My scene is a set of points that find the shortest path between themselves using the A\* algorithm. There is a maze / fractal displayed on the screen. The maze can be changed using the “t” key. It can be changed between 3 different maze options. Using the arrow keys the user can increase or decrease the recursion level.

The extensions I implemented were.

Stochastic L system

* On L Systems Housing and Maze, the generation of a new box is a production, and it not make a new box / line is also a production

Parametric

* On my housing L system, I made it parametric. The length of a line continually shrinks, so I have my lines drawn and moved len \* pow(2^k) distance. This saved me a LOT of compute. Previously I had used the production ex. “r -> rr” instead which worked, but exponentially increased the number of lines I had to rasterize.
* Noted problem, my primitive method does not work for a depth over 9. I blocked the user from going over a depth past 9, and a depth of 9 would be too small to render on my sketch regardless.

Interaction

* You can move the points for the start / goal of the pathfinding algorithm using the mouse.
  + To switch between modes, click “S” for the start, and “G” for the goal.
* You can increase the recursion level by pressing the up arrow, and decrease by pressing the down arrow.
* You can also toggle the discovered path of the pathfinding algorithm with the “D” key.

Additional - different L-system in your scene.

* I found the Hilbert Curve really interesting so I added that to my program
* Not needed for grading, but I thought it was fun to implement.

**L System 1 (Housing)**

Alphabet: U, r, d, l, u, [, ], x, y, z, q

Axiom: U

Rules:

U -> [U r0 U d0 U l0 U u0]

// this is inserted 9 times, to give it a 90% chance of selection

U -> [x1 y1 z1 q1]

// this is inserted once, to give it a 10% chance of selection

Instructions:

| U | Nothing |
| --- | --- |
| [ | push location onto stack |
| ] | pop location from stack |
| rk | move right k lengths |
| dk | move down k lengths |
| lk | move left k lengths |
| uk | move up k lengths |
| xk | move up and draw k lengths |
| yk | move down and draw k lengths |
| zk | move left and draw k lengths |
| qk | move up and draw k lengths |

// I could have reused certain alphabet symbols to use less, but I found breaking up drawing and moving to be more readable.

**L System 2 (Hilbert)**

Alphabet: A, B, +, -, F

Axiom: A

Rules:

A -> +BF-AFA-FB+

B -> -AF+BFB+FA-

Instructions:

| A | Nothing |
| --- | --- |
| B | Nothing |
| + | Turn CW |
| - | Turn CCW |
| F | Move forward and draw line |

// due to not using rotations, I instead track the rotation in an integer where

// 0 right

// 1 down

// 2 left

// 3 right

// I think there may be a better solution, however this worked well for this only right angle L system.

**L System 3 (Maze?)**

Alphabet: F, f, L, l, R, r, S, s, [, ]

Axiom: [F]rsssss[F]rssssss[F]rssssssss[F]rssssssss[F]rsss[F]rss[F]rs[F]

Rules:

F -> fL

F -> fR

F -> fS

F -> [F][F]

F -> fF // inserted 3x

L -> lL

L -> lR

L -> lS

L -> lF // inserted 3x

R -> rL

R -> rR

R -> rS

R -> rF // inserted 3x

S -> sL

S -> sR

S -> sS

S -> sF // inserted 3x

Instructions:

|  | Rotate CW |
| --- | --- |
|  | Rotate CCW |
|  | Add line, move forward |
|  | Move forward (skip) |
| [ | push location onto stack |
| ] | pop location from stack |

This is a basic maze generator. I added it because I needed an original L system. I wanted to have a series of progressing lines, with gaps to generate a maze. It produces reasonable results.