**Graph Sandbox Development Journal**

*“To model interconnectedness…”*

**V0.1 Features**

**V1.0 Features**

* Quiz feature where you can rearrange nodes to by hand for each step of an algorithm and check whether you’ve got the correct arrangement.
* Pseudocode for each algorithm with different lines highlighted as you step through it.

**Classes & inheritance**

Node (position, rendering, mouse interactions) -> BinaryNode (implements 2 children, insertions), GraphNode (implements neighbors, possibly edge weights).

We want support for directed and undirected graphs/edges too. I’m starting to think it might be better to keep the tree and graph projects separate, or at least keep their code fairly separate in this project. Trees and graphs don’t have as much in common as I had thought.

BinaryTree, AVLTree, TwoThreeTree, RedBlackTree, Heap, Tree, Graph, etc., inheritance can be configured however makes sense later.

Somehow the algorithms such as insert and remove need to be configured for each class. Not sure how to structure that at the moment.

**November 25, 2020**

I simplified and improved on the outline drawing by checking diagonal pixels and using an abs() node to check both sides of

**December 7, 2021**

A good way to deal with instructions that need to be generated once and then executed in any order later would be to generate sentences from an alphabet and store them in a string. Well, they need not be strings – lists would also suffice. But instead of storing the function calls themselves it would probably work just fine to encode them into a list. You’d need each symbol to have a reverse symbol which maps to the reverse function but that’s fine. You could either generate a reverse version of the list at generation time or just lookup the reverse versions at runtime when reversing.