Here is a backup file for my scripts

**RABBIT**

//Head//

polySphere -r 1 -sx 10 -sy 10 -ax 0 1 0 -cuv 2 -ch 1 ;

move -r 0 16 0 ;

scale -r 2.8 2.8 2.8 ;

select -r pSphere1.f[73] pSphere1.f[79] ;

polyExtrudeFacet -constructionHistory 1 -keepFacesTogether 1 -pvx -7.748603825e-008 -pvy 18.28809561 -pvz -0.449139019 -divisions 1 -twist 0 -taper 1 -off 0 -thickness 0 -smoothingAngle 30 pSphere1.f[73] pSphere1.f[79];

// Result: polyExtrudeFace5 //

setAttr "polyExtrudeFace5.localTranslate" -type double3 0 0 8.145757 ;

//Body//

polySphere -r 1 -sx 10 -sy 10 -ax 0 1 0 -cuv 2 -ch 1 ;

move -r 0 8 0 ;

scale -r 3.5 3.5 3.5 ;

select -r pSphere2.vtx[91] ;

select -tgl pSphere2.vtx[86] ;

select -tgl pSphere2.vtx[85] ;

select -add pSphere2.vtx[80:89] ;

move -r 0 2.63889 0 ;

select -r pSphere2.f[8] ;

select -tgl pSphere2.f[4] ;

PolyExtrude;

hilite pSphere2.f[4] pSphere2.f[8] ;

selectMode -component ;

select -r pSphere2.f[4] pSphere2.f[8] ;

polyExtrudeFacet -constructionHistory 1 -keepFacesTogether 1 -pvx -1.043081284e-007 -pvy 4.9198713 -pvz 0.604610227 -divisions 1 -twist 0 -taper 1 -off 0 -thickness 0 -smoothingAngle 30 pSphere2.f[4] pSphere2.f[8];

// Result: polyExtrudeFace6 //

move -r 0 -3.744942 0 ;

scale -r -p -2.08616e-007cm 1.174928cm 0.60461cm 1 1e-005 1 ;

//Arms//

polyCylinder -r 1 -h 2 -sx 10 -sy 1 -sz 1 -ax 0 1 0 -rcp 0 -cuv 3 -ch 1;

move -r 0 12.51 0 ;

scale -r 0.457 8.353 0.457 ;

rotate -r 0 0 90 ;

//Right foot & Left foot//

polySphere -r 1 -sx 10 -sy 10 -ax 0 1 0 -cuv 2 -ch 1 ;

move -r -1.413 1.137 1.775 ;

scale -r 1 1 2.822 ;

polySphere -r 1 -sx 10 -sy 10 -ax 0 1 0 -cuv 2 -ch 1 ;

move -r 1.413 1.137 1.775 ;

scale -r 1 1 2.822 ;

//Right & Left hands//

polySphere -r 1 -sx 10 -sy 10 -ax 0 1 0 -cuv 2 -ch 1 ;

move -r -7.939 12.535 0 ;

scale -r 1.752 0.601 1.219 ;

polySphere -r 1 -sx 10 -sy 10 -ax 0 1 0 -cuv 2 -ch 1 ;

move -r 7.939 12.535 0 ;

scale -r 1.752 0.601 1.219 ;

//Tail//

polySphere -r 1 -sx 10 -sy 10 -ax 0 1 0 -cuv 2 -ch 1 ;

move -r 0 7.6 -3.6 ;

//Nose//

polySphere -r 1 -sx 10 -sy 10 -ax 0 1 0 -cuv 2 -ch 1 ;

move -r 1 15.809 2.725 ;

scale -r 0.523 0.302 0.28 ;

//Right & Left Eyes//

polySphere -r 1 -sx 10 -sy 10 -ax 0 1 0 -cuv 2 -ch 1 ;

move -r -0.814 16.623 2.505 ;

scale -r 0.276 0.499 0.159 ;

rotate -r -14.799 -12.884 3.371 ;

polySphere -r 1 -sx 10 -sy 10 -ax 0 1 0 -cuv 2 -ch 1 ;

move 0.814 16.623 2.505 ;

scale -r 0.276 0.499 0.159 ;

rotate -r -14.799 12.884 3.371 ;

**STAR RANDOMIZATION**

proc SpiralStars(int $numStars, float $rotation) {

string $sels [] = `ls -sl`;

string $sel = $sels[0];

for($i=0; $i<$numStars; $i++)

{

string $newSel [];

$newSel = `duplicate -rr $sel` ;

$sel = $newSel [0];

move -r -os -wd 0 0.95 0 $newSel [0];

rotate -r -os -fo 0 $rotation 0 $newSel [0];

}

}

SpiralStars(20, 25);

**STAR RANDOMIZATION2.0-Spiral**

proc SpiralStars(int $numStars, float $rotation) {

string $sels [] = `ls -sl`;

string $sel = $sels[0];

for($i=0; $i<$numStars; $i++)

{

string $newSel [];

$newSel = `duplicate -rr $sel` ;

$sel = $newSel [0];

move -r -os -wd 2 2 2 $newSel [0];

rotate -r -os -fo 0 $rotation 0 $newSel [0];

}

}

SpiralStars(15, 25);

**STAR RANDOMIZATION3.0-Cluster**

//Randomization Script//

proc Randomize(int $num, int $xRange, int $yRange, int $zRange) {

string $sels[] = `ls -sl` ;

for ($i=0; $i<size($sels); $i++) {

for ($j=0; $j<$num; $j++) {

float $xPos = `rand ($xRange\*-1) $xRange` ;

float $yPos = `rand ($yRange\*-1) $yRange` ;

float $zPos = `rand ($zRange\*-1) $zRange` ;

string $obj[] = `duplicate -rr $sels[$i]` ;

move $xPos $yPos $zPos $obj[0] ;

rotate `rand 360` `rand 360` `rand 360` $obj[0] ;

}

}

}

Randomize (50, 50, 50, 50);

**RENAME SCRIPT**

proc Rename (string $name)

{

string $sels[] = `ls -sl`;

string $name = "Test\_###\_Geo";

string $buffer[];

int $numTokens;

$numTokens = `tokenize $name "#" $buffer`;

if($numTokens ! = 2) {

error("Too much hash!");

}

for($i=0; $i<size($sels); $i++)

{

string $newName;

$newName = $buffer [0] + ($i+ 1) + $buffer [1]

rename $sels[$i] $newName;

}

print ($buffer);

}

Rename("WakeUp\_#\_ClassIsAlmostOver);

global proc ControlWindow() {

//builds the window UI

//assign object name for window UI object

string $mWindow = "ctrlWindow";

//delete the window if it has already been created

if(`window -exists $mWindow` == true)

deleteUI -window $mWindow;

//create the main window and and column for controls

$mWindow = `window -title "Control Creator" -sizeable 0 $mWindow`;

string $mColumn = `columnLayout -adjustableColumn 1 -columnAttach "both" 2 -parent $mWindow`;

//create buttons and fields

string $circleBtn = `button -parent $mColumn -label "Circle"`;

string $squareBtn = `button -parent $mColumn -label "Square"`;

string $cubeBtn = `button -parent $mColumn -label "Cube"`;

string $txtField = `textFieldGrp -parent $mColumn -label "Control Name" -placeholderText "Enter name"`;

string $floatField = `floatSliderGrp -parent $mColumn -label "Size" -minValue 0 -maxValue 10 -field 1 -value 1`;

//use the edit flag on each button to add the commands. could not assign previously when built

//because the text and float field groups had not yet been created.

button -e -command ("CreateControl(\"circle\", \"" + $txtField + "\", \"" + $floatField + "\")") $circleBtn;

button -e -command ("CreateControl(\"square\", \"" + $txtField + "\", \"" + $floatField + "\")") $squareBtn;

button -e -command ("CreateControl(\"cube\", \"" + $txtField + "\", \"" + $floatField + "\")") $cubeBtn;

window -e -width 400 -height 125 $mWindow;

showWindow($mWindow);

}

global proc string CreateControl(string $type, string $nameField, string $scaleField) {

//creates a control of specified type

//query controls to get field values and assign to a variable

float $scale = `floatSliderGrp -q -value $scaleField`;

string $name = `textFieldGrp -q -text $nameField`;

//generate generic name if none is provided in text field

if($name == "")

$name = "Generic";

$name = $name + "\_Ctrl";

//create an additional variable for the control

string $ctrl;

//create specified control type

if ($type == "circle") {

string $tCtrl[] = `circle -c 0 0 0 -nr 0 1 0 -sw 360 -r 1 -d 3 -ut 0 -tol 0.01 -s 8 -ch 1`;

$ctrl = $tCtrl[0];

}

else if($type == "square") {

$ctrl = `curve -d 1 -p 1 0 -1 -p -1 0 -1 -p -1 0 1 -p 1 0 1 -p 1 0 -1 -k 0 -k 1 -k 2 -k 3 -k 4`;

}

else if($type == "cube") {

$ctrl = `curve -d 1 -p 0.5 0.5 0.5 -p -0.5 0.5 0.5 -p -0.5 -0.5 0.5 -p 0.5 -0.5 0.5 -p 0.5 0.5 0.5 -p 0.5 0.5 -0.5 -p 0.5 -0.5 -0.5 -p 0.5 -0.5 0.5 -p 0.5 -0.5 -0.5 -p -0.5 -0.5 -0.5 -p -0.5 -0.5 0.5 -p -0.5 0.5 0.5 -p -0.5 0.5 -0.5 -p -0.5 -0.5 -0.5 -p -0.5 0.5 -0.5 -p 0.5 0.5 -0.5 -k 0 -k 1 -k 2 -k 3 -k 4 -k 5 -k 6 -k 7 -k 8 -k 9 -k 10 -k 11 -k 12 -k 13 -k 14 -k 15` ;

}

//scale the object globally using the float field value

scale -r $scale $scale $scale $ctrl;

//rename control and delete history

$ctrl = `rename $ctrl $name`;

delete -ch $ctrl;

//return control name

return $ctrl;

}

global proc CreateToolbox() {

string $mWindow = "myTooBox";

if(`window -exists $mWindow`)

deleteUI -window $mWindow;

$mWindow = `window -cc "CloseToolUI()" $mWindow`;

$mColLayout = `columnLayout -parent $mWindow`;

$ctrlBtn = `button -label "Controls" -c "ControlBtn()"`;

showWindow($mWindow);

}

global proc ControlsBtn() {

source myControls;

ControlWindow();

}

global proc CloseToolUI() {

if(`window -exists "ctrlWindow"`)

deleteUI -window "ctrlWindow";

}

CreateToolbox();

// Create a new template.

//

if (`uiTemplate -exists ExampleTemplate`) {

deleteUI -uiTemplate ExampleTemplate;

}

uiTemplate ExampleTemplate;

button -defineTemplate ExampleTemplate

-width 100 -height 40;

frameLayout -defineTemplate ExampleTemplate

-borderVisible true -labelVisible false;

// Create a window and apply the template.

//

string $window = `window`;

setUITemplate -pushTemplate ExampleTemplate;

columnLayout -rowSpacing 5;

frameLayout;

columnLayout;

button -label "One";

button -label "Two";

button -label "Three";

setParent ..;

setParent ..;

frameLayout;

columnLayout;

button -label "Red";

button -label "Green";

button -label "Blue";

setParent ..;

setParent ..;

setUITemplate -popTemplate;

showWindow $window;

//Double parent contraints

parentConstraint -sr x -sr y -sr z -mo;

parentConstraint -st x -st y -st z -mo;

//

string $sels[] = `ls -sl`;

for ($sel in $sels) {

setAttr ($sel + ".translateY") 0;

setAttr ($sel + ".translateZ") 0;

setAttr ($sel + ".jointOrientX") 0;

setAttr ($sel + ".jointOrientY") 0;

}

**//Create FK Groups and Controls//**

SelectHierarchy;

string $theJoints[] = `ls -sl`;

int $theArraySize = size($theJoints);

for ( $i = 0 ; $i < $theArraySize; $i++ )

{

$theCircleName = $theJoints[$i] + "\_CTRL";

circle -n $theCircleName;

$theCircle = `ls -sl`;

string $theGroupName = $theJoints[$i] + "\_GRP";

group -n $theGroupName;

$theGroup = `ls -sl`;

select $theJoints[$i];

matchTransform $theGroupName $theJoints[$i];

select $theGroup;

if($i>0){

$theParent = `listRelatives - p $theJoints[$i]`;

$thePrevCircle = $theParent[0] + "\_CTRL";

select -tgl $thePrevCircle;

parent;

}

parentConstraint $theCircleName $theJoints[$i];

scaleConstraint -sc off $theCircleName $theJoints[$i];

}

**// Renamer//**

proc string[] Rename(string $name)

{

string $sels[] = `ls -sl`;

string $buffer[];

int $numTokens;

string $returnNames[];

$numTokens = `tokenize $name "#" $buffer`;

if($numTokens != 2){

error("ERROR!");

}

for($i=0; $i<size($sels); $i++){

string $newName;

$newName = $buffer[0] + ($i+1) + $buffer[1];

stringArrayInsertAtIndex(size($returnNames),

$returnNames,

`rename $sels[$i] $newName`);

}

return $returnNames;

}

Rename("FK\_#\_Joint");

**//Freeze Transforms//**

makeIdentity -apply true -t 1 -r 1 -s 1 -n 0 -pn 1;

**//Delete History//**

delete -ch;

**///Python FK Hierarchy Build///**

import maya.cmds as cmds

cmds.SelectHierarchy()

theJoints = cmds.ls(sl = True)

theArraySize = len(theJoints)

for i in range(len(theJoints)):

theCircleName = theJoints[i] + "\_CTRL"

cmds.circle(n = theCircleName)

theCircle = cmds.ls (sl = True)

theGroupName = theJoints[i] + "\_GRP"

cmds.group (n = theGroupName)

theGroup = cmds.ls(sl = True)

cmds.select(theJoints[i])

cmds.matchTransform(theGroupName, theJoints[i])

cmds.select(theGroup)

if(i>0):

theParent = cmds.listRelatives(theJoints[i], p = True)

thePrevCircle = theParent[0] + "\_CTRL"

cmds.select(thePrevCircle, tgl = True)

cmds.parent()

cmds.parentConstraint(theCircleName, theJoints[i])

cmds.scaleConstraint(theCircleName, theJoints[i], sc = False)

**//Python Broken FK build//**

import maya.cmds as cmds

cmds.SelectHierarchy()

theJoints = cmds.ls(sl = True)

theArraySize = len(theJoints)

# Create Master Control

masterControl = theJoints[0] + "\_MasterControl"

cmds.spaceLocator(n = masterControl, p = [0, 0, 0])

cmds.matchTransform (masterControl, theJoints[0])

cmds.addAttr (masterControl, ln = "Follow", at = "enum", en = "Master:Hierarchy:")

cmds.setAttr (masterControl + ".Follow", e = True, keyable = True)

# Create Group for Master Control

masterGroup = cmds.group(n = theJoints[0] + "\_MasterGroup")

# These are listed arrays

theGroups = []

theControls = []

# This function will group together the controls and groups, constrain and name them appropriately, and establish set driven keys

for i in range(len(theJoints)):

theCircleName = theJoints[i] + "\_CTRL"

cmds.circle(n = theCircleName)

theCircle = cmds.ls (sl = True)

theControls.extend(theCircle)

theGroupName = theJoints[i] + "\_GRP"

cmds.group (n = theGroupName)

theGroup = cmds.ls(sl = True)

theGroups.extend(theGroup)

cmds.select(theJoints[i])

cmds.matchTransform(theGroupName, theJoints[i])

cmds.select(theGroup)

if(i==0):

translate = cmds.parentConstraint(masterControl, theGroup, sr = ["x", "y", "z"], mo = 1)

rotateable = cmds.parentConstraint(masterControl, theGroup, st = ["x", "y", "z"], mo = 1)

else:

theParent = cmds.listRelatives(theJoints[i], p = True)

thePrevCircle = theParent[0] + "\_CTRL"

translate = cmds.parentConstraint(masterControl, theControls [i-1], theGroup, sr = ["x", "y", "z"], mo = 1)

rotateable = cmds.parentConstraint(masterControl, theControls [i-1], theGroup, st = ["x", "y", "z"], mo = 1)

constraints = [translate[0], rotateable[0]]

for j in range (2):

cmds.setAttr(masterControl + ".Follow", j)

for k in range (len(constraints)):

if k == 0:

reverse = 1

else :

reverse = 0

cmds.setAttr(constraints [j] + "." + masterControl + "W0", reverse)

cmds.setAttr(constraints [j] + "." + theControls [i-1] + "W1", k)

cmds.setDrivenKeyframe(constraints [j] + "." + masterControl + "W0", cd = masterControl + ".Follow")

cmds.setDrivenKeyframe(constraints [j] + "." + theControls [i-1] + "W1", cd = masterControl + ".Follow")

cmds.parentConstraint(theCircleName, theJoints[i])

cmds.scaleConstraint(theCircleName, theJoints[i], sc = False)

cmds.select(theGroups, r = True)

brokenGroup = cmds.group(p = masterGroup, n = theJoints[0] + "\_Broken FK GROUP")

cmds.scaleConstraint(masterControl, brokenGroup)