MEL Scripting
Variables MEL

float $param;
int $counter;
string $name;
vector $position;
Example

- // Declare the variables...
- float $foo;
- float $bar;
- int $baz;
- // Now assign values...
- $test = 5.4;
- // value of $test is 5.4
- $bar = $test + 2;
- // value of $bar is 7.4
- $baz = $bar;
- // the value of $baz is 7
- // MEL converted the value to an integer.
Printing

• print(500);
• print("Hello world!\n");
• print("The time is now " + $time);
Random Number

• rand(1000);
  • // Result: 526.75028 //

• rand(100,200);
  • // Result: 183.129179 //
Arrays

• To declare an array variable, use:
  • the keyword of the type which this array will hold,
  • then the variable name,
  • add square brackets ([]) to the end of the variable name.

• int $ari[];

• Set Initial size
  • float $arf[4];
  • string $temp[3];
• $arf[2] = 45.646;
• $temp[50] = "Glonk!";

• $rip = {1, 2, 3, 4};
• string $hats = {"blue", "red", "black"};
• string $shoes[3] = {"black", "brown", "blue suede"};
Size of array

- int $scores[];
- // Declared as a zero element array.
- $scores[150] = 3;
- // Now a 151 element array.
- $scores[200] = 5;
- // Now a 201 element array.

- string $hats[] = {"blue", "red", "black"};
- clear($hats);
- print(size($hats)); // 0
3D vector

• vector $roger = <<3.0, 7.7, 9.1>>;
• vector $more = <<4.5, 6.789, 9.12356>>;

• vector $test = <<3.0, 7.7, 9.1>>;
• print($test.x)
  // 3.0
• print($test.y)
  // 7.7
• print($test.z)
  // 9.1
Conditionals, Loops

if ($x > 10) {
    print("It's greater than 10!\n");
    print("Run!!!\n");
} else {
    print("It's not above 10.\n");
    print("It's safe... for now.\n");
}
For loop

int $i;
for ($i = 10; $i > 0; $i--)
{
    print($i+"...
");
}
print("Blastoff!!!");
Attributes

sphere -name "Brawl";
print(getAttr("Brawl.scaleY"));
float $ys = `getAttr Brawl.scaleY``;
setAttr("Brawl.scaleY", $ys * 2);
Procedure

global proc float square(float $x) {
    return $x * $x;
}

square(5.0);
25
How to model trees / plants?
L-systems

- Developed in 1968 by Aristid Lindenmayer
- Studying the growth of Algae

- Rewriting strings of symbols by other strings of characters

- Alphabet: A B
- Start: A
- Rule: (A -> AB) (B -> A)
Example 1: Algae, explained

<table>
<thead>
<tr>
<th>n=0</th>
<th>A</th>
<th>start (axiom/initiator)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=1</td>
<td>A B</td>
<td>the initial single A spawned into AB by rule (A + AB), rule (B + A) couldn't be applied</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
<td></td>
</tr>
<tr>
<td>n=2</td>
<td>A B A</td>
<td>former string AB with all rules applied, A spawned into AB again, former B turned into A</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
<td></td>
</tr>
<tr>
<td>n=3</td>
<td>A B A A B A</td>
<td>note all A's producing a copy of themselves in the first place, then a B, which turns ...</td>
</tr>
<tr>
<td></td>
<td>/ \</td>
<td></td>
</tr>
<tr>
<td>n=4</td>
<td>A B A A B A</td>
<td>... into an A one generation later, starting to spun/repeat/recurse then</td>
</tr>
</tbody>
</table>


Can we model plant growth?
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Move forward one step, drawing the path of motion</td>
</tr>
<tr>
<td>b</td>
<td>Move forward one step, but do not draw</td>
</tr>
<tr>
<td>[</td>
<td>Begin a branch</td>
</tr>
<tr>
<td>]</td>
<td>End a branch</td>
</tr>
<tr>
<td>+</td>
<td>Increase angle of direction by amount $\theta$</td>
</tr>
<tr>
<td>-</td>
<td>Decrease angle of direction by amount $\theta$</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>F</strong></td>
<td>Move forward one step, drawing the path of motion</td>
</tr>
<tr>
<td><strong>b</strong></td>
<td>Move forward one step, but do not draw</td>
</tr>
<tr>
<td></td>
<td>Begin a branch</td>
</tr>
<tr>
<td></td>
<td>End a branch</td>
</tr>
<tr>
<td>+</td>
<td>Increase angle of direction by amount $\theta$</td>
</tr>
<tr>
<td>-</td>
<td>Decrease angle of direction by amount $\theta$</td>
</tr>
</tbody>
</table>
Procedural Modeling

- Procedural modeling is an umbrella term for a number of techniques in computer graphics to create 3D models and textures from sets of rules.