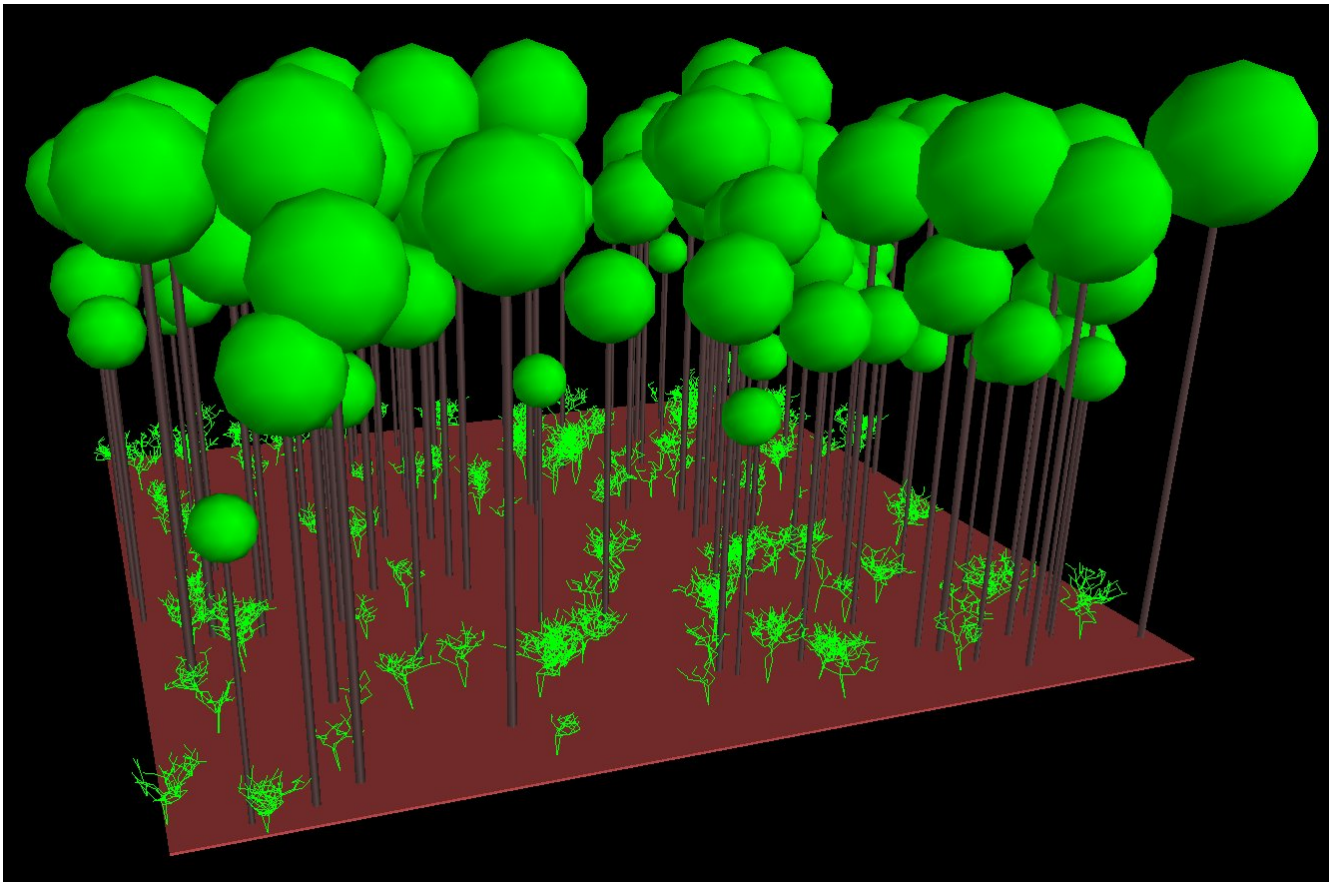
Lab 11: Using VPython

For this lab, you will create a 3D forest using VPython, something like the following:



For this, you'll need to create four things:

- The ground, which will be a large flat box, centered at 0,0,0
- The tree trunks, which will be cylinders, pointing up (along the y axis), of reasonably long length. These will need to be positioned randomly in the x and z directions, but all starting at the same height (y=0). They should also have random lengths.
- The treetops, which will be green spheres. One of these should appear at the top of each trunk. The sizes should vary, preferably with the height of the trunk.
- Undergrowth, which can be generated using the technique from class. This should be scattered around the forest floor. It is not required.

Installing VPython

- On Windows or Mac, you can download VPython from http://www.vpython.org/. Note that a specific version is required to work with Python 3. On Mac, if you have Python 2.x installed from the previous pygame assignment, feel free to use a newer version of vpython.
- On Linux, look for a package named python-visual. On any Debian-based system (Ubuntu included), just type `sudo aptitude install python-visual`. On Gentoo, install dev-python/visual. Other distributions should be similar.

Creating the ground

The ground will be a large, flat box. Take a look at the documentation at:
http://www.vpython.org/contents/docs/visual/index.html

You will want to center your box at 0,0,0 (set `pos` to `(0,0,0)`). For other dimensions, I recommend height of 1, length of 400, and width of 400. The color should be some sort of brown. I rather like `(.8,.3,.3)`, which is a sort of reddish brown.

Creating the trees

You'll need a couple of things to make a tree:

- `sphere` generates a sphere given arguments of `pos` (the position of the center of the sphere), `radius`, and `color`.
- `cylinder` generates a cylinder given arguments of `pos` (the position of the starting point of the cylinder), `radius`, `color`, and `axis`, which is the direction and length of the cylinder. To make a cylinder which extends upward 100 units, you would set this to `(0,100,0)`.
- `random()` gives a random number between 0 and 1. You can import it from the `random` package (`from random import random`).

I recommend an algorithm something like this:

```
for i in range(0,100):
   find a random value for the x coordinate (t_x)
   find a random value for the z coordinate (t_z)
   find a random height for the tree (height)
   make a cylinder with a position of (t_x, 0, t_z) and length height
   make a sphere at position (t_x, height, t_z)
```

There is nothing to preclude one tree from forming directly on top of another, but in a real forest, trees often grow very close to each other as well. The random values for x and y should be in the range covered by the ground. You can generate a random number between -200 and 200 like this:

```
x = 200 - random() * 400
```

You might also want a bigger window at this point. This would make a 1600x1200 window (adjust to suit your monitor):

```
# The following must be at the BEGINNING of your program!
display(height=1200,width=1600)
```

Alternatively, you could use fullscreen mode:

```
display(fullscreen=True)
```

Extra Credit: Adding undergrowth

The forest should look reasonably good at this point, with variable height trees and ground. However, adding undergrowth can improve it quite a bit. There is a file called forest_brush.py in the labs area of the class webpage. It draws 100 green bushes of random heights. You can simply paste this into your program at the end (minus the imports and the display setting), but you might have to adapt the bush size to fit your forest.

You might notice some slowdown at this point. VPython uses OpenGL (Open Graphics Library), which along with DirectX is one of the leading 3D development libraries. This allows it to use your video card to speed drawing. Many laptops and inexpensive desktop computers use integrated video cards with poor 3D performance. If your forest is very slow, reduce the size or amount of undergrowth. Commercial video games have been carefully optimized for maximum performance, and thus are capable of impressive graphics on even low-end and obsolete video cards.

Completing the undergrowth is worth 25% extra credit on this lab.