
m.getAttr(side + '_PrimaryCoverts_1.rotateZ'), r=True ) #Rotate it

        m.scale( 1, 1, 1, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_PrimaryCoverts_Con_2')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        tempPrimaryCov= int(numPrimaryCoverts) #typecast to an integer

        #Create first primary coverts control and move into position
        m.duplicate('ArrowConBase', n= side + '_PrimaryCoverts_Con_1')
        m.showHidden( side + '_PrimaryCoverts_Con_1' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_PrimaryCoverts_Con_1_GRP' )
        m.parent( side + '_PrimaryCoverts_Con_1', side + '_PrimaryCoverts_Con_1_GRP' )
        m.select( side + '_PrimaryCoverts_Con_1_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        #Fix rotation so it doesnt need the 20+
        m.rotate( m.getAttr(side + '_PrimaryCoverts_' + repr(tempPrimaryCov) + '.rotateX'),  m.getAttr(side +
```

```
'_PrimaryCoverts_' + repr(tempPrimaryCov) + '.rotateY'), m.getAttr(side + '_PrimaryCoverts_' + repr(tempPrimaryCov) + '.rotateZ'), r=True ) #Rotate it

        m.scale( 1.3, 1.3, 1.3, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_PrimaryCoverts_Con_1')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        ### Alulas ###
        #Create second alula control and move into position
        m.duplicate('ArrowConBase', n= side + '_Alula_Con_2')
        m.showHidden( side + '_Alula_Con_2' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_Alula_Con_2_GRP' )
        m.parent( side + '_Alula_Con_2', side + '_Alula_Con_2_GRP' )
        m.select( side + '_Alula_Con_2_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_Alula_1.rotateX'),  m.getAttr(side + '_Alula_1.rotateY'), m.getAttr(side +
'_Alula_1.rotateZ'), r=True ) #Rotate it

        m.scale( 1, 1, 1, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_Alula_Con_2')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        tempAlula= int(numAlulas) #typecast to an integer

        #Create first alula control and move into position
        m.duplicate('ArrowConBase', n= side + '_Alula_Con_1')
        m.showHidden( side + '_Alula_Con_1' )
        #Create a null group above the control for use on wing folding
        m.group( em=True, name=side + '_Alula_Con_1_GRP' )
        m.parent( side + '_Alula_Con_1', side + '_Alula_Con_1_GRP' )
        m.select( side + '_Alula_Con_1_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        m.rotate( m.getAttr(side + '_Alula_' + repr(tempAlula) + '.rotateX'),  m.getAttr(side + '_Alula_' + repr(tempAlula)
+ '.rotateY'), m.getAttr(side + '_Alula_' + repr(tempAlula) + '.rotateZ'), r=True ) #Rotate it

        m.scale( 1.3, 1.3, 1.3, r=True)
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_Alula_Con_1')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        IKFKControlSetup()        #Build IK FK Switch

        #Parent Wrist rotation to the Wrist con
        m.select(side + '_Wrist_CON', r=True)
        m.select(side + '_Wing_3_JNT_IK', tgl=True)
        m.orientConstraint(weight=1, mo=True)

        #Correctly Parent control system
        m.parent(side + '_Wrist_Con_GRP', side + '_Shoulder_CON' )
        m.parent(side + '_Wing_Pole_CON', side + '_Shoulder_CON' )

        #Constrain the pole vector
        m.poleVectorConstraint( side + '_Wing_Pole_CON', side + '_Wing_IK' )


        #Parent Joints to the Shoulder control
```

```
        m.select(side + '_Shoulder_CON', r=True)
        m.select(side + '_Wing_1_JNT', tgl=True)
        m.pointConstraint(weight=1, mo=True)

        m.select(side + '_Shoulder_CON', r=True)
        m.select(side + '_Wing_1_JNT_FK', tgl=True)
        m.parentConstraint(weight=1, mo=True)

        m.select(side + '_Shoulder_CON', r=True)
        m.select(side + '_Wing_1_JNT_IK', tgl=True)
        m.pointConstraint(weight=1, mo=True)

        #Parent FK controls to shoulder control
        m.parent(side + '_Wing2_FK_Con_GRP', side + '_Shoulder_CON' )
        m.parent(side + '_Wing3_FK_Con_GRP', side + '_Wing2_FK_CON' )

        m.select(cl=True)

#Creates the IK and FK controls
def IKFKControlSetup():
        #Create IK control
        m.select(d=True)
        m.ikHandle( sj=side + '_Wing_1_JNT_IK', ee=side + '_Wing_3_JNT_IK', p=2, w=.5, sol='ikRPsolver') #Step 11
        m.rename(side + '_Wing_IK')

        #Parent wrist Con to the IK
        m.select(side + '_Wrist_CON', r=True)
        m.select(side + '_Wing_IK', tgl=True)
        m.pointConstraint(weight=1, mo=True)

        #Orient constrain joint hierarchy for transition skeleton to IK and FK skeletons
        m.select(side + '_Wing_1_JNT_IK', r=True)
        m.select(side + '_Wing_1_JNT_FK', tgl=True)
        m.select(side + '_Wing_1_JNT', tgl=True)
        m.orientConstraint(weight=1, mo=True)

        m.select(side + '_Wing_2_JNT_IK', r=True)
        m.select(side + '_Wing_2_JNT_FK', tgl=True)
        m.select(side + '_Wing_2_JNT', tgl=True)
        m.orientConstraint(weight=1, mo=True)

        m.select(side + '_Wing_3_JNT_IK', r=True)
        m.select(side + '_Wing_3_JNT_FK', tgl=True)
        m.select(side + '_Wing_3_JNT', tgl=True)
        m.orientConstraint(weight=1, mo=True)

        #Create Switch attribute
        m.addAttr( side + '_Shoulder_CON', longName='IK_FK_Switch', defaultValue=0.0, minValue=0.0, maxValue=1.0)
        m.setAttr(side + '_Shoulder_CON.'+ 'IK_FK_Switch', k=1) #Make it keyable

        #Set switch to alternate between IK and FK orient constraints using set driven keys
        m.setAttr(side + '_Wing_1_JNT_orientConstraint1.' + side + '_Wing_1_JNT_IKW0', 1 )
        m.setAttr(side + '_Wing_1_JNT_orientConstraint1.' + side + '_Wing_1_JNT_FKW1', 0 )

        m.setDrivenKeyframe(side + '_Wing_1_JNT_orientConstraint1.' + side + '_Wing_1_JNT_IKW0', cd= side +
'_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe(side + '_Wing_1_JNT_orientConstraint1.' + side + '_Wing_1_JNT_FKW1', cd= side +
'_Shoulder_CON.IK_FK_Switch' )
```

```
        m.setAttr(side + '_Wing_2_JNT_orientConstraint1.' + side + '_Wing_2_JNT_IKW0', 1 )
        m.setAttr(side + '_Wing_2_JNT_orientConstraint1.' + side + '_Wing_2_JNT_FKW1', 0 )

        m.setDrivenKeyframe(side + '_Wing_2_JNT_orientConstraint1.' + side + '_Wing_2_JNT_IKW0', cd=side +
'_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe(side + '_Wing_2_JNT_orientConstraint1.' + side + '_Wing_2_JNT_FKW1', cd=side +
'_Shoulder_CON.IK_FK_Switch' )

        m.setAttr(side + '_Wing_3_JNT_orientConstraint1.' + side + '_Wing_3_JNT_IKW0', 1 )
        m.setAttr(side + '_Wing_3_JNT_orientConstraint1.' + side + '_Wing_3_JNT_FKW1', 0 )

        m.setDrivenKeyframe(side + '_Wing_3_JNT_orientConstraint1.' + side + '_Wing_3_JNT_IKW0', cd=side +
'_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe(side + '_Wing_3_JNT_orientConstraint1.' + side + '_Wing_3_JNT_FKW1', cd=side +
'_Shoulder_CON.IK_FK_Switch' )

        m.setAttr(side + '_Shoulder_CON.IK_FK_Switch', 1 )
        m.setAttr(side + '_Wing_1_JNT_orientConstraint1.' + side + '_Wing_1_JNT_IKW0', 0 )
        m.setAttr(side + '_Wing_1_JNT_orientConstraint1.' + side + '_Wing_1_JNT_FKW1', 1 )

        m.setDrivenKeyframe(side + '_Wing_1_JNT_orientConstraint1.' + side + '_Wing_1_JNT_IKW0', cd=side +
'_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe(side + '_Wing_1_JNT_orientConstraint1.' + side + '_Wing_1_JNT_FKW1', cd=side +
'_Shoulder_CON.IK_FK_Switch' )


        m.setAttr(side + '_Wing_2_JNT_orientConstraint1.' + side + '_Wing_2_JNT_IKW0', 0 )
        m.setAttr(side + '_Wing_2_JNT_orientConstraint1.' + side + '_Wing_2_JNT_FKW1', 1 )

        m.setDrivenKeyframe(side + '_Wing_2_JNT_orientConstraint1.' + side + '_Wing_2_JNT_IKW0', cd=side +
'_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe(side + '_Wing_2_JNT_orientConstraint1.' + side + '_Wing_2_JNT_FKW1', cd=side +
'_Shoulder_CON.IK_FK_Switch' )

        m.setAttr(side + '_Wing_3_JNT_orientConstraint1.' + side + '_Wing_3_JNT_IKW0', 0 )
        m.setAttr(side + '_Wing_3_JNT_orientConstraint1.' + side + '_Wing_3_JNT_FKW1', 1 )

        m.setDrivenKeyframe(side + '_Wing_3_JNT_orientConstraint1.' + side + '_Wing_3_JNT_IKW0', cd=side +
'_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe(side + '_Wing_3_JNT_orientConstraint1.' + side + '_Wing_3_JNT_FKW1', cd=side +
'_Shoulder_CON.IK_FK_Switch' )

        #Create FK Controllers
        m.circle( nr=(4, 0, 0), c=(0,0,0), r= 5, n=side + '_Wing2_FK_CON' )
        m.group( em=True, name=side + '_Wing2_FK_Con_GRP' )
        m.parent( side + '_Wing2_FK_CON', side + '_Wing2_FK_Con_GRP' )
        m.select( side + '_Wing2_FK_Con_GRP')
        m.move( Loc2Coord[0], Loc2Coord[1], Loc2Coord[2], ws=True )
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.circle( nr=(4, 0, 0), c=(0,0,0), r= 5, n=side + '_Wing3_FK_CON' )
        m.group( em=True, name=side + '_Wing3_FK_Con_GRP' )
        m.parent( side + '_Wing3_FK_CON', side + '_Wing3_FK_Con_GRP' )
        m.select( side + '_Wing3_FK_Con_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        #Parent the controls to the joint
        m.select(side + '_Wing2_FK_CON', r=True)
        m.select(side + '_Wing_2_JNT_FK', tgl=True)
```

```
        m.orientConstraint(weight= 1, mo=True)

        #Parent the controls to the joint
        m.select(side + '_Wing3_FK_CON', r=True)
        m.select(side + '_Wing_3_JNT_FK', tgl=True)
        m.orientConstraint(weight= 1, mo=True)

        #set visibility of IK/FK switch
        m.setAttr( side + '_Shoulder_CON.IK_FK_Switch', 0 )

        #Visibility for IK mode
        m.setAttr( side + '_Wing2_FK_Con_GRP.visibility', 0 )
        m.setAttr( side + '_Wing3_FK_Con_GRP.visibility', 0 )
        m.setAttr( side + '_Wing_1_JNT_FK.visibility', 0 )
        m.setAttr( side + '_Wrist_Con_GRP.visibility', 1 )
        m.setAttr( side + '_Wing_1_JNT_IK.visibility', 1 )
        m.setAttr( side + '_Wing_Pole_CON.visibility', 1 )

        m.setDrivenKeyframe( side + '_Wing2_FK_Con_GRP.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe( side + '_Wing3_FK_Con_GRP.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe( side + '_Wrist_Con_GRP.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe( side + '_Wing_1_JNT_IK.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe( side + '_Wing_1_JNT_FK.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )

        m.setDrivenKeyframe( side + '_Wing_Pole_CON.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )


        #Visibility for FK mode
        m.setAttr( side + '_Shoulder_CON.IK_FK_Switch', 1 )
        m.setAttr( side + '_Wing2_FK_Con_GRP.visibility', 1 )
        m.setAttr( side + '_Wing3_FK_Con_GRP.visibility', 1 )
        m.setAttr( side + '_Wing_1_JNT_FK.visibility', 1 )
        m.setAttr( side + '_Wrist_Con_GRP.visibility', 0 )
        m.setAttr( side + '_Wing_1_JNT_IK.visibility', 0 )
        m.setAttr( side + '_Wing_Pole_CON.visibility', 0 )

        m.setDrivenKeyframe( side + '_Wing2_FK_Con_GRP.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe( side + '_Wing3_FK_Con_GRP.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe( side + '_Wrist_Con_GRP.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe( side + '_Wing_1_JNT_IK.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe( side + '_Wing_1_JNT_FK.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )
        m.setDrivenKeyframe( side + '_Wing_Pole_CON.visibility', cd= side + '_Shoulder_CON.IK_FK_Switch' )


        #Set control back to Default 0 (sets it to default IK mode)
        m.setAttr( side + '_Shoulder_CON.IK_FK_Switch', 0 )

        m.select(cl=True)

#Main controls setup
def mainControls():
        ### Create Feathers Tips Control ###
        tempPrim = repr(int(numPrimaries)) #Typecast to an int repr to cast as a string
        m.duplicate('CurveConBase', n= side + '_TipFeathers_CON')
        m.showHidden( side + '_TipFeathers_CON' )
        #Make two groups so one can be used for the wingfold control
        m.group( em=True, name=side + '_TipFeathers_GRP2' )
        m.parent( side + '_TipFeathers_CON', side + '_TipFeathers_GRP2' )
        m.group( em=True, name=side + '_TipFeathers_GRP' )
```

```
        m.parent( side + '_TipFeathers_GRP2', side + '_TipFeathers_GRP' )
        m.select( side + '_TipFeathers_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        #Rotate control into position based on the Primary control
        m.rotate( m.getAttr(side + '_Primary_'+ tempPrim +'.rotateX'), m.getAttr(side + '_Primary_'+ tempPrim
+'.rotateY'), m.getAttr(side + '_Primary_'+ tempPrim +'.rotateZ'), r=True ) #Rotate it
        m.scale( 1.2, 1.2, 1.2, r=True) #Scale it
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_TipFeathers_CON')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_TipFeathers_GRP2')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        ### Feathers Middle Control ###
        #Calculate where the control should sit based on where the primary and secondary controls are.
        PrimaryPos = [m.getAttr(side + '_Primary_1.rotateX'), m.getAttr(side + '_Primary_1.rotateY'), m.getAttr(side +
'_Primary_1.rotateZ')]
        SecondaryPos = [m.getAttr(side + '_Secondary_1.rotateX'), m.getAttr(side + '_Secondary_1.rotateY'),
m.getAttr(side + '_Secondary_1.rotateZ')]
        HalfwayPos = [(PrimaryPos[0]-SecondaryPos[0])/2.0, (PrimaryPos[1]-SecondaryPos[1])/2.0, (PrimaryPos[2]-
SecondaryPos[2])/2.0]

        m.duplicate('CurveConBase', n= side + '_MiddleFeathers_CON')
        m.showHidden( side + '_MiddleFeathers_CON')
        #Make two groups so one can be used for the wingfold control
        m.group( em=True, name=side + '_MiddleFeathers_GRP2' )
        m.parent( side + '_MiddleFeathers_CON', side + '_MiddleFeathers_GRP2' )
        m.group( em=True, name=side + '_MiddleFeathers_GRP' )
        m.parent( side + '_MiddleFeathers_GRP2', side + '_MiddleFeathers_GRP' )
        m.select( side + '_MiddleFeathers_GRP')
        m.move( Loc3Coord[0], Loc3Coord[1], Loc3Coord[2], ws=True )
        #Rotate control into position based on the earlier calculations
        m.rotate( SecondaryPos[0] + HalfwayPos[0], SecondaryPos[1] + HalfwayPos[1], SecondaryPos[2] +
HalfwayPos[2] )  #Rotate it in between the two controls
        m.scale( 1, 1, 1, r=True) #Scale it
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_MiddleFeathers_CON')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)

        ### Feathers End Control ###
        tempSec = repr(int(numSecondaries)) #Pfffft typcasting!

        m.duplicate('CurveConBase', n= side + '_EndFeathers_CON')
        #Make two groups so one can be used for the wingfold control
        m.group( em=True, name=side + '_EndFeathers_GRP2' )
        m.parent( side + '_EndFeathers_CON', side + '_EndFeathers_GRP2' )
        m.group( em=True, name=side + '_EndFeathers_GRP' )
        m.parent( side + '_EndFeathers_GRP2', side + '_EndFeathers_GRP' )
        m.select( side + '_EndFeathers_GRP')
        m.showHidden( side + '_EndFeathers_CON' )
        m.move( Loc2Coord[0], Loc2Coord[1], Loc2Coord[2], ws=True )
        #Rotate control into position based on the Secondary control
        m.rotate( m.getAttr(side + '_Secondary_'+ tempSec +'.rotateX'), m.getAttr(side + '_Secondary_'+ tempSec
+'.rotateY'), m.getAttr(side + '_Secondary_'+ tempSec +'.rotateZ'), r=True ) #Rotate it
        m.scale( 0.8, 0.8, 0.8, r=True) #Scale it
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
        m.select( side + '_EndFeathers_CON')
        m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
```

```
	#Delete Extraneous Flex Controls
	if (side == "L"):
		m.delete( side + '_TipFeathers_CON|Wing_Flex_2', side + '_EndFeathers_CON|Wing_Flex_1') #Remove
the extra control
	elif (side == "R"):
		m.delete( side + '_TipFeathers_CON|Wing_Flex_1', side + '_EndFeathers_CON|Wing_Flex_2') #Remove
the extra control

	#Name the flex controls properly
	if (side == "L"):
		m.rename( side + '_TipFeathers_CON|Wing_Flex_1', side + '_Tip_Flex_Feathers')
		m.rename( side + '_MiddleFeathers_CON|Wing_Flex_1', side + '_Mid_R_Flex_Feathers')
		m.rename( side + '_MiddleFeathers_CON|Wing_Flex_2', side + '_Mid_L_Flex_Feathers')
		m.rename( side + '_EndFeathers_CON|Wing_Flex_2', side + '_End_Flex_Feathers')
	elif (side == "R"):
		m.rename( side + '_TipFeathers_CON|Wing_Flex_2', side + '_Tip_Flex_Feathers')
		m.rename( side + '_MiddleFeathers_CON|Wing_Flex_2', side + '_Mid_R_Flex_Feathers')
		m.rename( side + '_MiddleFeathers_CON|Wing_Flex_1', side + '_Mid_L_Flex_Feathers')
		m.rename( side + '_EndFeathers_CON|Wing_Flex_1', side + '_End_Flex_Feathers')

	#Parent controls under the joints for proper movement (even between IK and FK)
	m.select(side + '_Wing_3_JNT', r=True)
	m.select(side + '_TipFeathers_GRP', tgl=True)
	m.parentConstraint(weight=1, mo=True)

	m.select(side + '_Wing_3_JNT', r=True)
	m.select(side + '_MiddleFeathers_GRP', tgl=True)
	m.parentConstraint(weight=1, mo=True)

	m.select(side + '_Wing_2_JNT', r=True)
	m.select(side + '_EndFeathers_GRP', tgl=True)
	m.parentConstraint(weight=1, mo=True)

	#Parent The minor controls
	m.parent( side + '_Secondary_Con_2_GRP', side + '_MiddleFeathers_CON' )
	m.parent( side + '_Secondary_Con_1_GRP', side + '_EndFeathers_CON' )

	m.parent( side + '_Primary_Con_2_GRP', side + '_MiddleFeathers_CON' )
	m.parent( side + '_Primary_Con_1_GRP', side + '_TipFeathers_CON' )

	m.parent( side + '_SecondaryCoverts_Con_2_GRP', side + '_MiddleFeathers_CON' )
	m.parent( side + '_SecondaryCoverts_Con_1_GRP', side + '_EndFeathers_CON' )

	m.parent( side + '_PrimaryCoverts_Con_2_GRP', side + '_MiddleFeathers_CON' )
	m.parent( side + '_PrimaryCoverts_Con_1_GRP', side + '_TipFeathers_CON' )

	m.parent( side + '_MedianCoverts_Con_2_GRP', side + '_MiddleFeathers_CON' )
	m.parent( side + '_MedianCoverts_Con_1_GRP', side + '_EndFeathers_CON' )

	m.parent( side + '_Alula_Con_1_GRP', side + '_TipFeathers_CON' )
	m.parent( side + '_Alula_Con_2_GRP', side + '_MiddleFeathers_CON' )

	#Setup visibility Controls on the Overall Controls
	### Tip Feathers Controls ###
	#Create Control Attributes
	m.addAttr(  side + '_TipFeathers_CON', longName='GroupControls', defaultValue=0.0, minValue=0.0,
maxValue=10.0)
	m.setAttr( side + '_TipFeathers_CON.GroupControls', k=1) #Make it keyable
```

```
        #Key them for on and off
        m.setAttr( side + '_TipFeathers_CON.GroupControls', 0 )
        m.setAttr( side + '_Alula_Con_1.visibility', 0 )
        m.setAttr( side + '_Primary_Con_1.visibility', 0 )
        m.setAttr( side + '_PrimaryCoverts_Con_1.visibility', 0 )

        m.setDrivenKeyframe( side + '_Alula_Con_1.visibility', cd=  side + '_TipFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_Primary_Con_1.visibility', cd=  side + '_TipFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_PrimaryCoverts_Con_1.visibility', cd=  side +
'_TipFeathers_CON.GroupControls' )

        m.setAttr( side + '_TipFeathers_CON.GroupControls', 1 )
        m.setAttr( side + '_Alula_Con_1.visibility', 1 )
        m.setAttr( side + '_Primary_Con_1.visibility', 1 )
        m.setAttr( side + '_PrimaryCoverts_Con_1.visibility', 1 )

        m.setDrivenKeyframe( side + '_Alula_Con_1.visibility', cd=  side + '_TipFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_Primary_Con_1.visibility', cd=  side + '_TipFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_PrimaryCoverts_Con_1.visibility', cd=  side +
'_TipFeathers_CON.GroupControls' )

        ### Middle Feathers Controls ###
        #Create Control Attributes
        m.addAttr(  side + '_MiddleFeathers_CON', longName='GroupControls', defaultValue=0.0, minValue=0.0,
maxValue=10.0)
        m.setAttr( side + '_MiddleFeathers_CON.GroupControls', k=1) #Make it keyable

        #Key them for on and off
        m.setAttr( side + '_MiddleFeathers_CON.GroupControls', 0 )
        m.setAttr( side + '_Alula_Con_2.visibility', 0 )
        m.setAttr( side + '_Primary_Con_2.visibility', 0 )
        m.setAttr( side + '_PrimaryCoverts_Con_2.visibility', 0 )
        m.setAttr( side + '_MedianCoverts_Con_2.visibility', 0 )
        m.setAttr( side + '_Secondary_Con_2.visibility', 0 )
        m.setAttr( side + '_SecondaryCoverts_Con_2.visibility', 0 )

        m.setDrivenKeyframe( side + '_Alula_Con_2.visibility', cd=  side + '_MiddleFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_Primary_Con_2.visibility', cd=  side + '_MiddleFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_PrimaryCoverts_Con_2.visibility', cd=  side +
'_MiddleFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_MedianCoverts_Con_2.visibility', cd=  side +
'_MiddleFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_Secondary_Con_2.visibility', cd=  side + '_MiddleFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_SecondaryCoverts_Con_2.visibility', cd=  side +
'_MiddleFeathers_CON.GroupControls' )

        m.setAttr( side + '_MiddleFeathers_CON.GroupControls', 1 )
        m.setAttr( side + '_Alula_Con_2.visibility', 1 )
        m.setAttr( side + '_Primary_Con_2.visibility', 1 )
        m.setAttr( side + '_PrimaryCoverts_Con_2.visibility', 1 )
        m.setAttr( side + '_MedianCoverts_Con_2.visibility', 1 )
        m.setAttr( side + '_Secondary_Con_2.visibility', 1 )
        m.setAttr( side + '_SecondaryCoverts_Con_2.visibility', 1 )

        m.setDrivenKeyframe( side + '_Alula_Con_2.visibility', cd=  side + '_MiddleFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_Primary_Con_2.visibility', cd=  side + '_MiddleFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_PrimaryCoverts_Con_2.visibility', cd=  side +
'_MiddleFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_MedianCoverts_Con_2.visibility', cd=  side +
```

```
'_MiddleFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_Secondary_Con_2.visibility', cd=  side + '_MiddleFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_SecondaryCoverts_Con_2.visibility', cd=  side +
'_MiddleFeathers_CON.GroupControls' )

        ### End Feathers Controls ###
        #Create Control Attributes
        m.addAttr(  side + '_EndFeathers_CON', longName='GroupControls', defaultValue=0.0, minValue=0.0,
maxValue=10.0)
        m.setAttr( side + '_EndFeathers_CON.GroupControls', k=1) #Make it keyable

        #Key them for on and off
        m.setAttr( side + '_EndFeathers_CON.GroupControls', 0 )
        m.setAttr( side + '_MedianCoverts_Con_1.visibility', 0 )
        m.setAttr( side + '_Secondary_Con_1.visibility', 0 )
        m.setAttr( side + '_SecondaryCoverts_Con_1.visibility', 0 )

        m.setDrivenKeyframe( side + '_MedianCoverts_Con_1.visibility', cd=  side +
'_EndFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_Secondary_Con_1.visibility', cd=  side + '_EndFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_SecondaryCoverts_Con_1.visibility', cd=  side +
'_EndFeathers_CON.GroupControls' )

        m.setAttr( side + '_EndFeathers_CON.GroupControls', 1 )
        m.setAttr( side + '_MedianCoverts_Con_1.visibility', 1 )
        m.setAttr( side + '_Secondary_Con_1.visibility', 1 )
        m.setAttr( side + '_SecondaryCoverts_Con_1.visibility', 1 )

        m.setDrivenKeyframe( side + '_MedianCoverts_Con_1.visibility', cd=  side +
'_EndFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_Secondary_Con_1.visibility', cd=  side + '_EndFeathers_CON.GroupControls' )
        m.setDrivenKeyframe( side + '_SecondaryCoverts_Con_1.visibility', cd=  side +
'_EndFeathers_CON.GroupControls' )


        #Set controls to default 0 (Invisible)
        m.setAttr( side + '_TipFeathers_CON.GroupControls', 0 )
        m.setAttr( side + '_MiddleFeathers_CON.GroupControls', 0 )
        m.setAttr( side + '_EndFeathers_CON.GroupControls', 0 )


        # Blendshape Controls #
        ### Tip Feathers Blendshape Controls ###
        #Create Control Attributes
        m.addAttr(  side + '_TipFeathers_CON', longName='FlexControls', defaultValue=0.0, minValue=0.0,
maxValue=10.0)
        m.setAttr( side + '_TipFeathers_CON.FlexControls', k=1) #Make it keyable

        m.setAttr( side +  '_TipFeathers_CON.FlexControls', 0 )
        m.setAttr(side + '_Tip_Flex_Feathers.visibility', 0 )
        m.setDrivenKeyframe( side + '_Tip_Flex_Feathers.visibility', cd=  side + '_TipFeathers_CON.FlexControls' )
        m.setAttr( side +  '_TipFeathers_CON.FlexControls', 1 )
        m.setAttr( side + '_Tip_Flex_Feathers.visibility', 1 )
        m.setDrivenKeyframe( side + '_Tip_Flex_Feathers.visibility', cd=  side + '_TipFeathers_CON.FlexControls' )

        ### Middle Feathers Blendshape Controls ###
        #Create Control Attributes
        m.addAttr(  side + '_MiddleFeathers_CON', longName='FlexControls', defaultValue=0.0, minValue=0.0,
maxValue=10.0)
```

```
        m.setAttr( side + '_MiddleFeathers_CON.FlexControls', k=1) #Make it keyable

        m.setAttr( side +  '_MiddleFeathers_CON.FlexControls', 0 )
        m.setAttr(side + '_Mid_R_Flex_Feathers.visibility', 0 )
        m.setAttr(side + '_Mid_L_Flex_Feathers.visibility', 0 )
        m.setDrivenKeyframe( side + '_Mid_R_Flex_Feathers.visibility', cd=  side +
'_MiddleFeathers_CON.FlexControls' )
        m.setDrivenKeyframe( side + '_Mid_L_Flex_Feathers.visibility', cd=  side +
'_MiddleFeathers_CON.FlexControls' )

        m.setAttr( side +  '_MiddleFeathers_CON.FlexControls', 1 )
        m.setAttr(side + '_Mid_R_Flex_Feathers.visibility', 1 )
        m.setAttr(side + '_Mid_L_Flex_Feathers.visibility', 1 )
        m.setDrivenKeyframe( side + '_Mid_R_Flex_Feathers.visibility', cd=  side +
'_MiddleFeathers_CON.FlexControls' )
        m.setDrivenKeyframe( side + '_Mid_L_Flex_Feathers.visibility', cd=  side +
'_MiddleFeathers_CON.FlexControls' )

        ### End Feathers Blendshape Controls ###
        #Create Control Attributes
        m.addAttr(  side + '_EndFeathers_CON', longName='FlexControls', defaultValue=0.0, minValue=0.0,
maxValue=10.0)
        m.setAttr( side + '_EndFeathers_CON.FlexControls', k=1) #Make it keyable

        m.setAttr( side +  '_EndFeathers_CON.FlexControls', 0 )
        m.setAttr(side + '_End_Flex_Feathers.visibility', 0 )
        m.setDrivenKeyframe( side + '_End_Flex_Feathers.visibility', cd=  side + '_EndFeathers_CON.FlexControls' )
        m.setAttr( side +  '_EndFeathers_CON.FlexControls', 1 )
        m.setAttr(side + '_End_Flex_Feathers.visibility', 1 )
        m.setDrivenKeyframe( side + '_End_Flex_Feathers.visibility', cd=  side + '_EndFeathers_CON.FlexControls' )

        #Set controls to default 0 (Invisible)
        m.setAttr( side + '_TipFeathers_CON.FlexControls', 0 )
        m.setAttr( side + '_MiddleFeathers_CON.FlexControls', 0 )
        m.setAttr( side + '_EndFeathers_CON.FlexControls', 0 )


        #Parent Main groups        properly
        m.parent( side + '_Wing_1_JNT',  'Joints' )
        m.parent( side + '_Wing_1_JNT_IK',  'Joints' )
        m.parent( side + '_Wing_1_JNT_FK',  'Joints' )
        m.parent( side + '_Shoulder_Con_GRP',  'Controls' )
        m.parent( side + '_TipFeathers_GRP',  'Controls' )
        m.parent( side + '_MiddleFeathers_GRP',  'Controls' )
        m.parent( side + '_EndFeathers_GRP',  'Controls' )
        #Parent Feather Groups properly
        m.parent( side + '_Primary_GRP',  'FeatherGroups' )
        m.parent( side + '_PrimaryCoverts_GRP',  'FeatherGroups' )
        m.parent( side + '_Secondary_GRP',  'FeatherGroups' )
        m.parent( side + '_SecondaryCoverts_GRP',  'FeatherGroups' )
        m.parent( side + '_Alula_GRP',  'FeatherGroups' )
        m.parent( side + '_MedianCoverts_GRP',  'FeatherGroups' )

        m.parent(side + '_Wing_IK',  'NoXForm' )


#MOTION SYSTEM SETUP

#Setup the Feather Groups            - Function takes in the feather name, number of feathers in the group, controls 1 and 2
```

```
#and the wing joint that the group should be parented to
def featherSetup(featherNamePAS, numFeathersPAS, controlAPAS, controlBPAS,  wingJointPAS):
	NAME = side + featherNamePAS
	i = 1

	#Create main group and move it to the rotation point
	m.group( em=True, name= NAME+ 'GRP' )
	m.move(m.getAttr(side+'_Wing_3.translateX'), m.getAttr(side+'_Wing_3.translateY'),
m.getAttr(side+'_Wing_3.translateZ' ), ws=True)

	#Create Groups for the feathers
	while (i < (numFeathersPAS + 1)):
		FeatherName = NAME + repr(i)
		GRPName = NAME + repr(i) + "_GRP"
		m.group( em=True, name= GRPName ) #create a null group for each feather
		tester = m.xform( FeatherName +'.rotatePivot', t=True, q=True, ws=True) #Get the position of the feather's
rotate point
		m.move(tester[0], tester[1], tester[2], ws=True) #Move the group to the feathers rotate point so they're the
same
		m.makeIdentity(apply=True, t=1, r=1, s=1, n=0)
		m.parent( FeatherName, GRPName ) #Parent feather to it's group
		m.parent( GRPName, NAME+'GRP' ) #Parent feather under proper main group
		i=i+1
	m.select(cl=True)

	j = 1
	amount1=0.0
	amount2=1.0
	incriment= 1.0/numFeathersPAS #Incriment by the percentage of the number of feathers

	#while loop takes each feather and orient constrains it to the two controls based on it's position in a wing
	#For example if feather 5 is in the center of the wing, it would be constrained 50% to control A and 50% to
	#control B.  Whereas Feather 2 may be closer to A, so it'd be 90% to A and 10% to B.
	#This is what caused the fanning motion of the feathers in a group between each control.
	while (j < (numFeathersPAS + 1)):
		m.select(NAME + controlBPAS, r=True)
		m.select(NAME + repr(j) + '_GRP', tgl=True)
		m.orientConstraint(weight= amount2, mo=True)
		amount2=amount2 - incriment

		m.select(NAME + controlAPAS, r=True)
		m.select(NAME + repr(j) + '_GRP', tgl=True)
		m.orientConstraint(weight= amount1, mo=True)
		amount1=amount1 + incriment

		j=j+1

	#Parent wing group to the joint for proper feather movement
	m.select(side + wingJointPAS, r=True)
	m.select(NAME+ 'GRP', tgl=True)
	m.parentConstraint(weight= amount1, mo=True)

#Set the side to build based on which radio button is selected
def setSide():
	global side

	#Get the side from the radio button
	lButton = m.radioButton('leftButton', q=True, select = True)
	rButton = m.radioButton('rightButton', q=True, select = True)
```

```
	#Set which side was chosen
	if lButton ==True:
		side = "L"
	elif rButton ==True:
		side = "R"

	#Set the coordinates of the joint locators
	global Loc1Coord, Loc2Coord, Loc3Coord, Loc4Coord
	Loc1Coord = m.xform( side + '_Wing_1', t=True, q=True, ws=True)    #Shoulder location coordinates
	Loc2Coord = m.xform( side + '_Wing_2', t=True, q=True, ws=True)    #Elbow location coordinates
	Loc3Coord = m.xform( side + '_Wing_3', t=True, q=True, ws=True)    #Wrist location coordinates
	Loc4Coord = m.xform( side + '_Wing_4', t=True, q=True, ws=True) #Fingertips location coordinates

	return side

#Set number of feathers input by the user
def setNumOfFeathers():
	global numPrimaries
	numPrimaries = m.textField('numPrimariesField', q= True, text=True) #Get the number of feathers from the GUI
	numPrimaries = float(numPrimaries)		#recast as a float since it reads in as a string

	global numSecondaries
	numSecondaries = m.textField('numSecondariesField', q= True, text=True) #Get the number of feathers from the GUI
	numSecondaries = float(numSecondaries)    #recast as a float since it reads in as a string

	global numPrimaryCoverts
	numPrimaryCoverts = m.textField('numPrimaryCovertsField', q= True, text=True) #Get the number of feathers from the GUI
	numPrimaryCoverts = float(numPrimaryCoverts)      #recast as a float since it reads in as a string

	global numSecondaryCoverts
	numSecondaryCoverts = m.textField('numSecondaryCovertsField', q= True, text=True) #Get the number of feathers from the GUI
	numSecondaryCoverts = float(numSecondaryCoverts)		#recast as a float since it reads in as a string

	global numMedianCoverts
	numMedianCoverts = m.textField('numMedianCovertsField', q= True, text=True) #Get the number of feathers from the GUI
	numMedianCoverts = float(numMedianCoverts)       #recast as a float since it reads in as a string

#Skeleton Setup
def generateSkeleton():

	#Create three skeletons- One for the IK, one for the FK and one for between
	#The between one is the one that is skinned to.
	m.select(cl=True)
	# create the between skeleton for the wing
	m.joint( n = side + '_Wing_1_JNT', p =  [m.getAttr(side + '_Wing_1.translateX'), m.getAttr(side + '_Wing_1.translateY'), m.getAttr(side + '_Wing_1.translateZ' )] )
	m.joint( n = side + '_Wing_2_JNT', p =  [m.getAttr(side + '_Wing_2.translateX'), m.getAttr(side + '_Wing_2.translateY'), m.getAttr(side + '_Wing_2.translateZ' )] )
	m.joint( side + '_Wing_1_JNT', e = 1, zso = 1, oj = 'xyz', sao = 'yup')
	m.joint( n = side + '_Wing_3_JNT', p =  [m.getAttr(side + '_Wing_3.translateX'), m.getAttr(side + '_Wing_3.translateY'), m.getAttr(side + '_Wing_3.translateZ' )] )
	m.joint( side + '_Wing_2_JNT', e = 1, zso = 1, oj = 'xyz', sao = 'yup')
	m.joint( n = side + '_Wing_4_JNT', p =  [m.getAttr(side + '_Wing_4.translateX'), m.getAttr(side + '_Wing_4.translateY'), m.getAttr(side + '_Wing_4.translateZ' )] )
```

```
	m.joint( side + '_Wing_3_JNT', e = 1, zso = 1, oj = 'xyz', sao = 'yup')
	m.select(cl=True)

	m.select(cl=True)
	# create the IK skeleton for the wing
	m.joint( n = side + '_Wing_1_JNT_IK', p =  [m.getAttr(side + '_Wing_1.translateX'), m.getAttr(side + 
'_Wing_1.translateY'), m.getAttr(side + '_Wing_1.translateZ' )] )
	m.joint( n = side + '_Wing_2_JNT_IK', p =  [m.getAttr(side + '_Wing_2.translateX'), m.getAttr(side + 
'_Wing_2.translateY'), m.getAttr(side + '_Wing_2.translateZ' )] )
	m.joint( side + '_Wing_1_JNT_IK', e = 1, zso = 1, oj = 'xyz', sao = 'yup')
	m.joint( n = side + '_Wing_3_JNT_IK', p =  [m.getAttr(side + '_Wing_3.translateX'), m.getAttr(side + 
'_Wing_3.translateY'), m.getAttr(side + '_Wing_3.translateZ' )] )
	m.joint( side + '_Wing_2_JNT_IK', e = 1, zso = 1, oj = 'xyz', sao = 'yup')
	m.joint( n = side + '_Wing_4_JNT_IK', p =  [m.getAttr(side + '_Wing_4.translateX'), m.getAttr(side + 
'_Wing_4.translateY'), m.getAttr(side + '_Wing_4.translateZ' )] )
	m.joint( side + '_Wing_3_JNT_IK', e = 1, zso = 1, oj = 'xyz', sao = 'yup')
	m.select(cl=True)

	# create the FK skeleton for the wing
	m.joint( n = side + '_Wing_1_JNT_FK', p =  [m.getAttr(side + '_Wing_1.translateX'), m.getAttr(side + 
'_Wing_1.translateY'), m.getAttr(side + '_Wing_1.translateZ' )] )
	m.joint( n = side + '_Wing_2_JNT_FK', p =  [m.getAttr(side + '_Wing_2.translateX'), m.getAttr(side + 
'_Wing_2.translateY'), m.getAttr(side + '_Wing_2.translateZ' )] )
	m.joint( side + '_Wing_1_JNT_FK', e = 1, zso = 1, oj = 'xyz', sao = 'yup')
	m.joint( n = side + '_Wing_3_JNT_FK', p =  [m.getAttr(side + '_Wing_3.translateX'), m.getAttr(side + 
'_Wing_3.translateY'), m.getAttr(side + '_Wing_3.translateZ' )] )
	m.joint( side + '_Wing_2_JNT_FK', e = 1, zso = 1, oj = 'xyz', sao = 'yup')
	m.joint( n = side + '_Wing_4_JNT_FK', p =  [m.getAttr(side + '_Wing_4.translateX'), m.getAttr(side + 
'_Wing_4.translateY'), m.getAttr(side + '_Wing_4.translateZ' )] )
	m.joint( side + '_Wing_3_JNT_FK', e = 1, zso = 1, oj = 'xyz', sao = 'yup')
	m.select(cl=True)

#GUI button function 1
#Call the generation functions when the button to generate feathers is pressed
def generateFeathers(self):
	#Set the side and number of feathers before beginning
	setSide()
	setNumOfFeathers()

	if (m.objExists(side +'_Primary_1') == 0): #Check to make sure function hasn't been run yet by
													#checking to see
if the first feather exists
		#check to make sure all objects needed for the program are there
		existanceCheck()

		if (exists == 1):
			#Check to make sure the values the user input are within bounds
			checkUserInput()

			if (userInputCheck == 1):
				#check to see if startup function has been run by checking the existance of the controls 
group
				if m.objExists('Controls'):
					print "Startup already run. Skipping."
				else:
					runAtStartup()
					print "Startup complete."

				print "All parts accounted for, generating feathers."
```

```
                                #Run feather generation functions
                                buildPrimaries()
                                buildSecondaries()
                                buildSecondaryCoverts()
                                buildPrimaryCoverts()
                                buildMedianCoverts()
                                buildAlulas()
                        else:
                                print "Cannot generate feathers: Missing integral parts. Check Script Editor for details."
                else:
                        print "Feathers have already been generated for this side."

#GUI button function 2
#Call the generation functions when the button to generate the motion system is pressed
def generateMotionSys(self):
        if (m.objExists(side +'_Wrist_CON') == 0):#Check to make sure function hasn't been run yet by
                                                                                                #checking to see
if the Wrist Control exists
                if m.objExists(side + '_Primary_1'):  #Check to make sure the generation function HAS been run by
                                                                                                #checking to see
if the first feather exists, if so, continue on
                        checkLocators() #Make sure locators are properly placed
                        generateSkeleton() #Create the skeleton ...
                        controlsSetup() #...and the controls

                        #Setup each of the groups of feathers
                        featherSetup('_Primary_', numPrimaries, 'Con_1', 'Con_2', '_Wing_3_JNT')
                        featherSetup('_Secondary_', numSecondaries + numTertials, 'Con_1', 'Con_2', '_Wing_2_JNT')
                        featherSetup('_PrimaryCoverts_', numPrimaryCoverts, 'Con_1', 'Con_2', '_Wing_3_JNT')
                        featherSetup('_SecondaryCoverts_', numSecondaryCoverts, 'Con_1', 'Con_2', '_Wing_2_JNT')
                        featherSetup('_MedianCoverts_', numMedianCoverts, 'Con_1', 'Con_2', '_Wing_2_JNT')
                        featherSetup('_Alula_', numAlulas, 'Con_1', 'Con_2', '_Wing_3_JNT')

                        mainControls()    #Setup the main controls

                        #check to see if Blendshapes are included. If they are, create controls for them, if not, don't add
them.
                        checkBlendshapesExistance(side + '_Primary_BLND')

                        if (blednshapesExist == 1):
                                #Create blendshape controls on the primary feathers
                                makeBlends('_Primary_', numPrimaries)
                                blendshapeSys('_Primary_', numPrimaries, '_Tip_Flex_Feathers',
'_Mid_L_Flex_Feathers')
                                print side + " Primary flex Added."
                                m.delete(side + '_Primary_BLND') #Delete the leftover blendshape
                        else:
                                print side + " Primary Blendshape missing. Didn't add primary feather flex."

                        checkBlendshapesExistance(side + '_Secondary_BLND')
                        if (blednshapesExist == 1):
                                #Create blendshape controls on the secondary feathers
                                makeBlends('_Secondary_', numSecondaries+numTertials)
                                blendshapeSys('_Secondary_', numSecondaries+numTertials, '_End_Flex_Feathers',
'_Mid_R_Flex_Feathers')
                                print side + " Secondary flex Added."
                                m.delete(side + '_Secondary_BLND') #Delete the leftover blendshape
                        else:
                                print side + " Secondary Blendshape missing. Didn't add secondary feather flex."
```

```
                foldControls() #Create controls for wingfolding

                cleanup() #Run cleanup
                print side + " Motion system generated."
        else:
                print "Cannot Generate motion system. Generate feathers first."
    else:
        print side + " Motion system already created"

#Function to create the blendshapes for each feather
def makeBlends(featherNamePAS, numFeathersPAS):
    NAME = side + featherNamePAS
    i = 1

    #While function creates a blendshape for each feather.
    while (i < (numFeathersPAS+1)):
        m.select(NAME + 'BLND', r=True)
        m.select(NAME + repr(i), tgl=True)

        m.blendShape(n= NAME + repr(i) + '_BLNDSHP')

        i=i+1

#Function to create the system for the blendshapes.
#Works similarly to the feather controls- each blendshape is keyed in proportionally to it's position in
#the wing.
def blendshapeSys(featherNamePAS, numFeathersPAS, control1PAS, control2PAS):
    NAME = side + featherNamePAS
    numFeathers = numFeathersPAS
    control1 = control1PAS
    control2 = control2PAS

    i = 1
    stepAmount = 1.0/numFeathers #Step incriment for each blednshape- Percentage by number of feathers
    j = stepAmount
    k = 1

    #Create nodes to do the math for the blendshapes
    while (i < (numFeathers+1)):
        #Create node structure
        m.createNode( 'multiplyDivide', n= NAME + repr(i) + '_MDNode1' ) #creates plus min avg node
        m.createNode( 'multiplyDivide', n= NAME + repr(i) +'_MDNode2' ) #creates plus min avg node
        m.createNode( 'plusMinusAverage', n= NAME + repr(i) +'_PMNode' ) #creates plus min avg node

        #First multiplication- Multiplies the distance control 2 is moved and the blendshape
        m.setAttr( NAME + repr(i) + '_MDNode1.input1X', -k )
        m.connectAttr( side + control2 + '.translateY',  NAME + repr(i) +'_MDNode1.input2X' )

        m.connectAttr( NAME + repr(i) + '_MDNode1.outputX',  NAME + repr(i) + '_PMNode.input1D[1]' )

        #Second Multiplication- Multiplies the distance control 1 is moved and the blendshape
        m.setAttr( NAME + repr(i) + '_MDNode2.input1X', -j )
        m.connectAttr( side + control1 + '.translateY',  NAME + repr(i) + '_MDNode2.input2X' )

        m.connectAttr( NAME + repr(i) + '_MDNode2.outputX',  NAME + repr(i) + '_PMNode.input1D[2]' )

        m.connectAttr(  NAME + repr(i) + '_PMNode.output1D', NAME + repr(i) + '_BLNDSHP.' + NAME +
'BLND' )        #Add the results
```

```
                #Incriment all the variables
                i = i + 1
                j = j + stepAmount
                k = k - stepAmount

        m.select(cl=True)

#Function that creates the wingfold control
def foldControls():
        #Add control Atribute
        m.addAttr( side + '_Shoulder_CON', longName='_AutoWingfold', defaultValue=0.0, minValue=0.0,
maxValue=10.0)
        m.setAttr( side + '_Shoulder_CON.' + '_AutoWingfold', k=1 )#Make it keyable

        #Set the default positions for all the controls. Use the GRP so that it causes no problems with the controls.
        m.setAttr(side + '_Shoulder_CON.' + '_AutoWingfold', 0 )
        m.setAttr(side + '_Shoulder_Con_GRP.rotateX', 0 )
        m.setAttr(side + '_Shoulder_Con_GRP.rotateY', 0 )
        m.setAttr(side + '_Shoulder_Con_GRP.rotateZ', 0 )
        m.setAttr(side + '_Wing2_FK_Con_GRP.rotateX', 0 )
        m.setAttr(side + '_Wing2_FK_Con_GRP.rotateY', 0 )
        m.setAttr(side + '_Wing2_FK_Con_GRP.rotateZ', 0 )
        m.setAttr(side + '_Wing3_FK_Con_GRP.rotateX', 0 )
        m.setAttr(side + '_Wing3_FK_Con_GRP.rotateY', 0 )
        m.setAttr(side + '_Wing3_FK_Con_GRP.rotateZ', 0 )

        m.setAttr(side + '_MiddleFeathers_GRP2.rotateX', 0 )
        m.setAttr(side + '_MiddleFeathers_GRP2.rotateY', 0 )
        m.setAttr(side + '_MiddleFeathers_GRP2.rotateZ', 0 )
        m.setAttr(side + '_EndFeathers_GRP2.rotateX', 0 )
        m.setAttr(side + '_EndFeathers_GRP2.rotateY', 0 )
        m.setAttr(side + '_EndFeathers_GRP2.rotateZ', 0 )
        m.setAttr(side + '_TipFeathers_GRP2.rotateX', 0 )
        m.setAttr(side + '_TipFeathers_GRP2.rotateY', 0 )
        m.setAttr(side + '_TipFeathers_GRP2.rotateZ', 0 )

        m.setAttr(side + '_Secondary_Con_2_GRP.rotateX', 0 )
        m.setAttr(side + '_Secondary_Con_2_GRP.rotateY', 0 )
        m.setAttr(side + '_Secondary_Con_2_GRP.rotateZ', 0 )
        m.setAttr(side + '_SecondaryCoverts_Con_2_GRP.rotateX', 0 )
        m.setAttr(side + '_SecondaryCoverts_Con_2_GRP.rotateY', 0 )
        m.setAttr(side + '_SecondaryCoverts_Con_2_GRP.rotateZ', 0 )
        m.setAttr(side + '_MedianCoverts_Con_2_GRP.rotateX', 0 )
        m.setAttr(side + '_MedianCoverts_Con_2_GRP.rotateY', 0 )
        m.setAttr(side + '_MedianCoverts_Con_2_GRP.rotateZ', 0 )

        #Set the driven keys for those position
        m.setDrivenKeyframe(side + '_Shoulder_Con_GRP.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Shoulder_Con_GRP.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Shoulder_Con_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing2_FK_Con_GRP.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing2_FK_Con_GRP.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing2_FK_Con_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing3_FK_Con_GRP.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing3_FK_Con_GRP.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing3_FK_Con_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_MiddleFeathers_GRP2.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_MiddleFeathers_GRP2.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
```

```
		m.setDrivenKeyframe(side + '_MiddleFeathers_GRP2.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_TipFeathers_GRP2.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_TipFeathers_GRP2.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_TipFeathers_GRP2.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_EndFeathers_GRP2.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_EndFeathers_GRP2.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_EndFeathers_GRP2.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_Secondary_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_Secondary_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_Secondary_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_SecondaryCoverts_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_SecondaryCoverts_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_SecondaryCoverts_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_MedianCoverts_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_MedianCoverts_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_MedianCoverts_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_Primary_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_Primary_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_Primary_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_PrimaryCoverts_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_PrimaryCoverts_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_PrimaryCoverts_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
		m.setDrivenKeyframe(side + '_Alula_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_Alula_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
		m.setDrivenKeyframe(side + '_Alula_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )


		if (side == "L"):
			#Set the end positions for all the controls. Use the GRP so that it causes no problems with the controls.
			m.setAttr(side + '_Shoulder_CON.' + '_AutoWingfold', 10 )
			m.setAttr(side + '_Shoulder_Con_GRP.rotateX', 0)
			m.setAttr(side + '_Shoulder_Con_GRP.rotateY', 68.065)
			m.setAttr(side + '_Shoulder_Con_GRP.rotateZ', 0)
			m.setAttr(side + '_Wing2_FK_Con_GRP.rotateX', 0)
			m.setAttr(side + '_Wing2_FK_Con_GRP.rotateY', -113.657)
			m.setAttr(side + '_Wing2_FK_Con_GRP.rotateZ', 0)
			m.setAttr(side + '_Wing3_FK_Con_GRP.rotateX', 0)
			m.setAttr(side + '_Wing3_FK_Con_GRP.rotateY', 150.692)
			m.setAttr(side + '_Wing3_FK_Con_GRP.rotateZ', 0)

			m.setAttr(side + '_MiddleFeathers_GRP2.rotateX', 0)
			m.setAttr(side + '_MiddleFeathers_GRP2.rotateY', 0)
			m.setAttr(side + '_MiddleFeathers_GRP2.rotateZ', 0)
			m.setAttr(side + '_EndFeathers_GRP2.rotateX', 0)
			m.setAttr(side + '_EndFeathers_GRP2.rotateY', 0)
			m.setAttr(side + '_EndFeathers_GRP2.rotateZ', 0)
			m.setAttr(side + '_TipFeathers_GRP2.rotateX', 0)
			m.setAttr(side + '_TipFeathers_GRP2.rotateY', 0)
```

```
        m.setAttr(side + '_TipFeathers_GRP2.rotateZ', 0)

        m.setAttr(side + '_Secondary_Con_2_GRP.rotateX', -2.956)
        m.setAttr(side + '_Secondary_Con_2_GRP.rotateY', -82.188)
        m.setAttr(side + '_Secondary_Con_2_GRP.rotateZ', 2.069)
        m.setAttr(side + '_SecondaryCoverts_Con_2_GRP.rotateX', -2.956)
        m.setAttr(side + '_SecondaryCoverts_Con_2_GRP.rotateY', -82.188)
        m.setAttr(side + '_SecondaryCoverts_Con_2_GRP.rotateZ', 2.069)
        m.setAttr(side + '_MedianCoverts_Con_2_GRP.rotateX', -0.919)
        m.setAttr(side + '_MedianCoverts_Con_2_GRP.rotateY', -64.074)
        m.setAttr(side + '_MedianCoverts_Con_2_GRP.rotateZ', -0.034)

        m.setAttr(side + '_Primary_Con_2_GRP.rotateX', 0)
        m.setAttr(side + '_Primary_Con_2_GRP.rotateY', -65.324)
        m.setAttr(side + '_Primary_Con_2_GRP.rotateZ', 0)
        m.setAttr(side + '_PrimaryCoverts_Con_2_GRP.rotateX', 0)
        m.setAttr(side + '_PrimaryCoverts_Con_2_GRP.rotateY', -65.324)
        m.setAttr(side + '_PrimaryCoverts_Con_2_GRP.rotateZ', 0)
        m.setAttr(side + '_Alula_Con_2_GRP.rotateX', 0)
        m.setAttr(side + '_Alula_Con_2_GRP.rotateY', -24.427)
        m.setAttr(side + '_Alula_Con_2_GRP.rotateZ', 0)
if (side == "R"):
        #Set the default positions for all the controls. Use the GRP so that it causes no problems with the controls.
        m.setAttr(side + '_Shoulder_CON.' + '_AutoWingfold', 10 )
        m.setAttr(side + '_Shoulder_Con_GRP.rotateX', 0)
        m.setAttr(side + '_Shoulder_Con_GRP.rotateY', -68.065)
        m.setAttr(side + '_Shoulder_Con_GRP.rotateZ', 0)
        m.setAttr(side + '_Wing2_FK_Con_GRP.rotateX', 0)
        m.setAttr(side + '_Wing2_FK_Con_GRP.rotateY', 113.657)
        m.setAttr(side + '_Wing2_FK_Con_GRP.rotateZ', 0)
        m.setAttr(side + '_Wing3_FK_Con_GRP.rotateX', 0)
        m.setAttr(side + '_Wing3_FK_Con_GRP.rotateY', -150.692)
        m.setAttr(side + '_Wing3_FK_Con_GRP.rotateZ', 0)

        m.setAttr(side + '_MiddleFeathers_GRP2.rotateX', 0)
        m.setAttr(side + '_MiddleFeathers_GRP2.rotateY', 0)
        m.setAttr(side + '_MiddleFeathers_GRP2.rotateZ', 0)
        m.setAttr(side + '_EndFeathers_GRP2.rotateX', 0)
        m.setAttr(side + '_EndFeathers_GRP2.rotateY', 0)
        m.setAttr(side + '_EndFeathers_GRP2.rotateZ', 0)
        m.setAttr(side + '_TipFeathers_GRP2.rotateX', 0)
        m.setAttr(side + '_TipFeathers_GRP2.rotateY', 0)
        m.setAttr(side + '_TipFeathers_GRP2.rotateZ', 0)

        m.setAttr(side + '_Secondary_Con_2_GRP.rotateX', -2.956)
        m.setAttr(side + '_Secondary_Con_2_GRP.rotateY', 82.188)
        m.setAttr(side + '_Secondary_Con_2_GRP.rotateZ', -2.069)
        m.setAttr(side + '_SecondaryCoverts_Con_2_GRP.rotateX', -2.956)
        m.setAttr(side + '_SecondaryCoverts_Con_2_GRP.rotateY', 82.188)
        m.setAttr(side + '_SecondaryCoverts_Con_2_GRP.rotateZ', -2.069)
        m.setAttr(side + '_MedianCoverts_Con_2_GRP.rotateX', -0.919)
        m.setAttr(side + '_MedianCoverts_Con_2_GRP.rotateY', 64.074)
        m.setAttr(side + '_MedianCoverts_Con_2_GRP.rotateZ', 0.034)

        m.setAttr(side + '_Primary_Con_2_GRP.rotateX', 0)
        m.setAttr(side + '_Primary_Con_2_GRP.rotateY', 65.324)
        m.setAttr(side + '_Primary_Con_2_GRP.rotateZ', 0)
        m.setAttr(side + '_PrimaryCoverts_Con_2_GRP.rotateX', 0)
        m.setAttr(side + '_PrimaryCoverts_Con_2_GRP.rotateY', 65.324)
```

```
                m.setAttr(side + '_PrimaryCoverts_Con_2_GRP.rotateZ', 0)
                m.setAttr(side + '_Alula_Con_2_GRP.rotateX', 0)
                m.setAttr(side + '_Alula_Con_2_GRP.rotateY', 24.427)
                m.setAttr(side + '_Alula_Con_2_GRP.rotateZ', 0)

        #Set the driven keys for those position
        m.setDrivenKeyframe(side + '_Shoulder_Con_GRP.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Shoulder_Con_GRP.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Shoulder_Con_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing2_FK_Con_GRP.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing2_FK_Con_GRP.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing2_FK_Con_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing3_FK_Con_GRP.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing3_FK_Con_GRP.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Wing3_FK_Con_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_MiddleFeathers_GRP2.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_MiddleFeathers_GRP2.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_MiddleFeathers_GRP2.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_TipFeathers_GRP2.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_TipFeathers_GRP2.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_TipFeathers_GRP2.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_EndFeathers_GRP2.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_EndFeathers_GRP2.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_EndFeathers_GRP2.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Secondary_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Secondary_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Secondary_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_SecondaryCoverts_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_SecondaryCoverts_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_SecondaryCoverts_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_MedianCoverts_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_MedianCoverts_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_MedianCoverts_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Primary_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Primary_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Primary_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_PrimaryCoverts_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_PrimaryCoverts_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_PrimaryCoverts_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' +
'_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Alula_Con_2_GRP.rotateX', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Alula_Con_2_GRP.rotateY', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )
        m.setDrivenKeyframe(side + '_Alula_Con_2_GRP.rotateZ', cd= side + '_Shoulder_CON.' + '_AutoWingfold' )

        m.setAttr(side + '_Shoulder_CON.' + '_AutoWingfold', 0 ) #Set the default position to 0

#Function that cleans up any extra stuff that needs to be done such as locking controlers,
#hiding things, locking visibility, deleting extras...
def cleanup():
```

```
#IKFK Switch Cleanup
i=0

#Create array variables for simplicities sake
scale = [".sx", ".sy", ".sz"]
translate = [".translateX", ".translateY", ".translateZ"]
rotate = [".rotateX", ".rotateY", ".rotateZ"]

while (i<3):
        # Lock an attribute to prevent further modification on FK controls
        m.setAttr(side + '_Wing2_FK_CON' + scale[i], k=0, lock=True)
        m.setAttr(side + '_Wing2_FK_CON' + translate[i], k=0, lock=True)

        m.setAttr(side + '_Wing3_FK_CON' + scale[i], k=0, lock=True)
        m.setAttr(side + '_Wing3_FK_CON' + translate[i], k=0, lock=True)

        m.setAttr(side + '_Shoulder_CON' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_Wrist_CON' + scale[i], lock=True, k=0 )

        #Lock spread controls
        m.setAttr(side + '_TipFeathers_CON' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_MiddleFeathers_CON' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_EndFeathers_CON' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_TipFeathers_CON' + translate[i], lock=True, k=0 )
        m.setAttr(side + '_MiddleFeathers_CON' + translate[i], lock=True, k=0 )
        m.setAttr(side + '_EndFeathers_CON' + translate[i], lock=True, k=0 )

        #Lock Flex controls
        m.setAttr(side + '_Tip_Flex_Feathers' + rotate[i], lock=True, k=0 )
        m.setAttr(side + '_Mid_L_Flex_Feathers' + rotate[i], lock=True, k=0 )
        m.setAttr(side + '_Mid_R_Flex_Feathers' + rotate[i], lock=True, k=0 )
        m.setAttr(side + '_End_Flex_Feathers' + rotate[i], lock=True, k=0 )
        m.setAttr(side + '_Tip_Flex_Feathers' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_Mid_L_Flex_Feathers' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_Mid_R_Flex_Feathers' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_End_Flex_Feathers' + scale[i], lock=True, k=0 )

        #Lock Pole vector
        m.setAttr(side + '_Wing_Pole_CON' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_Wing_Pole_CON' + rotate[i], lock=True, k=0 )

        #Lock low level controls
        m.setAttr(side + '_Primary_Con_1' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_Primary_Con_1' + translate[i], lock=True, k=0 )
        m.setAttr(side + '_Primary_Con_2' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_Primary_Con_2' + translate[i], lock=True, k=0 )
        m.setAttr(side + '_PrimaryCoverts_Con_1' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_PrimaryCoverts_Con_1' + translate[i], lock=True, k=0 )
        m.setAttr(side + '_PrimaryCoverts_Con_2' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_PrimaryCoverts_Con_2' + translate[i], lock=True, k=0 )

        m.setAttr(side + '_Secondary_Con_1' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_Secondary_Con_1' + translate[i], lock=True, k=0 )
        m.setAttr(side + '_Secondary_Con_2' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_Secondary_Con_2' + translate[i], lock=True, k=0 )
        m.setAttr(side + '_SecondaryCoverts_Con_1' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_SecondaryCoverts_Con_1' + translate[i], lock=True, k=0 )
        m.setAttr(side + '_SecondaryCoverts_Con_2' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_SecondaryCoverts_Con_2' + translate[i], lock=True, k=0 )
```

```
        m.setAttr(side + '_MedianCoverts_Con_1' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_MedianCoverts_Con_1' + translate[i], lock=True, k=0 )
        m.setAttr(side + '_MedianCoverts_Con_2' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_MedianCoverts_Con_2' + translate[i], lock=True, k=0 )

        m.setAttr(side + '_Alula_Con_1' + scale[i], lock=True, k=0 )
        m.setAttr(side + '_Alula_Con_1' + translate[i], lock=True, k=0 )

        i=i+1


m.setAttr( side + '_Wing_1_JNT.visibility', 0 )                 #Turn off visibility for the main joint set
m.setAttr( side + '_Wing_IK.visibility', 0 )            #Turn off visibility for the IK

#lock visibility on feather controls
m.setAttr(side + '_MedianCoverts_Con_1.visibility', lock=True, k=0 )
m.setAttr(side + '_MedianCoverts_Con_2.visibility', lock=True, k=0 )
m.setAttr(side + '_SecondaryCoverts_Con_1.visibility', lock=True, k=0 )
m.setAttr(side + '_SecondaryCoverts_Con_2.visibility', lock=True, k=0 )
m.setAttr(side + '_Secondary_Con_1.visibility', lock=True, k=0 )
m.setAttr(side + '_Secondary_Con_2.visibility', lock=True, k=0 )
m.setAttr(side + '_PrimaryCoverts_Con_1.visibility', lock=True, k=0 )
m.setAttr(side + '_PrimaryCoverts_Con_2.visibility', lock=True, k=0 )
m.setAttr(side + '_Primary_Con_1.visibility', lock=True, k=0 )
m.setAttr(side + '_Primary_Con_2.visibility', lock=True, k=0 )
m.setAttr(side + '_Alula_Con_1.visibility', lock=True, k=0 )
m.setAttr(side + '_Alula_Con_2.visibility', lock=True, k=0 )

#Lock Flex controls and visibility on flex controls
m.setAttr(side + '_Tip_Flex_Feathers.translateX', lock=True, k=0 )
m.setAttr(side + '_Mid_L_Flex_Feathers.translateX', lock=True, k=0 )
m.setAttr(side + '_Mid_R_Flex_Feathers.translateX', lock=True, k=0 )
m.setAttr(side + '_Tip_Flex_Feathers.translateZ', lock=True, k=0 )
m.setAttr(side + '_Mid_L_Flex_Feathers.translateZ', lock=True, k=0 )
m.setAttr(side + '_Mid_R_Flex_Feathers.translateZ', lock=True, k=0 )
m.setAttr(side + '_End_Flex_Feathers.translateX', lock=True, k=0 )
m.setAttr(side + '_End_Flex_Feathers.translateZ', lock=True, k=0 )
m.setAttr(side + '_Tip_Flex_Feathers.visibility', lock=True, k=0 )
m.setAttr(side + '_Mid_L_Flex_Feathers.visibility', lock=True, k=0 )
m.setAttr(side + '_Mid_R_Flex_Feathers.visibility', lock=True, k=0 )
m.setAttr(side + '_End_Flex_Feathers.visibility', lock=True, k=0 )

if (side == "L"):
        m.setAttr(side + '_Mid_R_Flex_Feathers.translateZ', lock=True, k=0 )
if (side == "R"):
        m.setAttr(side + '_Mid_L_Flex_Feathers.translateZ', lock=True, k=0 )

# Set Limits on Feather curl
m.select( side + '_Tip_Flex_Feathers')
m.transformLimits( ty=(-1.3, 1.3), ety=(True, True))
m.select( side + '_Mid_L_Flex_Feathers')
m.transformLimits( ty=(-1.3, 1.3), ety=(True, True))
m.select( side + '_Mid_R_Flex_Feathers')
m.transformLimits( ty=(-1.3, 1.3), ety=(True, True))
m.select( side + '_End_Flex_Feathers')
m.transformLimits( ty=(-1.3, 1.3), ety=(True, True))

#Delete extraneous
```

```
	i=1
	while (i < 5):
		m.delete(side + '_Wing_' + repr(i))
		i=i+1

	#If the motion system function has been run, delete the extraneous bases
	if m.objExists('L_Wrist_CON'):
		if m.objExists('R_Wrist_CON'):
			m.delete('L_PrimaryBase')
			m.delete('R_PrimaryBase')
			m.delete('L_SecondaryBase')
			m.delete('R_SecondaryBase')
			m.delete('R_PrimaryCovertsBase')
			m.delete('L_PrimaryCovertsBase')
			m.delete('R_SecondaryCovertsBase')
			m.delete('L_SecondaryCovertsBase')
			m.delete('R_MedianCovertsBase')
			m.delete('L_MedianCovertsBase')
			m.delete('R_AlulaBase')
			m.delete('L_AlulaBase')
			m.delete('TertialBase')
			m.delete('CONBASE')
			m.delete('BoxConBase')
			m.delete('ArrowConBase')
			m.delete('MainConBase')
			m.delete('MoveAllConBase')
			m.delete('CurveConBase')

	m.select(cl=True)

#This function checks to make sure the integral parts of the rig are there
#If any are missing it will throw an error and not allow the code to proceed
def existanceCheck():
	global exists
	exists = 1

	#check to see if all the feathers exist
	if (m.objExists('R_PrimaryBase') == 0):
		exists = 0
		print "R Primary missing"

	if (m.objExists('L_PrimaryBase') == 0):
		exists = 0
		print "L Primary missing"

	if (m.objExists('L_PrimaryCovertsBase') == 0):
		exists = 0
		print "L Primary Covert missing"

	if (m.objExists('R_PrimaryCovertsBase') == 0):
		exists = 0
		print "R Primary Covert missing"

	if (m.objExists('R_SecondaryBase') == 0):
		exists = 0
		print "R Secondary missing"

	if (m.objExists('L_SecondaryBase') == 0):
		exists = 0
```

```
		print "L Secondary missing"

if (m.objExists('R_SecondaryCovertsBase') == 0):
		exists = 0
		print "R Secondary Coverts missing"

if (m.objExists('L_SecondaryCovertsBase') == 0):
		exists = 0
		print "L Secondary Coverts missing"

if (m.objExists('R_MedianCovertsBase') == 0):
		exists = 0
		print "R Median Coverts missing"

if (m.objExists('L_MedianCovertsBase') == 0):
		exists = 0
		print "L Median Coverts missing"

if (m.objExists('R_AlulaBase') == 0):
		exists = 0
		print "R Alula missing"

if (m.objExists('L_AlulaBase') == 0):
		exists = 0
		print "L Alula missing"

if (m.objExists('TertialBase') == 0):
		exists = 0
		print "Tertial missing"

#Check for Controls existance
if (m.objExists('CONBASE') == 0):
		exists = 0
		print "CONBASE missing"

if (m.objExists('BoxConBase') == 0):
		exists = 0
		print "BoxConBase missing"

if (m.objExists('ArrowConBase') ==0):
		exists = 0
		print "ArrowConBase missing"

if (m.objExists('MainConBase') == 0):
		exists = 0
		print "MainConBase missing"

if (m.objExists('MoveAllConBase') == 0):
		exists = 0
		print "MoveAllConBase missing"

if (m.objExists('CurveConBase') == 0):
		exists = 0
		print "CurveConBase missing"

if (m.objExists('Wing_Flex_1') == 0):
		exists = 0
		print "Wing_Flex_1 missing"
```

```
	if (m.objExists('Wing_Flex_2') == 0):
		exists = 0
		print "Wing_Flex_2 missing"

	#Check Locators
	i=1
	while (i < 5):
		if (m.objExists(side + '_Wing_' + repr(i)) ==0):
			exists = 0
			print side + '_Wing_' + repr(i) + " missing"

		i = i+1

#Function to check if blendshapes exist
def checkBlendshapesExistance(blendshapeNamePAS):
	global blednshapesExist
	blednshapesExist = 1

	#check to see if all the feathers exist
	if (m.objExists(blendshapeNamePAS) == 0):
		blednshapesExist = 0

#Function that checks if the locators are located all in the same plane or not
def checkLocators():
	#Check Locators
	setSide()

	#Find out what plane the y coordinates are in
	tempVar = Loc1Coord[1]
	x=1

	if (Loc2Coord[1] != tempVar):
		print "_____Second locator not in plane_______"
		x=0

	if (Loc3Coord[1] != tempVar):
		print "_____Third locator not in plane_______"
		x=0

	if (Loc4Coord[1] != tempVar):
		print "_____Fourth locator not in plane_______"
		x=0

	if (x != 1):
		print "Warning, One or more locators are not in the same plane. This could cause unexpected results in the
rig."

#Function to make sure the user hasn't given any numbers that are out of bounds such as 0 or 1.
#The minimum number of feathers is set to three.
def checkUserInput():
	global userInputCheck
	userInputCheck =1

	if (numPrimaries < 3):
		userInputCheck = 0
		print "Too few Primaries"

	if (numSecondaries < 3):
		userInputCheck = 0
```

```
                print "Too few Secondaries"

        if (numPrimaryCoverts < 3):
                userInputCheck = 0
                print "Too few Primary Coverts"

        if (numSecondaryCoverts < 3):
                userInputCheck = 0
                print "Too few Secondary Coverts"

        if (numMedianCoverts < 3):
                userInputCheck = 0
                print "Too few Secondary Coverts"

#Call the program on startup
featherGenerator()
```