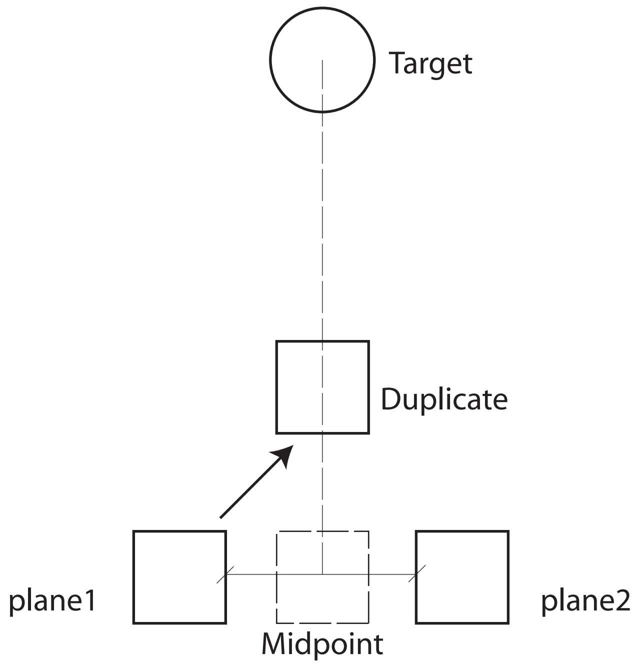
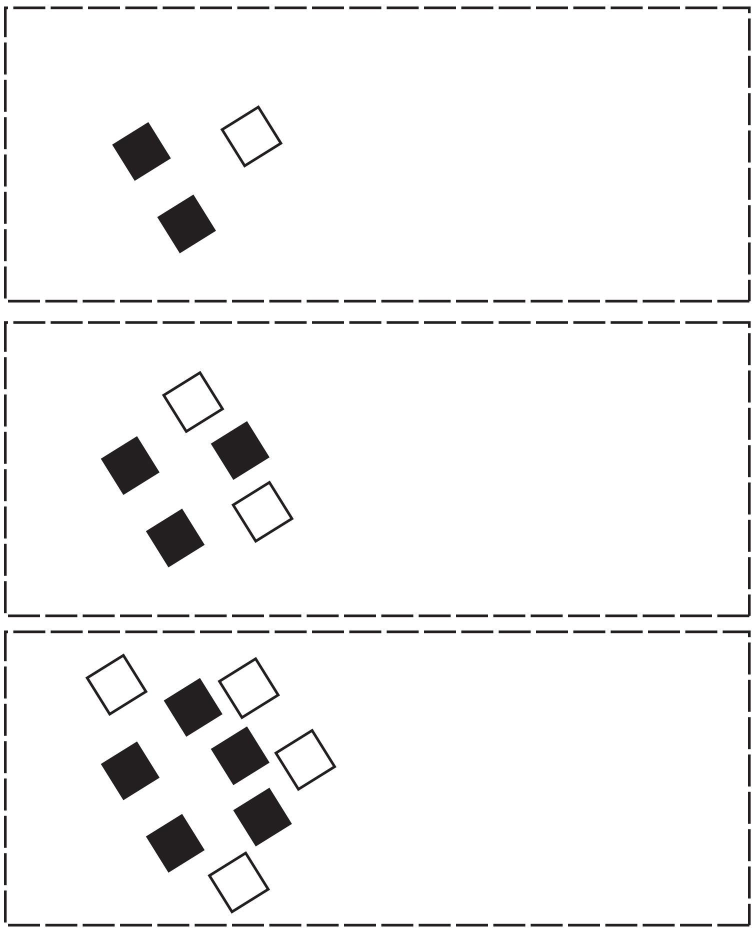
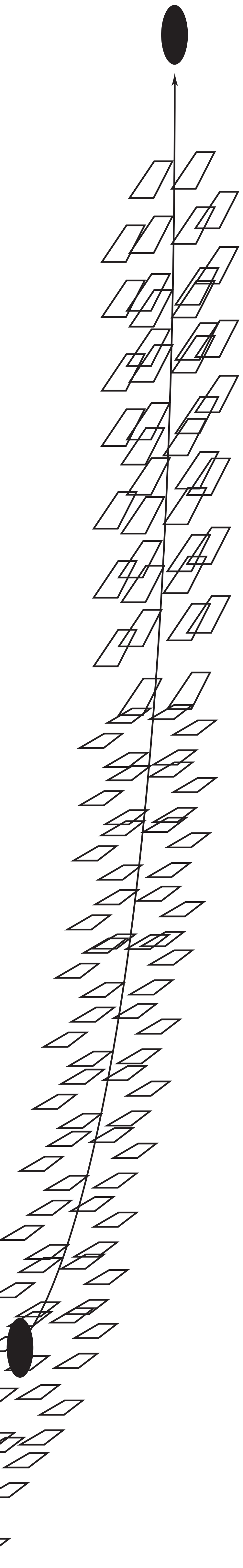
1 basic replication



2 exponential growth



3 follows animated target



```
//Tracking script
//
//Written by: Sky Milner
//Date: 3-03-08
//
//This script duplicates selected objects based on the
//location of a target.
//
//Setup:
//First make an object and call it "Target", Animate target as you like, make 6 nurbsPlanes
//
//
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


string $Selection[] = {"nurbsPlane1", "nurbsPlane2", "nurbsPlane2", "nurbsPlane3",
"nurbsPlane3","nurbsPlane4","nurbsPlane4","nurbsPlane5","nurbsPlane5","nurbsPlane6", "nurbsPlane6","nurbsPlane1"};
int $SelSize = size($Selection);
int $SquareCounter = 0;

for ($i = 0; $i < 50; ++$i) {

  string $TempSelection[];
  int $CurrentNumber = 0;
  int $SelectionCounter = 0;

  do {

    //++++++++++++++Duplicate and Morph an object between 2 initial planes and target++++++++++++++++++++

    //load objects and get values
    string $FirstObject = $Selection[$CurrentNumber];
    string $SecondObject = $Selection[$CurrentNumber+1];

    float $XYZfirst[] = `pointPosition ($FirstObject + ".cv [0] [0]")` ;
    vector $VecFirst = <<$XYZfirst[0], $XYZfirst[1], $XYZfirst[2]>> ;

    float $XYZsecond[] = `pointPosition ($SecondObject + ".cv [0] [0]")` ;
    vector $VecSecond = <<$XYZsecond[0], $XYZsecond[1], $XYZsecond[2]>> ;

    float $XYZtarget[] = `pointPosition ("Target.cv [0] [0]")`;
    vector $VecTarget = <<$XYZtarget[0], $XYZtarget[1], $XYZtarget[2]>>;


    $Between = $VecFirst - $VecSecond;

    //Duplicate and move into position

    duplicate -rr -n ("series"+$i+"square"+$SquareCounter) $Selection[($CurrentNumber)];

    string $Duplicate = ("series"+$i+"square"+$SquareCounter);


    //setAttr ($Duplicate+ ".translate") ($XYZfirst[0] + $Between.x/(-2)) ($XYZfirst[1] + $Between.y / (-2) ) ($XYZfirst[2] + $Between.z / (-2));

                xform  -r -t ($Between.x/(-2)) ($Between.y / (-2)) ($Between.z / (-2)) $Duplicate;


    float $XYZduplicate[] = `pointPosition ($Duplicate + ".cv [0] [0]")`;
    vector $Vecduplicate = <<$XYZduplicate[0], $XYZduplicate[1], $XYZduplicate[2]>>;

    $Between2 = $VecTarget - $Vecduplicate;

    //setAttr ($Duplicate+ ".translate") ($XYZduplicate[0] + $Between2.x/4) ($XYZduplicate[1] + $Between2.y/2) ($XYZduplicate[2] + $Between2.z/4);

                xform  -r -t ($Between2.x/4) ($Between2.y/2) ($Between2.z/4) $Duplicate;


    //Scale Duplicate

    float $scalefactor = ((mag($Between2)*.01)+.75);

    //setAttr ($Duplicate+ ".scaleX") $scalefactor;
    //setAttr ($Duplicate+ ".scaleY") $scalefactor;
    //setAttr ($Duplicate+ ".scaleZ") $scalefactor;


    //Manipulate cv[0]

    float $PointX = `getAttr ($Duplicate + ".cv[0].xValue")`;
    float $PointY = `getAttr ($Duplicate + ".cv[0].yValue")`;
    float $PointZ = `getAttr ($Duplicate + ".cv[0].zValue")`;

    vector $VecPoint = <<$PointX, $PointY, $PointZ>>;

    $Between3 = $VecTarget - $VecPoint;

    setAttr ($Duplicate+ ".cv[0].xValue") ($PointX + $Between3.x/(10));
    setAttr ($Duplicate+ ".cv[0].yValue") ($PointY + $Between3.y/(10));
    setAttr ($Duplicate+ ".cv[0].zValue") ($PointZ + $Between3.z/(10));

    //setAttr ($Duplicate + ".objectColor") $SquareCounter;

    $TempSelection [$SelectionCounter] = $FirstObject;
    $TempSelection [$SelectionCounter+1] = $Duplicate;
    $TempSelection [$SelectionCounter+2] = $SecondObject;
    $TempSelection [$SelectionCounter+3] = $Duplicate;


    $CurrentNumber = $CurrentNumber+2;
    $SelectionCounter = $SelectionCounter + 4;
    $SquareCounter++;
  } while ($CurrentNumber < $SelSize);

  currentTime $i;

  $Selection = $TempSelection;
                $SelSize = size($Selection);
                if ($SelSize > 100) {
                            for ($s = 0; $s < 100; ++$s) {
                            $Selection[$s] = $Selection[($Selsize - 100 +$s)];
                            }

                $SelSize = 100; }


}


//assigning transparencies
int $iii = 0;
string $jump[] = `ls -tr "series*"`;
int $sizejump = size($jump);
do {

    string $nicklambert = "nicklambert"+$iii;
    string $nicklambertSG = "nicklambert"+$iii+"SG";
    string $nicklambertoutcolor = "nicklambert"+$iii+".outColor";
    string $nicklambertSGsS = "nicklambert"+$iii+"SG.surfaceShader";
    string $nicklamberttrans = "nicklambert"+$iii+".transparency";

    shadingNode -asShader lambert -n $nicklambert;
    renderCreateBarCB -asShader "surfaceShader" lambert;
    sets -renderable true -noSurfaceShader true -empty -name $nicklambertSG;
    connectAttr -f $nicklambertoutcolor $nicklambertSGsS;

    currentTime $iii;
    setAttr $nicklamberttrans -type double3 1 1 1 ;
    setKeyframe $nicklamberttrans;
    float $iiiend = $iii + 40;
    currentTime $iiiend;
    setAttr $nicklamberttrans -type double3 0 0 0 ;
    setKeyframe $nicklamberttrans;
              float $iiiend2 = $iii + 80;
    currentTime $iiiend2;
    setAttr $nicklamberttrans -type double3 .70 .70 .70 ;
    setKeyframe $nicklamberttrans;


    select -r ("*square"+$iii) ;
    sets -e -forceElement $nicklambertSG;

$iii=$iii + 1;
}

while ($iii < $sizejump);

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

//assigns color

float $temp = 0;

for ($i=0; $i<=9; ++$i) {
string $series[] = `ls -tr ("series" + $i + "square*")`;
int $seriessize = size($series);
//print ($seriessize + "<--");
int $s = 0;
float $upper = 0;

do {
                string $Shapes[] = `listRelatives($series[$s])`;
                string $ShadGrp[] = `listConnections ($Shapes[0])`;
                string $Shader[] = `listConnections ($ShadGrp[0])`;
                string $CurShader = $Shader[4];
                setAttr ($CurShader+".color") -type "double3" $upper $temp $temp;
                $upper = ($s * .1);
                $s++;
  } while ($s < $seriessize);

if ($temp == 0) {
$temp = 1; } else {
$temp = 0;}
print ($temp + "<<<<");

}
```

Generative Scripting [Spring 2008] Instructor [Nicholas Pisca] Designer [Sky Milner]