Turtle Graphics in 3D

My Final assignment was aimed at porting aspects of turtle graphics to 3D. In process of doing so, my aims were:

- **My main idea is to do this in JavaScript and HTML**. This was done to gain some level of interaction with WebGL. In my experience, it seems to follow most practices established by OpenGL. Understandably, work wasn't done directly on WebGl, but a wrapper called Three.js that augments, and eases use.
- **Some ability to draw 3D primitives**. This was attempted. Unfortunately, since work was started in Vector graphics, leveraging any sort of reasonably high triangle count primitives was just not possible as WebGL is not capable to performing Vector graphics. Instead it had to be done directly to a SVG canvas, which while capable, can't handle reasonably sized primitives. To avoid spending more time, and at least put a deliverable forward, this was sacrificed.
- ***Possibly*** **do this as a vector graphic**. This links to the second point. SVG graphics tend to be inversely proportional to primitive count. As a result, even a small number of primitives can tank performance significantly. The reasoning here seems to be that SVG on browsers isn't hardware accelerated. While there is a working prototype with it, performance is rather bad.

During this process, I have attempted to not make the turtle require compilation and be sort of interpreted. A result of doing that, and my basic programming skills is that it is designed more like a text parser and doesn't have direct access to the language like the Python Turtle or LOGO graphics would have. This did turn out to be restricting in the variety of commands I could pass to the language without reworking everything.

In doing this, I settled on a few main functions that I wanted to support. These were :

1. Move N units : Done using Forward X command with X being a float (negative or positive)
2. Turn/Rotate(AXIS,ANGLE) : Done using Rotate (AXIS) (ANGLE). Axis can take two values : X and Y. This is meant to emulate the latitude and longitude way of positioning with X controlling the longitude and Y the latitude. The angle represents the inclination in degrees. Both Axis and Angle can be made negative to deal with left/right rotations. While a third axis of rotation could be used, since figures here are mostly symmetrical along one of their axes, it is of little use.
3. Up/Down : A basic command of Turtle, useful for positioning without drawing to the screen.
4. Repeat (Count) **,** [Sequence,] : Repeat a series of actions for n counts. Actions are separated by a comma (,).
5. Curve(Angle, Radius) : This can be accomplished using a suitable Repeat Command. So, Repeat 32, Forward 3, Rotate X 11.25 would create a circle (not made of smooth splines arguably, but still)
6. Recurse (Start) (Intervals), [Sequence,] : Recurse on a sequence. Regrettably, this is a hack at best and isn't perfect. While it works, there is still work required to draw figures like trees/leaves.

Inspiration for some parts of the design (specifically the commands required) came from Tom Verhoeff (Verhoeff, 2009). Inspiration was also taken from Cheloniidae (Tipping, n.d.) who made a Java version of a 3D Turtle-like program.