CIS 6600: Advanced Topics in Computer Graphics and Animation

Homework Assignment 2 (Spring 2024)

Developing Maya Plug-ins Using MEL and C++

Due: Feb. 5, 2024

The goal of this assignment is to help you setup your development environment and gain familiarity with the Maya Plug-in API. It will cover creating commands, custom nodes, and MEL GUIs.

1. L-Systems

Plug-ins allow developers to add new functionality to Maya. To illustrate this point, we will add the ability to create plant models based on the L-Systems described in "The Algorithmic Beauty of Plants" by Przemyslaw Prusinkiewicz and Arstid Lindenmeyer [1]. L-Systems, or Lindenmeyer systems, are very good at representing self-similar organisms and are popular for modeling artificial life, such as plants and cells. L-Systems consist of an input grammar that describes the structure of the system and a set of rules for interpreting the strings generated by the grammar. Below is an example grammar:

$$\omega : F - F - F - F$$

$$\rho : F \Rightarrow F - F + F + FF - F - F + F$$

Each iteration, the L-System produces a string using the rules from the grammar. For the example above, ω is the start symbol. Our first string is thus $F - F - F - F$. On the next iteration, we apply the production rule ρ to the string from the last iteration. In other words, we replace every instance of F from the string $F - F - F - F$ with $F - F + F + FF - F - F + F$.

To visualize the output from our L-Systems, we will interpret each output string character as a command for a simple drawer called a turtle. The state of the turtle is given by (world position, UP, LEFT, FORWARD). The following table enumerates the commands used in this assignment:

F	Move forward a step of length d. Draw a line segment.
f	Move forward a step of length d without drawing a line.
+	Rotate around the UP axis by angle θ
-	Rotate around the UP axis by angle -θ
&	Rotate around the LEFT axis by angle θ
^	Rotate around the LEFT axis by angle -θ
\	Rotate around the FORWARD axis by angle θ
/	Rotate around the FORWARD axis by angle -θ
_	Rotate around the UP axis by angle 180
[	Push the current state of the turtle onto the stack
]	Set the turtle to the top of the stack. Pop the stack

Table 1: Table of Turtle Commands

The pictures below visualize the fractal given by our example grammar using theta equal to 90 degrees for all turns.

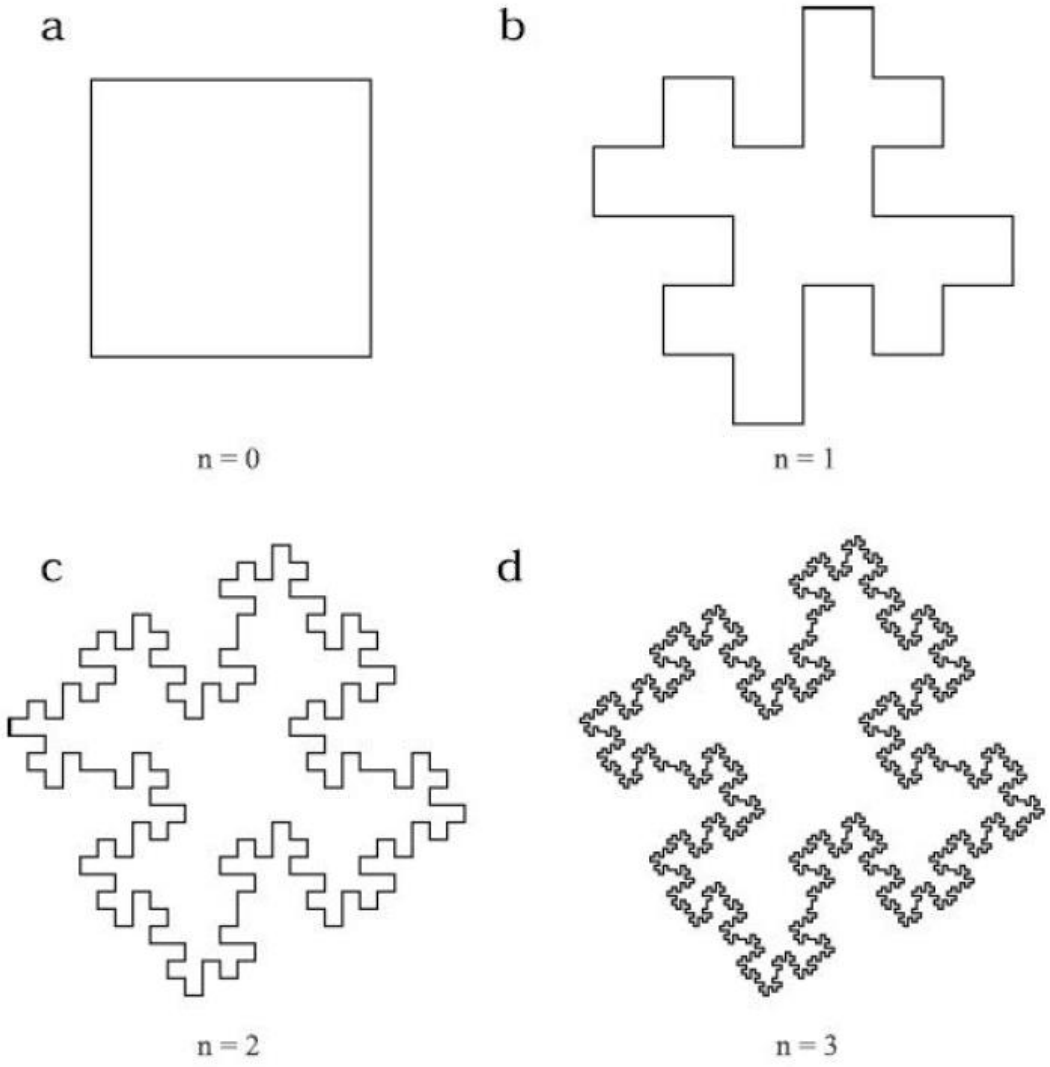

Figure 1: The geometry corresponding to our example L-System.

The assignment includes a simple L-System implementation for you to integrate into Maya. The implementation reads in a grammar and for a requested iteration number, runs the turtle and returns the corresponding geometry. The assignment includes a stand alone C++ example in the L-System project.

Reference:

[1] “The Algorithmic Beauty of Plants" by Przemyslaw Prusinkiewicz and Arstid Lindenmeyer is available free at http://algorithmicbotany.org/papers/#abop

2. Assignment Description

2.1 [5 pts] Getting Started

Check your development environment.

a) Try compiling the skeleton code. Open MyMaya.sln. If MS Visual Studio cannot find the Maya include files, open the properties dialog from the Project menu. Then check the paths under the C/C++ settings and in the Linker settings.

b) Try loading the command plug-in. From Maya, open the Plug-in Manager from the menu Window -> Settings/Preferences. Browse for your .mll file. Check the Script Editor (Window -> General Editors) for the line:

```
{\small loadPluginCallback "<path_to_plugin_project>/LSystemCmd.mll" "plug-in";}
```

Verify that the code loaded correctly. The skeleton code contains a stub in LSystem-Cmd::doIt() which prints “IMPLEMENT ME!" to the output window. Make sure the message appears in the output window.

2.2 [40 pts] Create an L-System command

Let's start by making the L-System callable from MEL. We will create a command which visualizes the positions output by the L-System with Maya geometry.

a) [10 Pts] Replace the stub in LSystemCmd::doIt() so it outputs geometry. A straight forward approach might be to extrude nurbs circles around curves, for example,

```
\begin{tiny}
curve -d 1 -p <branch.start.x> <branch.start.y> <branch.start.z>
    -p <branch.end.x> <branch.end.y> <branch.end.z> -k 0 -k 1 -name curve1;
select -r nurbsCircle1 curve1 ;
extrude -ch true -rn false -po 1 -et 2 -ucp 1 -fpt 1 -upn 1 -rotation 0
    -scale 1 -rsp 1 "nurbsCircle1" "curve1" ;
\end{tiny}
```

For this part of the assignment, simply adding curves will suffice (though they won't be visible when you render the scene).

b) [*15 Pts*] Add command line arguments. Extend your command to take the following command line arguments: default step size (double), default angle in degrees (double), grammar (string), and number of iterations (int). The project comes with several example grammars in the /plants directory.

c) [*15 Pts*] Create a GUI. In a MEL script implement a GUI which contains the following:

(i) Browse button for opening txt files containing grammars

(ii) floatSliderGrp for number of iterations

(iii) floatSliderGrp for default angle

(iv) floatSliderGrp for default step

(v) (Optional) ScrollField widget for entering grammars (if the user opens a file, try loading it into the scrolledit)

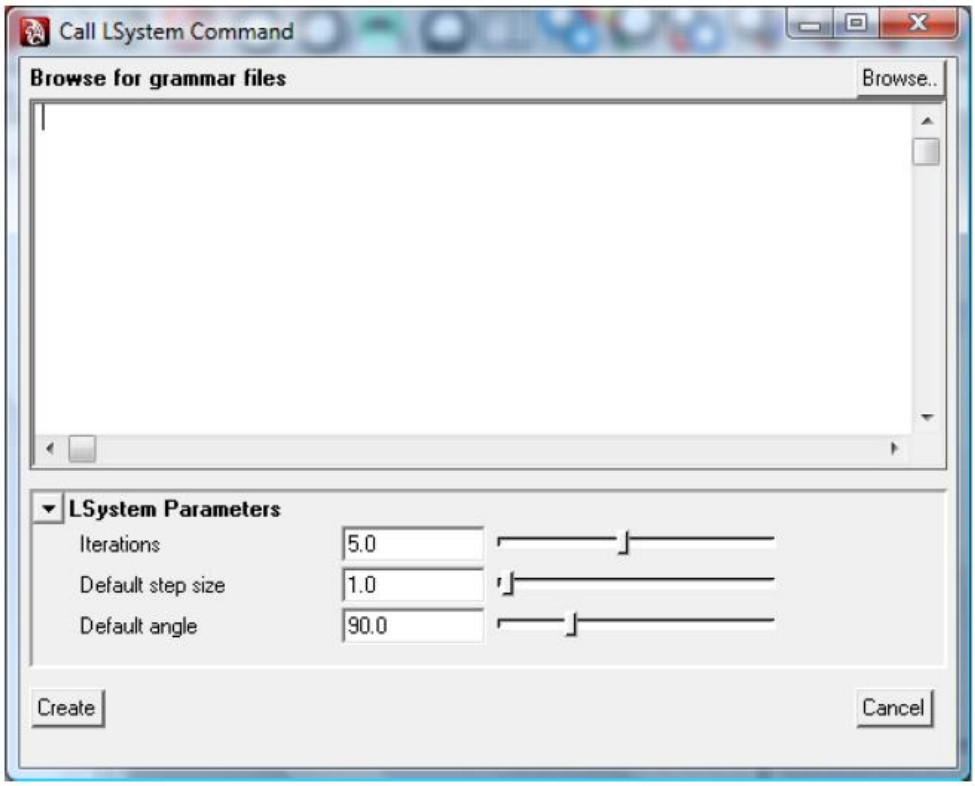

Figure 2: An example dialog.

Add a menu item which opens your dialog to the Maya menu bar. Give the menu an option such as “LSystem Command". When the user presses Ok, the dialog should invoke your LSystemCmd. Try printing out the parameters from the MEL script and from the C++ plug-in to ensure values are passed correctly. You can run a MEL script from initializePlugin() to auto-register your MEL dialog boxes and menus, for example, try the following:

```
char buffer[2048];
sprintf_s(buffer, 2048, "source \"%s/MyPluginDialog\";", plugin.loadPath());
MGlobal::executeCommand(buffer, true);
```

2.3 [50 pts] Create an L-System node

We'd like to now modify our plug-in so that the L-System geometry is dynamic, capable of updating each frame. To accomplish this, we will create a new node which inputs the current time and outputs an MFnMesh.

a) [*10 Pts*] Implement a class L-SystemNode derived from MPxNode (use the example animCubeNode from Maya devkit as a guide). It should contain a constructor, virtual destructor, static creator function, and define a unique Node Id. Your virtual initialize function should define the following:

   (i) Create and add an input MFnNumericAttribute for default angle

   (ii) Create and add an input MFnNumericAttribute for default step size

   (iii) Create and add an input MFnTypedAttribute for the grammar file. You can specify files relative to the plug-in .mll location using MFnPlugin::loadPath().

   (iv) Create and add an input MFnUnitAttribute for the input time

   (v) Create and add a MFnTypedAttribute for the output geometry

   Add code to initializePlugin() and uninitializePlugin() to register your new node. Now, when you load your plug-in, Maya should find both LSystemCmd and LSystemNode. In the MEL script window, type the following to test

```
createNode LSystemNode;
```

b) [*35 Pts*] Implement LSystemNode::compute(). This function should i) get called whenever an attribute, such as time, changes, and ii) then recompute the output mesh. We have included a utility class for creating cylinders which you may use to help create the mesh geometry. Don't forget to connect all the input attributes to the geometry! Test that changing every attribute from the attribute panel works correctly!

c) [*5 Pts*] Add an additional menu option called \LSystem Node..." under your L-System menu for creating your new node. Use the following MEL scrip to get create your and setup your node to start:

```
createNode transform -n LSystem1;
createNode mesh -n LSystemShape1 -p LSystem1;
sets -add initialShadingGroup LSystemShape1;
createNode LSystemNode -n LSystemNode1;
connectAttr time1.outTime LSystemNode1.time;
connectAttr LSystemNode1.outputMesh LSystemShape1.inMesh;
```

After running the above script, check that the created nodes are correct in Maya's Hypergraph and attribute windows. Note that readable or writeable node attributes automatically appear in the node's attribute window (note: don't make output values writable!). Also, be careful not to

needlessly re-initialize the L-System (which will clear the L-System's cached values from previous calls). Below are some screenshots to compare against.

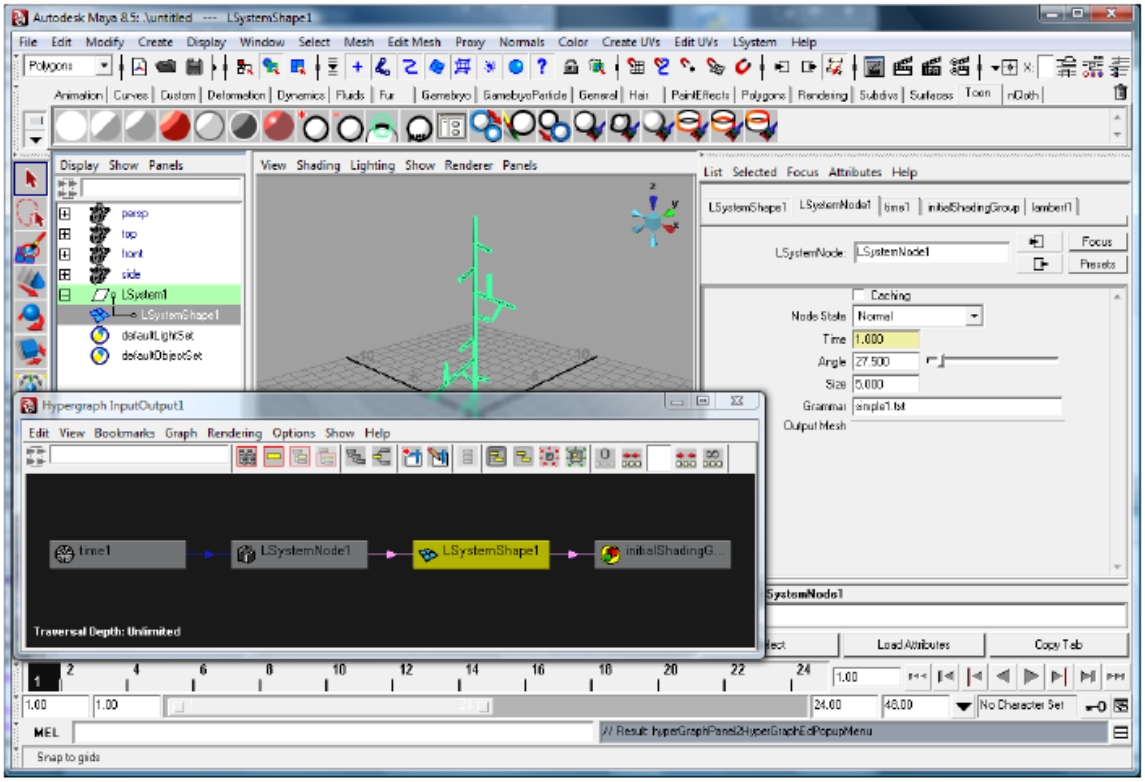

Figure 3: Screenshot after loading L-SystemNode.

2.4 [5 pts] Submission

(a) ZIP your .cpp, .mel, .h, .vcproj, and .sln files and submit to Canvas. For full credit, your plug-in must build without modifications for VisualStudio2022. To grade your assignment, the grader will compile your source, load LSystem.mll into Maya, and then run 'Create LSystem' and 'Create Dynamic LSystem' from the menu.

(b) Briefly write-up which questions you completed in a file called README.txt. Don't forget to point out any extra credits attempted! Include any images from your assignment that you're proud of!