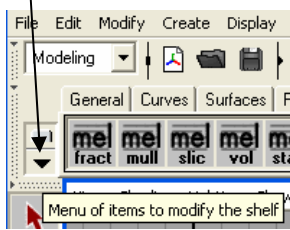
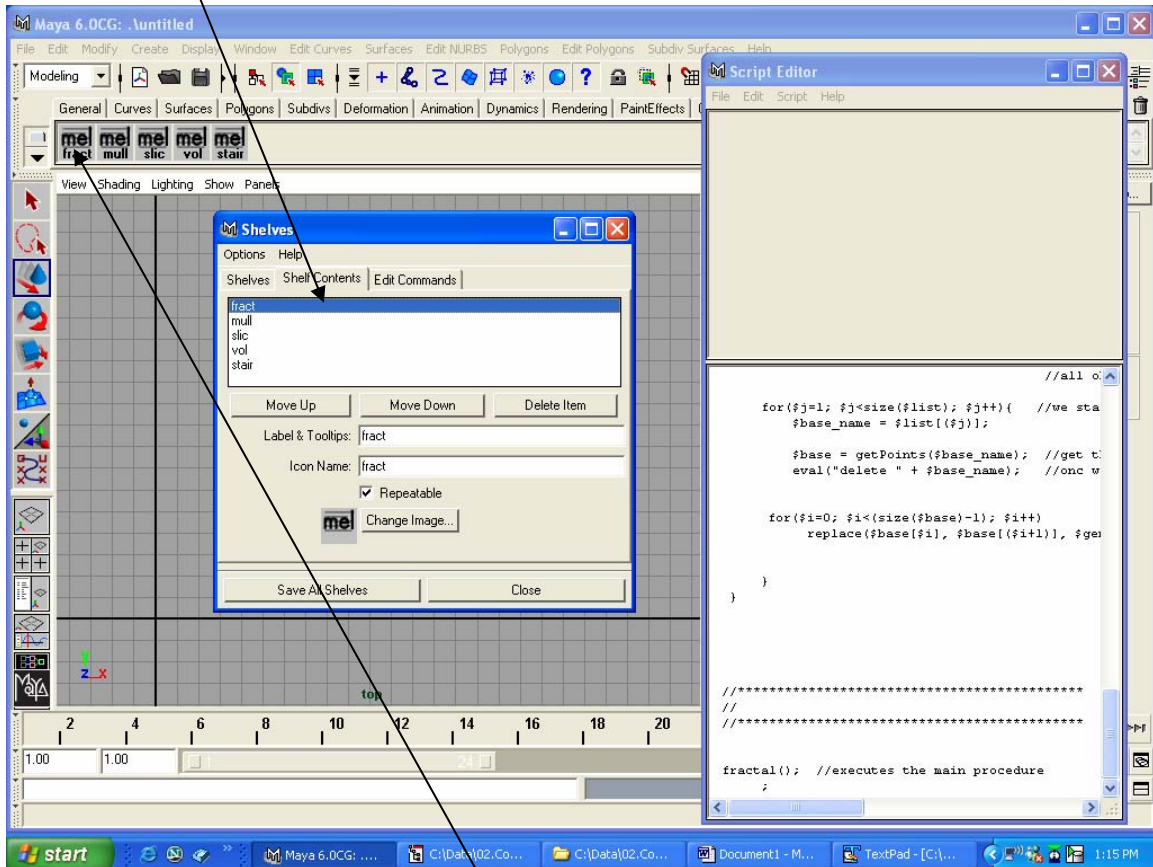


Fractal (L-system) manual

Copy the code (at the end of this document) and paste it MEL script editor. Select it. Click on the shelf editor



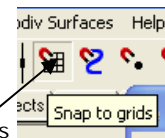
Then create a new shelf with the selected code:

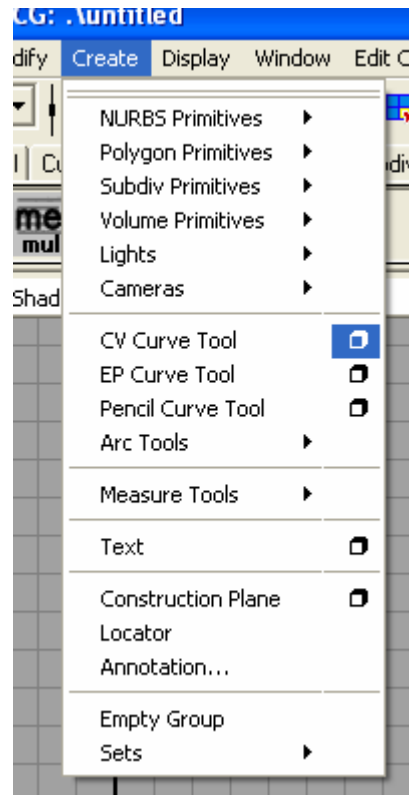


You should see a new icon in the custom tag folder. This is your fractal command: Everytime you click on it you should have the command executed.

How to use the fractal command:

Start with an empty space. Use Panels->Orthographic->Top and Snap to grids



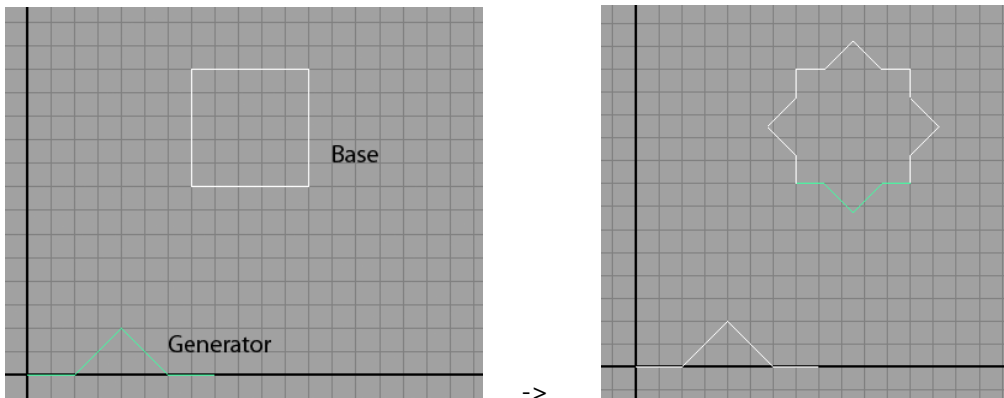


Select Create->CV Curve Tool->(Options) and then select degree 1

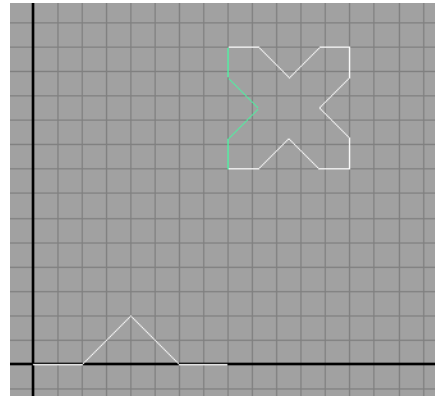
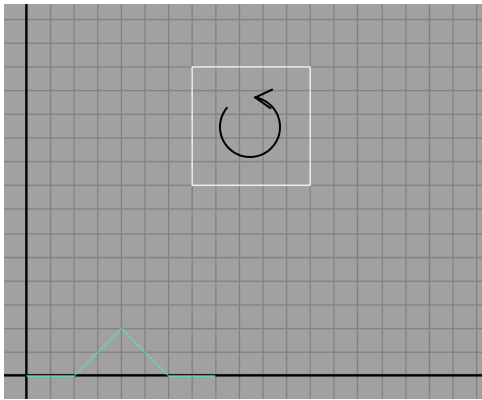
Experimentation:

First create a generator curve. There are two rules: 1) the generator must be called curve1 (default) and 2) the generator will start at from the origin 0,0,0 regardless of where your curve starts.

Then create a Base curve (direction is important)

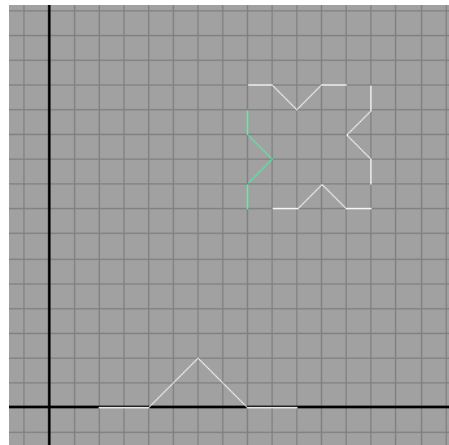
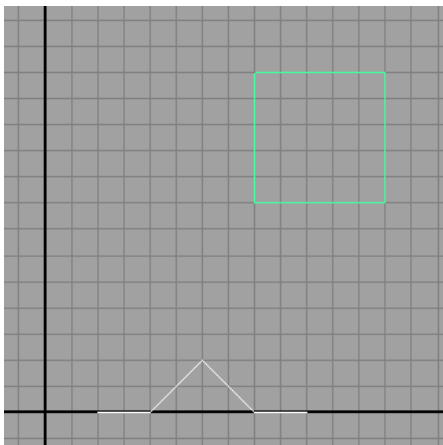


If the base was produced counter-clock wise then



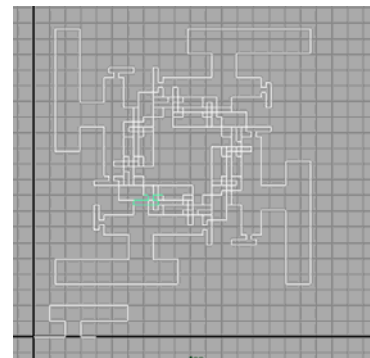
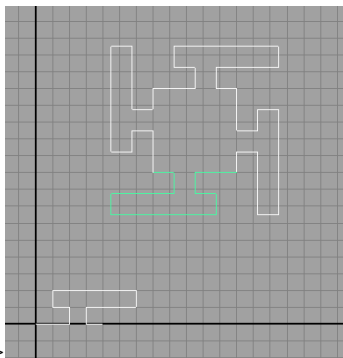
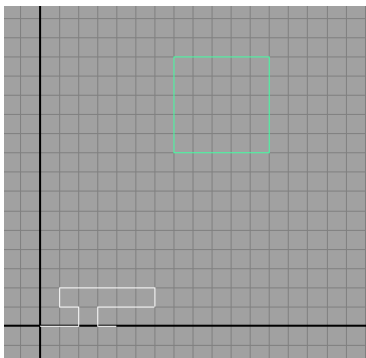
->

In this example the generator is offset from the origin



->

In this example the generator is longer than its own base



->

Code (to copy and paste)

```

/*****
// 2311 Algorithmic Architecture
//
// L-system 3D: Replacement operation
//
// Copyright Kostas Terzidis
//
// The L-system, as a generative system, consists of an initial state
// of a shape (the base) and one or more generators. The generator,
// from the practical point of view, is a production rule: replace each
// and every line segment of the base with the shape of the generator.
//
/*****

/*****
// the is procedure fits a curve ($shape[])
// between two points $start and $end
/*****
proc fit(vector $start, vector $end, vector $shape[]){

    // make a curve out of the array points in shape[]
    string $points = "";
    $point = $shape[0];

    for($i=0; $i<size($shape); $i++){
        $point = $shape[$i];
        $points += " -p " + $point;
    }

    eval("curve -d 1 " + $points); //a 1-d curve but can be changed to 2-d

    //scale to fit
    $temp_end = $end - $start; // move the points to the origin
    float $mag = mag($temp_end); // get the magnitude of their length
    float $amag = mag($point); // get the magnitude of the length of the base segment
    float $scale_factor = $mag/$amag; //get the scaling factor
    scale $scale_factor $scale_factor $scale_factor; // scale

    //rotate to fit
    $a = eval("angleBetween -euler -v1 1 0 0 -v2 " + $temp_end.x + " " + $temp_end.y + " " + $temp_end.z);
    if((abs((float)$a[0])<0.001) && (abs(180-(float)$a[1])<0.001) && (abs((float)$a[2])<0.001)){ $a[0]=0; $a[1]=0; $a[2]=180;}
    print($end.x + " - " + $start.x + " = " + $temp_end.x + "\n");
    print($end.y + " - " + $start.y + " = " + $temp_end.y + "\n");
    print($end.z + " - " + $start.z + " = " + $temp_end.z + "\n");
    print(" " + $a[0] + " " + $a[1] + " " + $a[2] + "\n");

    rotate $a[0] $a[1] $a[2]; // rotate in 3D space

    //move to fit
    move ($start.x) ($start.y) ($start.z); // move back to the original location

    refresh; //refresh the screen to see the replacement as it happens

}

```

```

//*****
// given a name of a curve will return back an array with the control points
//*****
proc vector[] getPoints(string $curve_name){

    //get the number of spans
    $numSpans = eval("getAttr " + $curve_name + ".spans");
    vector $points[]; //make a vector array to collect the points

    for($i=0; $i<($numSpans+1); $i++){
        $point = eval("pointPosition " + $curve_name + ".cv[" + $i + "]");//get the cvs
        float $x = $point[0];
        float $y = $point[1];
        float $z = $point[2];
        vector $v = <<$x, $y, $z>>;
        $points[$i] = $v; //store the values in points[]
    }

    return $points; //return the array
}

//*****
// Main procedure. Runs the fractal process
// note: it is global so can be sourced or called during a session
//*****
global proc fractal(){

    //make an array for the generator
    vector $generator[];

    //populate the array with the curve1 (generator) points
    $generator = getPoints("curve1");

    //make an array for the base
    vector $base[];

    string $list[];
    $list = `ls -transforms "curve*"; //returns an array with the names of
        //all objects in the scene that match the word "curve*"

    for($j=1; $j<size($list); $j++){ //we start at 1 because 0 is the generator
        $base_name = $list[$j];

        $base = getPoints($base_name); //get the points
        eval("delete " + $base_name); //once we have the points we erase the shape since we will replace it

        for($i=0; $i<(size($base)-1); $i++){
            fit($base[$i], $base[$i+1], $generator); //replace the current (i) and next (i+1) points with the generator's array
        }
    }

}

//*****
//
//*****

fractal(); //executes the main procedure
;

```