

Appendix A

(source code)

```

proc int powers (int $x, ix Page 1 15/03/2007nt $power)
{
int $z = $x;
  for($i=0; $i<$power; $i++)
  {
    $x = $z*$x;
  }
return $x;
}

proc animate(int $case)
{

  if($case == 1)
  {
currentTime 1;
    for ($i = 1; $i <= 2; $i++)
    {
      string $subCurve_select = "subCurve" + $i*2 + ".maxValue";
      setAttr $subCurve_select 0.0001;
      setKeyframe $subCurve_select;
    }

currentTime 50;
    for ($i = 1; $i <= 2; $i++)
    {
      string $subCurve_select = "subCurve" + $i*2 + ".maxValue";
      setAttr $subCurve_select 1;
      setKeyframe $subCurve_select;
    }
  }

  if($case == 2)
  {
int $arr1 [2] = {1,4};
int $arr2 [4] = {2,3,5,6};

currentTime 1;
    for ($i = 0; $i <= 1; $i++)
    {
      string $subCurve_select = "subCurve" + $arr1[$i]*2 + ".maxValue";
      setAttr $subCurve_select 0.0001;
      setKeyframe $subCurve_select;
    }

currentTime 50;
    for ($i = 0; $i < 2; $i++)
    {
      string $subCurve_select = "subCurve" + $arr1[$i]*2 + ".maxValue";
      setAttr $subCurve_select 1;
      setKeyframe $subCurve_select;
    }

    for ($i = 0; $i < 4; $i++)
    {
      string $subCurve_select = "subCurve" + $arr2[$i]*2 + ".maxValue";
      setAttr $subCurve_select 0.0001;
      setKeyframe $subCurve_select;
    }

currentTime 100;
    for ($i = 0; $i < 4; $i++)
    {
      string $subCurve_select = "subCurve" + $arr2[$i]*2 + ".maxValue";
      setAttr $subCurve_select 1;
      setKeyframe $subCurve_select;
    }
  }

  if($case == 3)
  {
int $arr1 [2] = {1,8};
int $arr2 [4] = {2,5,9,12};
int $arr3 [8] = {3,4,6,7,10,11,13,14};
  }
}

```

```

/*1st branches*/
for ($i = 0; $i < 2; $i++)
{
    currentTime 1;
    string $subCurve_select = "subCurve" + $arr1[$i]*2 + ".maxValue";
    setattr $subCurve_select 0.0001;
    setKeyframe $subCurve_select;
    currentTime 50;
    string $subCurve_select = "subCurve" + $arr1[$i]*2 + ".maxValue";
    setattr $subCurve_select 1;
    setKeyframe $subCurve_select;
}

/*2nd branches*/
for ($i = 0; $i < 4; $i++)
{
    currentTime 50;
    string $subCurve_select = "subCurve" + $arr2[$i]*2 + ".maxValue";
    setattr $subCurve_select 0.0001;
    setKeyframe $subCurve_select;
    currentTime 100;
    string $subCurve_select = "subCurve" + $arr2[$i]*2 + ".maxValue";
    setattr $subCurve_select 1;
    setKeyframe $subCurve_select;
}

/*3rd branches*/
for ($i = 0; $i < 8; $i++)
{
    currentTime 100;
    string $subCurve_select = "subCurve" + $arr3[$i]*2 + ".maxValue";
    setattr $subCurve_select 0.0001;
    setKeyframe $subCurve_select;
    currentTime 150;
    string $subCurve_select = "subCurve" + $arr3[$i]*2 + ".maxValue";
    setattr $subCurve_select 1;
    setKeyframe $subCurve_select;
}
}

if($case == 4)
{
    int $arr1 [2] = {1,16};
    int $arr2 [4] = {2,9,17,24};
    int $arr3 [8] = {3,6,10,13,18,21,25,28};
    int $arr4 [16] = {4,5,7,8,11,12,14,15,19,20,22,23,26,27,29,30};

/*1st branches*/
for ($i = 0; $i < 2; $i++)
{
    currentTime 1;
    string $subCurve_select = "subCurve" + $arr1[$i]*2 + ".maxValue";
    setattr $subCurve_select 0.0001;
    setKeyframe $subCurve_select;
    currentTime 50;
    string $subCurve_select = "subCurve" + $arr1[$i]*2 + ".maxValue";
    setattr $subCurve_select 1;
    setKeyframe $subCurve_select;
}

/*2nd branches*/
for ($i = 0; $i < 4; $i++)
{
    currentTime 50;
    string $subCurve_select = "subCurve" + $arr2[$i]*2 + ".maxValue";
    setattr $subCurve_select 0.0001;
    setKeyframe $subCurve_select;
    currentTime 100;
    string $subCurve_select = "subCurve" + $arr2[$i]*2 + ".maxValue";
    setattr $subCurve_select 1;
    setKeyframe $subCurve_select;
}

/*3rd branches*/
for ($i = 0; $i < 8; $i++)
{
    currentTime 100;
    string $subCurve_select = "subCurve" + $arr3[$i]*2 + ".maxValue";
    setattr $subCurve_select 0.0001;

```

```

        setKeyframe $subCurve_select;
currentTime 150;
        string $subCurve_select = "subCurve" + $arr3[$i]*2 + ".maxValue";
        setAttr $subCurve_select 1;
        setKeyframe $subCurve_select;
    }
/*4th branches*/
    for ($i = 0; $i < 16; $i++)
    {
        currentTime 150;
        string $subCurve_select = "subCurve" + $arr4[$i]*2 + ".maxValue";
        setAttr $subCurve_select 0.0001;
        setKeyframe $subCurve_select;
        currentTime 200;
        string $subCurve_select = "subCurve" + $arr4[$i]*2 + ".maxValue";
        setAttr $subCurve_select 1;
        setKeyframe $subCurve_select;
    }
}
if($case == 5)
{
    int $arr1 [2] = {1,32};
    int $arr2 [4] = {2,17,33,48};
    int $arr3 [8] = {3,10,18,25,34,41,49,56};
    int $arr4 [16] = {4,7,11,14,19,22,26,29,35,38,42,45,50,53,57,60};
int $arr5 [32] = {5,6,12,13,8,9,15,16,20,21,23,24,27,28,30,31,36,37,39,40,43,44,46,47,51,52,54,55,58,59,61,62};

    for ($i = 0; $i < 2; $i++)
    {
        currentTime 1;
        string $subCurve_select = "subCurve" + $arr1[$i]*2 + ".maxValue";
        setAttr $subCurve_select 0.0001;
        setKeyframe $subCurve_select;
        currentTime 50;
        string $subCurve_select = "subCurve" + $arr1[$i]*2 + ".maxValue";
        setAttr $subCurve_select 1;
        setKeyframe $subCurve_select;
    }

    for ($i = 0; $i < 4; $i++)
    {
        currentTime 50;
        string $subCurve_select = "subCurve" + $arr2[$i]*2 + ".maxValue";
        setAttr $subCurve_select 0.0001;
        setKeyframe $subCurve_select;
        currentTime 100;
        string $subCurve_select = "subCurve" + $arr2[$i]*2 + ".maxValue";
        setAttr $subCurve_select 1;
        setKeyframe $subCurve_select;
    }

    for ($i = 0; $i < 8; $i++)
    {
        currentTime 100;
        string $subCurve_select = "subCurve" + $arr3[$i]*2 + ".maxValue";
        setAttr $subCurve_select 0.0001;
        setKeyframe $subCurve_select;
        currentTime 150;
        string $subCurve_select = "subCurve" + $arr3[$i]*2 + ".maxValue";
        setAttr $subCurve_select 1;
        setKeyframe $subCurve_select;
    }

    for ($i = 0; $i < 16; $i++)
    {
        currentTime 150;
        string $subCurve_select = "subCurve" + $arr4[$i]*2 + ".maxValue";
        setAttr $subCurve_select 0.0001;
        setKeyframe $subCurve_select;
        currentTime 200;
        string $subCurve_select = "subCurve" + $arr4[$i]*2 + ".maxValue";
        setAttr $subCurve_select 1;
        setKeyframe $subCurve_select;
    }
}

```

```

    }

    for ($i = 0; $i < 32; $i++)
    {
        currentTime 200;
        string $subCurve_select = "subCurve" + $arr5[$i]*2 + ".maxValue";
        setAttr $subCurve_select 0.0001;
        setKeyframe $subCurve_select;
        currentTime 250;
        string $subCurve_select = "subCurve" + $arr5[$i]*2 + ".maxValue";
        setAttr $subCurve_select 1;
        setKeyframe $subCurve_select;
    }
}

```

```

/*animate the visibility, to hide little stubs*/
int $hi = powers(2,$case) -2;

for ($i = 1; $i <= $hi; $i++)
{
    string $subCurve = "subCurve" + $i*2 + ".maxValue";
    string $vis = "extrudedSurface" + $i + ".visibility";
    float $valCheck = `getAttr $subCurve`;

```

```

currentTime 0;
$valCheck = `getAttr $subCurve`;
    if ($valCheck <= 0.0001)
    {
        setAttr $vis 0;
        setKeyframe $vis;
    }

```

```

currentTime 2;
$valCheck = `getAttr $subCurve`;
    if ($valCheck > 0.0001)
    {
        setAttr $vis 1;
        setKeyframe $vis;
    }

```

```

currentTime 50;
$valCheck = `getAttr $subCurve`;
    if ($valCheck <= 0.0001)
    {
        setAttr $vis 0;
        setKeyframe $vis;
    }

```

```

currentTime 51;
$valCheck = `getAttr $subCurve`;
    if ($valCheck > 0.0001)
    {
        setAttr $vis 1;
        setKeyframe $vis;
    }

```

```

currentTime 100;
$valCheck = `getAttr $subCurve`;
    if ($valCheck <= 0.0001)
    {
        setAttr $vis 0;
        setKeyframe $vis;
    }

```

```

currentTime 101;
$valCheck = `getAttr $subCurve`;
    if ($valCheck > 0.0001)
    {
        setAttr $vis 1;
    }

```

```

        setKeyframe $vis;
    }

    currentTime 150;
    $valCheck = `getAttr $subCurve`;
    if ($valCheck <= 0.0001)
    {
        setAttr $vis 0;
        setKeyframe $vis;
    }

    currentTime 151;
    $valCheck = `getAttr $subCurve`;
    if ($valCheck > 0.0001)
    {
        setAttr $vis 1;
        setKeyframe $vis;
    }

    currentTime 200;
    $valCheck = `getAttr $subCurve`;
    if ($valCheck <= 0.0001)
    {
        setAttr $vis 0;
        setKeyframe $vis;
    }

    currentTime 201;
    $valCheck = `getAttr $subCurve`;
    if ($valCheck > 0.0001)
    {
        setAttr $vis 1;
        setKeyframe $vis;
    }
}

/*Extrude the circles along the corresponding curves*/
proc animateExtrude(int $n , int $rec)
{
    /*initiate counter to make sure this is only ran once*/
    global int $count=0;
    $count = $count + 1;

    if ($count == 1)
    {
        /*create array for storing branch numbers*/
        int $branch_order_arr[ 30 ];

        for ($i = 1; $i <=$n; $i ++)
        {
            string $curve_select = "curve" + $i;
            string $circle_select = "nurbsCircle" + $i;

            /*add extra control vertices to the curve*/
            rebuildCurve -rt 0 -s 1 $curve_select;

            /*move the control vertices to create a better flow*/
            for($j = 1; $j <= 2; $j++)
            {
                string $sub_curve_select = $curve_select + ".cv[" + $j + "]";

                float $x_move = rand (-0.1, 0.1);
                float $y_move = rand (0, 0.2);
                float $z_move = rand (-0.1, 0.1);

                select $sub_curve_select;
                move -r $x_move $y_move $z_move;
            }
        }

        /*Extrusion*/
        select $circle_select $curve_select;
        extrude -rn true -scale 0.6;
    }
}

```

```

    }
/*Animation*/
animate($rec);
}

$count = 0;
}

proc vector reflect(vector $i, vector $j)
{
    $i = unit($i);
    $j = unit($j);
    return $i - 2 * dot($i, $j) * $j;
}

proc float[] aim (vector $vec)
{
    float $out[2];
    float $xAngle;
    float $zAngle;
    float $xyLength;
    float $vecLength;

    $xyLength = sqrt(($vec.x) * ($vec.x) + ($vec.y) * ($vec.y));
    $vecLength = sqrt(($vec.x) * ($vec.x) + ($vec.y) * ($vec.y) + ($vec.z) * ($vec.z));

    if($xyLength == 0)
        $zAngle = 0;
    else
        $zAngle = acos(($vec.y)/$xyLength);

    $xAngle = acos($xyLength/$vecLength);

    $xAngle = ($vec.z) > 0 ? $xAngle : -$xAngle;
    $out[0] = $xAngle;

    $zAngle = ($vec.x) > 0 ? -$zAngle : $zAngle;
    $out[1] = $zAngle;
    return $out;
}

proc Vine (int $recursion,
    vector $base,
    vector $direction,
    float $branchAngle,
    float $branchLength,
    float $initialWidth,
    int $deg, string $obj, string $cam)
{
    /*Create the cameras necessary for the projections*/
    /*set the counter*/
    global int $c = 0;
    $c = $c+1;

    /*check the recursion depth*/
    if ($recursion <= 0)
        return;

    /*calculate the radius of the circle representing the branch radius*/
    float $radius = sin(deg_to_rad($branchAngle)) * $branchLength;

    /*find a random vector on the surface of the circle*/
    vector $random;

    float $x = rand(-$radius, $radius);
    float $y = 0;
    float $z = rand(-$radius, $radius);

```

```

$random = <<$x, $y, $z>>;

/*
$random = unit ($random);
$random = $random * $radius;*/

/*translate the circle up the branch*/
$random = $random + <<0, $branchLength, 0>>;

/*rotate the vector by the brach angle only x & z matter*/
float $angle[2] = aim ($direction);
float $x;          /*rotates in the ZY plane*/
float $z;          /*rotates in the XY plane*/

vector $x_axis = <<1, 0, 0>>;
vector $y_axis = <<0, 1, 0>>;
vector $z_axis = <<0, 0, 1>>;

$random = rot($random, $x_axis, $angle[0]);
$random = rot($random, $z_axis, $angle[1]);

$random = <<($random.x), ($random.y), ($random.z)>>;

/*draw the line*/
vector $end = $base + $random;
curve -d 1      -p ($base.x) ($base.y) ($base.z)
               -p ($end.x) ($end.y) ($end.z);

/*Create circles along the vine in order to use extrude later*/
circle -ch on -o on -nr 0 1 0 ;
scale $initialWidth $initialWidth $initialWidth;
move -r ($base.x) ($base.y) ($base.z) ;

/*Find the angle by which the circle must be tilted*/
float $angleCircle[2] = aim($random);
float $x_degrees = rad_to_deg($angleCircle[0]);
float $y_degrees = rad_to_deg($angleCircle[1]);

rotate -r $x_degrees 0 $y_degrees;

/*recurse*/
Vine($recursion - 1, $end, $direction, $branchAngle, $branchLength * 0.6, $initialWidth * 0.6, $deg, $obj, $cam);

/*REFLECT branch*/
$random = reflect ( -$random, $direction );
$random = <<($random.x), ($random.y), ($random.z)>>;

vector $end2= $base + $random;

curve -d 1      -p ($base.x) ($base.y) ($base.z)
               -p ($end2.x) ($end2.y) ($end2.z);

/*Create circles along the vine in order to use extrude later*/
circle -ch on -o on -nr 0 1 0 ;
scale $initialWidth $initialWidth $initialWidth;
move -r ($base.x) ($base.y) ($base.z) ;

/*Find the angle by which the circle must be tilted*/
float $angleCircle[2] = aim($random);
float $x_degrees = rad_to_deg($angleCircle[0]);
float $y_degrees = rad_to_deg($angleCircle[1]);

rotate -r $x_degrees 0 $y_degrees;

/*recurse*/
Vine($recursion - 1, $end2, $direction, $branchAngle, $branchLength * 0.6, $initialWidth * 0.6,
$deg, $obj, $cam);

```



```

/*Animate Extrude along the curves*/
/*check that current recursion is the last one, and that all the branches are already drawn*/
int $c1;

int $b_check = `strcmp "no_obj" $obj`;

$c1 = powers(2, $deg) - 1;

    if ($c == powers(2, $deg)-1)
    {
        global int $flag = 0;
        lookThru $cam;

        if ($flag == 0 && $b_check != 0)
        {

            /*project curves onto object and duplicate projected curves*/
            for ($i = 1; $i < powers(2, $deg)-1; $i++)
            {
                string $select = "curve" + $i;
                string $select2 = $obj + "->projectionCurve" + $i + "_1";

                select $select $obj;
                projectCurve -un false;

                duplicateCurve -ch 1 -rn 0 -local 0 $select2;
                DeleteHistory;

                int $persp = `strcmp "persp" $cam`;
                int $top = `strcmp "top" $cam`;
                int $bot = `strcmp "bot" $cam`;
                int $front = `strcmp "front" $cam`;
                int $back = `strcmp "back" $cam`;
                int $side = `strcmp "side" $cam`;
                int $left = `strcmp "left" $cam`;

                if($top == 0)
                    move 0 0.1 0;
                if($bot == 0)
                    move 0 -0.1 0;
                if($front == 0)
                    move 0 0 0.1;
                if($back == 0)
                    move 0 0 -0.1;
                if($side == 0)
                    move 0.1 0 0;
                if($left == 0)
                    move -0.1 0 0;

                $flag = $flag + 1;
            }
        }
    }
    global int $flag2 = 0;

    /*delete original & projected curves and rename the new ones*/
    if ($flag2 == 0 && $b_check != 0)
    {
        for ($i = 1; $i < powers(2, $deg)-1; $i++)
        {
            string $select = $obj + "->projectionCurve" + $i;
            string $select2 = "curve" + $i;
            string $select3 = "duplicatedCurve" + $i;

            select $select $select2;
            delete;

            select $select3;
            rename $select2;

            $flag2 = $flag2 + 1;
        }
    }

```

```

    }
}

global int $flag3 = 0;
/*move circles down to the new curves*/
if ($flag3 == 0 && $b_check != 0)
{
    for ($i = 1; $i < powers(2, $deg)-1; $i++)
    {
        string $select = "nurbsCircle" + $i;
        string $select2 = "curve" + $i + ".cv[0]";

        vector $move = `pointPosition $select2`;

        select $select;
        move ($move.x) ($move.y) ($move.z);

        $flag3 = $flag3 + 1;
    }
}

if($c == $c1)
{
    animateExtrude($c-1 , $deg);

    $flag = 0;
    $flag1 = 0;
    $flag2 = 0;
    $flag3 = 0;
    $c = 0;
}

//switchModelView persp;
lookThru persp;
}
}

proc vineTest()
{
    global string $recursion;
    global string $angle;
    global string $width;

    global string $pos;
    global string $length;
    global string $direction;
    global string $cameras;

    /*Create the extra cameras needed for the projections*/

    /*query the values*/
    int $rec = `intSliderGrp -query -value $recursion`;
    int $len = `intFieldGrp -query -value1 $length`;

    float $ang = `floatSliderGrp -query -value $angle`;
    float $wid = `floatSliderGrp -query -value $width`;

    int $move_x = `intFieldGrp -query -value1 $pos`;
    int $move_y = `intFieldGrp -query -value2 $pos`;
    int $move_z = `intFieldGrp -query -value3 $pos`;

    int $dir_x = `intFieldGrp -query -value1 $direction`;
    int $dir_y = `intFieldGrp -query -value2 $direction`;
    int $dir_z = `intFieldGrp -query -value3 $direction`;

    vector $move = <<$move_x, $move_y, $move_z>>;
    vector $dir = <<$dir_x, $dir_y, $dir_z>>;

    $cam = `textScrollList -query -si $cameras`;
    print "$cam is: ";
    print $cam;
}

```

```

$objj = `ls -sl`;

global int $run = 0;
global string $no_delete[];
string $delete[];

/*make sure objects that already exist in the scene do not get deleted*/
if ($run == 0)
{
    select -all;
    $no_delete = `ls -dag`;
    $run = $run + 1;
}

$delete = `ls -dag`;

select $delete;
select -d $no_delete;
delete;

/*Create the cameras for the projections*/
camera;
rename "bot";
setAttr "bot.rotateX" 90;
setAttr "bot.visibility" 0;

camera;
rename "back";
setAttr "back.rotateY" 180;
setAttr "back.visibility" 0;

camera;
rename "left";
setAttr "left.rotateY" 90;
setAttr "left.visibility" 0;

Vine($rec, $move, $dir, $ang, $len, $wid, $rec, $obj[0], $cam[0]);
}

proc vineTest2()
{
    global string $recursion;
    global string $angle;
    global string $width;

    global string $pos;
    global string $length;
    global string $direction;

    /*query the values*/
    int $rec = `intSliderGrp -query -value $recursion`;
    int $len = `intFieldGrp -query -value1 $length`;

    float $ang = `floatSliderGrp -query -value $angle`;
    float $wid = `floatSliderGrp -query -value $width`;

    int $move_x = `intFieldGrp -query -value1 $pos`;
    int $move_y = `intFieldGrp -query -value2 $pos`;
    int $move_z = `intFieldGrp -query -value3 $pos`;

    int $dir_x = `intFieldGrp -query -value1 $direction`;
    int $dir_y = `intFieldGrp -query -value2 $direction`;
    int $dir_z = `intFieldGrp -query -value3 $direction`;

    vector $move = <<$move_x, $move_y, $move_z>>;
    vector $dir = <<$dir_x, $dir_y, $dir_z>>;

    string $obj = "no_obj";
    string $cam = "persp";

    global int $run = 0;
    global string $no_delete[];

```

```

global string $no_mat[];
string $delete[];

/*make sure objects that already exist in the scene do not get deleted*/
if ($run == 0)
{
    select -all;
    $no_delete = `ls -dag`;
    $run = $run + 1;
}

$delete = `ls -dag`;

select $delete;
select -d $no_delete;
delete;

/*select $no_delete;
connectAttr -f lambert2.outColor lambert2SG.surfaceShader;*/

Vine($rec, $move, $dir, $ang, $len, $wid, $rec, $obj, $cam);
}
x      Page 1115/03/2007/*Procedures reversing the surface direction*/
proc reverseU()
{
    $obj = `ls -sl`;
    reverseSurface -d 0 -ch 1 -rpo 1 $obj;
    select $obj;
}
proc reverseV()
{
    $obj = `ls -sl`;
    reverseSurface -d 1 -ch 1 -rpo 1 $obj;
    select $obj;
}

/*create the GUI*/
string $window = `window -title "Vine Creator v1.0"
                    -widthHeight 100 55`;
columnLayout -adjustableColumn true;

$recursion = `intSliderGrp -label "Vine Recursion Depth" -field true -min 1 -max 5 -value 3 -step 1`;
$angle = `floatSliderGrp -label "Branch Angle" -field true -min 0 -max 90 -value 30`;
$width = `floatSliderGrp -label "Width" -field true -min 0 -max 1 -value 0.1`;

$length = `intFieldGrp -numberOfFields 1 -label "Branch Length" -value1 2`;
$pos = `intFieldGrp -numberOfFields 3 -label "Initial Position" -value1 0 -value2 0 -value3 0`;
$direction = `intFieldGrp -numberOfFields 3 -label "Direction" -value1 0 -value2 0 -value3 1`;

button -label "Create Vines" -command "vineTest2";

$cameras = `textScrollList -numberOfRows 4 -allowMultiSelection false
    -append "persp"      -append "top"      -append "bot"
    -append "front"      -append "back"     -append "side"
    -append "left"
    -selectItem "persp"
    -showIndexedItem 1`;

button -label "ReverseSurfaceDir U" -command "reverseU";
button -label "ReverseSurfaceDir V" -command "reverseV";

button -label "Create Vines on Selected Surface" -command "vineTest";
button -label "Exit" -command ("deleteUI -window " + $window);

window -edit -widthHeight 400 325 $window;
showWindow $window;

```