Maya Mel

From bernie's

Contents

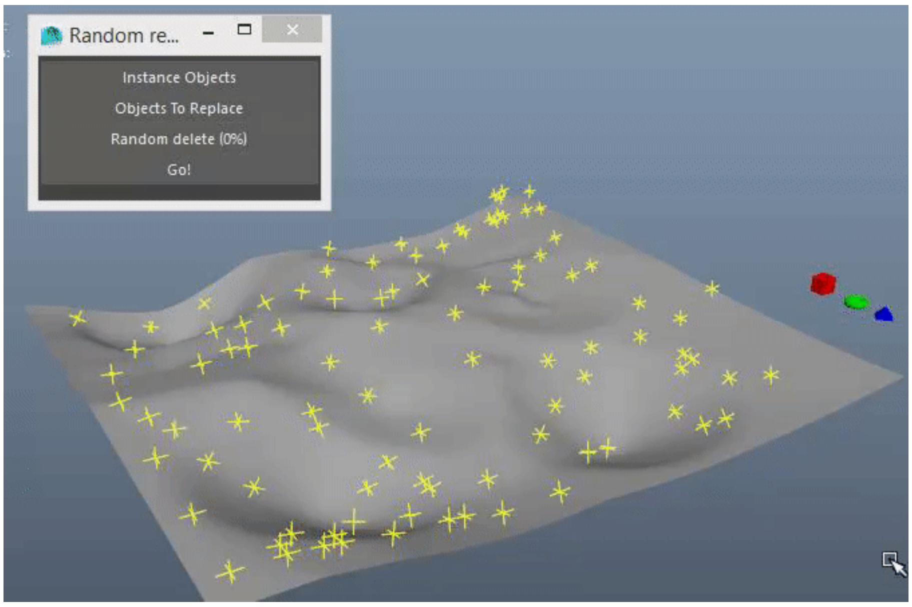
- 1 My mels
  - 1.1 Easy install tip
  - 1.2 Random replace with instances
  - 1.3 add equidistant locators on an edge selection
  - 1.4 vray add user ids on selection shapes
  - 1.5 vray add random user scalar attribute to selection
  - 1.6 vray add random colorful material id per shader
  - 1.7 Select all objects constraining an object
  - 1.8 Average skin weights from selected vertices to last vertex
  - 1.9 Show all maya icons
  - 1.10 add edge loops from single selected rings edges
  - 1.11 Hide things that are not in control balls
  - 1.12 Add .455 gamma to fileTextures
  - 1.13 Add camera center of interest locator
  - 1.14 Select faces by angle (slow-ish)
  - 1.15 Select similar shells
  - 1.16 Select similar to last object
  - 1.17 Stupid Pos/Rot/Scale Export & Stupid shader assignement export
  - 1.18 Save selection to temp text file
  - 1.19 Save weights to text file
  - 1.20 Simple incremental save w/ prompt
  - 1.21 Simple renderview batch
  - 1.22 Particle mover
  - 1.23 Turn all nucleuses on/off
  - 1.24 Photometric camera & studio lights
  - 1.25 Save render view image to Desktop as png
  - 1.26 add miLabels
  - 1.27 Place circle at base of paths
  - 1.28 'Julienne cut': multi polycut across shapes
  - 1.29 Scale/Rot/Move Manip to screenSpace (camera angle)
  - 1.30 Locators on normals
  - 1.31 Uniform weight for shells
  - 1.32 Recursive Parent
  - 1.33 Reconnect stuff
    - 1.33.1 Reconnect follicles
    - 1.33.2 Reconnect motionpaths
  - 1.34 Zero Out
  - 1.35 Select influenced bones / get skinned meshes from bone
  - 1.36 Quick Change wire color
  - 1.37 Match shape
  - 1.38 Delete groups w/o children
  - 1.39 Emit from facing ratio
  - 1.40 Random Color Viewport (à la 3dsmax)
  - 1.41 Set viewport color on selected obj/vertices
  - 1.42 postFrame status
  - 1.43 Tracking -> Maya to After
    - 1.43.1 Maya Mel
    - 1.43.2 Script After (Panel)
  - 1.44 Get and keyframe object speed
  - 1.45 Duplicates current cam and tear off copy
  - 1.46 Charge un dossier de XPM dans le renderview
  - 1.47 Reverse Selection
  - 1.48 Bake To Clamp Keys
  - 1.49 Merge to shape
  - 1.50 Linear Align Objects
    - 1.50.1 Space attributes blender
  - 1.51 PNG snapshot of current cam
  - 1.52 Right click in shelf (<maya2010)
  - 1.53 Remove useless keys
  - 1.54 Create ID Shaders
  - 1.55 ikSpline stretch
  - 1.56 Curve Text Captions
  - 1.57 Rounding numbers in mel
  - 1.58 Super Snapshot
  - 1.59 Particules or Objects to Curve
  - 1.60 Selection to shelf 2
- 2 Snippets / Pastebins
  - 2.1 Sur le net
  - 2.2 Annotate (arrow from obj to obj)
  - 2.3 Random rotates
  - 2.4 Random select 1/2
  - 2.5 Random translates
  - 2.6 Array procs
  - 2.7 Edge to Loc
  - 2.8 Find Centroid
  - 2.9 Faces to obj
  - 2.10 Remove useless edges

My mels

Easy install tip

```
//stolen from advanced skeleton file and other scripts
//basically call 'whatIs' on a procedure that you just sourced so you know when the current mel file is
//useful for dragging and dropping files in viewport for installation

global proc installDir (){}
string $p = `whatIs installDir`;

string $fullPath=`substring $p 25 999`;
string $buffer[];
string $slash="/";
if (`gmatch $p "*\\\\*"`)//sourced from ScriptEditor
        $slash="\\";
int $numTok=`tokenize $fullPath $slash $buffer`;
int $numLetters=size($fullPath);
int $numLettersLastFolder=size($buffer[$numTok-1]);
string $scriptLocation=`substring $fullPath 1 ($numLetters-$numLettersLastFolder)`;
print $scriptLocation;
```

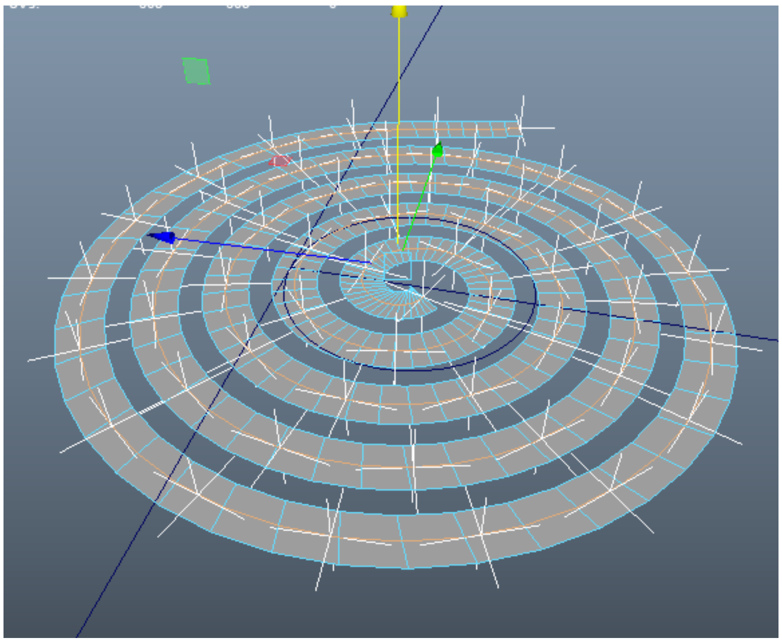
Random replace with instances



```
global float $deletepercentage;

if(`objExists("instances_set")`){delete "instances_set";};
if(`objExists("replacingobjs_set")`){delete "replacingobjs_set";};

string $window = `window -menuBar true -title "Random replace"`;
columnLayout -adjustableColumn true;
$a = `button -label "Instance Objects"`;
button -e -command ("setSet(\"instances_set\",\""+$a+"\",\"Instance Objects\")") $a;
$b = `button -label "Objects To Replace"`;
button -e -command ("setSet(\"replacingobjs_set\",\""+$b+"\",\"Objects To Replace\")") $b;
$c = `button -label ("Random delete ("+$deletepercentage+"%)")`;
button -e -command ("setDelete(\""+$c+"\")") $c;
button -label "Go!" -command ("replaceWithInstances;deleteUI -window \""+$window+"\";");
showWindow $window;
proc setSet(string $name,string $button,string $buttonString){
    string $selectedElmts[] = `ls -sl -tr`;
    if(`objExists $name`){
        delete $name;
    }
    sets -n $name $selectedElmts;
    button -e -label ($buttonString+" ("+size($selectedElmts)+")") $button;
    select -cl;
}
proc setDelete(string $btn){
    global float $deletepercentage;
    string $text;
    string $result = `promptDialog
    -title "Delete random %"
    -message "Set chance of being deleted (0-100):"
    -button "OK" -button "Cancel"
    -defaultButton "OK" -cancelButton "Cancel"
    -dismissString "Cancel"`;

    if ($result == "OK") {
        $text = `promptDialog -query -text`;
        $deletepercentage = float($text);
        button -e -label ("Random delete("+$deletepercentage+"%)") $btn;
    }
}

global proc replaceWithInstances(){
    global float $deletepercentage;
    string $instances[] = `listConnections -s 1 -d 0 -p 0 -c 0 "instances_set"`;
    string $objs[] = `listConnections -s 1 -d 0 -p 0 -c 0 "replacingobjs_set"`;
    int $end = size($objs)-1;
    global string $gMainProgressBar;  // This is defined on maya startup
    progressBar -edit -beginProgress -isInterruptable true -status "Replacing with instances" -maxValue $end $gMainProgressBar;
    refresh -su 1;
    for($i = 0;$i<=$end;$i++){
        if(`progressBar -query -isCancelled $gMainProgressBar`)
        break;
        if(`rand 1` > ($deletepercentage/100)){
            select -r $objs[$i] $instances[int(floor(rand(size($instances))))] ;
            replaceObjects 1 1 1 1;
        }else{
            delete $objs[$i];
        }
        progressBar -edit -step 1 $gMainProgressBar;
    }
    progressBar -edit -endProgress $gMainProgressBar;
        refresh -su 0;
}
```

add equidistant locators on an edge selection

via a path animation



```
proc locatorsOnEdge(int $numberOfLocators){
    cycleCheck -e off;
    int $clones = $numberOfLocators;
    string $objs[] = {};
    string $edg[] = `ls -sl`;
    string $ob[] = `ls -sl -o`;
    string $ptc[] = `polyToCurve -form 2 -degree 3`;
    string $lo[] = `spaceLocator -n ($ob[0]+"_loc")`;
    string $pa = `pathAnimation -fractionMode true -follow true -followAxis x -upAxis y -worldUpType "vector" -worldUpVector 0 1 0 -inverseUp false -inverseFront false -bank false $pt
    disconnectAttr ($pa+"_uValue.output") ($pa+".uValue");
    for($i = 0;$i<$clones;$i++){
        setAttr ($pa+".uValue") (($i*1.0)/($clones-1));

        refresh;
    string $copy[] = `duplicate $lo[0]`;
    $objs[size($objs)]=$copy[0];

    }
    delete $lo;
    delete $ptc;
    select -r $objs;
    cycleCheck -e on;
}
locatorsOnEdge(3);
```

vray add user ids on selection shapes

```
//no error checking or multiple ids cause i haven't need it so far
string $text;
string $result = `promptDialog
    -title "Add vray ids"
    -message "id value:"
    -button "OK" -button "Cancel"
    -defaultButton "OK" -cancelButton "Cancel"
    -dismissString "Cancel"`;

if ($result == "OK") {
    int $text = `promptDialog -query -text`;
    for($o in `ls -sl -l`){
        string $r[] = `listRelatives -s -f $o`;
        for($sh in $r){
            print $sh;
            vray addAttributesFromGroup $sh vray_objectID 1;
            setAttr ($sh+".vrayObjectID") (int($text));
        }
    }
}
```

vray add random user scalar attribute to selection

```
//works on parent groups too, good for instances !
$usrAttr = "mixvar";
for($o in `ls -sl`){
    vray addAttributesFromGroup $o "vray_user_attributes" 1;
    setAttr ($o+".vrayUserAttributes") -type "string" ($usrAttr+"="+rand(1));
}
```

vray add random colorful material id per shader

```
for($o in `ls -sl`){
    if(!attributeExists("vrayColorId",$o)){ //prevents fucking up old ids
            vray addAttributesFromGroup $o "vray_material_id" 1;
            vector $v = `hsv_to_rgb <<rand(1),1,1>>`;
            setAttr ($o+".vrayColorId") -type "double3" ($v.x) ($v.y) ($v.z);
    }
}
```

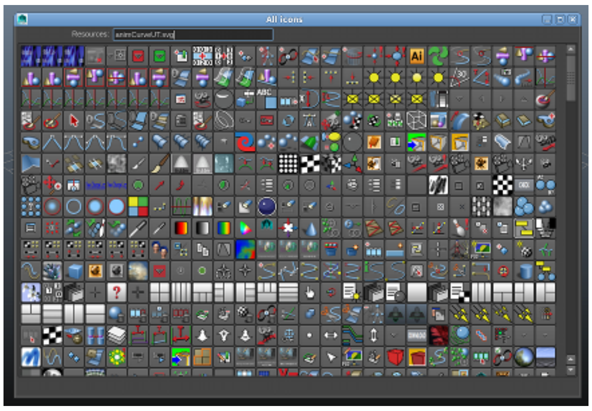
Select all objects constraining an object

```
string $selection[] = {};
for($c in `listRelatives -type "constraint"`){
    $nt = `nodeType $c`;
    $tl = eval($nt+ " -q -tl "+$c);
    $selection = stringArrayCatenate($selection , $tl);
}
select $selection;
```

Average skin weights from selected vertices to last vertex

```
proc averageWeights(){
    if(`selectPref -q -tso` == 0){
        select -cl;
        selectPref -tso 1;
        warning "\Track selection order\" has been checked in your prefs (Settings>Selection), start the procedure again!";
    }else{
        string $vertices[] = `ls -os -fl`;
        string $selObj[] = `ls -o -sl`;
        if(size($vertices)>0){
            string $sck = `findRelatedSkinCluster($selObj[0])`;
            string $influences[] = {};
            float $influencesVals[] = {};
            int $influenceVerticesCount = size($vertices)-1;
            for($i=0;$i<$influenceVerticesCount;$i++){

                $vertex = $vertices[$i];

                string $skinInfluences[]=`skinPercent -ib 0.000001 -q -t $sck $vertex`;
                float $sknVals[] = `skinPercent -ib 0.000001 -q -v  $sck $vertex`;
                for($k=0;$k<size($skinInfluences);$k++){
                    //ok, mel's handling of dictionaries or whatever it's called is 100% retarded, but the python implementation of skinPercent is a piece of shit too.
                    int $curListItem = stringArrayFind($skinInfluences[$k],0,$influences);
                    if($curListItem==-1){
                        int $listSize = size($influences);
                        $influences[$listSize] = $skinInfluences[$k];
                        $influencesVals[$listSize] = $sknVals[$k];
                    }else{
                        $influencesVals[$curListItem] += $sknVals[$curListItem];
                    }
                }
                //print("\n-----------\n"+$vertex+":\n");print("\n----------------\n");print($influences);print($influencesVals);

            }
            $eval = "skinPercent -zri 1";
            string $targetVertex = $vertices[size($vertices)-1];
            for($i=0;$i<size($influences);$i++){
                if($influencesVals[$i] > 0){
                    $eval += " -tv "+$influences[$i]+" "+$influencesVals[$i]/$influenceVerticesCount;
                }
            }

            $eval += " "+$sck+" "+$targetVertex+";\r\n";
            eval($eval);
        }
    }
}
averageWeights();
```

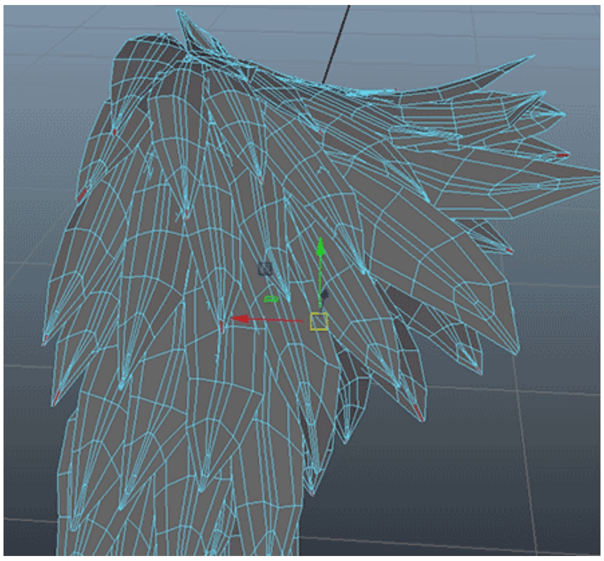
Show all maya icons



```
if (`window -exists AllIconsWin`) deleteUI AllIconsWin;
if (`windowPref -exists AllIconsWin`) windowPref -remove AllIconsWin;
```

```
string $window = `window -title "All icons" -rtf 1 -widthHeight 840 550 AllIconsWin`;

columnLayout -adj 1;
$labels =  `textFieldGrp -label "Resources: " -text  "click icons to get icon names"`;
$scrollLayout = `scrollLayout -verticalScrollBarThickness 16 -h 500`;
rowColumnLayout -numberOfColumns 25;

string $icons[] = `resourceManager -nameFilter "*"`;
for($icon in $icons){
    nodeIconButton -w 32 -h 32 -style "iconOnly" -command ("displayname $labels \""+$icon+"\"") -image1 $icon;
}
showWindow $window;

proc displayname(string $textfield,string $string){
    textFieldGrp -edit -text $string $textfield;
}
```

add edge loops from single selected rings edges



```
string $singleEdges[] = {};
string $sel[] = `ls -sl -fl`;

for($o in $sel){
    select -r $o;
    SelectEdgeRingSp;
    polySplitRing -ch 0;
    string $sel2[] = `ls -sl -fl -hd 1`;
    $singleEdges[size($singleEdges)] = $sel2[0];
}

select -r $singleEdges;
```

Hide things that are not in control balls

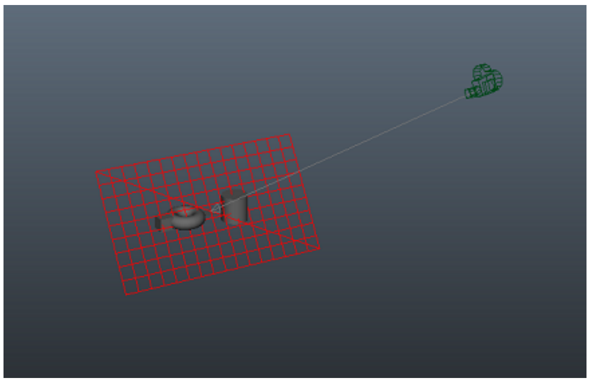
```
global string $objs;
global string $controlsObjs;
$objs = "";
$controlsObjs = "";

string $window = `window -menuBar true -title "Hide stuff with balls" "johny"`;
columnLayout -adjustableColumn true;
button -label "Select objects to be hidden" -command "set(0)";
button -label "Select balls to control" -command "set(1)";
showWindow $window;


proc set(int $n){
    global string $objs;
    global string $controlsObjs;
    string $names[] = {"objs","control"};
    $createSetResult = `sets -name $names[$n]`;
    if($n==0){
        $objs = $createSetResult;
    }
    if($n==1){
        $controlsObjs = $createSetResult;
    }
    print($objs+"------"+$controlsObjs);
    if($n){
        string $expr = "";
        string $controls[] = `listConnections -s 1 -d 0 -p 0 -c 0 $controlsObjs`;
        string $updates = "";
        for($o in $controls){
            $updates += $o+".tx +"+$o+".sx +";
        }
        $updates += "0";
        for($o in $controls){
            $expr += $o+".shearXY = 0*("+$updates+");\n";
        }






        $expr += "string $objs[] = `listConnections -s 1 -d 0 -p 0 -c 0 \""+$objs+"\"`;\n";
        $expr += "string $controls[] = `listConnections -s 1 -d 0 -p 0 -c 0 \""+$controlsObjs+"\"`;\n";
        $expr += "float $controlCenters[] = {};\n";
        $expr += "for($o in $controls){\n";
        $expr += "    float $center[] = `xform -ws -q -t $o`;\n";
        $expr += "    $i = size($controlCenters);\n";
        $expr += "    $controlCenters[$i] = $center[0];\n";
        $expr += "    $controlCenters[$i+1] = $center[1];\n";
        $expr += "    $controlCenters[$i+2] = $center[2];\n";
        $expr += "    $controlCenters[$i+3] = getAttr($o+\".sx\");\n";
        $expr += "}\n";
        $expr += "for($o in $objs){\n";
        $expr += "    float $c[] = `xform -ws -q -bb $o`;\n";
        $expr += "    $c = {($c[0]+$c[3])/2,($c[1]+$c[4])/2,($c[2]+$c[5])/2};\n";
        $expr += "    int $visib = false;\n";
        $expr += "    for($i = 0;$i<size($controlCenters);$i+=4){\n";
        $expr += "        vector $distVect = <<$controlCenters[$i]-$c[0],$controlCenters[$i+1]-$c[1],$controlCenters[$i+2]-$c[2]>>;\n";
        $expr += "        float $dist = `mag($distVect)`;\n";
        $expr += "        if($dist<abs($controlCenters[$i+3])){\n";
        $expr += "            $visib = true;\n";
        $expr += "        }\n";
        $expr += "    }\n";
        $expr += "    setAttr($o+\".visibility\") $visib;   \n";
        $expr += "} \n";

        expression -s $expr "objs";
        deleteUI "johny";
    }
}
```

Add .455 gamma to fileTextures

```
//todo: make it so you don't have to load the filetexture
string $fileTextures[] = `ls -sl`;
for($o in $fileTextures){
    $c = `listConnections -p 1 -d 1 ($o+".outColor")`;
    string $gamma = `shadingNode -asUtility gammaCorrect -n ($o+"_gammaCorrect")`;
    setAttr ($gamma + ".gammaX") 0.455;
    setAttr ($gamma + ".gammaY") 0.455;
    setAttr ($gamma + ".gammaZ") 0.455;
    connectAttr -force ($o + ".outColor") ($gamma + ".value");
    connectAttr -force ($gamma + ".outValue") ($c);
}
```

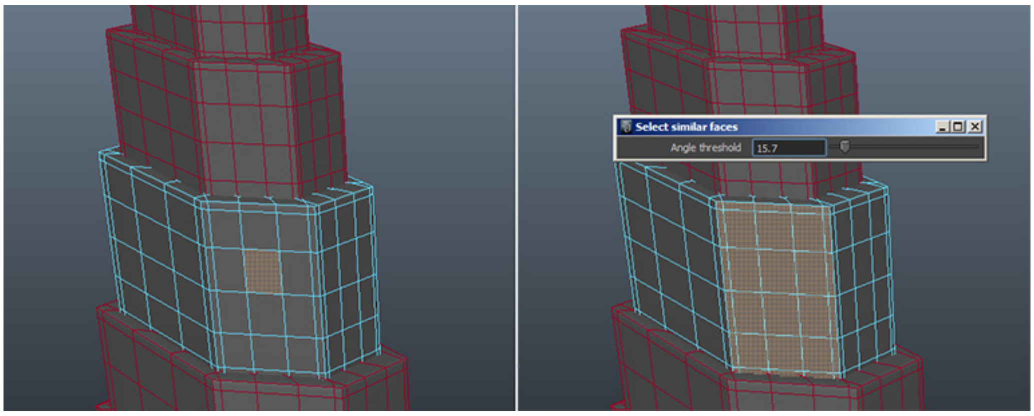
Add camera center of interest locator

better python version: zDepth Control Tool



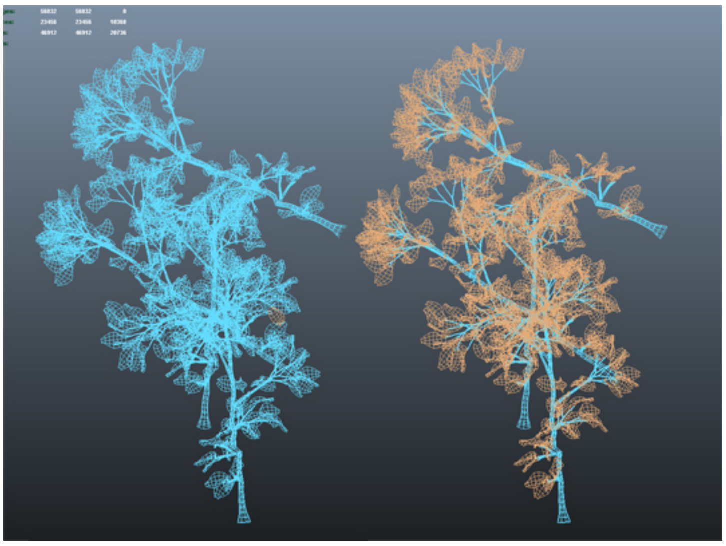
```
source channelBoxCommand;
string $camz[] = `ls -sl`;
if(size($camz)==1){
    string $shapes[] = `listRelatives -f $camz[0]`;
    if(`nodeType $shapes[0]` == "camera"){
        string $sl[] = {};
        string $curve = "curve -d 1 ";
        $w = 16;
        $h = 9;
        for($i=0;$i<=$w;$i++){
            $curve += "-p "+($i-$w/2.0)+" "+(($i%2)*$h-$h/2.0)+" 0 ";
            $curve += "-p "+($i-$w/2.0)+" "+((1-$i%2)*$h-$h/2.0)+" 0 ";
        }
        for($i=0;$i<=$h;$i++){
            $curve += "-p "+(($i%2)*$w-$w/2.0)+" "+($i-$h/2.0)+" 0 ";
            $curve += "-p "+((1-$i%2)*$w-$w/2.0)+" "+($i-$h/2.0)+" 0 ";
        }
        $curve += "-n focusedPlane";

        $sl[0] = eval($curve);

        setAttr -lock true ($sl[0]+".tx");
        setAttr -lock true ($sl[0]+".ty");
        setAttr ($sl[0]+".overrideEnabled") 1;
        setAttr ($sl[0]+".overrideColor") 13;
        string $group = `group -w -n ($camz[0]+"_focusLocator") $sl[0]`;


        string $l[] = `listRelatives -c $group`;
        $sl[0] = $group+"|"+$l[0];

        string $pcs[] = `parentConstraint -weight 1 $camz[0] $group`;
        setAttr($pcs[0]+".target[0].targetOffsetRotateY") 180;
        float $distance  = `getAttr ($shapes[0]+".centerOfInterest")`;
        setAttr($sl[0]+".tz") ($distance);

        $lc = `listConnections ($shapes[0]+".centerOfInterest")`;
        if(size($lc)>0 && `nodeType $lc` == "animCurveTL"){
            int $ck = `copyKey -at "centerOfInterest" -time ":" $shapes[0]`;
            CBdeleteConnection ($shapes[0]+".coi");
            if($ck < 1){
                warning("Can't copy animation, sorry");
            }else{
                pasteKey -at "tz" $sl[0];
                print("Pasted "+$ck+" center of interest keyframes to the focusPlane");
                catchQuiet("delete $lc");
            }
        }
        expression -s ($shapes[0]+".centerOfInterest=abs("+$sl[0]+".tz)") -o $camz[0] -ae 1 -uc all ;
        string $an = `annotate -tx "" -p 0 0 0 $sl[0]`;
        string $anPar[] = `listRelatives -p $an`;

        parent -r $anPar[0] $camz[0];
        setAttr ($an+".overrideDisplayType") 1;
        setAttr ($an+".overrideEnabled") 1;
        setAttr ($an+".overrideColor") 13;
        select -r $sl;

    }else{
        warning "select camera only";
    }
}else{
    warning "select a camera";
}
```

Select faces by angle (slow-ish)



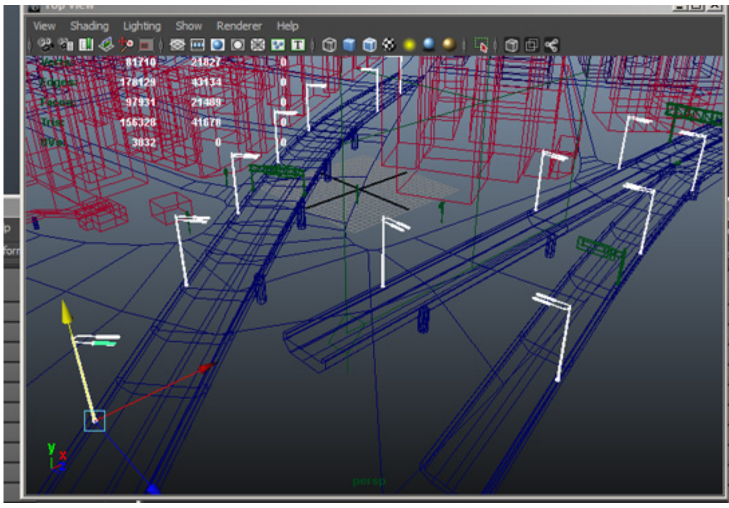
```
//super hackish code eww eww eww
global string $selectedFace;
string $selected[] = `ls -sl`;
$selectedFace = $selected[0];


string $window = `window -title "Select similar faces"`;
columnLayout;
floatSliderGrp -label "Angle threshold   " -field true
    -minValue 0 -maxValue 180
    -cc "growSel"
    -value 0 "slidAngle";
showWindow $window;

proc growSel(){
    global string $selectedFace;
    string $newSelection[] = `ls -sl`;
    if(size($newSelection)==1){
        $selectedFace = $newSelection[0];
    }
    float $threshold = `floatSliderGrp  -q -v "slidAngle"`;
    growByAngle($selectedFace,$threshold);
}

proc growByAngle(string $face, float $threshold){
    vector $base = norm($face);
    string $newSele[] = {};
    float $olds = 0;
    float $news = 1;
    int $round = 0;
    while($olds != $news || $round > 999){
        $round++;
        $olds = size(`ls -sl`);
        if($olds == 0){
            select -r $face;
        }
        GrowPolygonSelectionRegion;
        string $grow[] = `ls -sl -fl`;

        for($face in $grow){
           vector $fAngle = norm($face);
           float $result[] = `angleBetween -euler -v1 ($base.x) ($base.y) ($base.z) -v2 ($fAngle.x) ($fAngle.y) ($fAngle.z)`;
           if((abs($result[0])+abs($result[1])+abs($result[2]))<$threshold){
               $newSele[size($newSele)] = $face;
           }
        }
        select -r $newSele;
        $news = size(`ls -sl`);
    }
}



proc vector norm( string $face )
{
  //from Joseph A. Hansen (Beyond Games).
  vector $normal;
  float $x;
  float $y;
  float $z;

  string $pins[] = `polyInfo -fn $face`;
  string $pin = $pins[0];
  string $tokens[];
  int $numTokens = `tokenize $pin " " $tokens`;

  if ( ( $numTokens > 3 ) && ( $tokens[0] == "FACE_NORMAL" ) )
  {
    $x = ($tokens[$numTokens-3]);
    $y = ($tokens[$numTokens-2]);
    $z = ($tokens[$numTokens-1]);

    $normal = << $x, $y, $z >>;
    $normal = `unit $normal`;
  }
  return $normal;
}
```

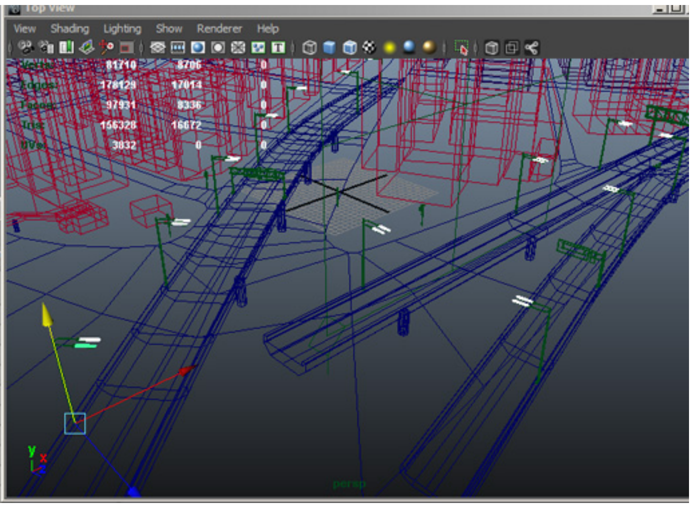
Select similar shells



```
//not tested on really heavy meshes
//
//selects shells with same number of faces
//expects 1 shell to be selected in face mode
//change numberOfTests according to mesh density

int $numberOfTests = 5000;  //change according to mesh density and distribution, this took approx 10s on a 25k poly object
int $useMoreThanFacesCount = 1; //change to 1 if you want to match using UVs/edge, can be nice if you have same poly count but different UVs/edges count
int $debug = 0;

string $selectionList[] = {};



string $sele[] = `ls -sl -fl`;
string $obj = `match "^[^\.]*" $sele[0]`;

float $matchCount = size($sele);

string $extra[] = `polyListComponentConversion -fv -fe -ff -fvf -te`; //uvs, change -tuv to -te to compare with edges instead of UVs
select -r $extra;
float $matchCountExtra = size(`ls -fl -sl`);
print($debug?("faces: "+$matchCount+" uvs: "+$matchCountExtra+"\n-------------------------------------------------\n"):"");

string $faces[] = `ls -fl ($obj+".f[*]")`;
$s = size($faces);

for($i=0;$i<=$s;$i+=($s/$numberOfTests)){

    $shl = `polySelect -asSelectString -ets $i $obj`;
    string $shell[] = `ls -sl -fl`;
    float $shellSize = size($shell);

    float $shellUVs = 0;
    if($useMoreThanFacesCount){
        string $extraCount[] = `polyListComponentConversion -fv -fe -ff -fvf -te`; //uvs, change -tuv to -te to compare with edges instead of UVs
        select -r $extraCount;
        $shellUVs = size(`ls -sl -fl`);
        print($debug?("faces: "+$shellSize+" uvs: "+$shellUVs+"\n"):"");
    }

    if($shellSize ==$matchCount && ($shellUVs== $matchCountExtra || $useMoreThanFacesCount==0 ) ){
        for($j = 0;$j<size($shl);$j++){
            $selectionList[size($selectionList)] = $shl[$j];
        }
    }
}

select -r $selectionList;
```

Select similar to last object





Selects the objects that are similar to the last of the current selection, in the current selection, based on face counts (useful when cleaning up geo)

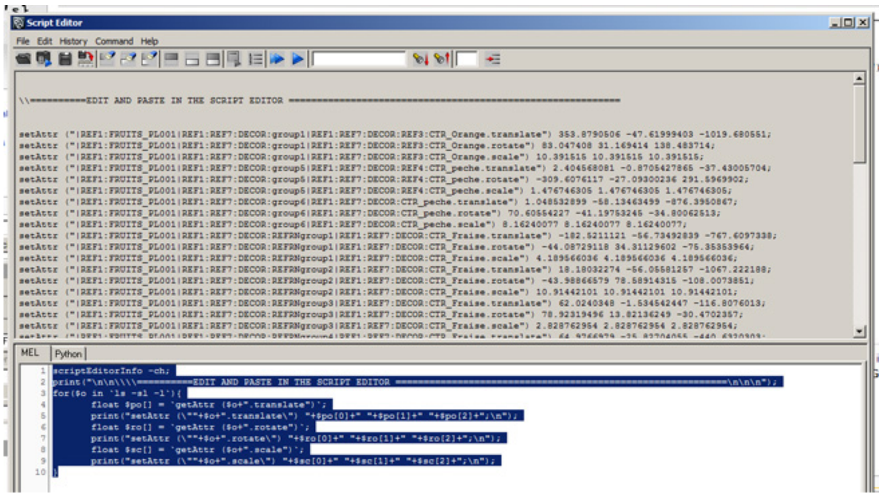
```
string $selection[] = `ls -sl`;
string $newSelection[];
int $faces[] = `polyEvaluate -f $selection[size($selection)-1]`;
for($obj in $selection){
    int $pe[] = `polyEvaluate -f $obj`;
    if($pe[0] == $faces[0]){
        $newSelection[size($newSelection)] = $obj;
    }
}
select -r $newSelection;
```

Select shapes according to wire color

```
string $os[] = `ls -dag -s -sl`;
$last = $os[size($os)-1];
int $color = `getAttr ($last+".overrideColor")`;
string $lists[] = {};
for($obj in $os){
    if(`getAttr ($obj+".overrideColor")` == $color){
        $lists[size($lists)] = $obj;
    }
}
select -r $lists;
```

Stupid Pos/Rot/Scale Export & Stupid shader assignement export

```
proc writeToTempTextFileAndOpen(string $textToWrite){
    //cross platform write to temp textfile
    string $file = `file -q -sn -shn`;
    int $r = `rand 1000 10000`;
    $file = ($file=="")?"tmp_maya_file."+$r+".txt":$file+"."+$r+".txt";
    string $myScriptDir = `internalVar -utd`;
    string $tmpfile = $myScriptDir+$file;
    int $outFileId = fopen($tmpfile,"w");
    if ($outFileId == 0 ) {
        scriptEditorInfo -ch;
        print $textToWrite;
        warning ("\n\n//  Could not open output file " + $tmpfile + " wrote to the script editor window instead.");
    }else{
        fprint $outFileId $textToWrite;
        fclose $outFileId;
        if(`about -win` == 1){
            system("load " + `toNativePath($tmpfile)`);     //win
        }else if(`about -li` == 1){
            system("gedit "+$tmpfile+">/dev/null 2>&1 &");  //linux
        }else if(`about -mac` == 1){
            system("TextEdit "+$tmpfile+">/dev/null 2>&1 &");  //macOs, untested
        }else{
            scriptEditorInfo -ch;
            print $textToWrite;
            warning ("\n\n//  Could not figure out system, wrote to the script editor window instead.");
        }
    }
}
```



```
\\=========EDIT AND PASTE IN THE SCRIPT EDITOR ============================================================

setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:group1|REF1:REF7:DECOR:REF3:CTR_Orange.translate") 353.8790506 -47.61999403 -1019.680551;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:group1|REF1:REF7:DECOR:REF3:CTR_Orange.rotate") 83.047408 31.169414 138.483714;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:group1|REF1:REF7:DECOR:REF3:CTR_Orange.scale") 10.391515 10.391515 10.391515;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:group5|REF1:REF7:DECOR:REF4:CTR_peche.translate") 2.404568081 -0.8705427865 -37.43005704;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:group5|REF1:REF7:DECOR:REF4:CTR_peche.rotate") -309.6076117 -27.09300236 291.5969902;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:group5|REF1:REF7:DECOR:REF4:CTR_peche.scale") 1.476746305 1.476746305 1.476746305;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:group6|REF1:REF7:DECOR:CTR_peche.translate") 1.048532899 -58.13463499 -876.3950867;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:group6|REF1:REF7:DECOR:CTR_peche.rotate") 70.60554227 -41.19753245 -34.80062513;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:group6|REF1:REF7:DECOR:CTR_peche.scale") 8.16240077 8.16240077 8.16240077;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:REFRNgroup1|REF1:REF7:DECOR:CTR_Fraise.translate") -182.5211121 -56.73492839 -767.6097338;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:REFRNgroup1|REF1:REF7:DECOR:CTR_Fraise.rotate") -44.08729118 34.31129602 -75.35353964;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:REFRNgroup1|REF1:REF7:DECOR:CTR_Fraise.scale") 4.189566036 4.189566036 4.189566036;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:REFRNgroup2|REF1:REF7:DECOR:CTR_Fraise.translate") 18.18032274 -56.05581257 -1067.222188;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:REFRNgroup2|REF1:REF7:DECOR:CTR_Fraise.rotate") -43.98866579 78.58914315 -108.0073851;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:REFRNgroup2|REF1:REF7:DECOR:CTR_Fraise.scale") 10.91442101 10.91442101 10.91442101;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:REFRNgroup3|REF1:REF7:DECOR:CTR_Fraise.translate") 62.0240348 -1.534542447 -116.8076013;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:REFRNgroup3|REF1:REF7:DECOR:CTR_Fraise.rotate") 78.92319496 13.82136249 -30.4702357;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:REFRNgroup3|REF1:REF7:DECOR:CTR_Fraise.scale") 2.828762954 2.828762954 2.828762954;
setAttr ("|REF1:FRUITS_PL001|REF1:REF7:DECOR:REFRNgroup4|REF1:REF7:DECOR:CTR_Fraise.translate") 64.9766979 -25.82704055 -440.6320303;
```

```
scriptEditorInfo -ch;
print("\n\n\\\\=========EDIT AND PASTE IN THE SCRIPT EDITOR ============================================================\n\n\n");
for($o in `ls -sl -l`){
        float $po[] = `getAttr ($o+".translate")`;
        print("setAttr (\""+$o+".translate\") "+$po[0]+" "+$po[1]+" "+$po[2]+";\n");
        float $ro[] = `getAttr ($o+".rotate")`;
        print("setAttr (\""+$o+".rotate\") "+$ro[0]+" "+$ro[1]+" "+$ro[2]+";\n");
        float $sc[] = `getAttr ($o+".scale")`;
        print("setAttr (\""+$o+".scale\") "+$sc[0]+" "+$sc[1]+" "+$sc[2]+";\n");
}
```

```
//output translate, rotate, scale for the current frame
scriptEditorInfo -ch;
print("\n\n//=========EDIT AND PASTE IN THE SCRIPT EDITOR ============================================================\n\n\n");
proc outputTransforms(){
    for($o in `ls -sl -l`){
    float $po[] = `getAttr ($o+".translate")`;
    print("setAttr (\""+$o+".translate\") "+$po[0]+" "+$po[1]+" "+$po[2]+";\n");
    float $ro[] = `getAttr ($o+".rotate")`;
    print("setAttr (\""+$o+".rotate\") "+$ro[0]+" "+$ro[1]+" "+$ro[2]+";\n");
    float $sc[] = `getAttr ($o+".scale")`;
    print("setAttr (\""+$o+".scale\") "+$sc[0]+" "+$sc[1]+" "+$sc[2]+";\n");
    }
}
evalDeferred("outputTranforms");
```

shaders:

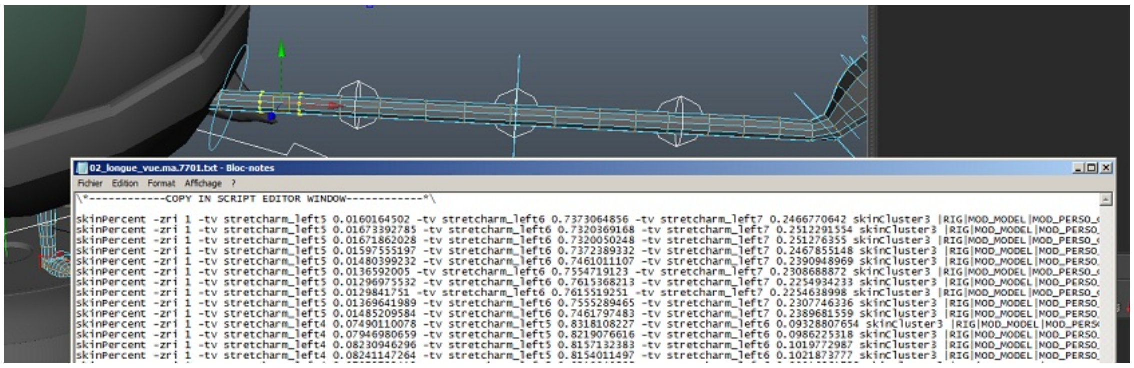
```
//for now face assignement doesn't work
scriptEditorInfo -ch;
proc listShadersAssignement(){
    print("\n\n//=========EDIT AND PASTE IN THE SCRIPT EDITOR ============================================================\n\n\n");
    for($o in `ls -sl -l`){
        string $shapes[] = `listRelatives -s  -f $o`;
        if(`nodeType $shapes[0]` == "mesh"){
            for($p in  `listConnections $shapes[0]`){
                $type = `nodeType $p`;
                if($type == "shadingEngine"){
                    $u = "";
                    for($i = 1;$i<=40 - size($p);$i++){
                        $u += " ";
                    }
                    print("\nsets -forceElement "+$p+""+$u+""+$o+";");
                    break; //stops at first shader found
                }
            }

        }
    }
}
evalDeferred("listShadersAssignement");
```

Save selection to temp text file

useful if you're between mayas and are switching selections between two similar scenes

```
string $myScriptDir = `internalVar -utd`;
string $tmpfile = $myScriptDir+"tmp_sel.txt";
int $outFileId = fopen($tmpfile,"w");
if ($outFileId != 0) {
    string $sel[] = `ls -sl`;
    $selstring = "select -add "+stringArrayToString($sel," ");
    fprint $outFileId $selstring;
    fclose $outFileId;
    exec("notepad "+$tmpfile);
}
```

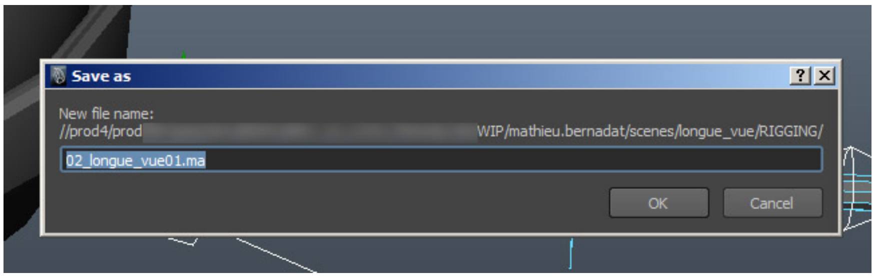
Save weights to text file


02_longue_vue.ma.7701.txt - Bloc-notes
Fichier Edition Format Affichage ?

```
\*------------COPY IN SCRIPT EDITOR WINDOW------------*\

skinPercent -zri 1 -tv stretcharm_left5 0.0160164502 -tv stretcharm_left6 0.7373064856 -tv stretcharm_left7 0.2466770642 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left5 0.01673392785 -tv stretcharm_left6 0.7320369168 -tv stretcharm_left7 0.2512291554 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left5 0.01671862028 -tv stretcharm_left6 0.7320050248 -tv stretcharm_left7 0.251276355 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left5 0.01597555197 -tv stretcharm_left6 0.7372389332 -tv stretcharm_left7 0.2467855148 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left5 0.01480399232 -tv stretcharm_left6 0.7461011107 -tv stretcharm_left7 0.2390948969 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left5 0.0136592005 -tv stretcharm_left6 0.7554719123 -tv stretcharm_left7 0.2308688872 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left5 0.01296975532 -tv stretcharm_left6 0.7615368213 -tv stretcharm_left7 0.2254934233 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left5 0.0129841751 -tv stretcharm_left6 0.7615519251 -tv stretcharm_left7 0.2254638998 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left5 0.01369641989 -tv stretcharm_left6 0.7555289465 -tv stretcharm_left7 0.2307746336 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left5 0.01485209584 -tv stretcharm_left6 0.7461797483 -tv stretcharm_left7 0.2389681559 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left4 0.07490110078 -tv stretcharm_left5 0.8318108227 -tv stretcharm_left6 0.09328807654 skinCluster3 |RIG|MOD_MODEL|MOD_PERS
skinPercent -zri 1 -tv stretcharm_left4 0.07946980659 -tv stretcharm_left5 0.8219076616 -tv stretcharm_left6 0.0986225318 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left4 0.08230946296 -tv stretcharm_left5 0.8157132383 -tv stretcharm_left6 0.1019772987 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
skinPercent -zri 1 -tv stretcharm_left4 0.08241147264 -tv stretcharm_left5 0.8154011497 -tv stretcharm_left6 0.1021873777 skinCluster3 |RIG|MOD_MODEL|MOD_PERSO_
```

```
    scriptEditorInfo -ch; //run first if linux

    string $vertices[] = `ls -sl -fl -l`;
    string $selObj[] = `ls -o -sl`;
    if(size($vertices)>0){
        string $sck = `findRelatedSkinCluster($selObj[0])`;



        string $eval = "//------------COPY IN SCRIPT EDITOR WINDOW------------//\r\n\r\n";
        for($vertex in $vertices){
            string $skinInfluences[]=`skinPercent -ib 0.000001 -q -t $sck $vertex`;
            float $sknVals[] = `skinPercent -ib 0.000001 -q -v  $sck $vertex`;
            $eval += "skinPercent -zri 1";
            for($i=0;$i<size($skinInfluences);$i++){
                if($sknVals[$i] > 0){
                    $eval += " -tv "+$skinInfluences[$i]+" "+$sknVals[$i];
                }
            }
            $eval += " "+$sck+" "+$vertex+";\r\n";
        }


        if(`about -win` == 1){
            string $file = `file -q -sn -shn`;
            int $r = `rand 1000 10000`;
            $file = ($file=="")?"tmp_skin_weights_maya."+$r+".txt":$file+"."+$r+".txt";
            string $myScriptDir = `internalVar -utd`;
            string $tmpfile = $myScriptDir+$file;
            int $outFileId = fopen($tmpfile,"w");
            if ($outFileId == 0 ) {
                scriptEditorInfo -ch
                print $eval;
                warning ("\n\n//  Could not open output file " + $tmpfile + " wrote to the script editor window instead.");
            }else{
                fprint $outFileId $eval;
                fclose $outFileId;
                system("load " + `toNativePath($tmpfile)`); //win only
            }
        }else{

           print($eval);
            print "\n\n//  Wrote to the script editor window instead.";
        }
    }else{
        warning "select vertices to extract and save weights from";
    }
```

Simple incremental save w/ prompt



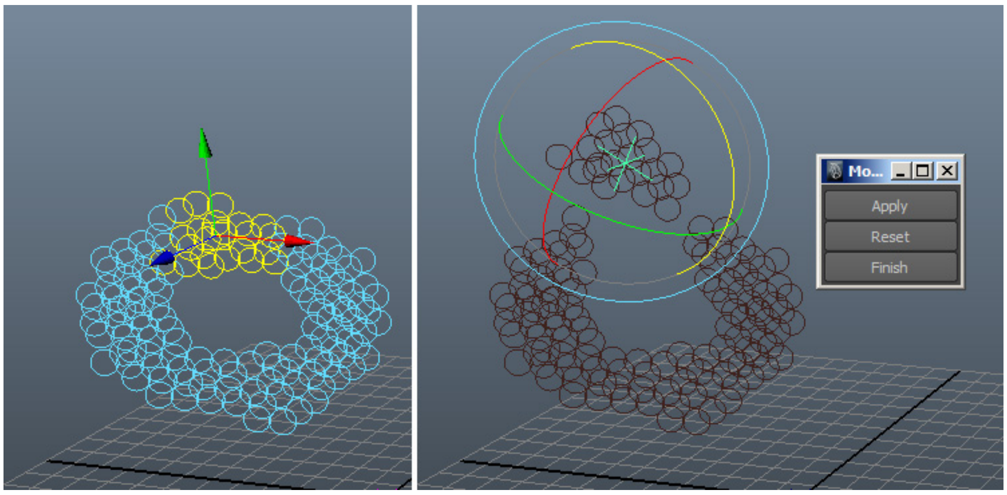
New file name:
//prod4/prod[redacted]WIP/mathieu.bernadat/scenes/longue_vue/RIGGING/
02_longue_vue01.ma

```
proc incrementAndSave(){
    //takes the current scene, increments (or adds '01' suffix if none found). Padding set to 2
    //so many regexes because I suck at this
    string $path = `file -q -loc`;
    $path = `match "^.*/" $path`;
    string $file = `file -q -sn -shn`;
    $ext = fileExtension($file);
    $file = `match "^[^\.]*" $file`;
    string $suffix = `match "[0-9]+$" $file`;
    $file = `match ".*[^0-9]" $file`;
    int $suff = 1;
    if($suffix != ""){
        $suff = $suffix;
        $suff++;
    }
    $suffix = ($suff<10)?"0"+$suff:$suff;
    $fileprompt = `promptDialog -text ($file+$suffix+"."+$ext) -title "Save as" -message "New file name:" -button "OK" -button "Cancel" -defaultButton "OK" -cancelButton "Cancel"`;
    if($fileprompt == "OK"){
        $scene = `promptDialog -q -text`;
        $type = (`tolower $ext` == "ma")?"mayaAscii":"mayaBinary";
        file -rename ($path+$scene); file -save -type $type;

    }
}
incrementAndSave;
```

Simple renderview batch

```
//if batch rendering only works in viewport. Uses rendersettings except for padding set to 4. Make sure you're rendering 32bit in the viewport
proc string pad(float $n){return ($n<1000)? ( ($n<100)?( ($n<10)?"000"+$n:"00"+$n ):"0"+$n ) : $n+"";}
proc batchrenderview(){
    $f_str = `getAttr "defaultRenderGlobals.startFrame"`;
    $f_end = `getAttr "defaultRenderGlobals.endFrame"`;
    $c = `lookThru -q`;
    $c = "Render "+($f_end-$f_str+1)+" frames with camera '"+$c+"' ?";
    $button = `confirmDialog -title "Confirm" -message $c -button "OK" -button "Cancel" -defaultButton "OK"    -cancelButton "Cancel" -dismissString "nope"`;
    if($button == "OK"){
        setAttr "defaultRenderGlobals.outFormatControl" 0;
        setAttr "defaultRenderGlobals.animation" 1;
        setAttr "defaultRenderGlobals.putFrameBeforeExt" 1;
        setAttr "defaultRenderGlobals.extensionPadding" 4;
        $log = 0;
        $dir = "";
        for($i = $f_str;$i<=$f_end;$i++){
            currentTime $i;
            renderIntoNewWindow render;
            string $f = pad($i);
            string $tempf[] = `renderSettings -gin $f -fpt`;
            string $tof[] = `renderSettings -gin $f -fp`;
            $dir = `match "^.*/" $tof[0]`;
            sysFile -md $dir;
            $log += `sysFile -ren $tof[0] $tempf[0]`;
        }
        warning($log+" files copied to     "+toNativePath($dir));

    }
}
batchrenderview;
```

Particle mover

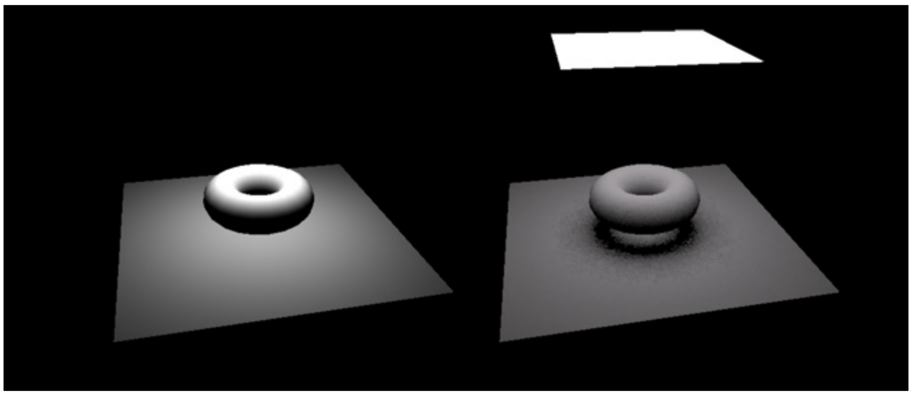


```
//select particles that you want to move
//launch script, move locator around, scale & rotate
//hit remove once you're happy. Initial state is set.
//
// doesn't yet work on particles systems that have been moved around

global string $gPart;
global string $gLoc;
global float $gPos[];
global float $gIds[];

string $particles[] = `ls -sl -fl`;
string $partName = `match "^[^\.]*" $particles[0]`;
string $partiName[] = `listRelatives -s $partName`;

float $ids[] = `getParticleAttr -a 1 -at id $particles`;
float $positions[] = `getParticleAttr -a 1 -at position $particles`;
float $locpos[] = `getParticleAttr -at position $particles`;
/*setToolTo moveSuperContext;
float $centerPosi[] = `manipMoveContext -q -position Move`;*/
select -cl;
string $loc[] = `spaceLocator`;
/*parent $loc[0] $partName;
setToZero($loc[0]);
parent -w $loc[0];*/

//move -r $centerPosi[0] $centerPosi[1] $centerPosi[2];
move -r $locpos[0] $locpos[1] $locpos[2];
makeIdentity -apply true -t 1 -r 0 -s 0 -n 0 $loc[0];

$gPart = $partiName[0];
$gLoc = $loc[0];
$gPos = $positions;
$gIds = $ids;

string $window = `window -menuBar true -title "Move Particles"`;
columnLayout -adjustableColumn true;
button -label "Apply" -command moveParticles;
button -label "Reset" -command ("setToZero(\""+$loc[0]+"\")");
button -label "Finish" -command ("delete "+$loc[0]+"; deleteUI -window "+$window+"; select -r "+stringArrayToString($particles," "));
showWindow $window;

proc moveParticles(){
    global string $gPart;
    global string $gLoc;
    global float $gPos[];
    global float $gIds[];

    string $locator = $gLoc;
    string $part = $gPart;
    float $pids[] = $gIds;
    float $posis[] = $gPos;

    string $sel[] = `ls -sl`;
    float $m[] = `xform -query -matrix $locator`;
    for($i = 0;$i<size($pids);$i++){
        float $po[] = {$posis[$i*3],$posis[$i*3+1],$posis[$i*3+2]};
        $po =  pointMatrixMult($po,$m);
        $po = {$po[0]+$m[12],$po[1]+$m[13],$po[2]+$m[14]};
        select -r ($part+".pt["+$pids[$i]+"]");
        setParticleAttr -vv $po[0] $po[1]$po[2] -at position;
    }
    select -r $gPart;
    catchQuiet(performSetNClothStartState(1));
    catchQuiet(saveInitialState($gPart));
    select -r $sel;
}
proc setToZero(string $obj){
    setAttr ($obj+".translateX") 0;
    setAttr ($obj+".translateY") 0;
    setAttr ($obj+".translateZ") 0;
    setAttr ($obj+".rotateX") 0;
    setAttr ($obj+".rotateY") 0;
    setAttr ($obj+".rotateZ") 0;
    setAttr ($obj+".scaleX") 1;
    setAttr ($obj+".scaleY") 1;
    setAttr ($obj+".scaleZ") 1;
    moveParticles();
}
```

Turn all nucleuses on/off

```
string $nuc[] = `ls "*nucleus*"`; //assumes it hasn't been renamed
if(size($nuc)>0)
{
    $mode = `getAttr ($nuc[0]+".enable")`;
    $mode = ($mode)?0:1;

    for($i=0;$i<size($nuc);$i++)
    {
        setAttr ($nuc[$i]+".enable") $mode;
        print("All nucleuses "+(($mode)?"on":"off"));
    }
}else{
    print("No nucleus found");
}
```

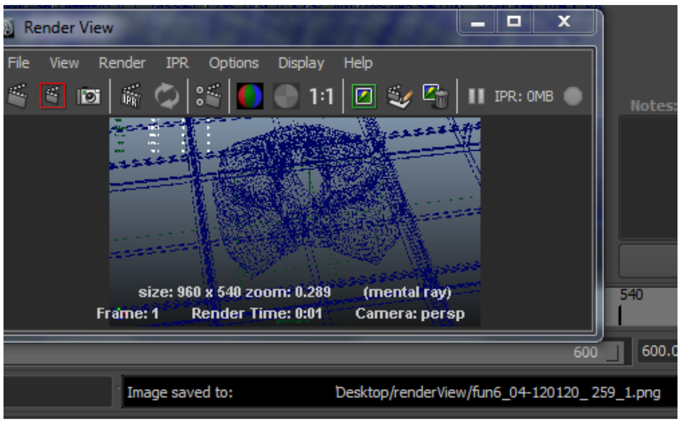
Photometric camera & studio lights



```
//same as manually adding a simple lens exposure; select camera first
if(nodeType(listRelatives(`ls -sl`))=="camera"){
    $cam = listRelatives(`ls -sl`);
    $node = `createNode "mia_exposure_simple"`;
    connectAttr -f ($node+".message") ($cam[0]+".miLensShader");
}else{
    warning "Select a camera";
}
```

```
//creates an area light with photo studio shader & shadows
select -cl;
defaultAreaLight 1 1 1 1 0 0 0 0 0 1 0;
string $sel[] = `ls -sl`;
$ali = $sel[0];
setAttr ($ali+".areaLight") 1;
setAttr ($ali+".areaVisible") 1;
$pl = `createNode  mia_portal_light`;
connectAttr -force ($pl+".message") ($ali+".mentalRayControls.miLightShader");
setAttr ($pl+".visible") 1;
setAttr ($pl+".use_custom_environment") 1;
$bb= `createNode  mib_blackbody`;
connectAttr -force ($bb+".message") ($pl+".custom_environment");
```

Save render view image to Desktop as png



size: 960 x 540 zoom: 0.289 (mental ray)
Frame: 1 Render Time: 0:01 Camera: persp

Image saved to: Desktop/renderView/fun6_04-120120_ 259_1.png

Save following proc as saveRenderViewToDesktop.mel in prefs/scripts:

```
global proc saveRenderViewToDesktop(){
    string $folderName = "renderView";
    string $fileName = `file -q -sceneName`;
    string $scene = `match "[^/\\]*$" $fileName`;
    $scene = `match "^[^\.]*" $scene`;
    string $desktop = getenv("USERPROFILE")+"/Desktop";
    string $d = system("echo %DATE%-%TIME%");
    string $datetime = substring($d,9,10)+substring($d,4,5)+substring($d,1,2)+"_"+substring($d,12,13)+substring($d,15,16)+"_"+substring($d,18,18);
    sysFile -makeDir ($desktop+"/"+$folderName+"/"); // Windows
    string $path = $desktop+"/"+$folderName+"/"+$scene+"-"+$datetime;
    int $imf = `getAttr "defaultRenderGlobals.imageFormat"`;
    setAttr "defaultRenderGlobals.imageFormat" 32;
    catch(`renderWindowSaveImageCallback "renderView" $path "image"`);
    setAttr "defaultRenderGlobals.imageFormat" $imf;
    print("Image saved to: "+$path+".png");
}
```

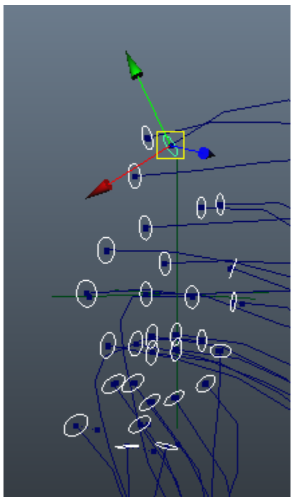
Add to \MayaPath\scripts\others\renderWindowPanel.mel (around line 3413, look for *iconTextButton -i1 "rvRemoveIt.png"*)

```
     iconTextButton -i1 "editRenderPass.png" -width $iconSize -height $iconSize
        -command ("saveRenderViewToDesktop");
```

add miLabels

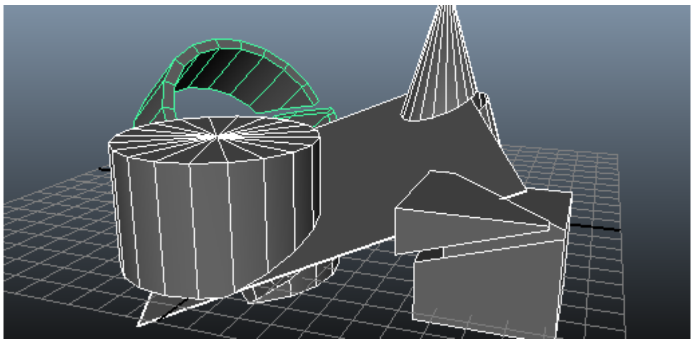
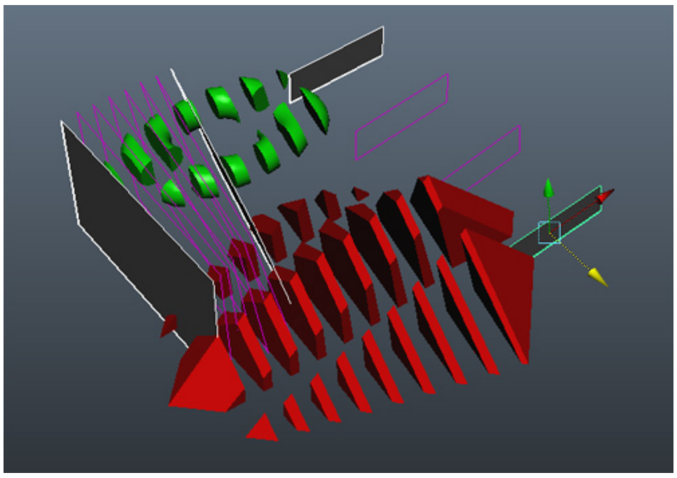
```
addmilabel;
global proc addmilabel(){
    string $objs[] = `ls -sl`;
    string $pd = `promptDialog -m "mi Label"`;
    int $mil = `promptDialog -q -text`;
    for($obj in $objs){
        $l = `listAttr -st "miLabel" $obj`;
        if($l[0] != "miLabel"){

            $cmd = `addAttr -at short -longName miLabel -defaultValue $mil $obj`;
            catchQuiet(`setAttr -k on ($obj+".miLabel")`);
        }
    }
}
```

Place circle at base of paths



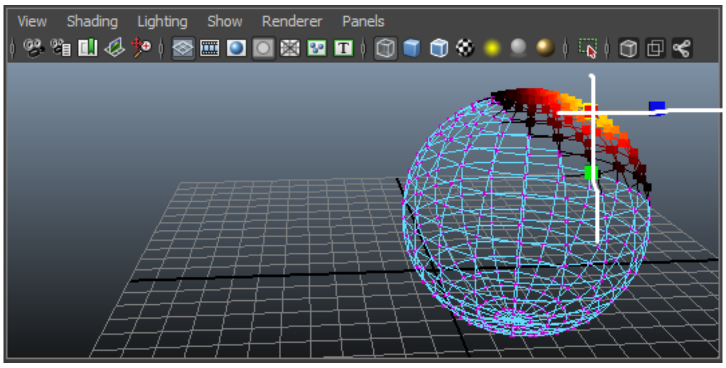
```
//places a circle at the base of a path
//select curve and run

source generateChannelMenu.mel;
source channelBoxCommand.mel;
string $objs[] = `ls -sl`;
string $select[] = {};
for($obj in $objs){

    string $circle[] = `circle -ch on -o on -nr 1 0 0 -r .5`;
    float $e = `playbackOptions -query -maxTime`;
    float $s = `playbackOptions -query -minTime`;
    string $pa = `pathAnimation -fractionMode true -follow true -followAxis x -upAxis y -worldUpType "vector" -worldUpVector 0 1 0 -inverseUp false -inverseFront false -bank false -st
    CBdeleteConnection($pa+".u");
    $select[size($select)]=$circle[0];

}

select -r $select;
```

'Julienne cut': multi polycut across shapes

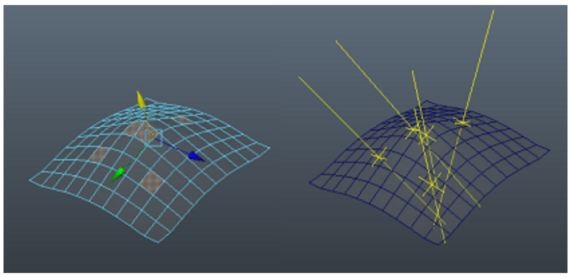
```
//adds as many polycuts as there are objects
//with one nurbs controller
//only works in worldspace
proc string cutObjs(){
    string $objs[] = `ls -sl -tr -o`;
    string $controlPlane[];
    if(size($objs) > 0){
        setToolTo moveSuperContext;
        vector $centerPos = `manipMoveContext -q -position Move`;
        float $bbox[] = `xform -q -ws -bb $objs[0]`;
        float $x = $bbox[3]-$bbox[0];
        float $y = $bbox[4]-$bbox[1];
        float $z = $bbox[5]-$bbox[2];
        $controlPlane = `nurbsPlane -w ($x*1.1) -lr ($y/$x*1.1) -ax 0 0 1 -n "ctrlPlane"`;
        move -r -wd ($centerPos.x) ($centerPos.y) ($centerPos.z);
        addAttr -ln "Detach"  -at bool  $controlPlane[0];
        setAttr -e-keyable true ($controlPlane[0]+".Detach");
        addAttr -ln "Delete"  -at bool  $controlPlane[0];
        setAttr -e-keyable true ($controlPlane[0]+".Delete");
        addAttr -ln "polyCutOffset"  -at double3  $controlPlane[0];
        addAttr -ln "polyCutOffsetX"  -at double -p polyCutOffset  $controlPlane[0];;
        addAttr -ln "polyCutOffsetY"  -at double -p polyCutOffset  $controlPlane[0];;
        addAttr -ln "polyCutOffsetZ"  -at double -p polyCutOffset  $controlPlane[0];;
        setAttr -type double3   ($controlPlane[0]+".polyCutOffset") 0 0 0;
        setAttr -e-keyable true ($controlPlane[0]+".polyCutOffset");
        setAttr -e-keyable true ($controlPlane[0]+".polyCutOffsetX");
        setAttr -e-keyable true ($controlPlane[0]+".polyCutOffsetY");
        setAttr -e-keyable true ($controlPlane[0]+".polyCutOffsetZ");

        string $polycuts[];
        for($t in $objs){
            string $polyCutTool[] = `polyCut  -ws 1  -cd "X" -ch 1 $t`;
            $polycuts[size($polycuts)] = $polyCutTool[0];
           connectAttr -f ($controlPlane[0]+".translate") ($polyCutTool[0]+".cutPlaneCenter");
           connectAttr -f ($controlPlane[0]+".rotate") ($polyCutTool[0]+".cutPlaneRotate");
           connectAttr -f ($controlPlane[0]+".polyCutOffset") ($polyCutTool[0]+".extractOffset");
           connectAttr -f ($controlPlane[0]+".Detach") ($polyCutTool[0]+".extractFaces");
           connectAttr -f ($controlPlane[0]+".Delete") ($polyCutTool[0]+".deleteFaces");

        }
    return $controlPlane[0];
    }else{
    return false;
    }

}

//////////////////////////////////////////////////////////////////////////////


proc multiCut(int $cutnumbers){
    //requires cutObjs proc above
    //expects a number of cuts (>2)
    string $ctrlplanes[];
    string $objs[] = `ls -sl -tr -o`;

    for($i = 0; $i < $cutnumbers; $i++){
        select -r $objs;
        $ctrlplanes[size($ctrlplanes)] = `cutObjs`;
    }
    if($cutnumbers > 2){
        $s = $ctrlplanes[0];
        $e = $ctrlplanes[$cutnumbers-1];
        $m = $cutnumbers-1;

        string $exp = "";

        for($i = 1; $i < $cutnumbers-1; $i++){
            $c = $ctrlplanes[$i];
            $shp = `listRelatives -s $c`;
            setAttr ($shp[0]+".overrideEnabled") 1;
            setAttr ($shp[0]+".overrideShading") 0;


            $exp += $c + ".tx = ( " + $s + ".tx * " + ( $m - $i ) + " + " + $e + ".tx * " + $i +" ) / " + $m + ";\n";
            $exp += $c + ".ty = ( " + $s + ".ty * " + ( $m - $i ) + " + " + $e + ".ty * " + $i +" ) / " + $m + ";\n";
            $exp += $c + ".tz = ( " + $s + ".tz * " + ( $m - $i ) + " + " + $e + ".tz * " + $i +" ) / " + $m + ";\n";

            $exp += $c + ".rx = ( " + $s + ".rx * " + ( $m - $i ) + " + " + $e + ".rx * " + $i +" ) / " + $m + ";\n";
            $exp += $c + ".ry = ( " + $s + ".ry * " + ( $m - $i ) + " + " + $e + ".ry * " + $i +" ) / " + $m + ";\n";
            $exp += $c + ".rz = ( " + $s + ".rz * " + ( $m - $i ) + " + " + $e + ".rz * " + $i +" ) / " + $m + ";\n";
            $exp += "\r";
        }
        expression -s $exp -o $s -ae 1 -uc all ;
        for($i = 1; $i < $cutnumbers; $i++){
            connectAttr -f ( $s + ".Detach" ) ( $ctrlplanes[$i] + ".Detach" );
            connectAttr -f ( $s + ".polyCutOffset")  ( $ctrlplanes[$i] + ".polyCutOffset" );
        }
        $z = getAttr ($ctrlplanes[0]+".translateZ");
        setAttr ($ctrlplanes[$m]+".translateZ") ($z+2);
    }
    select -r $ctrlplanes[0];
}

multiCut 6;
```

Scale/Rot/Move Manip to screenSpace (camera angle)



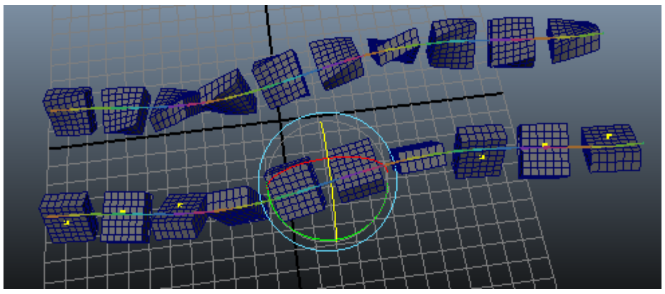
```
string $objs[] = `ls -sl`;
string $obj = `match "^[^\.]*" $objs[size($objs)-1]`;
//thnx NathanN @cgtalk vv
setToolTo moveSuperContext;
vector $centerPos = `manipMoveContext -q -position Move`;
$tmp = `group -em`;
move -r ($centerPos.x) ($centerPos.y) ($centerPos.z ) $tmp;
$cam = `lookThru -q`;
$ac = `aimConstraint -offset 0 0 0 -weight 1 -aimVector 1 0 0 -upVector 0 1 0 -worldUpType "objectrotation" -worldUpVector 0 1 0 -worldUpObject $cam $cam $tmp`;
float $ro[] = `xform -q -ws -ro $tmp`;
delete $ac $tmp;
$ro[0] = deg_to_rad($ro[0]);
$ro[1] = deg_to_rad($ro[1]);
$ro[2] = deg_to_rad($ro[2]);
select -r $objs;
if(`nodeType $objs[0]`!="transform"){
    hilite $obj;
    setSelectMode components Components;
}
setToolTo scaleSuperContext;
manipScaleContext -e -ah 3 -useManipPivot 0 -useObjectPivot 0 -oa $ro[0] $ro[1] $ro[2] -m 6 Scale;
manipMoveContext -e -oa $ro[0] $ro[1] $ro[2] -m 6 Move;
manipRotateContext -e -oa $ro[0] $ro[1] $ro[2] -m 6 Rotate;
```

Locators on normals



```
string $objs[] = `ls -sl -fl`;
setToolTo moveSuperContext;
$manipMode = `manipMoveContext -q -mode Move`;
manipMoveContext -e -mode 9 Move;
for($o in $objs){
    select -r $o;
    vector $centerPos = `manipMoveContext -q -position Move`;
    vector $centerRot = `manipMoveContext -q -oa Move`;
    string $loca[] = `spaceLocator -n "normalLoc"`;
    setAttr ($loca[0]+".tx") ($centerPos.x);
    setAttr ($loca[0]+".ty") ($centerPos.y);
    setAttr ($loca[0]+".tz") ($centerPos.z);
    setAttr ($loca[0]+".rx") (rad_to_deg($centerRot.x));
    setAttr ($loca[0]+".ry") (rad_to_deg($centerRot.y));
    setAttr ($loca[0]+".rz") (rad_to_deg($centerRot.z));
    setAttr ($loca[0]+".localScaleX") 2;
    setAttr ($loca[0]+".localScaleY") 0.2;
    setAttr ($loca[0]+".localScaleZ") 0.2;
    setAttr ($loca[0]+".overrideEnabled") 1;
    setAttr ($loca[0]+".overrideColor") 17;


};
```

Uniform weight for shells



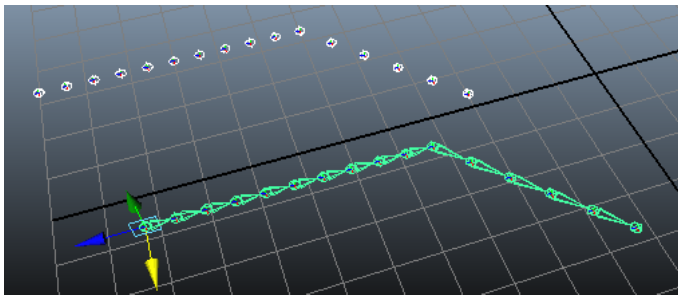
```
//some code from tiddlyspot
//if you have skin bound objects that need to have a uniform skinning
//ie all the vertices in the shell should get their weights from the selected vertex
string $vertices[] = `ls -sl -fl -l`;
for($vertex in $vertices){
    select -r $vertex;
    string $selObj[] = `ls -o -sl`;
    string $selPt[] = `ls -sl`;
    string $sck = `findRelatedSkinCluster($selObj[0])`;
    string $skinInfluences[]=`skinPercent -ib 0.000001 -q -t $sck $selPt[0]`;
    float $sknVals[]=`skinPercent -ib 0.000001 -q -v  $sck $selPt[0]`;
    polyConvertToShell;
    string $shell[] = `ls -sl`;
    string $eval = "skinPercent -zri 1";


    for($i=0;$i<size($skinInfluences);$i++){
        if($sknVals[$i] > 0){
            $eval += (" -tv "+$skinInfluences[$i]+" "+$sknVals[$i]);
        }
    }
    $eval += " "+$sck+" "+stringArrayToString($shell," ");
    eval($eval);
}
select -r $vertices;
```

quad face to single vertex selection

```
//select quads first
string $empty[] = {};
for($o in `ls -sl -fl`){
select -r $o;
ConvertSelectionToVertices;
string $v[] = `ls -sl -fl`;
    print($v);
    $empty[size($empty)] = $v[0];
}
select -r $empty;
```

```
//for multiple objects, with vtx 1
for($o in `ls -sl -fl -l`){
select -r ($o+".vtx[1]");
string $vertices[] = `ls -sl -fl -l`;
for($vertex in $vertices){
    select -r $vertex;
    string $selObj[] = `ls -o -sl`;
    string $selPt[] = `ls -sl`;
    string $sck = `findRelatedSkinCluster($selObj[0])`;
    string $skinInfluences[]=`skinPercent -ib 0.000001 -q -t $sck $selPt[0]`;
    float $sknVals[]=`skinPercent -ib 0.000001 -q -v  $sck $selPt[0]`;
    polyConvertToShell;
    string $shell[] = `ls -sl`;
    string $eval = "skinPercent -zri 1";


    for($i=0;$i<size($skinInfluences);$i++){
        if($sknVals[$i] > 0){
            $eval += (" -tv "+$skinInfluences[$i]+" "+$sknVals[$i]);
        }
    }
    $eval += " "+$sck+" "+stringArrayToString($shell," ");
    eval($eval);
}
//select -r $vertices;
}
```

Recursive Parent



```
//recursive parent
//useful if you select a bunch of joints in order and want them parented to each other, not to the last of the selection
string $objs[] = `ls -sl`;
for($i=size($objs)-1;$i>0;$i--){
    parent $objs[$i] $objs[($i-1)];
}
select -r $objs[0];
```

Reconnect stuff

Reconnect follicles

```
//select SHAPE then follicles
string $sel[] = `ls -sl`;
for($i=1;$i<size($sel);$i++){
    string $ch[] = `listRelatives -s $sel[$i]`;
    catchQuiet(`connectAttr -f ($sel[0]+".worldMatrix[0]") ($ch[0]+".inputWorldMatrix")`);
    catchQuiet(`connectAttr -f ($sel[0]+".outMesh") ($ch[0]+".inputMesh")`);
}
```

Reconnect motionpaths

Check out the Better python version

```
//select the curve to attach all motion paths to and run script
string $objs[] = `ls -sl -l`;
string $shape[] = `listRelatives -s -pa`;

for($o in `ls -l -type "motionPath"`){
    print($o+"\n");
    catchQuiet(`connectAttr -f ($shape[0]+".worldSpace[0]") ($o+".geometryPath")`);
}
```

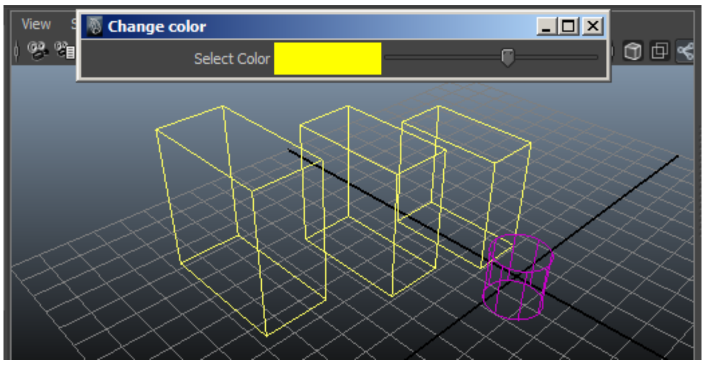
Zero Out

```
zero;
global proc zero(){
    string $sel[] = `ls -sl`;
    for($obj in $sel){
        select -cl;
        string $grandParents[] = `listRelatives -p`;
        string $zer = `group -em -w -n ($obj+"_ZERO")`;
        parent $zer $obj;
        setAttr ($zer+".rotateZ") 0;
        setAttr ($zer+".translateX") 0;
        setAttr ($zer+".translateY") 0;
        setAttr ($zer+".translateZ") 0;
        setAttr ($zer+".rotateX") 0;
        setAttr ($zer+".rotateY") 0;
        setAttr ($zer+".scaleZ") 1;
        setAttr ($zer+".scaleX") 1;
        setAttr ($zer+".scaleY") 1;
        if(size($grandParents)==0){
            parent -w $zer;
        }else{
        parent $zer $grandParents[0];
        }
        parent $obj $zer;
    }
    select -r $sel;
}
```

Select influenced bones / get skinned meshes from bone

```
//select a mesh or a bone
//selects all bones rigging the mesh or selects all meshes being rigged by this bone

string $sele[] = `ls -sl`;
if(`nodeType $sele[0]` == "joint"){
    select(`listConnections ($sele[0]+".worldMatrix")`);
    select(`listConnections -t "mesh"`);
}else{
    select(`findRelatedSkinCluster($sele[0])`);
    select -r (`listConnections ".matrix"`);
}
```

Quick Change wire color



```
//edit: works with joints
global string $cursel[];
proc string changeColor(){
    global string $cursel[];
    if(size(`ls -sl`)>0 || size($cursel)> 0){
        if(size(`ls -sl`)>0){
            $cursel = `ls -sl`;
        }
        select -cl;
        for($obj in $cursel){
            string $shap[] = `listRelatives -s -f $obj`;

            $v = `colorIndexSliderGrp -q -v "colorSlider"`;
            if($shap[0] == ""){
                setAttr ($obj+".overrideEnabled") 1;
                setAttr ($obj+".overrideColor") $v;
            }
            for($s in $shap){
                setAttr ($s+".overrideEnabled") 1;
                setAttr ($s+".overrideColor") $v;
            }
        }
    }
    return 1;
}
proc plop(){
    global string $cursel[];
    select -r $cursel;
    $cursel = {};
}

string $window = `window -title "Change color"`;
columnLayout;
string $slid = `colorIndexSliderGrp -label "Select Color" -min 0 -max 31 -cc "plop()" -dc "changeColor()" "colorSlider"`;
showWindow $window;
```

Match shape

Forces the shape of object A on object B through a blendshape. To match controllers.

```
//matches shape if same number of points.
pickWalk -d down;
string $sele[] = `ls -sl`;
string $bls[] = `blendShape`;
string $shortname = `match "[^|]*$" $sele[0]`;
setAttr ($bls[0]+".weight[0]") 1;
//setAttr ($bls[0]+"."+$shortname) 1;
delete -ch $sele[1];
```

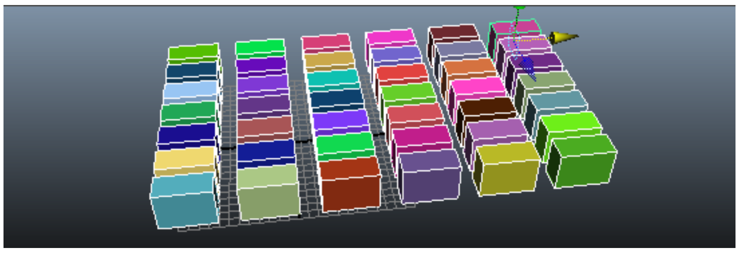
Delete groups w/o children

```
$deleteArray = {};
for($obj in `ls -l -tr`){
    $chld1 = `listRelatives $obj`;
    $chld2 = `listRelatives -s $obj`;
    if((size($chld1)+size($chld2)) == 0){
        $deleteArray[size($deleteArray)] = $obj;
    }
}
for($obj in $deleteArray){
    print(  "Deleted \""+$obj+"\" \n");
    delete $obj;
}
```

Emit from facing ratio

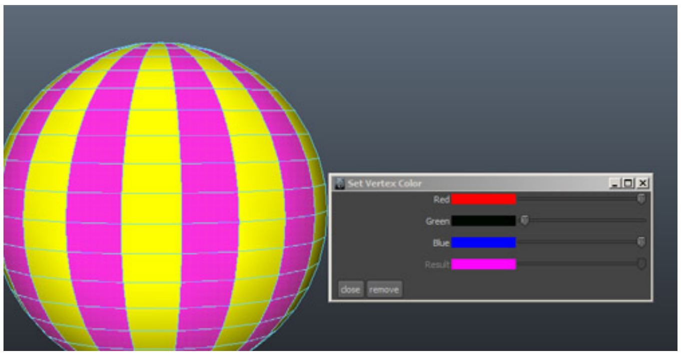
```
vector $bob = particleShape1.position;
vector $velo = particleShape1.velocity;
vector $prout = getAttr("camera1.translate");
vector $part = $prout-$bob;
$angle = (rad_to_deg(angle($velo,$prout))-90)/90;
$facingratio = (1-abs($angle));
//particleShape1.radiusPP = (1-abs($angle))/2;

if($facingratio <.8){
particleShape1.lifespanPP = 0;
}else{
particleShape1.lifespanPP = 1;
}
```

Random Color Viewport (à la 3dsmax)



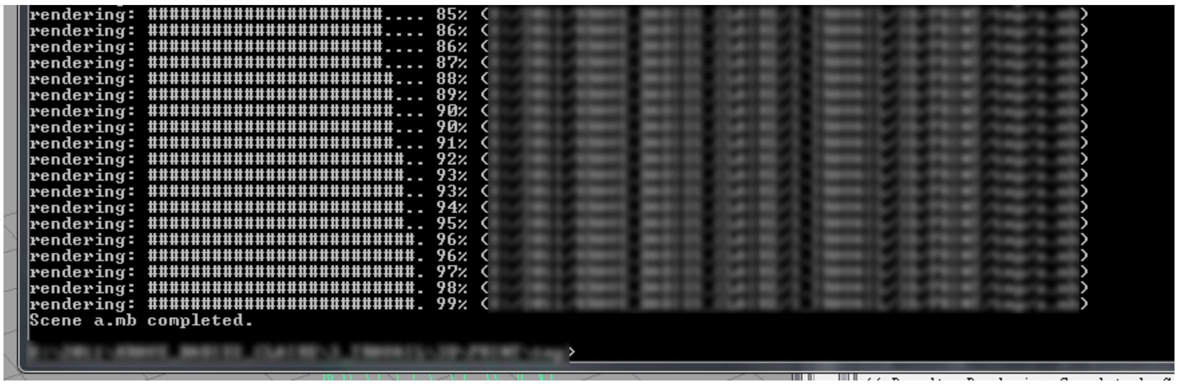
```
//creates differently colored objects using vertex color (doesn't affect shaders)
string $sel[] = `ls -sl`;
for($obj in $sel){
    select -cl;
    select -r $obj;
    $r = rand(0,1);
    $g = rand(0,1);
    $b = rand(0,1);
    polyColorPerVertex -r $r -g $g -b $b -a 1 -nun -cdo;
}
select -cl;
select -r $sel;
```

Set viewport color on selected obj/vertices



```
//if you want more details in the viewport (more info for animators). Does not affect shading
{
    proc chgcl(int $ch){

        if($ch){
            polyColorPerVertex -rem -nun;
            polyOptions -colorShadedDisplay false `ls -sl -o`;
        }else{
            float $r[] = `colorSliderGrp -q -rgb "col_r"`;
            float $g[] = `colorSliderGrp -q -rgb "col_g"`;
            float $b[] = `colorSliderGrp -q -rgb "col_b"`;
            colorSliderGrp -e -rgb ($r[0]) 0 0 "col_r";
            colorSliderGrp -e -rgb 0 ($g[1]) 0 "col_g";
            colorSliderGrp -e -rgb 0 0 ($b[2]) "col_b";
            colorSliderGrp -e -rgb ($r[0]) ($g[1]) ($b[2]) "col_result";
            polyColorPerVertex -rgb ($r[0]) ($g[1]) ($b[2]) -a 1 -nun -cdo;
        }
    }
    if (`window -exists vtxCol`) deleteUI vtxCol;
    if (`windowPref -exists vtxCol`) windowPref -remove vtxCol;
    string $window = `window -title "Set Vertex Color" vtxCol`;
    columnLayout -columnAttach "left" 5 -rowSpacing 10 -columnWidth 250;
    //columnLayout;
    colorSliderGrp -label "Red" -rgb 1 0 0 -cc "chgcl 0" "col_r";
    colorSliderGrp -label "Green" -rgb 0 1 0 -cc "chgcl 0" "col_g";
    colorSliderGrp -label "Blue" -rgb 0 0 1 -cc "chgcl 0" "col_b";
    colorSliderGrp -label "Result" -en 0 -rgb 1 1 1 "col_result";

    rowLayout -numberOfColumns 2;
    button -label "close" -command "deleteUI vtxCol";
    button -label "remove" -command "chgcl 1";
    showWindow;
}
```

postFrame status



```
rendering: #######################.... 85% ( [redacted] )
rendering: #######################.... 86% ( [redacted] )
rendering: #######################.... 86% ( [redacted] )
rendering: #######################.... 87% ( [redacted] )
rendering: ########################... 88% ( [redacted] )
rendering: ########################... 89% ( [redacted] )
rendering: ########################... 90% ( [redacted] )
rendering: ########################... 90% ( [redacted] )
rendering: ########################... 91% ( [redacted] )
rendering: #########################.. 92% ( [redacted] )
rendering: #########################.. 93% ( [redacted] )
rendering: #########################.. 93% ( [redacted] )
rendering: #########################.. 94% ( [redacted] )
rendering: #########################.. 95% ( [redacted] )
rendering: ##########################. 96% ( [redacted] )
rendering: ##########################. 96% ( [redacted] )
rendering: ##########################. 97% ( [redacted] )
rendering: ##########################. 98% ( [redacted] )
rendering: ##########################. 99% ( [redacted] )
Scene a.mb completed.
[redacted]
```

```
int $m=(((`currentTime -q`)-(`playbackOptions -q -min`))/(`playbackOptions -q -max`)*100);
$b= "rendering: ";
for($i=0;$i<=25;$i++){
$b += ($m/4<$i)?".":"#";
}
trace($b+" "+$m+"% ("+`file -q -sn`+")");

/// ou mettre ça dans un userSetup:
///
$t = "int $m=(((`currentTime -q`)-(`playbackOptions -q -min`))/(`playbackOptions -q -max`)*100);$b= \"rendering: \";for($i=0;$i<=25;$i++){$b += ($m/4<$i)?\".\":\"#\";}trace($b+\" \"+$
setAttr -type "string" defaultRenderGlobals.postRenderMel $t;


//oneliner
int $e=`playbackOptions -q -max`;int $s=`playbackOptions -q -min`;int $c=`currentTime -q`;int $m=($c-$s)/$e*100;$b= "rndr: ";for($i=0;$i<=25;$i++){$b += ($m/4<$i)?".":"#";}trace($b+"
```

Tracking -> Maya to After

Maya Mel

```
// transform un point 3d en un point 2d (pour eviter a retracker derriere)

//////////////////////////////////////////////////////
// Rob Bredow                                       //
// rob (at) 185vfx.com                              //
// http://www.185vfx.com/                           //
// Copyright 3/2002 Rob Bredow, All Rights Reserved //
//////////////////////////////////////////////////////

global string $camera = "";


proc float round(float $val,float $dec){
    $sign = `sign $val`;
    float $dec = `pow 10 $dec`;
    $val = (int) (($val + $sign*5/($dec*10)) * $dec);
    $val = ($val / $dec);
    return $val;
}

// Get a matrix
proc matrix screenSpaceGetMatrix(string $attr){
  float $v[]=`getAttr $attr`;
  matrix $mat[4][4]=<<$v[0], $v[1], $v[2], $v[3];
             $v[4], $v[5], $v[6], $v[7];
             $v[8], $v[9], $v[10], $v[11];
             $v[12], $v[13], $v[14], $v[15]>>;
 return $mat;
}

// Multiply the vector v by the 4x4 matrix m, this is probably
// already in mel but I cant find it.
proc vector screenSpaceVecMult(vector $v, matrix $m){
  matrix $v1[1][4]=<<$v.x, $v.y, $v.z, 1>>;
  matrix $v2[1][4]=$v1*$m;
  return <<$v2[0][0], $v2[0][1],  $v2[0][2]>>;
}

global proc int screenSpace()
{
global string $camera;
global float $noOfDecimals = 4;

$w = `getAttr "defaultResolution.width"`;
$h = `getAttr "defaultResolution.height"`;
float $fs = `playbackOptions -q -min`;
float $fe = `playbackOptions -q -max`;
float $ct = `currentTime -q`;
$fps = `currentTime -edit 1sec -u 0`;
currentTime -edit $ct;
string $scenepath = `file -q -sn`;
string $filename = `match "[^/\\]*$" $scenepath`;
$filename = `match "^[^\.]*" $filename`;
$filename = ($filename =="")?"untitled.ma":$filename;
$firstline = "//maya to after track file: Filename:"+$filename+" Start:"+$fs+" End:"+$fe+" Fps:"+$fps+" Width:"+$w+" Height:"+$h+"\n";

if($camera == ""){
    string $panel = `getPanel -withFocus`;
    $camera = `modelPanel -q -cam $panel`;
}

  string $dumpList[] = `ls -sl -fl`;
  print ("Dumping selection...("+$camera+")\n");

  string $pointWsFile = `fileDialog -m 1 -dfn "trackpoints.txt"`;
  int $outFileId = fopen($pointWsFile,"w");

  if ($outFileId == 0) {
    print ("Could not open output file " + $pointWsFile);
    return -1;
  }

  float $tx[],$ty[],$tz[];
fprint $outFileId $firstline;
for($dumpPt in $dumpList){
fprint $outFileId "\n";
  for ($f=$fs;$f<=$fe;$f++)
  {
    currentTime $f;

    // get the world space position of the point into a vector
    float $ptPosWs[] = `xform -q -ws -t $dumpPt`;
    vector $ptVecWs = <<$ptPosWs[0],$ptPosWs[1],$ptPosWs[2]>>;

    // Grab the worldInverseMatrix from cam_main
    matrix $cam_mat[4][4] = screenSpaceGetMatrix($camera+".worldInverseMatrix");

    // Multiply the point by that matrix
    vector $ptVecCs = screenSpaceVecMult($ptVecWs,$cam_mat);

    // Adjust the point's position for the camera perspective
    float $hfv = `camera -q -hfv $camera`;
    float $ptx = (($ptVecCs.x/(-$ptVecCs.z))/tand($hfv/2))/2.0+.5;
    float $vfv = `camera -q -vfv $camera`;
    float $pty = (($ptVecCs.y/(-$ptVecCs.z))/tand($vfv/2))/2.0+.5;

    float $ptz = $ptVecCs.z;

    float $ww = ($ptx*$w);
    float $hh = ($pty*$h);

    $hh = $h - $hh;
    $hh =   100.0/85*($hh-.5*$h)+.5*$h;     //nasty hack because somehow the output is scaled @ 85% in height, dunno why.

    $ww = round($ww,$noOfDecimals);
    $hh = round($hh,$noOfDecimals);


    $line = $f+"\t"+$ww+ "\t" +$hh + "\n";
    fprint $outFileId $line;
  }
}
    fclose $outFileId;
    currentTime -edit $ct;
    return 1;
};
```

Script After (Panel)

```
function createUI(thisObj) {
    var myPanel = ( thisObj instanceof Panel) ? thisObj : new Window("palette", "Maya Track 1.0",[100, 100, 300, 300]);

    myPanel.createOptPnl = myPanel.add('panel', [10,10,180,57], 'Create');
    myPanel.createOptPnl.toCtrl = myPanel.createOptPnl.add('radiobutton', [16,9,85,33], 'PntCtrls');
    myPanel.createOptPnl.toNull  = myPanel.createOptPnl.add('radiobutton', [95,9,160,33], 'Nulls');
    myPanel.createOptPnl.toCtrl.value = 1;

    myPanel.optionsOptPnl = myPanel.add('panel', [10,62,180,105], 'Option(s)');
    myPanel.optionsOptPnl.shiftKf = myPanel.optionsOptPnl.add('Checkbox', [16,7,160,30], 'Nudge 1 keyframe back');
    myPanel.optionsOptPnl.shiftKf.value = 1;

    impButton = myPanel.add("button", [10, 113, 105, 137], "Import From File");
    impButton.onClick = main;

    return myPanel;
}
function main(){
app.beginUndoGroup("Add Maya Track Points");

check = false;
       if (app.project != null) {
      if (app.project.activeItem != null) {
         if (app.project.activeItem instanceof CompItem) {
        check = true;
        comp = app.project.activeItem;

        file = openfile();

        name = "";
        width = 0;
        height = 0;
        fps = 0;
        start = 0;
        end =0;

        firstline = readLine(file);
        split = firstline.split(" ");
        for(string in split){
            newsplit = split[string].split(":");
            switch(newsplit[0]){
                case "Filename":
                name = newsplit[1];
                break;
                case "Start":
                start = newsplit[1];
                break;
                case "End":
                end = newsplit[1];
                break;
                case "Fps":
                fps = newsplit[1];
                break;
                case "Width":
                width = newsplit[1];
                break;
                case "Height":
                height = newsplit[1];
                break;
            }
        }
        check2 = true;
        if(fps != comp.frameRate || width != comp.width || height != comp.height){
            alertText = "The track file and comp don't match, proceed?\n\n";
            alertText += "Source\t\tRate\tWidth\tHeight\tStart\tEnd\t\n";
            alertText += "-----------------------------------------------------------------------------------\n";
            alertText += "Track File:\t"+fps+"\t"+width+"\t"+height+"\t"+start+"\t"+end+"\t\n";
            alertText += "After Comp:\t"+comp.frameRate+"\t"+comp.width+"\t"+comp.height+"\t"+app.project.displayStartFrame+"\t"+comp.duration/comp.frameDuration+"\t\n";

            check2 = confirm(alertText,0,"Proceed?");
            }
        }
    }
    if(check && check2){
        curIt = "";
        point = 0;
        processing = 0;
        //token = 0;
        while(c=readLine(file)){
            processing++;
            if(c == ">newpoint<")
            {
                point++;
                if(mayaTrackPnl.createOptPnl.toCtrl.value == 1)
                {
                    curIt = addPointControl("TrackPoint "+point);
                }else{
                    curIt = createNull("TrackPoint "+point,comp.duration);
                }
            }else{
                    info = (c.split("\t"));
                    nudge = mayaTrackPnl.optionsOptPnl.shiftKf.value;
                    curIt.setValueAtTime((info[0]-nudge)*comp.frameDuration,[info[1],info[2]]);
                    writeLn("Track Line: "+processing+" ("+point+")");
                }
        }
    }else{
        if(!check){
        alert("No comp opened... ");
        }else{
            alert("Error");
            }
    }
}
clearOutput();
file.close();
app.endUndoGroup();
}

function openfile(){
    var myFile = File.openDialog ("Select track file","*.txt");
    var fileOK = myFile.open("r","TEXT","????");
    if(fileOK){
        return myFile;
        }else{
            //alert("Problem opening file");
        return false;
    }
}

function readLine(myFile){
    if(!myFile.eof){
        c = myFile.readln();
        if(!c){
            return ">newpoint<";
        }else{
        return c;
        }
    }else{
        return false;
    }
}

function createNull(name,duration){
    theComp = app.project.activeItem;
    theLayers = theComp.layers;
    mynull = theLayers.addNull(duration);
    mynull.name = name;
    mynull.shy = 1;
return mynull.position;
}
function addPointControl(name){
    theComp = app.project.activeItem;
    theLayers = theComp.layers;
    trckL = theLayers.byName("mayatrack");
    if(trckL == null){
        trckL = theLayers.addNull();
        trckL.name = "mayatrack";
        trckL.position.setValueAtTime(0,[0,0]);
    }
    nuPnt = trckL.Effects.addProperty("ADBE Point Control");
    nuPnt.name = name;
return nuPnt.property(1);
}
var mayaTrackPnl = createUI(this);
```

Get and keyframe object speed

```
global proc keySpeed(){
    $initialTime = `currentTime -q`;
    $target = `ls -sl`;
    if(!objExists ($target[0]+".speed")){
        addAttr -ln "speed"  -at double $target[0];
        setAttr -e -keyable true {$target[0]+".speed"};
    }
    float $fs = `playbackOptions -q -min`;
    float $fe = `playbackOptions -q -max`;
    currentTime -e ($fs-1);
    $posA = getPos($target[0]);
    $posB = $posA;
    for($i = $fs;$i<=$fe;$i++){
        currentTime -e $i;
        $posA = getPos($target[0]);
        vector $magn = <<$posB[0]-$posA[0],$posB[1]-$posA[1],$posB[2]-$posA[2]>>;
        //print("\n"+$i+" "+mag($magn));
        float $speed = `mag($magn)`;
        setAttr ($target[0]+".speed") $speed;
        setKeyframe ($target[0]+".speed");
        $posB = $posA;
    }
    currentTime -e $initialTime;
}

global proc float[] getPos(string $obj){
    select -r $obj;
    setToolTo moveSuperContext;
    float $centerPosi[] = `manipMoveContext -q -position Move`;
    return $centerPosi;
}
keySpeed;
```

Duplicates current cam and tear off copy

```
string $panel = `getPanel -withFocus`;
string $cameraName = `modelPanel -q -cam $panel`;
$duplicate = `duplicate -n {$cameraName+"_duplicate"} $cameraName`;

string $window = `window -title $duplicate -wh 300 200 -tlb 0`;
paneLayout;
$pan = `modelPanel -cam  $duplicate -mbv 1`;
showWindow $window;
```

erase cam

```
string $panel = `getPanel -withFocus`;
string $cameraName = `modelPanel -q -cam $panel`;
catchQuiet(`delete $cameraName`);
```

Charge un dossier de XPM dans le renderview

```
global proc loadXPMs(string $path){
    string $filez[] = `getFileList -folder $path -filespec "*.xpm"`;
    for($file in $filez){
        renderWindowLoadImageCallback "renderView" ($path+$file) "image";
        renderWindowMenuCommand keepImageInRenderView renderView;
    }
}
global proc openDir( string $filename, string $fileType )
{
    loadXPMs($filename+"/");
}
fileBrowserDialog -m 4 -fc "openDir" -ft "directory" -an "Open_Dir";
```

Reverse Selection

```
// select: obj2 - obj3 --> $obj 3 - $obj2 - $obj1
//
string $sele = "";
for($obj in `ls -sl`){
     $sele = $obj+" "+$sele;
}
eval("select -r "+$sele);
```

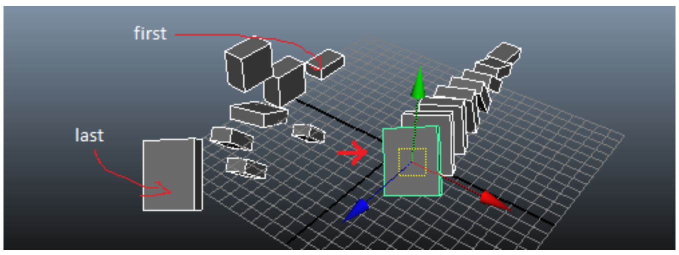
Bake To Clamp Keys

```
//Bake toutes les courbes d'anim en clamp et les déplaces de -0.5 dans le temps
//Pour Lionel
$from = `playbackOptions -q -ast`;
$to = `playbackOptions -q -aet`;
$range = $from+":"+$to;
string $sel[] = `ls`;
bakeResults -simulation true -t $range -sampleBy 1 -disableImplicitControl true -preserveOutsideKeys true -sparseAnimCurveBake false -removeBakedAttributeFromLayer false -bakeOnOverri
string $curves[] = `ls -type "animCurve"`;
selectKey -k $curves;
keyTangent -ott step;
keyframe -animation keys -option over -relative -timeChange (0 - .5) ;
```

Merge to shape

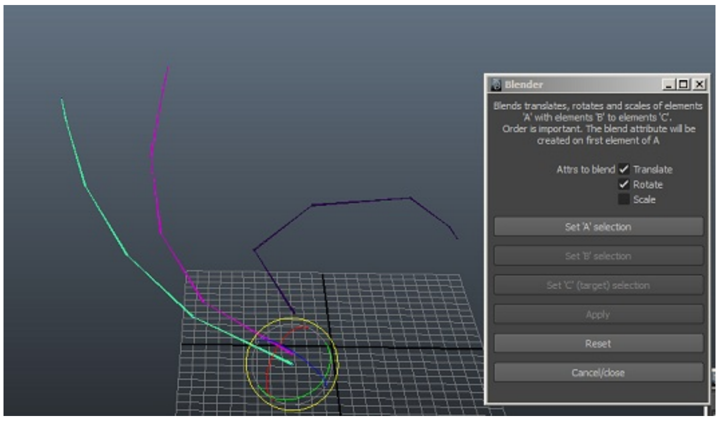
```
proc parentToShape(){

    //  bernie - mbernadat@gmail - feel free to take & modify ! v1
    //
    //  adds the current selected objects to the last object of the selection as shapes
    //  use it to add shapes to controlers. Renames the resulting shapes as "[Targetobject]Shape#"
    //
    //  ajoute les objets de la selection dans le dernier objet séléctionné (pour foutre des controlleurs
    //  sur des bones par exemple). Renomme les shapes resultantes en "[NomObjet]Shape#"


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

    if(size($sel) > 1){
        $target = $sel[size($sel)-1];
        for($a = 0;$a<size($sel)-1;$a++){
            if(size(`listRelatives -p $sel[$a]`) > 0){
                catch(`parent -w $sel[$a]`);
            }
            parent $sel[$a] $target;
            makeIdentity -apply true -t 1 -r 1 -s 1 -n 0;
            $shp = `listRelatives -shapes`;
            select $shp;
            select -add $target;
            parent -s -r;
        }
        delete(`listRelatives -typ "transform" $target`);
        select -r `listRelatives -shapes $target`;
        renameSelectionList($target+"Shape");
        select -r $target;
    }
}
parentToShape;
```

Linear Align Objects



```
arrange;
global proc arrange(){
    string $sel[] = `ls -sl`;
    $first = $sel[0];
    $size = size($sel);
    $last = $sel[$size-1];
    float $firstTr[] = `xform -q -r -t $first`;
    float $firstRt[] = `xform -q -r -ro $first`;
    float $firstSl[] = `xform -q -r -s $first`;
    float $lastTr[] = `xform -q -r -t $last`;
    float $lastRt[] = `xform -q -r -ro $last`;
    float $lastSl[] = `xform -q -r -s $last`;
    for($a = 0; $a<$size;$a++){
        $r = ($a*1.0)/($size-1);
        //super beau code  \o/
        setAttr ($sel[$a]+".translateX") ($firstTr[0]+$r*($lastTr[0]-$firstTr[0]));
        setAttr ($sel[$a]+".translateY") ($firstTr[1]+$r*($lastTr[1]-$firstTr[1]));
        setAttr ($sel[$a]+".translateZ") ($firstTr[2]+$r*($lastTr[2]-$firstTr[2]));
        setAttr ($sel[$a]+".rotateX") ($firstRt[0]+$r*($lastRt[0]-$firstRt[0]));
        setAttr ($sel[$a]+".rotateY") ($firstRt[1]+$r*($lastRt[1]-$firstRt[1]));
        setAttr ($sel[$a]+".rotateZ") ($firstRt[2]+$r*($lastRt[2]-$firstRt[2]));
        setAttr ($sel[$a]+".scaleX") ($firstSl[0]+$r*($lastSl[0]-$firstSl[0]));
        setAttr ($sel[$a]+".scaleY") ($firstSl[1]+$r*($lastSl[1]-$firstSl[1]));
        setAttr ($sel[$a]+".scaleZ") ($firstSl[2]+$r*($lastSl[2]-$firstSl[2]));
    }
}
```

Space attributes blender


Blends translates, rotates and scales of elements 'A' with elements 'B' to elements 'C'.
Order is important. The blend attribute will be created on first element of A

Attrs to blend ✓ Translate
✓ Rotate
Scale

```
{
    if (`window -exists blenderW`) deleteUI blenderW;
    if (`windowPref -exists blenderW`) windowPref -remove blenderW;
    string $window = `window -widthHeight 250 350 -title "Blender" blenderW `;
    columnLayout -columnAttach "both" 6 -rowSpacing 10 -columnWidth 250;
    text -ww 1 -l "Blends translates, rotates and scales of elements 'A' with elements 'B' to elements 'C'. \n Order is important. The blend attribute will be created on first element
    checkBoxGrp -numberOfCheckBoxes 3 -vr -label "Attrs to blend " -va3 1 1 1 -labelArray3 "Translate" "Rotate" "Scale" cbg;
    button -label "Set 'A' Selection" -command "selA" selAButton;
    button -label "Set 'B' Selection" -en 0 -command "selB" selBButton;

    button -label "Set 'C' (target) selection" -en 0 -command "selC" selCButton;
    button -label "Apply" -command "blendAttr" -en 0 blendButton;
    button -label "Reset" -command "resetAll" resetButton;
    button -label "Cancel/close" -command "cancelpr";
    showWindow $window;
    global string $A[];
    global string $B[];
    global string $C[];
    proc selA(){
        $objs = `ls -sl`;
        if(size($objs)!=0){
            button -e -en 0 -label ("Set 'A' Selection ("+size($objs[0])+")") "selAButton";
            button -e -en 1 "selBButton";
            $A = $objs;
            }
    }
    proc selB(){
        $objs = `ls -sl`;
        if(size($objs)!=0){
            button -e -en 0 -label ("Set 'B' Selection ("+size($objs[0])+")") "selBButton";
            button -e -en 1 "selCButton";
            $B = $objs;
            }
    }
    proc selC(){
        $objs = `ls -sl`;
        if(size($objs)!=0){
            button -e -en 0 -label ("Set 'C' (target) selection ("+size($objs[0])+")") "selCButton";
            button -e -en 1 "blendButton";
            $C = $objs;
            }
    }
    proc blendAttr(){
        $objs = $C;
        if(size($objs)!=0){

            int $trCB = `checkBoxGrp -q -v1 "cbg"`;
            int $roCB = `checkBoxGrp -q -v2 "cbg"`;
            int $scCB = `checkBoxGrp -q -v3 "cbg"`;

            string $at = `addAttr -ln "blender"  -at double  -min 0 -max 1 -dv (0.5) $A[0]`;
            setAttr -e-keyable true ($A[0]+".blender");

            for($i = 0; $i < size($B);$i++){
                string $ob_a = $A[$i];
                string $ob_b = $B[$i];
                string $ob_c = $objs[$i];
                if($trCB){
                    $md = `createNode blendColors`;
                    connectAttr -f ($ob_a+".translate") ($md+".color1");
                    connectAttr -f ($ob_b+".translate") ($md+".color2");
                    connectAttr -f ($A[0]+".blender") ($md+".blender");
                    connectAttr -f ($md+".output") ($ob_c+".translate");
                }
                if($roCB){
                    $md = `createNode blendColors`;
                    connectAttr -f ($ob_a+".rotate") ($md+".color1");
                    connectAttr -f ($ob_b+".rotate") ($md+".color2");
                    connectAttr -f ($A[0]+".blender") ($md+".blender");
                    connectAttr -f ($md+".output") ($ob_c+".rotate");
                }
                if($scCB){
                    $md = `createNode blendColors`;
                    connectAttr -f ($ob_a+".scale") ($md+".color1");
                    connectAttr -f ($ob_b+".scale") ($md+".color2");
                    connectAttr -f ($A[0]+".blender") ($md+".blender");
                    connectAttr -f ($md+".output") ($ob_c+".scale");
                }
            }
            select -r $A[0];
            resetAll();
        }else{
            warning "No objects selected";
        }
    }
    proc resetAll(){
        $A = {""};
        $B = {""};
        $C = {""};
        button -e -en 1 -label ("Set 'A' selection") "selAButton";
        button -e -en 0 -label ("Set 'B' selection") "selBButton";
        button -e -en 0 -label ("Set 'C' (target) selection") "selCButton";
        button -e -en 0 "blendButton";
    }
    proc cancelpr(){
        $A = {""};
        $B = {""};
        $C = {""};
        deleteUI blenderW;
    }
}
```

PNG snapshot of current cam

```
//saves a png snapshot of the current cam in the scene folder
string $sel[] = `ls -sl`;
select -d;
string $scenepath = `file -q -sn`;
string $filename = `match "[^/\\]*$" $scenepath`;
string $filename = `match "^[^\.]*" $filename`;
string $workdir = `substitute "/[^/]*$" $scenepath "/"`;
string $png = $workdir+$filename+".png";
$time = `currentTime -q`;
string $panel = `getPanel -withFocus`;
string $cameraName = `modelPanel -q -cam $panel`;
setAttr "defaultRenderGlobals.imageFormat" 32;

//SAVE DISPLAY OPTIONS
string $editorName = `getPanel -withFocus`;
$curState = `modelEditor -q -sts $editorName`;
string $modelEditorSettings = `match "^[^\//\;]*" $curState`;
$polyHud = `optionVar -q polyCountVisibility`;
$axisHud = `optionVar -q viewAxisVisibility`;
$frameHud = `optionVar -q currentFrameVisibility`;

//SET DISPLAY OPTIONS
setPolyCountVisibility 0;
setViewAxisVisibility 0;
setCurrentFrameVisibility 1;
string $optns = `modelEditor -e
                -useInteractiveMode 0
                -activeOnly 0
                -wireframeOnShaded 0
                -headsUpDisplay 1
                -selectionHiliteDisplay 1
                -xray 0
                -jointXray 0
                -activeComponentsXray 0
                -displayTextures 1
                -nurbsCurves 0
                -nurbsSurfaces 1
                -polymeshes 1
                -subdivSurfaces 1
                -planes 1
                -cameras 0
                -controlVertices 0
                -hulls 0
                -grid 0
                -joints 0
                -ikHandles 0
                -deformers 0
                -dynamics 0
                -fluids 0
                -hairSystems 0
                -follicles 0
                -nCloths 0
                -nRigids 0
                -dynamicConstraints 0
                -manipulators 0
                -handles 0
                -pivots 0 $editorName`;
setRendererInModelPanel hwRender_OpenGL_Renderer $editorName;
playblast -frame $time -format image -w 832 -h 468  -p 100 -cf $png;
setRendererInModelPanel base_OpenGL_Renderer $editorName;
setPolyCountVisibility $polyHud;
setViewAxisVisibility $axisHud;
setCurrentFrameVisibility $frameHud;
eval($modelEditorSettings);
select -r $sel;
print $png;
```

Right click in shelf (<maya2010)

```
//rajouter un flag -mi dans le mel de la shelf (ouvrir avec wordpad)
    shelfButton
        ......
        -style "iconOnly"
        -marginWidth 1
        -marginHeight 1
        -command "warning \"Clique droit!\";"
        -sourceType "mel"
        -actionIsSubstitute 0
        -commandRepeatable 1
        -mi "Menu droit 1" ("Commande")
        -mi "Menu droit 2" ("print(\"HI\")")
    ;
```

Remove useless keys

```
//removes useless keys for ppl who hit shift-s like monkeys

source generateChannelMenu.mel;
source channelBoxCommand.mel;
for($obj in `ls -sl`){
    float $testVar;
    float $testVar2;
    $from = `playbackOptions -q -ast`;
    $to = `playbackOptions -q -aet`;
    $safe = false;
    string $Attrs[] = `listAnimatable $obj`;
    for($Attr in $Attrs){
        $safe = false;
        $numbK = `keyframe -query -keyframeCount $Attr`;
        if($numbK > 0){
            $testVar = `getAttr -t $from $Attr`;
            $testVar2 = 0;
            for($i = $from;$i<=$to;$i++){
                $testVar2 = `getAttr -t $i $Attr`;
                    if($testVar2 != $testVar){
                    //print("Clés sur "+$Attr+"\n");
                    $safe = true;
                    break;
                }
                $testVar = $testVar2;
            }
            //print("\n"+($safe)?"\nAnim: "+$Attr:"\nNotAnim: "+$Attr);
        if(!$safe){
            CBdeleteConnection($Attr);
        }
        }
    }
}
```

Create ID Shaders

Crée des Surface Shaders pour les obj IDs chopper les objets et lancer le script (attention, il efface les shaders inutilisés) le script fout des surfaceShaders vachement colorés, en utilisant une liste specifique de couleur pour les objets les plus près de la caméra active on peut regler les min et max des Teinte Saturation Luminosité des autres

```
global proc createIDShaders(int $deleteUnusedShaders){


    string $forcedColors[] = {"1 0 0","0 1 0","0 0 1","1 1 1","0 1 1","1 1 0","1 0 1",".5 .5 .5"};  //'base' colors
    float $maxHue = 1;  //color hue
    float $minHue = 0;  //
    float $maxSat = 1;  //color saturation
    float $minSat =.8;  //
    float $maxVal = 1;  //color darkness
    float $minVal =.6;     //

    string $sel[] = `ls -sl`;
    $cam = getCurCam();
    $sortedobjects = sortFromClosestToFurthest($cam , $sel);
    float $oLen = `size($sortedobjects)`;
    float $cLen = `size($forcedColors)`;
    for($a = 0;$a<=$oLen-1;$a++){
        string $curColor;
        if($a<$cLen){
            $curColor = $forcedColors[$a];
        }else{
            $curColor = createRGBColorString($minHue,$minSat,$minVal,$maxHue,$maxSat,$maxVal);
        }
        //string $curColor = ($a<$cLen-1)?$forcedColors[$a]:createRGBColorString($minHue,$minSat,$minVal,$maxHue,$maxSat,$maxVal);
        $shader = createSurfShader($sortedobjects[$a],$curColor);
        connectShader($sortedobjects[$a],$shader);
    }
    if($deleteUnusedShaders){
        $cmd = `MLdeleteUnused`;
    }
    $cmd = `select -r $sel`;
}


proc string getCurCam(){
    string $currentPanel = `getPanel -withFocus`;
    string $cameraName = `modelPanel -q -cam $currentPanel`;
    return $cameraName;
}

proc float distanceFrom(string $from,string $to){
    float $worldPosFrom[] = `xform -q -ws -t $from`;
    float $worldPosTo[] = `xform -q -ws -t $to`;
    vector $diff = << $worldPosFrom[0]-$worldPosTo[0], $worldPosFrom[1]-$worldPosTo[1], $worldPosFrom[2]-$worldPosTo[2]>>;
    float $distance = `mag $diff`;
    return $distance;
}
proc string[] sortFromClosestToFurthest(string $fromObj, string $objectList[]){
    //needs "distanceFrom()" proc
    int $objssize = size($objectList);
    string $distances[];

    //build array with distances and object names to sort them easily later
    for($a= 0;$a<= $objssize-1;$a++){
        $distances[$a] = distanceFrom($fromObj,$objectList[$a])+"="+$objectList[$a];
        //print($distances[$a]+"\n");
    }

    //sort
    $distances = `sort $distances`;

    //clean array to have sorted object names only
    for($a= 0;$a<= $objssize-1;$a++){
        string $component = `substitute "^[^=]*\\=" $distances[$a] ""`;
        $distances[$a] = $component;
    }
    return $distances;
}

proc string createSurfShader(string $name,string $r_g_b){
    string $nuSurfShader = `shadingNode -asShader surfaceShader`;
    string $nuName = $name+"_IDShade";
    $nuSurfShader = `rename $nuSurfShader $nuName`;
    eval("setAttr "+$nuSurfShader+".outColor -type double3 "+$r_g_b);
    eval("sets -renderable true -noSurfaceShader true -empty -name "+$nuSurfShader+"SG");
    eval("connectAttr -f "+$nuSurfShader+".outColor "+$nuSurfShader+"SG.surfaceShader");
    return $nuSurfShader;
}
proc connectShader(string $obj, string $shader){
    string $shapes[] = `listRelatives -s -path $obj`;       //select shapes to prevent a parent selecting its children and giving them a shader
    $cmd = `select -r $shapes[0]`;                  //un-comment the next line  to keep this 'feature'
    //$cmd = `select -r $obj`;
    $cmd = "sets -e -forceElement "+$shader+"SG";
    print($cmd+"\n");
    eval($cmd);
}
proc string createRGBColorString(float $mH,float $mS, float $mV,float $MH,float $MS, float $MV){
    vector $nucolor = `rand <<$mH,$mS,$mV>> <<$MH,$MS,$MV>>`;
    $nucolor = `hsv_to_rgb $nucolor`;
    return $nucolor.x+" "+$nucolor.y+" "+$nucolor.z;
}
createIDShaders 1;
```

ikSpline stretch

```
string $objs[] = `ls -sl`;
string $curve = $objs[`size($objs)`-1];

string $arclen = `arclen -ch 1 $curve`;
string $multDiv = `createNode multiplyDivide -n "stretchMultiplier"`;
setAttr ($multDiv+".operation") 2;
connectAttr -f ($arclen+".arcLength") ($multDiv+".input1X");
float $val = `getAttr ($arclen+".arcLength")`;
setAttr ($multDiv+".input2X") $val;
int $l = `size($objs)`-1;
for($i=0;$i<$l;$i++){
    connectAttr -f ($multDiv+".outputX") ($objs[$i]+".scaleX");
}
```

Curve Text Captions

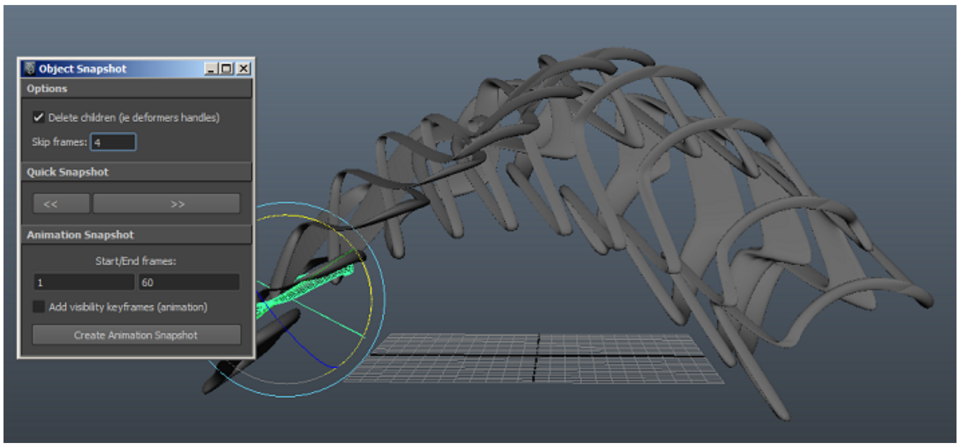
```
float $corr[] = {-1.25, -0.5, 0};
float $corrScale = .1;
$objs = `ls -sl`;
string $textobjs = "";
for($a=0;$a<=size($objs)-1;$a++){
    string $obj = $objs[$a];
    $text4 = `textCurves -ch 1 -f "Arial|h-11|w200|c0" -t $obj`;
    $t = $text4[0];
    $textobjs += " "+$t;
    eval("setAttr "+$t+".scaleX "+$corrScale);
    eval("setAttr "+$t+".scaleY "+$corrScale);
    eval("setAttr "+$t+".scaleZ "+$corrScale);
    makeIdentity -apply true -t 0 -r 0 -s 1 -n 0;
    parent -relative $t $obj;
    eval("setAttr "+$t+".translateX "+$corr[0]);
    eval("setAttr "+$t+".translateY "+$corr[1]);
    eval("setAttr "+$t+".translateZ "+$corr[2]);
}
eval("select -r "+$textobjs);
```

Rounding numbers in mel

Doesn't exist, use this:

```
 //round(30.3333,1) ==> 30.3
 proc float round(float $val,float $dec){

   $sign = `sign $val`;
   float $dec = `pow 10 $dec`;
   $val = (int) (($val + $sign*5/($dec*10)) * $dec);
   $val = ($val / $dec);
   return $val;
 }
```

Super Snapshot



Object Snapshot

Options

☑ Delete children (ie deformers handles)

Skip frames: 4

Quick Snapshot

<< >>

Animation Snapshot

Start/End frames:

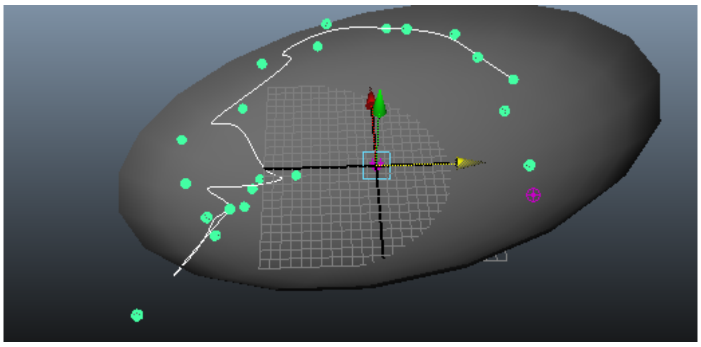
1 60

☐ Add visibility keyframes (animation)

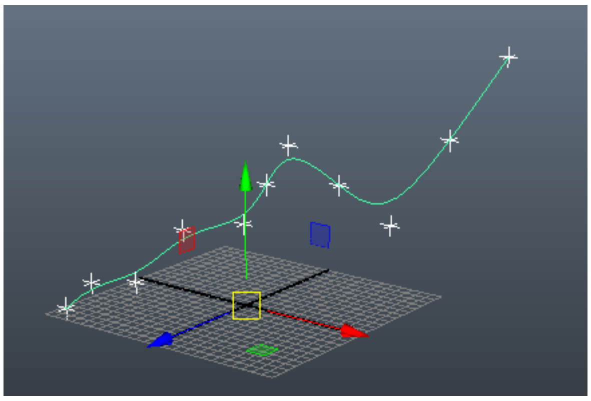
Create Animation Snapshot

```
//like animation snapshot but works with other types of shapes
superSnapshot;

global proc superSnapshot(){
    print "better animation snapshot tool by bernie@berniebernie.fr";
    if (`window -exists superSnapshotWindow`) deleteUI superSnapshotWindow;
    if (`windowPref -exists superSnapshotWindow`) windowPref -remove superSnapshotWindow;

    int $e = `playbackOptions -query -maxTime`;
    int $s = `playbackOptions -query -minTime`;

    string $window = `window -t "Object Snapshot" superSnapshotWindow`;

    columnLayout -adjustableColumn true -columnAlign "center";

        frameLayout -mh 10 -mw 10 -l "Options" ;

            checkBoxGrp -columnWidth2 100 165 -numberOfCheckBoxes 1 -label1 "Delete children (ie deformers handles)" -v1 true "sS_delChildren";
            rowLayout -nc 2;
                text -label "Skip frames:" -align "center";
                textField  -text 0 -w 50 "sS_skip";
                setParent..;
            //checkBoxGrp -columnWidth2 100 165 -numberOfCheckBoxes 1 -label1 "Parent to group in world" -v1 true;
            setParent..;

        frameLayout -mh 10 -mw 10 -l "Quick Snapshot" ;

            rowLayout -nc 2 -adjustableColumn 2 -columnAlign  2 "right";
                button -label "  <<          " -command ("superSnapshotProc(-1)");
                button -label "          >>  " -command ("superSnapshotProc(1)");
                setParent..;
            setParent..;

        frameLayout -mh 10 -mw 10 -l "Animation Snapshot" ;

            text -label "Start/End frames:" -align "center";

            rowLayout -nc 2;
                textField  -text $s "sS_start";
                textField  -text $e "sS_end";
            setParent..;



            checkBoxGrp -columnWidth2 100 165 -numberOfCheckBoxes 1 -label1 "Add visibility keyframes (animation)" -v1 false "sS_visib";

            button -label "Create Animation Snapshot" -command ("superSnapshotProc(0)");

    showWindow $window;
}

global proc superSnapshotProc(float $quickSnapshot){

    string $objects[] = `ls -sl -l`;

    if(size($objects)>=1){
        int $curFrame = `currentTime -q`;
        int $delChildren = `checkBoxGrp -q -v1 "sS_delChildren"`;
        int $sF = `textField -q -text "sS_start"`;
        int $eF = `textField -q -text "sS_end"`;
        int $skip = `textField -q -text "sS_skip"`;
        int $visibKey = `checkBoxGrp -q -v1 "sS_visib"`;

        string $parentGrp = "";
        if(`objExists($objects[0]+"_snapshots")`){
            $parentGrp = $objects[0]+"_snapshots";
        }else{
            $parentGrp = `group -em -n ($objects[0]+"_snapshots")`;
        }

        select -r $objects;

        if($quickSnapshot != 0){
            currentTime ($curFrame+$quickSnapshot+$quickSnapshot*$skip);
            string $newObjects[] = `duplicate -rc`;
            $newObjects = `ls -sl`;
            parent $newObjects $parentGrp;
            if($delChildren){
                catchQuiet(delete(`listRelatives -children -typ "transform" $newObjects`));
            }
        }else{
            for($i = $sF;$i<=$eF;$i+=(1+$skip)){
                select -r $objects;
                currentTime $i;
                string $newObjects[] = `duplicate -rc`;
                $newObjects = `ls -sl`;
                parent $newObjects $parentGrp;
                if($delChildren){
                    delete(`listRelatives -children -typ "transform" $newObjects`);
                }

                if($visibKey){
                    for($o in $newObjects){
                        eval("setKeyframe -time "+($i-1)+" -value 0 "+$o+".visibility");
                        eval("setKeyframe -time "+$i+" -value 1 "+$o+".visibility");
                        eval("setKeyframe -time "+($i+1)+" -value 0 "+$o+".visibility");
                    }
                }
            }
            currentTime $curFrame;
        }
    }else{
        warning ">>> No object(s) selected!";
    }
    select -r $objects;
}
```

Particules or Objects to Curve



```
//cree une courbe passant par toutes les partoches dans l'ordre des IDs
string $objs[] = `ls -sl`;
$rel = listRelatives($objs[0]);
if(objectType($rel)=="particle"){
    $ptc = $rel[0];
    $cnt = `getAttr($ptc+".count")`;
    $curve = "curve";
    float $ptAr[] = eval("getParticleAttr -at position -array true "+$ptc+".pt[0:"+($cnt-1)+"]");
    for($a = 0;$a<($cnt-2);$a++){
        $curve += " -p "+$ptAr[$a*3]+" "+$ptAr[$a*3+1]+" "+$ptAr[$a*3+2];
    }
    eval($curve);
}else{
    warning "select particle object";
}
```



```
$curve = "curve";

for($o in `ls -sl`){
    $x = `xform -q -ws -t $o`;
    print($x);
    $curve += (" -p "+$x[0]+" "+$x[1]+" "+$x[2]);
}
eval($curve);
```

Selection to shelf 2

```
// place that in a shelf button; or put it in a .mel file in your script directory and use a shelf button/shortcut that calls "selectionToShelf"
//
// takes your current controllers; shapes or whatever and saves the selection to the shelf, prompting you for a name
// if you are in an animation scene with lots of 'cloned' references (that share controller names) and choose the option 'All Characters'
// the created shelf button will select the controllers that are on the currently selected object using the namespace
//
// call "selectionToShelf 1" for a simpler interface

global proc selectionToShelf(int $simple){
    $option = "";
    if(!$simple){
    string $option = `confirmDialog -title "Selection to shelf" -message "Create a button for this unique character or all similar characters ?"
        -button "This character"
        -button "All characters" -defaultButton "All characters"
        -button "Cancel"
        -cancelButton "Cancel" -dismissString "No"`;
    }else{
        $option="This character";
    }
    if($option!="Cancel" && size(`ls -sl`)>0){
        string $sel[] = `ls -sl`;
        string $text;
        string $result = `promptDialog
            -title "Selection Name"
            -message "Name your shelf button (3-6 letters):"
            -button "OK" -button "Cancel"
            -defaultButton "OK" -cancelButton "Cancel"
            -dismissString "Cancel"`;
        $text = `promptDialog -query -text`;
        if ($result == "OK") {
            if($option=="This character"){
                $selstring = "select -add "+stringArrayToString($sel," ");
                textToShelf ($text, $selstring);
            }else{
                $namespace = `match ".*:" $sel[0]`;
                for($i=0;$i<size($sel);$i++){
                    $sel[$i] = `match "[^:.]*$" $sel[$i]`;
                }
                string $str= "__NS__"+stringArrayToString($sel," __NS__");
                $selstring = "string $sel[] = `ls -sl`;string $ns = `match \".*:\" $sel[0]`;string $se[] = stringToStringArray(substituteAllString(\""+$str+"\",\"__NS__\",$ns),\" \");
                textToShelf ($text, $selstring);
            }
        }
    }else{
        warning "Select one or more object";
    }
}
selectionToShelf 1;
```

Snippets / Pastebins

Create follicle on points (mFollicles): http://pastebin.com/raw.php?i=52GGufMt

3D point to 2d screenspace: http://pastebin.com/raw.php?i=C4nwpXLB

Sur le net

- Redistribue les edges proprement sur une surface
  - http://www.tgjay.com/htms/research/tgPolyRelax.htm

Annotate (arrow from obj to obj)

```
string $sl[] = `ls -sl`;
$from = $sl[1];
$to = $sl[0];

$an = `annotate -tx "" -p 0 0 0 $from`;
catchQuiet(`parent -r $an $to`);
setAttr ($an+".overrideDisplayType") 1;
setAttr ($an+".overrideEnabled") 1;
setAttr ($an+".overrideColor") 13;
```

Random rotates

```
for($o in `ls -sl`){
    setAttr($o+".rotate") (rand(-180,180)) (rand(-180,180)) (rand(-180,180));
}
```

Random select 1/2

```
string $sele[] = {};
for($o in `ls -sl`){
    if(rand(1)<.5){
        $sele[size($sele)] = $o;
    }
}
select -r $sele;
```

Random translates

```
for($o in `ls -sl`){

    vector $v = `sphrand(1)`;
    float $f[] = `getAttr($o+".translate")`;
    setAttr($o+".translate") ($v.x+$f[0]) ($v.y+$f[1]) ($v.z+$f[2]);
}
```

Array procs

```
//is value in array
proc int inArray(float $v,float $thearray[]){
    int $flag = 0;
    for($item in $thearray){
        if($item == $v){
            $flag = 1;
            break;
        }
    }
    return $flag;
}
```

Edge to Loc

```
string $edges[] = `ls -sl`; //expects edges
for($edge in $edges){
    select -r $edge;
    string $vtx[] = `polyListComponentConversion -ff -fe -fuv -fvf -tv`;
    $vtx = `ls -fl $vtx`;
    float $p1[] = `xform -q -ws -t  $vtx[0]`;
    float $p2[] = `xform -q -ws -t  $vtx[1]`;
    float $diff[] = {$p2[0]-$p1[0],$p2[1]-$p1[1],$p2[2]-$p1[2]};
    float $angls[] = `angleBetween -euler -v1 1 0 0 -v2 $diff[0] $diff[1] $diff[2]`; //works in my case

    $loc = `spaceLocator`;
    move -r (($p1[0]+$p2[0])/2) (($p1[1]+$p2[1])/2) (($p1[2]+$p2[2])/2) $loc;
    rotate -r -os $angls[0] $angls[1] $angls[2]  $loc;
}
```

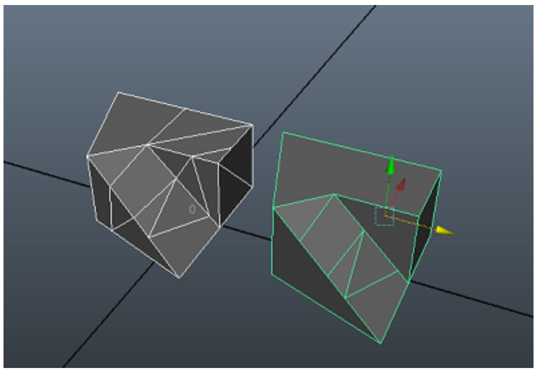
Find Centroid

```
//NathanN @cgtalk
setToolTo moveSuperContext;
vector $centerPos = `manipMoveContext -q -position Move`;
$b = `spaceLocator`;
move -r ($centerPos.x) ($centerPos.y) ($centerPos.z) $b;
```

Faces to obj

```
//selection of faces to objects
$selection = `ls -sl`;
select -cl;
for($obj in $selection){;
    string $no_component = `match "^[^\.]*" $obj`;
    select -add $no_component;
}
```

Remove useless edges



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

for($o in $sel){
    //thnx to NathanN
    catchQuiet(removeUselessEdge($o));
}
select -r $sel;
proc removeUselessEdge(string $obj){
    select -r $obj;
    string $mySelection[] = `ls -sl`;
    $softEdge = `polySoftEdge -angle 1 -ch 1 $mySelection`;
    ConvertSelectionToEdges;
    polySelectConstraint -m 3 -type 32768 -sm 2;
    string $selectEdges[] = `ls -sl`;
    delete $softEdge;
    polyDelEdge -cv on;
    polySelectConstraint -m 0 -type 0x0000 -sm 0;
}
```

Retrieved from "https://berniebernie.fr/w/index.php?title=Maya_Mel&oldid=306"

Category: Maya

- This page was last modified on 20 July 2018, at 12:42.