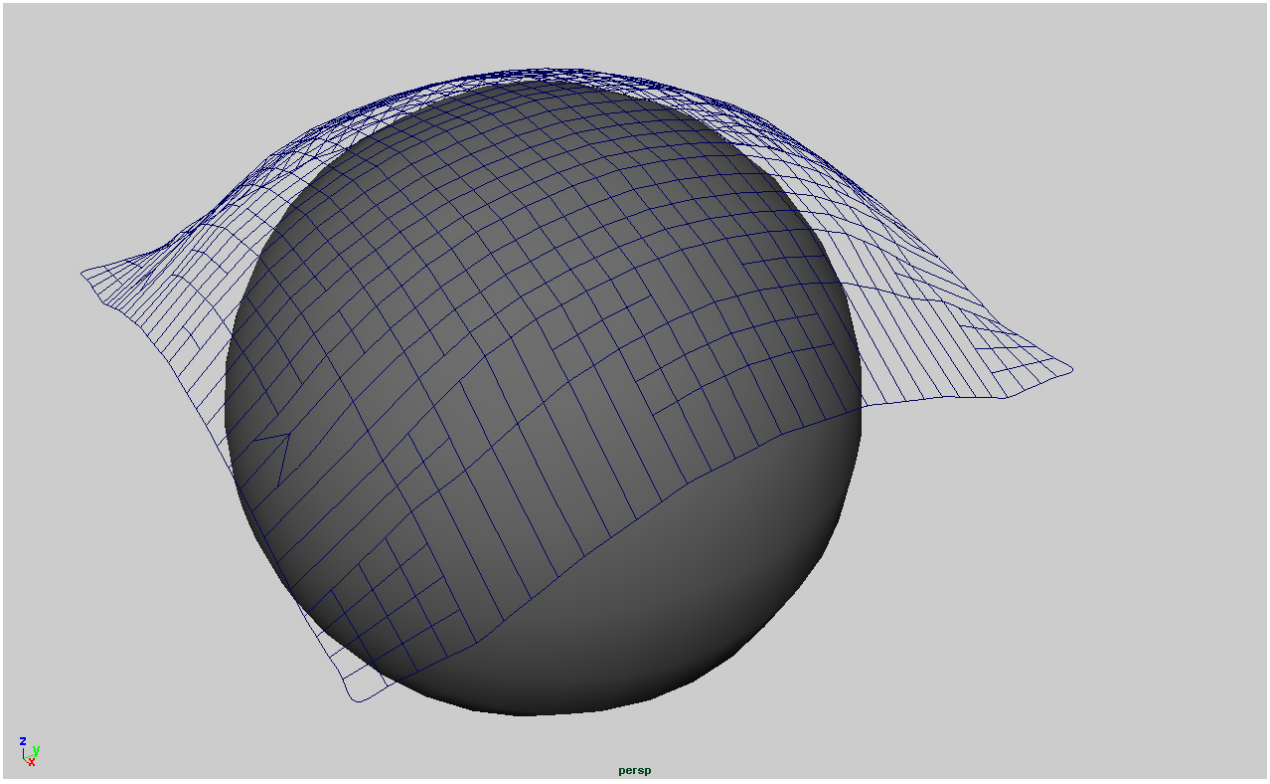


Help topics

MEL-command descriptions

degenr8

regenr8

regn8

repellor

attractor

GENR8 homepage

Installing GENR8

1. GENR8 is a plug-in for Alias|Wavefront's Maya. So if you do not have Maya installed on your computer, you need to install it before you can use GENR8. Also, it runs on Windows NT, so you may have to change hardware and/or operating system.
2. Copy the file GENR8.mll to c:\aw\maya3.0\bin\plugins (you may have to change the path if Maya is installed somewhere else on your harddrive).
3. Start Maya.
4. Go to Window->Settings/Preferences->Plug-in Manager. There you can check both load and auto-load (so that you do not have to do this every time you start Maya) for the GENR8.
5. Alternatively, you can go to Window->General Editors->Script Editor in the menu. In the white field, type: loadPlugin GENR8.

More Preparations

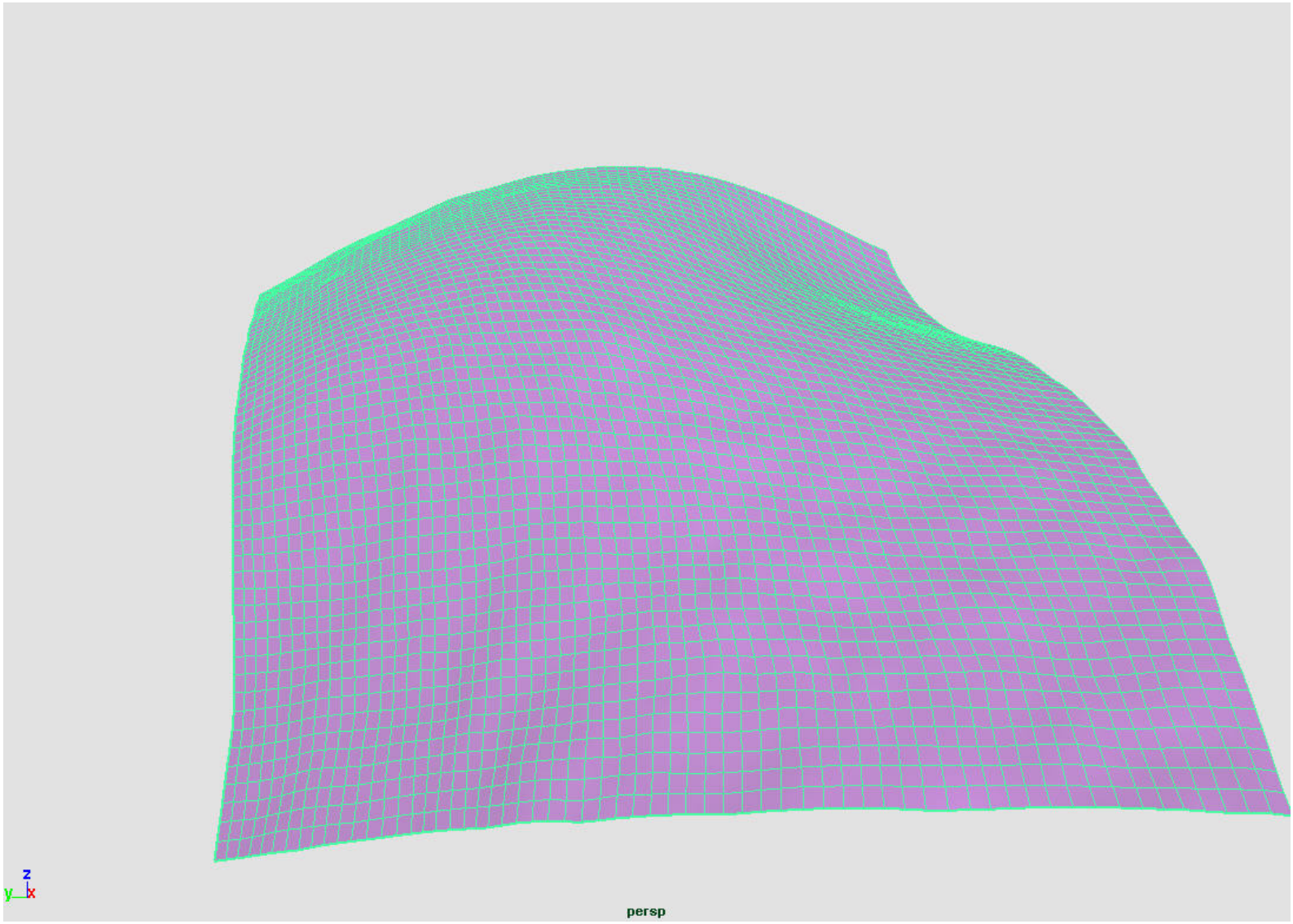
GENR8.mll contains several MEL-commands and if you are comfortable using MEL, you can use it in that way. However, there is also a GUI for some of the commands. If you do not want to be typing MEL-commands all the time, it is a good idea to create shelf-buttons. These buttons should suffice in most cases, but you might want to do some command that these buttons do not handle and then you must resort to the Command Line (or the Script Editor).

1. Type "genr8 -gui" on the Command Line (the white field in the lower left corner of the Maya-window.
2. Select the text with the mouse and draw it up to a free space on the shelf (next to the "Particle Tool" if you not previously edited the shelf).
3. Start the Shelf Editor, it can be found as the second item on the dropdown menu next to the shelf (the black arrow to the left of the "Curve with CVs")
4. Scroll down to the white field until you find the "genr8 -gui" and select it.
5. Change the overlay label to "genr8".
6. Repeat the above steps for "degenr8" (label "degenr8"), "repellor -g" (label "repellor") and "attractor -g" (label "attractor").

What is GENR8?

GENR8 is a design tool for generating surfaces. Surfaces are generated through a reactive growth process. The growth is governed by a grammar that holds the instructions for the growth process. The growth model is based on map L-systems. It is important to note that once an element of the surface has been grown, it is not fixed in space and may move as it is affected by the environment. The process is not entirely intuitive at first, since there are no fixed points during the growth.

The development of the surface is highly dependant on the environment, which can be defined by the user as well as a lot of parameters.



Map L-systems

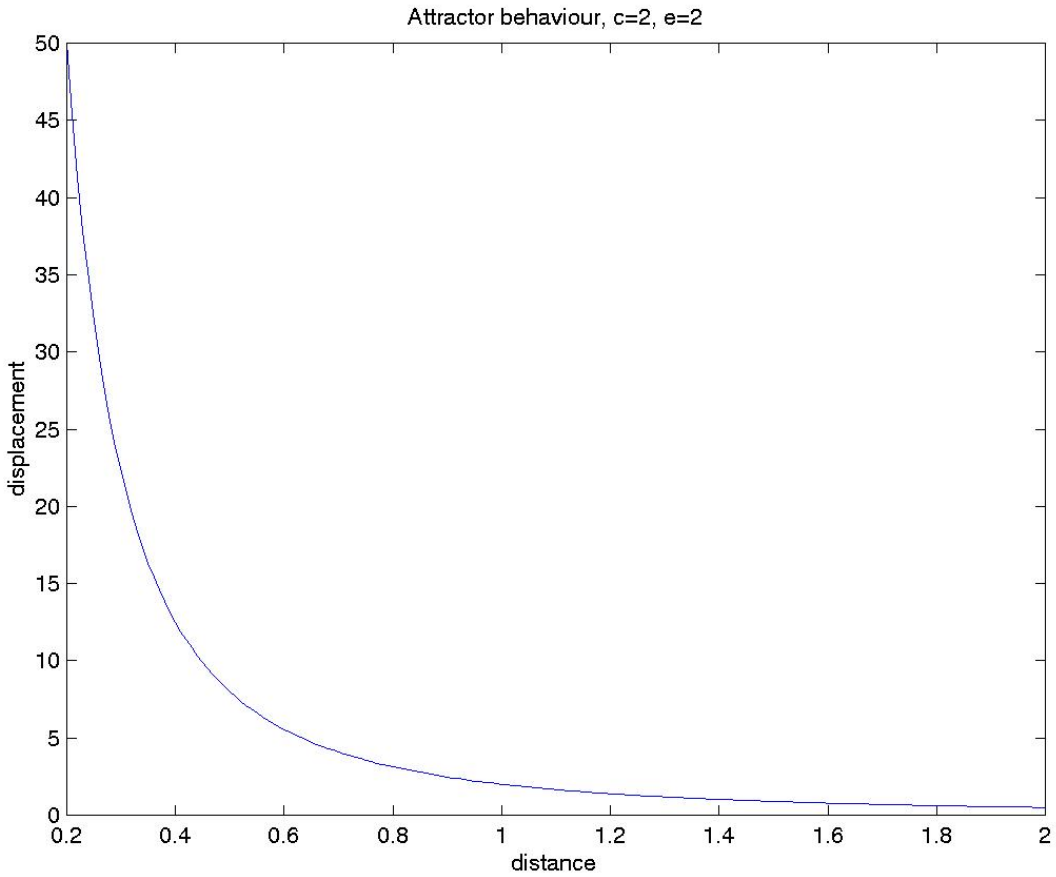
Map L-systems are version of the more familiar L-systems. Formally it can be seen as a method for rewriting strings (compare with the more famous Chomsky grammars). Map L-systems is a graphic interpretation of the strings as graphs and it is those graphs that are produced by GENR8. More about the growth model and map L-systems can be found in the thesis.

Setting up an Environment

The normal of the surface will initially be pointing in the z-direction, so it might be a good idea to change the settings in Maya so that its z-axis points upward. Do this from the menu Window->Setting/ Preferences->Preferences in the lefthand menu, choose Settings and there you will be able to change axis. Also it might be easier to view what is happening without the grid, turn it off on the Display-menu.

Attractors

Attractors act like magnets for the surface. As it grows it draws closer to the attractor. Attractors and repellors are represented as point charges and have to parameters, a constant and an exponent. The displacement from an attractor is *diplacement = c/d^e*, where *d* is the distance to the surface element, *c* is the constant and *e* is the exponent. This function has a singularity in *d*=0 and thus you get strange (and perhaps undesired) results if a surface element is close to the attractor.



Attractors are created in a separate layer, this makes it easy to toggle their visibility.

Repellors

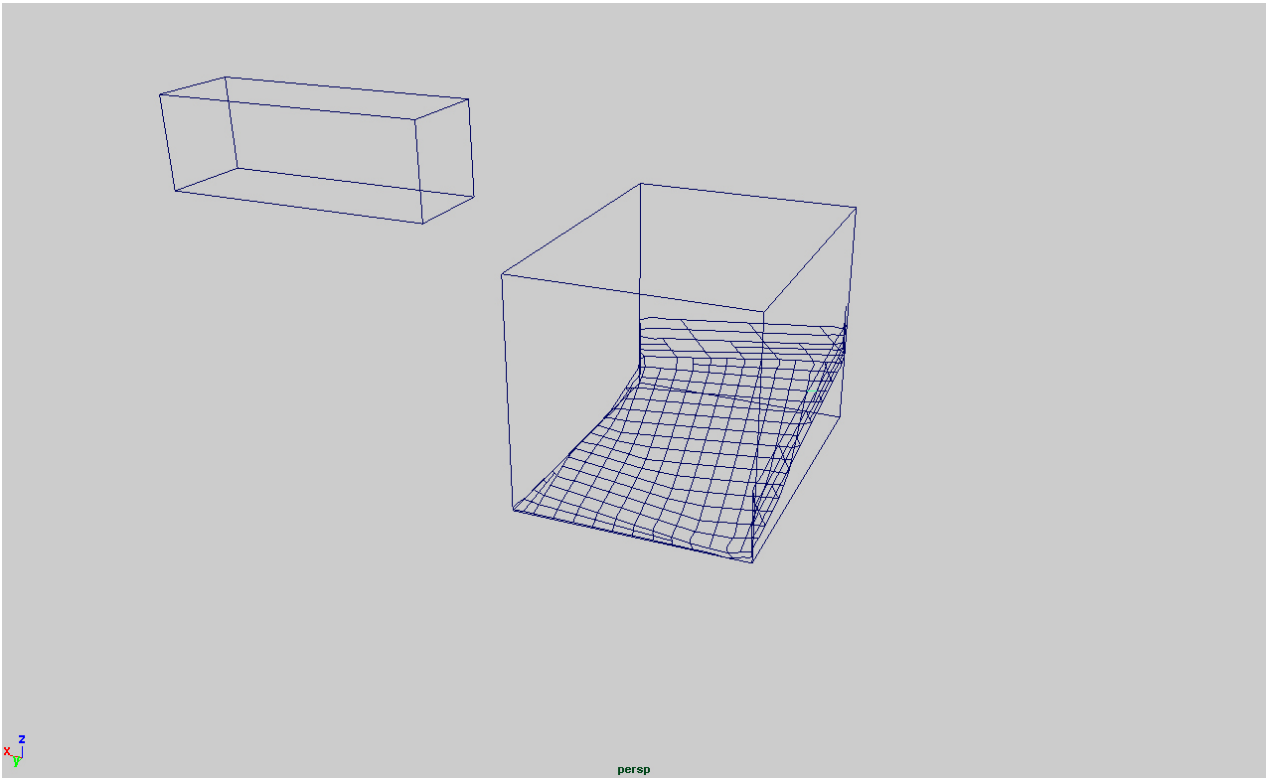
Repellors are similar to attractors, instead of drawing the surface closer, it is pushed away.

Gravity

Gravity is a force that acts uniformly throughout space in one direction (x, y, or z). You can set the gravity with the -g/gravity flag.

Boundaries

You can set boundaries for the growth by placing ordinary Maya-surfaces (polygons or nurbs) in the scene. Before starting genr8, you need to place the surfaces on the selection-list before running the command. There are three different wall behaviours defined, that you can choose between before the run.



Default

The growth is cut off so that the surface do not penetrate the boundary. During testing there have been cases where the, boundary does not stop the growth. The reason for this is most likely because Maya failed to detect the intersection

Cut Off

As the surface element hit the boundary, all movement is stopped. With the default behaviour, it slides along the boundary, but now it stops dead instead.

Soft Boudaries

This behaviour is only interesting when you have evolution. The surface can grow past the boundaries, but their fitness is penalized as they do so.

User Defined Seeds

The *seed* (starting surface) for GENR8 is by default a regular polygon. The user can control the length of the sides and the starting position of this polygon. However, GENR8 can also handle user defined seeds. You can draw a curve and select it before starting the run (you do not have to close the curve, GENR8 will do that for you). This curve will be tha starting point for the run, predicting the result is difficult though, since the assignment of types for the segments is randomized.

Since GENR8 uses Bsplines rather than nurbs, it is recommended that you use the "Curve with EPs" rather than the "Curve with CVs" tool. Otherwise, the output from GENR8 will have a different shape from the input. GENR8 assumes that the normal of your axiom has some component in the z-axis, if not things might go wrong.

Grammars

Surfaces are grown using a grammar. The grammar tells us how each surface element should be altered during each growth step.

There are three kinds of grammars in GENR8, predefined, user-defined and evolved.

It is recommended that you start by using the predefined grammars until you feel that you have understood the growth model and the environment. It is much easier to figure out the effects of the environment if you have a rough idea of what the outcome will be.

Predefined

Three-sided Tiles

This grammar produces a triangular surfaces that gets subdivided in to smaller and smaller triangles. The grammar is defined as:

```
   Edge0 + Edge1 + Edge2
   Edge0 -> Edge0 [ [ [ [ + Edge2 ] + + Edge1 ] - Edge1 ] - - Edge2 ]
Edge0
   Edge1 -> Edge1 [ [ [ [ + Edge0 ] + + Edge2 ] - Edge2 ] - - Edge0 ]
Edge1
   Edge2 -> Edge2 [ [ [ [ + Edge1 ] + + Edge0 ] - Edge0 ] - - Edge1 ]
Edge2
   Angle 60
   Sync
```

Four-sided Tiles

This grammar produces a square surface which is subdivided in to smaller squares. The grammar is:

```
   Edge0 + ~ Edge1 + ~ Edge0 + Edge1
   Edge0 -> Edge0 [ [ + Edge1 ] - Edge1 ] Edge0
   Edge1 -> Edge1 [ [ + Edge0 ] - Edge0 ] Edge1
   Angle 90
```

Koch curve

The Koch curve is an example of a fractal and it is a curve rather than a surface and it is not of much use

for generating surfaces. However, fractals are interesting in themselves and the ability to generate them comes "for free" with the growth model. The grammar is:

```
   Edge0 + Edge0 + Edge0
   Edge0 -> Edge0 - Edge0 + + Edge0 - Edge0
   Angle 60
```

Quadratic Koch curve

The quadratic Koch curve is a variant of the Koch curve.

```
   Edge0 + Edge0 + Edge0 + Edge0
   Edge0 -> Edge0 + Edge0 - Edge0 - Edge0 Edge0 + Edge0 + Edge0 -
Edge0
   Angle 90
```

User-defined

GENR8 can parse grammars that are defined by the user. The grammars used by GENR8 are expressed on Backus-Naur Form (BNF).The parser can handle all grammars that are generated from the following definition.

```
N = { RewriteRule, Predecessor, Successor, Modifier, Operator, Axiom,
Condition, LSystem, Segment }

T = { +, -, &, ^, \, /, ~, [, ], <, >, Edge, -> }

S = { <LSystem> }

P = {

<LSystem>       ::=     <Axiom> <RewriteRule> { <RewriteRule> }

<Axiom>         ::=     <Segment> [ "~" ] "+" <Segment> [ "~" ] "+"
<Segment> { [ "~" ] "+" <Segment> }

<RewriteRule>   ::=     <Predecessor> "->" <Successor> [ <Condition>]

<Predecessor>   ::=     <Segment> |
                        <Segment> "<" <Segment> |
                        <Segment> ">" <Segment> |
                        <Segment> "<" <Segment> ">" <Segment>
```

```
<Successor>     ::=     { <Modifier> } <Segment>

<Condition>     ::=     "If"

<Modifier>      ::=     { <Segment> } |
                        "+" <Modifier> "-" |
                        "-" <Modifier> "+" |
                        "&" <Modifier> "^" |
                        "^" <Modifier> "&" |
                        "\" <Modifier> "/" |
                        "/" <Modifier> "\" |
                        "~" <Modifer> |
                        "[" "[" "+" { <Operator> } <Segment> "]"
"-" { <Operator> } <Segment> "]"

<Operator>      ::=     "+"     |
                        "-"     |
                        "&"     |
                        "^"     |
                        "\"     |
                        "/"     |
                        "~"     |
```

Most of this information is unnecessary when writing your own grammar. Instructions for creating a grammar file can be found here.

Evolved

There are obviously an infinite set of possible grammars and in order to be able to search them somewhat more effectively, there is an evolutionary component that does this. When generating grammars the evolutionary algorithm uses a BNF that is very similar to the one described above. You can use one of the following.

Default

The default BNF has been specified to generate a lot of branches. This will in turn produce subdivisions, which in general is a desired property for a surface.

Reversible

This BNF produces grammars that can be inverted using regn8.

Symmetric

The symmetric BNF creates a grammar that produces a symmetric surface.

Probabilistic

This BNF produces grammars with probabilistic productions.

Grammar Files

By using the -lg grammar, you can specify your own grammar and have GENR8 do the drawing. Here is an example of a grammar file:

```
Edge0 + Edge1 + Edge2 + Edge3
// This is a comment, the parser ignores this text
Edge0 -> Edge3 [ + Edge0 ] Edge1
Edge1 -> Edge1
Edge2 -> Edge1 [ + Edge0 ] Edge1
Edge3 -> Edge2
// Another comment
Angle 90
Sync
BranchAngle 30
```

Before we discuss what each symbol means it is necessary to emphasize that the parser is very primitive, and you should be careful about getting the syntax correct. Each word should be separated by one space (' ') or tab (\t) and the lines should end with a return ('\n'). Unfortunately the comments must be on a separate line and you should not forget the white space after "//".

EdgeX	Segment type X.
<	Left-sensitive context.
>	Right-sensitive context.
+	Turn right by Angle.
-	Turn left by Angle.
&	Pitch down by Angle.
^	Pitch up by Angle.
\	Roll left by Angle.
/	Roll right by Angle.
~	Switch direction.
[	Push state on stack.
]	Pop state from stack.

The first line contains the axiom. The axiom must be a regular polygon and may only contain '+', '~' and "EdgeX". The example specifies a square, where each side has a different type. By default the direction of the first segment is (1,0,0) and unfortunately, this can not be changed.

Next comes the rewrite rules, they have the format: predecessor -> successor A predecessor can have four formats:

EdgeX

EdgeY < EdgeX

EdgeX > EdgeZ

EdgeY < EdgeX > EdgeZ

When GENR8 looks for successors and predecessors, any segment that comes directly before the segment is considered.

The '[' and ']' symbols are used for placing branches. A branch may be preceded by any number of operators, but it can only consist of one segment.

The angle is given in degrees (0-360) and it is preceded by the word "Angle", it specifies how much to turn, pitch or roll.

Branches can be joined either after each rewrite-rule has been applied (asyncronous), or when all the rules have been applied (syncronous). You may get different results for the same grammar depending on which method you choose. Default is asyncronous, so if you want syncronous growth, you simply type "Sync".

If you like, you can specify the -ba flag in the grammar instead of in the command line.

Probabilistic RewriteRule

GENR8 can handle probalistic rewriterules, for example

```
Edge0 + Edge1 + Edge0 + Edge1
Edge1 -> Edge0
      -> + Edge1 - - Edge1      Weight 2
Edge0 -> Edge1
      -> - Edge0 + + Edge0      Weight 2
Angle 45
```

The keyword Weight followed by an integer is used to control the relative weights of the possible successors when deciding which one to use. A Successor will get a Weight of 1 by default. Note that the parser can handle tabs as well as space when tokenizing. However, it is case-sensitive.

Rewrite systems

Genr8 has a feature for reading rewrite systems (or grammars) from a file. On this webpage are a few rewrite systems that I have found particularly interesting.

Handwritten rewrite systems

Genr8 can parse probabilistic rewrite systems. Here is an example of one that I think produces some interesting results. It differs very little from the predefined tiles 4 rewrite system.

```
Edge0 + ~ Edge1 + ~ Edge0 + Edge1
Edge0 -> Edge0 [ [ + + Edge1 ] - - Edge1 ] Edge0   Weight 4
      -> Edge0 - Edge0 + + Edge0 - Edge0
Edge1 -> Edge1 [ [ + + Edge0 ] - - Edge0 ] Edge1   Weight 4
      -> Edge1 \ Edge1 / / Edge1 \ Edge1
Angle 45
```

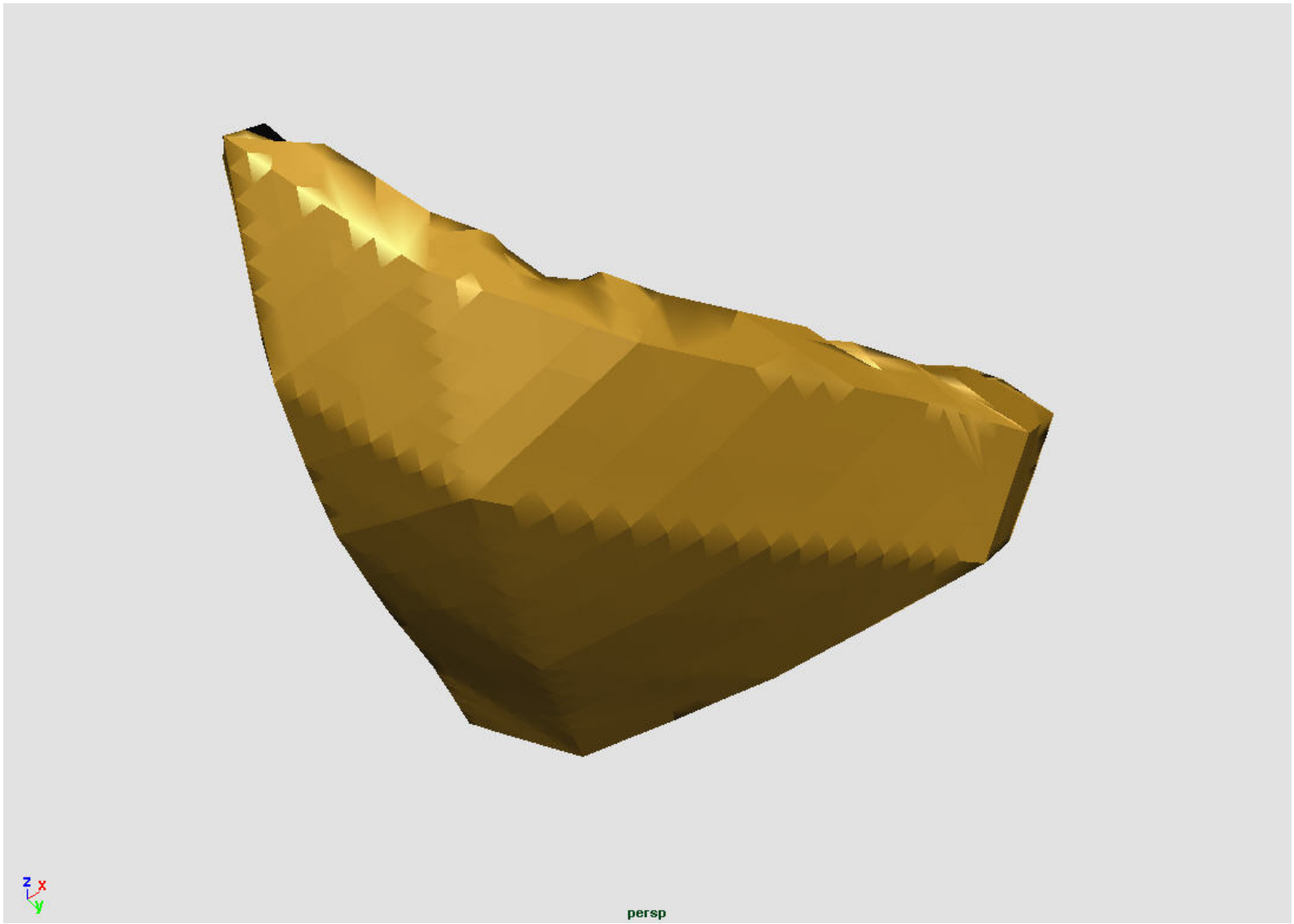
This rewrite system is time-dependent which means that different rules are applied depending on the age of the edge.

```
Edge0 + ~ Edge1 + ~ Edge0 + Edge1
Edge0_i -> Edge0_i+1 [ [ + Edge1_i ] - Edge1_i ] Edge0_i+1  If
i < 3
Edge0_i -> Edge0_i+1
Edge1_i -> Edge1_i+1 [ [ + Edge0_i ] - Edge0_i ] Edge1_i+1  If
i < 4
Edge1_i -> Edge1_i+1
Angle 90
```

Running GENR8

The growth of the surfaces are exponential. This means that each step takes approxiamtely twice as long as the previous. It is important to emphasize that this a very rapid growth and that you should be careful about using more than four steps.

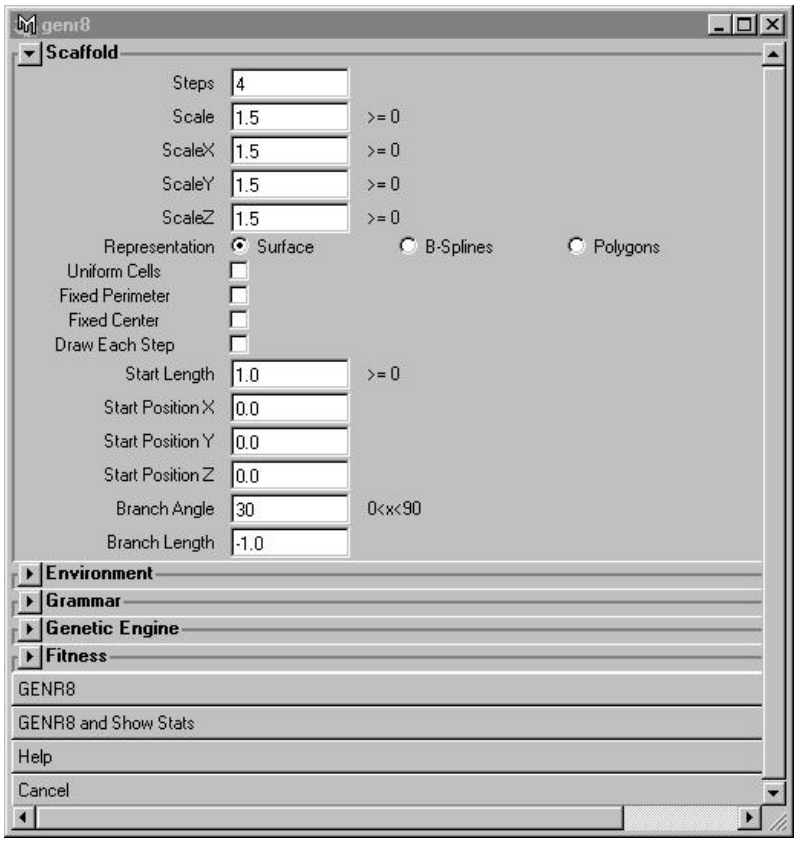
GENR8 is seamlessly integrated with Maya, so (except when during calcualtions) you can use any feature in Maya. If a run is taking to long time, it can be interrupted by pressing the ESC-key. Unfortunately, this only works for the evolutionary runs. If you did not follow my advice and started a run with the predefined grammars for 12 steps, then you must force Maya to quit with the Task Manager.



GUI

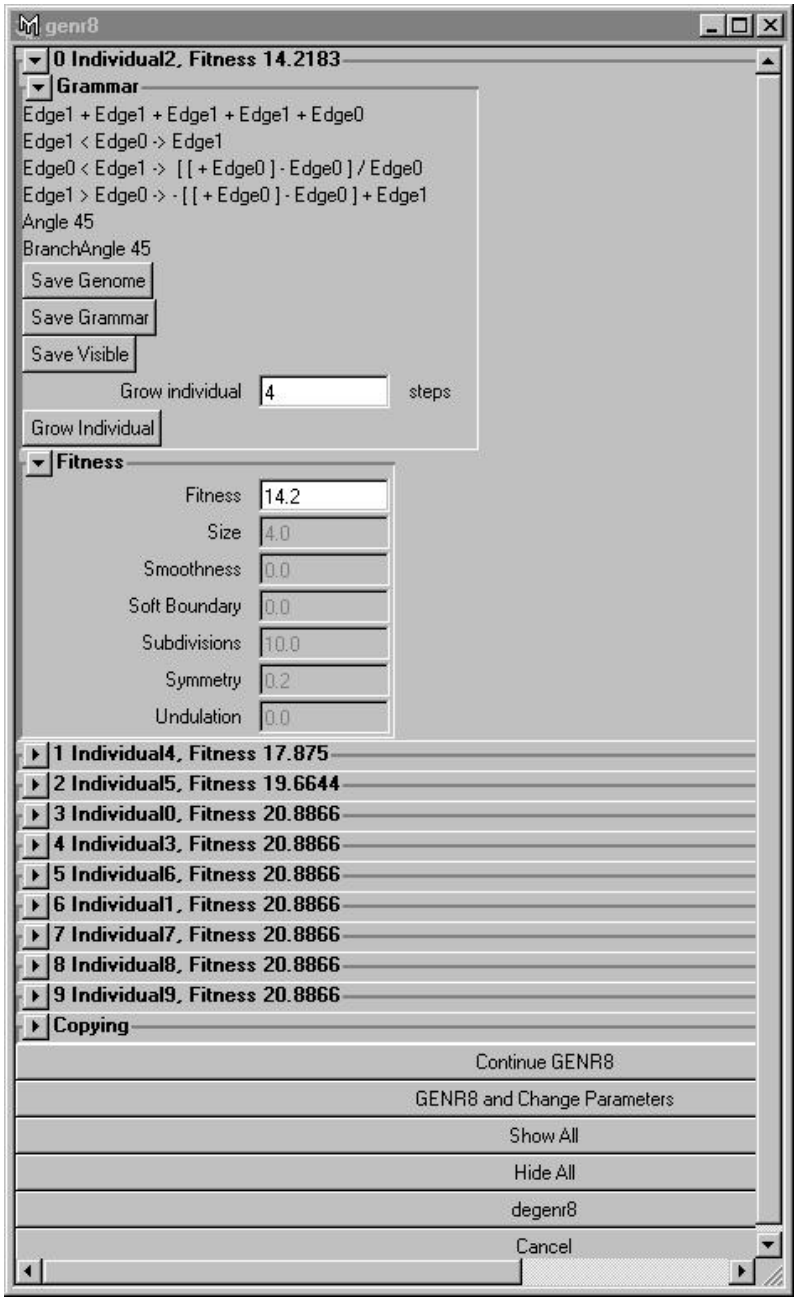
The GUI provides more user-friendly access to GENR8 for those who do not fancy the command-line alternative. When using the evolutionary part of GENR8, it is strongly advised to use the GUI since it facilitates the inspection of the population and provides interesting data.

When using the GUI, it is important to use the Cancel-button to close the windows. Otherwise, variable values may linger in the environment, resulting in unexpected behaviour. If you think that GENR8 behaves odd, start the GUI and check that all variables have proper values, correct them and close with the Cancel-button.



For a description of the parameters, read the help-file for the MEL-command. If you change a parameter value, it is important to hit the Tab-key, or click in another field with the mouse, afterwards for the change to take effect.

The output window pops up if you choose to "GENR8 and Show Stats". Each individual is drawn in a separate layer, and the visibility of that layer is toggled when the details of the individual is viewed. If you choose to "Continue GENR8", the evolution will be continued with the same parameters as before. If you choose to "GENR8 and Change Parameters", you will be able to change the parameters for the run. If you wish to continue with the same population as before, do not forget to check "Continue" under "Genetic Engine". The "Grow Individual" option can be used to study a surface in more detail. You will be able to see all the growth steps of that particular surface.


genr8
0 Individual2, Fitness 14.2183
Grammar
Edge1 + Edge1 + Edge1 + Edge1 + Edge0
Edge1 < Edge0 -> Edge1
Edge0 < Edge1 -> [[+ Edge0] - Edge0] / Edge0
Edge1 > Edge0 -> - [[+ Edge0] - Edge0] + Edge1
Angle 45
BranchAngle 45
Save Genome
Save Grammar
Save Visible
Grow individual 4 steps
Grow Individual
Fitness
Fitness 14.2
Size 4.0
Smoothness 0.0
Soft Boundary 0.0
Subdivisions 10.0
Symmetry 0.2
Undulation 0.0
1 Individual4, Fitness 17.875
2 Individual5, Fitness 19.6644
3 Individual0, Fitness 20.8866
4 Individual3, Fitness 20.8866
5 Individual6, Fitness 20.8866
6 Individual1, Fitness 20.8866
7 Individual7, Fitness 20.8866
8 Individual8, Fitness 20.8866
9 Individual9, Fitness 20.8866
Copying
Continue GENR8
GENR8 and Change Parameters
Show All
Hide All
degenr8
Cancel

Evolution

GENR8 features an evolutionary algorithm that searches a universe of grammars. The purpose of this is to create new and interesting surfaces. The user has high-level control of the evolution by setting the fitness criterion. The specific method used by GENR8 is called grammatical evolution(GE).

Troubleshooting

So GENR8 does not behave the way you want? Feeling frustrated about stupid computers and software? Beginning to question the sanity of the guy who developed GENR8? Getting frustrated by all the stupid messages? What is the meaning of Life? Why am I doing this? Is he ever going to stop writing stupid things?

Well, here is a FAQ to help you run GENR8 and produce the output you desire and answer some of the above questions.

How do I choose the parameter values?

As you see, there is a huge number of parameters and unfortunately I can not give much advice on how to choose a good set of values. The default values are really quite arbitrary and you might find much more interesting behaviours using a different set of parameters. I have only tried most of the features one at a time, so combining two may give interesting and unexpected results. The best advice is to experiment, the default values give sensible results but another set of values might suit your purpose

much better. Sorry, not much help on this one....

I get some crazy results as the surface comes close to an attractor!

Because of the singularity in the attractor equation, you get larger displacement as you get closer. Thus, you should not start too close to an attractor or place a boundary between the surface and the attractor. Repellors usually work better since they push the surface away, so you do not have to worry about things getting too close to the repellor.

Some of the lines are missing?

As far as I know there are two situations where GENR8 can fail to draw the lines that it is supposed to draw. One situation arises when the vertices are too close to each other, then Maya will not be able to draw the lines. However, the lines still exist in GENR8's internal model of the scaffold. This usually occurs beacause of environmental factors such as attractors and random noise. To combat this problem you should make sure that a lot of vertices do not get piled up in the same place.

The other case when lines will not be drawn as they are supposed to also occurs due to environmental factors. When two branches are to be connected across a region GENR8 makes sure that they have roughly the same direction before connecting them. If the external forces has distorted the scaffold too much, this test will fail and the region will not be subdivided (later it will be possible to adjust this tolerance).

This computaion takes too much time, can I go for a coffe break?

The growth is exponential and this means that each grwth step takes about twice as long time as the previous step. Thus the computation time increases VERY fast. Unless you are a trained professional and know what you are doing, you should never grow more than five steps!

If you are using the evolutionary algorithm, you can interrupt the run by hitting the Esc-key. Unfortunately, it is very hard to predict how long time the evolutionary run is going to take. In some cases you might get a surface with lots of subdivions that takes ages to compute and in the next run you might end up with simple boring loops. To some extent, this can be controlled by the fitness criterion.

GENR8 has an internal model of the environment, that speeds up computation by orders of magnitude. However, some things must be done through Maya (in particular drawing and detecting intersections) and this takes a lot of time. You can not do much about the drawing part (just watching one individual does not help) but if you use less boundaries, things might go a little faster.

More on setting up the environment.

Note that the environment can have great impact on a design and even more so on an entire run. Thus it is hard to say anything except for an empty environment. Also, the other parameters (especially the scaling) can produce a wide range of results.

If you define your own environment with boundaries and layers etc, do not give the Maya-objects names that start with "genr8", this might confuse GENR8 severly. It is a good idea to create a new layer and draw your boundary elements in that layer. In that way it is easy to toggle the visibility of the layer if you want to view the generated surface without the bounding elements obscuring the view. New layers are created with the button left of the Default layer.

Evolution is stupid. I can't believe that people think that humans were evolved from bacteria, when it is impossible to evolve a decent surface! How can I make it produce interesting results?

The evolutionary process tries to minimize the fitness of the individuals, so the fitness function is very important.

By default the subdivisions criteria has a high weight since that is usually a desired feature. Otherwise the system just creates surfaces that only consists of a perimeter. Since the size criteria depends on the start length, the scale and the number of steps, it is usually a good idea to decrease the size weight.

If you see something that looks intersesting but GENR8 is of a differing opinion (giving it a high fitness) you can coerce GENR8 into doing as you wish. You can edit the fitness value of individual designs. This will make sure that the surface is copied directly into the next generation and it's genetic material will be more prevalent in the next generation. The same effect can be achieved by copying the individual so that it occupies a larger portion of the population.

All the surfaces look the same and nothing happens as I run more generations!

The best solution is to increase the population size. The downside of this is that each generation takes longer time to evaluate. So if you are working on a slow computer, then there are a few other things to do. The easisest thing is to increase the mutation rate or the number of crossover points, creating more genetic variation. Another thing to do is to load a saved genome into the population to get some more variation.

What is a good population size?

The answer depends on what computer you are using. During development I was using a 350 MHz Pentium II with 395 MBytes of RAM. I could run a population with 50 individuals without getting bored (I usually only run 2-4 generations at a time and inspect the result in between). If the population size is

less than 20, it tends to converge quite fast. As a rule of thumb it is better to have a large population and only run it for a few generations than the other way around (many generations, few individuals). This allows for greater diversity and varialtion in the population.

The surfaces produced by the symmetric BNF aren't symmetric.

The most likely culprit is the environment. You may retort "But I have an empty environment, how could that be?". The answer lies in the *branchAngle* parameter. If the angle is large, pairs of branches that match badly might merge and thus the symmetry is lost since the branches were supposed to merge in another way. If you ask why this can happen, the answer is that the algorithm that performs the merging does the matching in a unpredictable (albeit deterministic) order.

I can't shade the surface that GENR8 produced.

For some obscure reason, there is no shading node associated with the Mesh created by GENR8. In order to shade it, you must do the following:

1. Open the Hypershade-window under Window->Hypershade.
2. Select the surface.
3. Right click on one of the shaders (eg lambert1).
4. Choose Assign initialShadingGroup To Selection.

The stupid plug-in does not work! Nothing happens as I type the command.

Ok, here's what to do (if meassure *i* doesn't solve the problem resort to *i+1*):

1. Unload the plug-in and then reload it (using the plug-in manager).
2. Restart Maya.
3. Re-install GENR8.
4. Re-start your computer.
5. Re-install Maya.

How do I get rid of the stupid messages? They really annoy me! By the way I do not like your sense of humour at all, it sucks as much as this !%&#$*:~ application!

Well there is not much to do about the stupid messages, they are part of the tool. My sense of humour is even harder to deal with, it pervades the whole system...

GENR8 Examples

In this section we present a few commands that can be used to create interesting surfaces. Hopefully, this can be of some help when you are learning how to use the system (I am afraid that it is more difficult to use than I intended to). The easiest way to use these commands is to copy the text to the script editor and hit Enter (you must use the Enter key at the numerical keyboard and not the ordinary one).

genr8 -kc
Comment: The Koch-curve (also known as the snow flake), fractals are just subset of what is possible with genr8.

genr8 -fp -sl 8 -n 5 -ba 70 -t 4 -r 0 0 3 -rn 0.1 -s 1.1
Comment: The surface behaves more like a membrane since the perimeter vertices are not allowed to move.

polyCube -w 3 -h 4 -d 3;
genr8 -t 3 -rn 0.1 -n 4;
Comment: Remember that you must select teh cube before the run. If you run the commands simultaneously, the cube is automatically selected.

polyCube -w 3 -h 4 -d 3;
genr8 -t 3 -n 5 -a 0 0 4 -g z -0.5;
Comment: Do not forget to select the cube! The center of the surface is pulled by the attractor until it hits the ceiling of the cube. The other parts are dragged down by gravity.

polySphere -r 5;
genr8 -t 4 -n 5 -ba 90 -g z -1 -sp 0 0 7 -sl 3 -s 1.3;
Comment: This an example where it works fine with a convex wall.

genr8 -t 4 -n 6 -ds -s 1.3 -r 0 0 -3 -r 0 4 0 -ba 90 -r 1 -3 -4 -r 2 2 5 -r -3 -2 1;
Comment: The final shape is far from the original square.

genr8 -t 3 -n 5 -g z -1 -r 0 0 -3;
Comment: The gravity is balanced by the repellor, producing a nice arc.

polyCube -w 3 -h 4 -d 3;
genr8 -t 4 -n 5 -r 3 3 3 -co;
Comment: Note how the parts of the surface that hit the walls of the cube stop moving because of the cut-off wall behaviour.

Name

genr8

Synopsis

genr8 [flags]

ReturnValue

Stupid message from the developer.

Description

Generate a scaffold using a reactive growth model in a user-defined environment

Flags

-n/steps unsigned	The number of growth steps. Note, since the growth is exponential, each step will take approximately twice as long as the previous. *Default:* 4
-s/scale double	How much the design will be scaled on each growth step. *Default:* 1.5.
-sx/scaleX double	How much the design will be scaled in the X-direction. *Default:* same as scale.
-sy/scaleY double	How much the design will be scaled in the Y-direction. *Default:* same as scale.
-sz/scaleZ double	How much the design will be scaled in the Z-direction. *Default:* same as scale.

-p/polygons	Draw polygons instead of nurbs. *Default:* off.
-fp/fixedPerimeter	Keep the perimeter fixed. If this is on, the perimeter will be fixed and the surface will behave more like a membrane.
-fc/fixedCenter	Keep the center fixed to the starting position.
-ds/drawEachStep	If this flag is set GENR8 will draw each growth step, if not it only draws the scaffold when it has finished growing it. If the populationSize is 1 and this flag is used, each step will be drawn in its own layer.
-sp/startPosition double double double	The coordinates for the centre at the start. *Default:* origin (0, 0, 0).
-sl/startLength double	Set the side length of the seed. The seed is always a regular polygon. *Default:* 1.
-ba/branchAngle unsigned	Set the angle (in degrees, ie between 0 and 90) for the cone that is used when trying to match branches. *Default:* 30.
-bl/branchLength double	Set a limit to the maximum distance for when branches can be connected. This is not an absolute value, but a multiple of the *characteristic length* of the scaffold. The characteristic length is defined as the startLength, multiplied by the scale for each step. A value of -1 means infinite range. *Default:* -1.
-g/gravity x|y|z double	Apply a gravitational force in the desired direction. The second argument gives the magnitude, where a negative value means in the negative direction. *Default:* no gravity.
-a/attractor double double double	Place an attractor at the given coordinates. This argument can be used multiple times.

-r/repellor double double double	Place a repellor at the given coordinates. This argument can be used multiple times.
-aw/activeWalls	If the walls should push back vertices growing towards it or simply cut them off if they try to move outside. If this is off and you have a small bounding box there is a risk that a lot of vertices will pile up at the walls.
-rn/randomNoise double	Add a noise component to the system. The growth will be affected by a random perturbation. *Default:* 0.
-rr/randomRewrite unsigned	This is similar to the -rn flag. Its value should be between 0 and 100 and it is the probability that a segment will get another type than the rewriterule indicated. *Default:* 0.
-wc/wallConstant double	Set c in the attractorequation, c/d^e. Only used when aw is on. *Default:* 50.
-we/WallExponent unsigned	Set e in the attractorequation, c/d^e. Only used when aw is on. *Default:* 2.
-co/cutOff	Determines the behaviour when a vertex hits a wall. If this is on, it stops dead at the wall, if off, it slides along the wall and continues to grow.
-rp/repellingPoints double unsigned	Make the vertices repelling, this means that there will be an internal feedback throughout the system. The vertices will move away from each other, making the cellsizes more uniform. This function has quadratic complexity, while the other functions are linear, thus this function could slow down the program significantly.
-rns/randomSeed unsigned	Set the seed of the random number generator to this value (good if you want to get the same run again).

-t/tiles 3|4	Use a predefined grammar that generates square tiles. If you use the four sided tiles, GENR8 will try to create a nurbs-surface. Since the two first steps are of degree one and the latter of degree three, the transition between these steps can look odd.
-kc/kochCurve	Use a predefined grammar to draw a Koch-curve (snowflake fractal).
-qk/quadraticKoch	Use a predefined grammar to draw a variant of the Koch curve.
-rv/reversible	If this flag is set, the grammar will be generated from a more restricted BNF. Then it will always be possible to reverse the grammar produced with regn8. This flag should also be used when you are loading a genome that was produced by regn8.
-syb/symmetricBNF	If this flag is set the BNF will be constrained so that it only produces grammars that generate symmetric scaffolds. The symmetric BNFs are a subset of the reversible BNFs.
-pr/probabilistic	If this flag is used, genr8 will produce grammars that have probabilistic rewrite rules.
-frr/fullRandomRewrite unsigned	If this flag is set the probabalistic rules will behave in a different way. If the flag is not used, every segment gets rewritten by the same rule (during each step), if it is set each segment will get a separate rule, which most likely results in a more random structure.
-mbd/maxBNFDepth unsigned	This limits the depth of the expansion of the grammar when interpreting the evolutionary algorithm interprets the genome. *Default:* 5.

-met/maxEdgeTypes unsigned	This enables you to set a maximum for the number of edge types that can be produced by the evolutionary algorithm.
-lg/loadGrammar string	Use a grammar that has been specified in the file in string. If you do not specify the entire path to the file, GENR8 looks for it in the same directory that GENR8 is stored in. For more info on how you should write your grammar file, please read this section.
-sg/saveGrammar string	Save the grammars used (all of them) in the file specified by string. GENR8 will automatically place it in the same directory as it is installed in if you do not specify the entire path.
-ps/populationSize unsigned	If you do not use the predefined tiles or fractals, GENR8 will do an evolutionary search for good grammars. This specifies the population size, and it must be at least 1 (even if you are using tiles). Each inidivual is drawn in its own layer. *Default:* 1.
-ge/generations unsigned	How many generations should be evolved. Each scaffold will grow steps times before being evaluated (and it may take a while if steps, populationSize or generations is high). *Default:* 0.
-mr/mutationRate unsigned	Set the mutation rate for the genetic engine. This is the probability that a gene is mutated and the value should be between 0 and 100. *Default:* 2.
-ts/tournamentSize unsigned	Set the size of the tournament used by the EA. A high value means more elitism. *Default:* 4.

-e/elites unsigned	Set the number of elites. Elites are automatically copied into the next generation. In order to maintain diversity, this value should be kept low. *Default:* 1.
-gl/genomeLength unsigned	Set the genomelength to this value, all individuals have the same length. Unless the genome is very short, the value does not really make a big difference. *Default:* 50.
-e/elites unsigned	Set the number of elites. Elites are automatically copied into the next generation. In order to maintain diversity, this value should be kept low. *Default:* 1.
-vi/viewIndividual unsigned	Just view the individual with rank unsigned instead of the entire population.
-ci/copyIndividual unsigned unsigned	The first argument is the rank of the individual to be copied and the second is the number of copies. The new copies be inserted at the end of the population. Note that you copy the individual with the specified rank and not the ID!
-sdp/seedPopulation unsigned unsigned	This is similar to the -ci flag. But instead of making exact copies, the copies will be mutated.
-c/continue	Continue with the same genetic material as in your last run. The genome is read from the file genome.txt. In this way, you can change your settings and parameters after running a few generations.
-lge/loadGenome string	Load genome from the file string.
-sge/saveGenome string	Save the genome in the file string. If you do not specify the whole path, it will be saved in the same directory as GENR8. By default, the genome is saved in the file genome.txt.

-sf/setFitness unsigned double	Set the fitness of individual unsigned to double. This is a good way to give the EA a kick in the direction you want it to go. This flag can be used multiple times.
-si/size double double	Desired size in the x- and y-direction. *Default:* 10 10.
-siw/sizeWeight double	Set the weight for the size fitness criterion. *Default:* 1.
-sm/smoothness double	This parameter is used when determining the fitness for a design. The value is how much you wish the z-values of the vertices to vary. 0 means that the surface is completly flat and a high value will give a rugged surface. *Default:* 0.
-smw/smoothnessWeight double	Set the weight for the smoothness fitness criterion. *Default:* 1.
-sbw/softBoundaryWeight double	If this weight is greater than 0, the boundaries will not stop growth, instead there will be a fitness penalty proportional to the number of vertices that are outside the boundaries. *Default:* 0.
-su/subdivisions double	This parameter should be between 0 and 1 and is the desired amount of subdivisions on the surface. A value of 1 means that there are no subdivisions and the lower the value is, the more subdivisions should be generated. *Default:* 0.
-suw/subdivisionsWeight double	Set the weight for the subdivisions fitness criterion. *Default:* 1.
-sy/symmetry double	Desired level of symmetry. The value should be between 0 and 1 where 0.5 represents a symmetrical design and the extreme values something totally unsymmetric. *Default:* 0.5.

-syw/symmetryWeight double	Set the weight for the symmetry fitness criterion. *Default:* 1.
-un/undulation double	The undulation fitness criteria specifies how much variation in the z-value we desire on a global level (smoothness is a local criterion). A value of 0 strives for a flat surface and positive values allows for more varition in z-values. *Default:* 0
-unw/undulationWeight double	Set the weight for the undulation fitness criterion. *Default:* 1.
-dy/displayStats	Use this flag to get the output window.

Examples

Related Commands

degenr8, regenr8, regn8, repellor, repellor

Name

degenr8

Synopsis

degenr8 [flags]

ReturnValue

Stupid message from the developer.

Description

Remove things from the Maya scene that were created by genr8. You should use this command for removing stuff created by genr8, attractor and repellor rather than deleting them by hand, since these commands create MEL-variables that can cause problems later.

Flags

-a/attractors	Just delete the attractors and the attractor layer.
-r/repellors	Just delete the repellors and the repellor layer.
-c/curves	Just delete the curves.
-l/layers	Just delete the layers. As a result all curves will migrate to the Default-layer.
-ug/unloadgenr8	Unload the GENR8 plugin after deleting.

Examples

Related Commands

degenr8, regenr8, regn8, repellor, repellor

Name

regenr8

Synopsis

regenr8 [flags]

ReturnValue

Stupid message from the developer.

Description

This command will make it easier to view the different individuals. If you want to, you can turn the visibility on and off in the menus, but this command provides an easier way. This command is built in to the GUI for genr8.

Flags

-a/allIndividuals boolean	Turn the visibility on or off for all individuals.
-s/showIndividual unsigned boolean	Turn the visibility on or off for individual unsigned.

Examples

Related Commands

degenr8, regenr8, regn8, repellor, repellor

Name

regn8

Synopsis

regn8 [flags]

ReturnValue

Stupid message from the developer.

Description

This command is part of the Interruption Interaction and Resumption (IIR) features of GENR8. It can be used to map a grammar to a genotype. This genotype can later be modified and used in a genr8 run.

It is important to note that the grammar that is to be inverted must be generated by a reversible BNF. If this is not the case, the produced genome will not produce the original grammar. It is important to use the -rv flag when using this genome with genr8.

Flags

-g/grammarFile string	The file that specifies the grammar that should be mapped to a genome. If the entire path is not specified, regn8 will look in the directory where it is stored.
-s/saveFile string	The file where the generated genome will be stored. If the entire path is not specified, regn8 will look in the directory where it is stored. NB Do not use the name genome.txt, since genr8 is using that name.

Examples

Related Commands

degenr8, regenr8, attractor, genr8, repellor

Name

repellor

Synopsis

repellor [flags]

ReturnValue

Stupid message from the developer.

Description

Use this command to place repellors (you may still place repellors using the flags for genr8). When you have placed the repellor, you may move it around just like an ordinary Maya-cylinder. If you use this command it is a good idea to remove the repellors with degenr8, since it cleans up the MEL-variables that are generated. If that is not done, things may behave strangely later. If you select a repellor when this command is executed, you will be able to edit the values of the repellor.

The repellors that are being drawn are ordinary polyCylinders and as long as you do not change the name of the repellor, feel free to alter the position and the size.

If you select a repellor and use the repellor command, you will be able to edit the parameters. Whenever you change a value, you should hit enter while the cursor still is in that field. If you edit a repellor, you will get the GUI, regardless of the presence of the -g flag.

Flags

-g/graphical	Turn on a gui to place the repellor.
-p/position double double double	Position of the repellor. Default is origin.
-c/constant double	Set the nominator in the repellor equation. Default is 10.

-e/exponent	The exponent for the repellor equation. Default is 2.

Examples

Related Commands

degenr8, regenr8, regn8, genr8, attractor

Name

attractor

Synopsis

attractor [flags]

ReturnValue

Stupid message from the developer.

Description

Use this command to place attractors (you may still place attractors using the flags for genr8). When you have placed the attractor, you may move it around just like an ordinary Maya-cylinder. If you use this command it is a good idea to remove the attractors with degenr8, since it cleans up the MEL-variables that are generated. If that is not done, things may behave strangely later. If you select a attractor when this command is executed, you will be able to edit the values of the attractor.

The attractors that are being drawn are ordinary polyCylinders and as long as you do not change the name of the attractor, feel free to alter the position and the size.

If you select a attractor and use the attractor command, you will be able to edit the parameters. Whenever you change a value, you should hit enter while the cursor still is in that field. If you edit a attractor, you will get the GUI, regardless of the presence of the -g flag.

Flags

-g/graphical	Turn on a gui to place the attractor.
-p/position double double double	Position of the attractor. Default is origin.
-c/constant double	Set the nominator in the attractor equation. Default is 10.

-e/exponent	The exponent for the attractor equation. Default is 2.

Examples

Related Commands

degenr8, regenr8, regn8, genr8, repellor