Bent and Bendy Plants

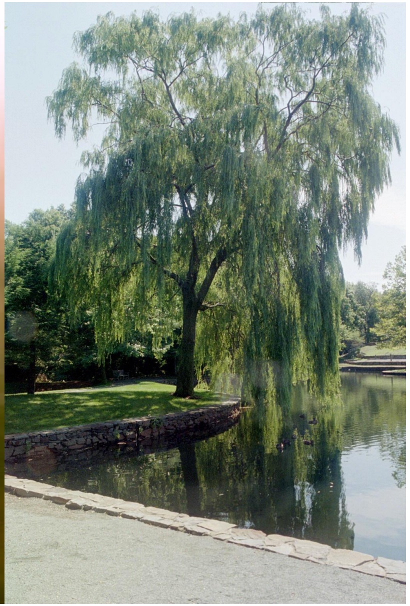


CS275 Project Progress Update

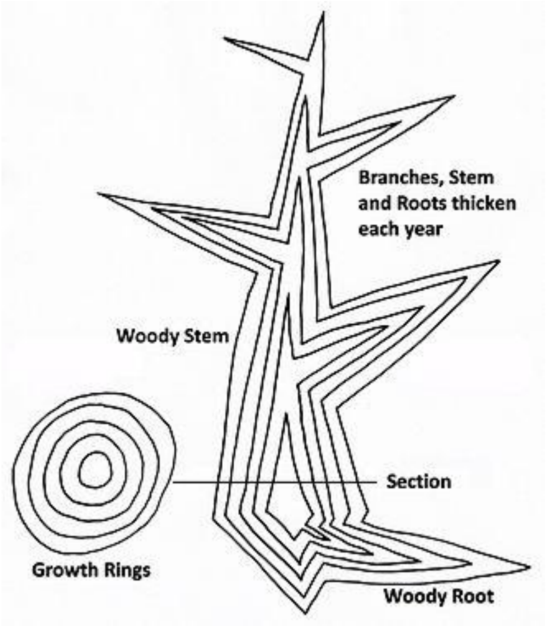
Pravin Visakan

Motivation: Bendy Plants





Motivation: Bent Trees



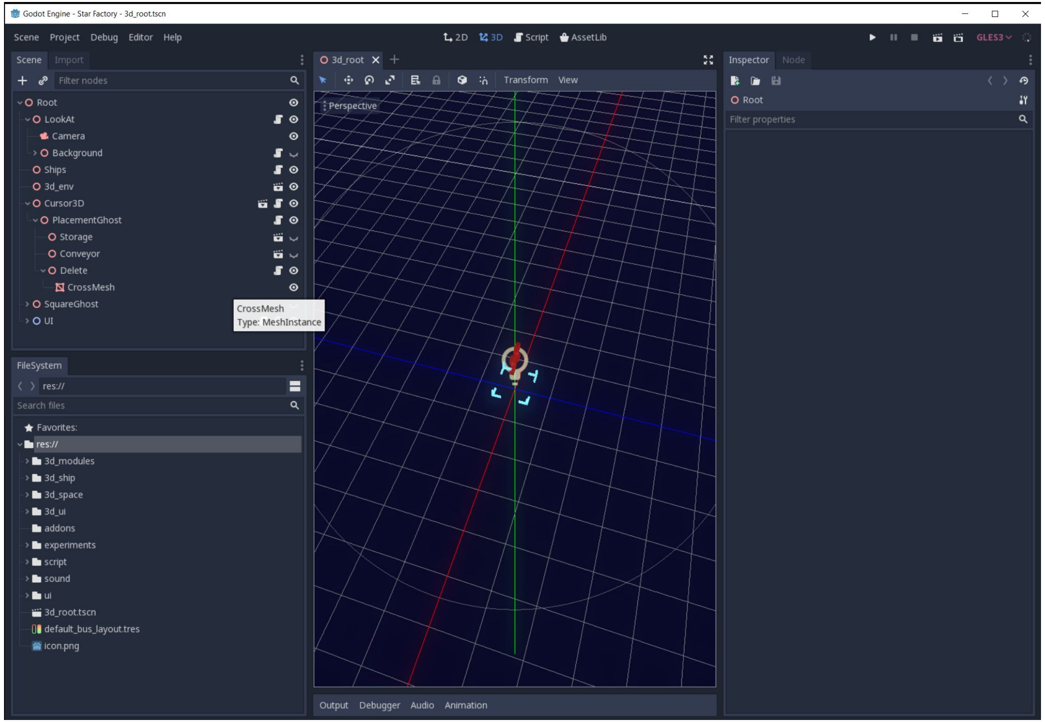
- Secondary growth produces stiffness in woody plants

The Idea

- Use traditional L-Systems to simulate plant growth
- Replace plant segments with physics-based soft models to produce bendy, flexible shapes
- Interweave L-System growth / production steps with physical simulation
- Introduce simulated forces (e.g.. wind) and stages of stiffening to grow bent secondary growth structures

Godot

- Standard high-level game engine
- Free software!
- Hierarchical scene structure and object-oriented language support
- Includes support for physics



L-Systems

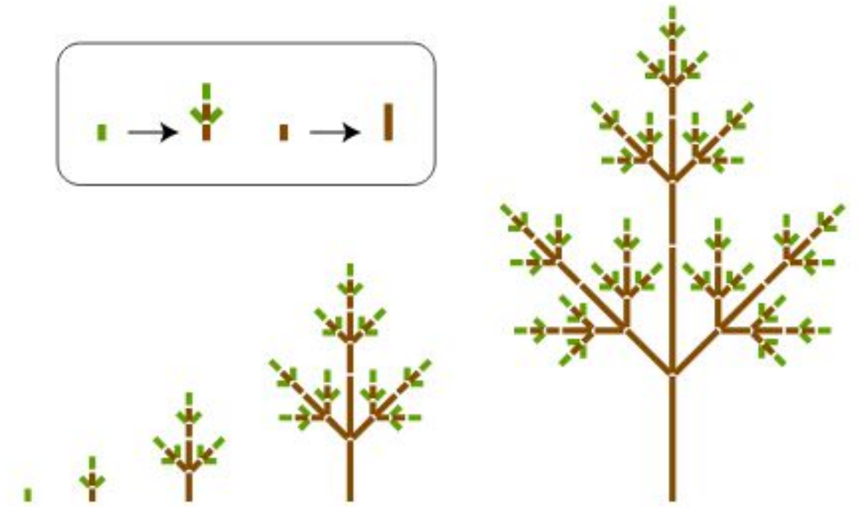


Fig. 8: The first five stages of the developmental of a simple branching structure modeled using the L-system in Table 4. The inset shows a graphical representation of the productions.

An L+C program equivalent to the symbolic notation in Table 4 is shown below:

```
module A;          // apex
module I(float);   // internode (length)

Axiom: SetWidth(0.4) A;

A :    produce I(1) SB Left(45) A EB SB Right(45) A EB I(1) A;

I(s) : produce I(2*s);

interpretation:
A    : produce SetColor(1) f(0.2) F(0.8);
I(s) : produce SetColor(2) f(0.2) F(s-0.2);
```

- CSG - esque models
- Start with an axiom, apply productions, get a string representation of a tree

From “Modeling plant development with L-systems” by Przemyslaw Prusinkiewicz, Mikolaj Cieslak, Pascal Ferraro, and Jim Hana

The Turtle Graphics Interpretation: Problematic

<table>
<tr><th>Turtle command</th><th colspan="2">Symbol</th><th colspan="2">L+C keyword</th></tr>
<tr><td>draw line segment</td><td colspan="2">F</td><td colspan="2">F</td></tr>
<tr><td>move without drawing a line</td><td colspan="2">f</td><td colspan="2">f</td></tr>
<tr><td>turn left | right</td><td>+</td><td>−</td><td>Left</td><td>Right</td></tr>
<tr><td>bend up | down</td><td>∧</td><td>&</td><td>Up</td><td>Down</td></tr>
<tr><td>roll to the left | right</td><td>\</td><td>/</td><td>RollL</td><td>RollR</td></tr>
<tr><td>start | end branch</td><td>[</td><td>]</td><td>SB</td><td>EB</td></tr>
<tr><td>set line width</td><td colspan="2">#</td><td colspan="2">SetWidth</td></tr>
<tr><td>set line color</td><td colspan="2">,</td><td colspan="2">SetColor</td></tr>
</table>

Table 3: Basic turtle commands in a symbolic notation and in the L+C language.

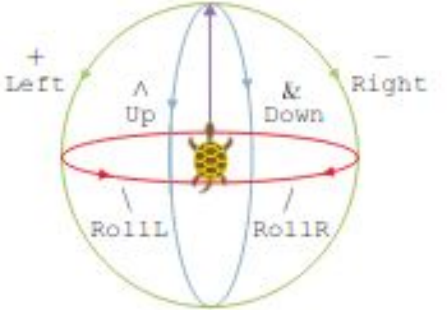


Fig. 7: Specification of turtle rotations in three dimensions.

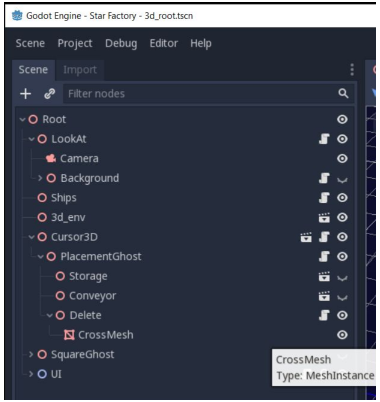
- Awkward mapping to programming
- Physics??

From “Modeling plant development with L-systems” by Przemyslaw Prusinkiewicz, Mikolaj Cieslak, Pascal Ferraro, and Jim Hana

An Object-Oriented Approach

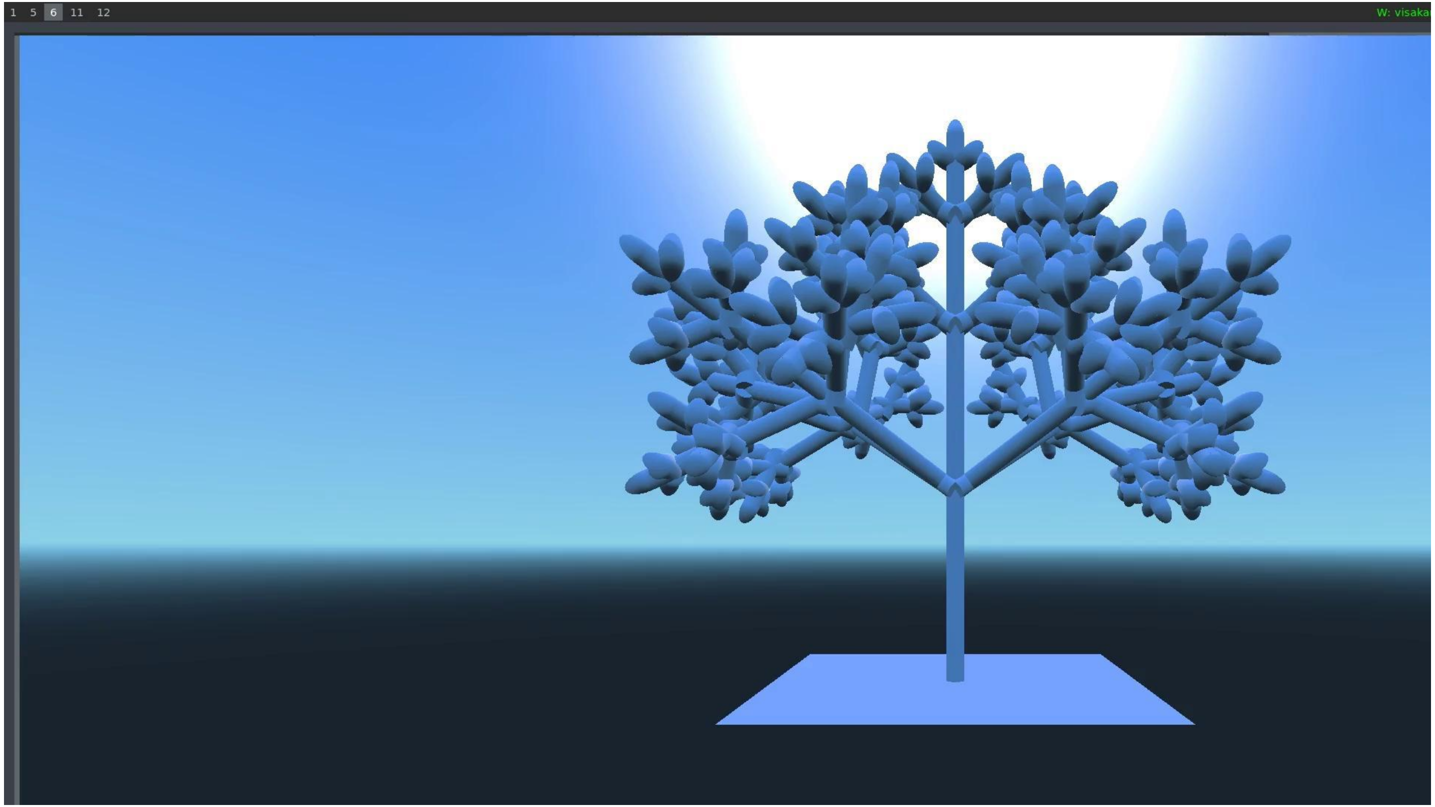
- Roughly equivalent, but easier to work with
- Uses stateful objects - easier organization of data
- Uses tree data-representation directly
- Integrates better with common design patterns
- Note: is less formal / probably less useful for plant studies; also, Python code is available

```
extends MeshInstance
class_name Symbol
# Generic "base" class for symbols

#Constructor
func _init():
	pass

# modifies the given list with this symbol advanced one iteration step
func grow():
	self.grow_children()

# recursive function, grows all child nodes, DFS order
func grow_children():
	var children = self.get_children()

	for child in children:
		child.grow()
```

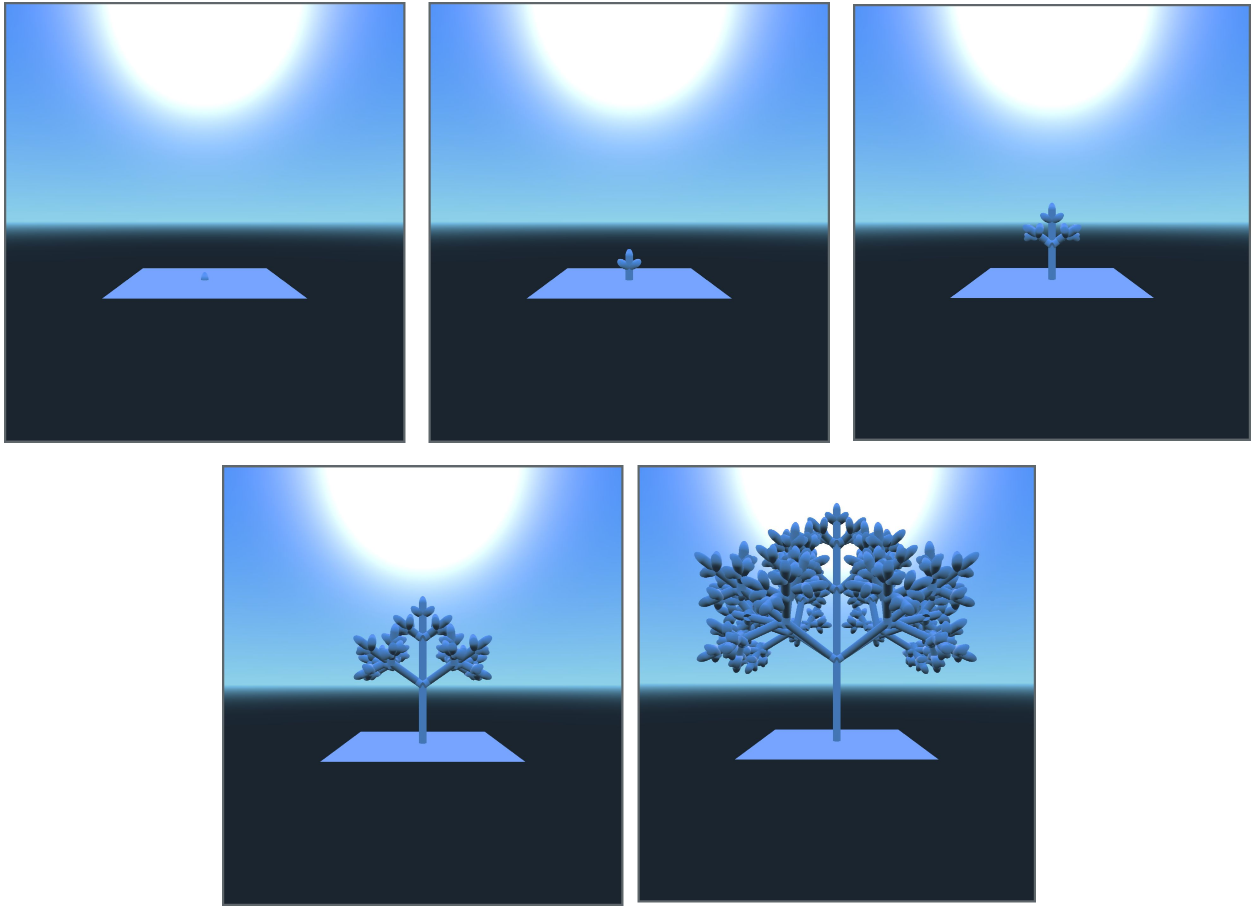



```
extends Symbol
class_name InterNode

# state variables
var age = 1;

func _init():
	var cylinder = CylinderMesh.new()
	cylinder.set_bottom_radius(0.5)
	cylinder.set_top_radius(0.5)
	self.set_mesh(cylinder)

# modifies the given list with this symbol advanced one iteration step
func grow():

	#debug
	print("debug")

	# update own size
	age += 1;
	var current_height = mesh.get_height()
	mesh.set_height(2*age)

	#update transform to compensate for increased size
	var parent = self.get_parent_spatial()
	var offset = 0

	#TODO make more generic wrt parent type; find a way to ref own class
	if !parent || parent is StartBranch:
		offset = 0
	else:
		offset = 1



	var translation_factor = 1 + offset

	self.translate(Vector3(0,translation_factor,0))

	# perform productions on children
	self.grow_children()
```

Progress So Far



Progress So Far



Future Steps

- Incorporate “soft body” physics elements into the object-oriented L-System
- Define object-oriented models based on real-world plants that display bendy characteristics - willows, flowering trees, etc.
- Quality of life features
    - Nicer aesthetics - color, materials, more complex meshes/shapes for intervening nodes
    - Performance optimizations
    - Branch culling - to prevent overcrowding / overlapping
    - Continuous growth - scale growth with delta_t instead of progressing at fixed intervals

The End!

Any Questions?