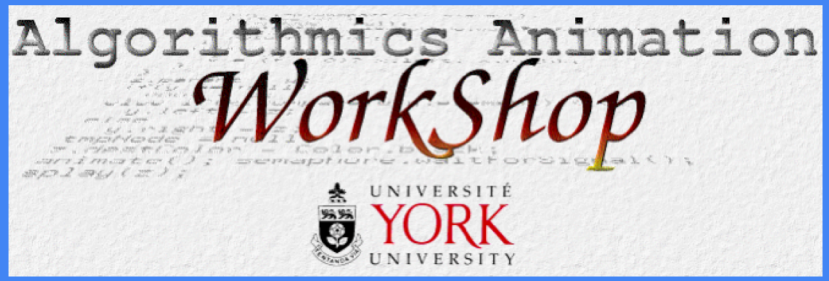
Algorithmics Animation Workshop (AAW)

Jay Karon and Nadav Hames

What is it?

- Interactive visualizations of data structures and algorithms
- Platform for students to learn from and contribute to
- Hosted on eecs.yorku.ca

The old site



DICTIONARIES:

- Hashing (by Hang Thi Anh Pham, 2001)
- Splay Tree (by Sotirios Stergiopoulos, 2001)
- Red Black Tree (by Sotirios Stergiopoulos, 2001)

PRIORITY QUEUES:

- Leftist and Skew Heaps (by Soheil Pourhashemi, 2007)
- Binomial Heap (by Sotirios Stergiopoulos, 2001)
- Fibonacci Heap (by Jason Huang Hu & Wei Wang, 2003)

DYNAMIC PROGRAMMING:

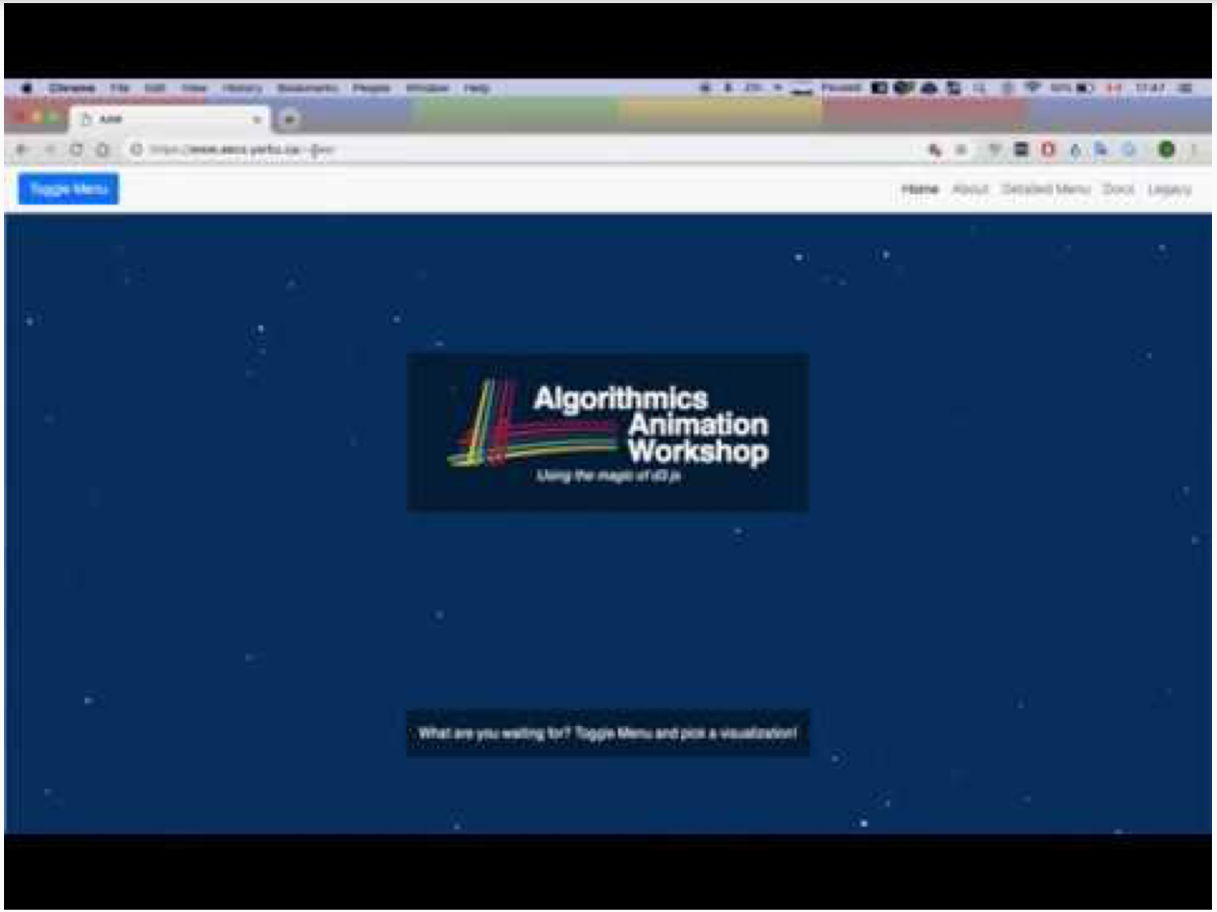
- Optimal Static Binary Search Tree (by Roman Gubarenko, 2005)

GRAPHS:

- Dijkstra - Single Source Shortest Paths (by Hai Feng Huang, 2005)
- Minimum Spanning Tree (by Dana Chiassi, 2005)

Problems		Goals
Antiquated Java applets	→	Make accessible
Little documentation	→	Thorough documentation
Drawing not abstracted	→	High-level abstraction
Java dependencies duplicated across project	→	Easily extendable, dependency sharing
High learning curve	→	Lower contributing difficulty
		Modern web design

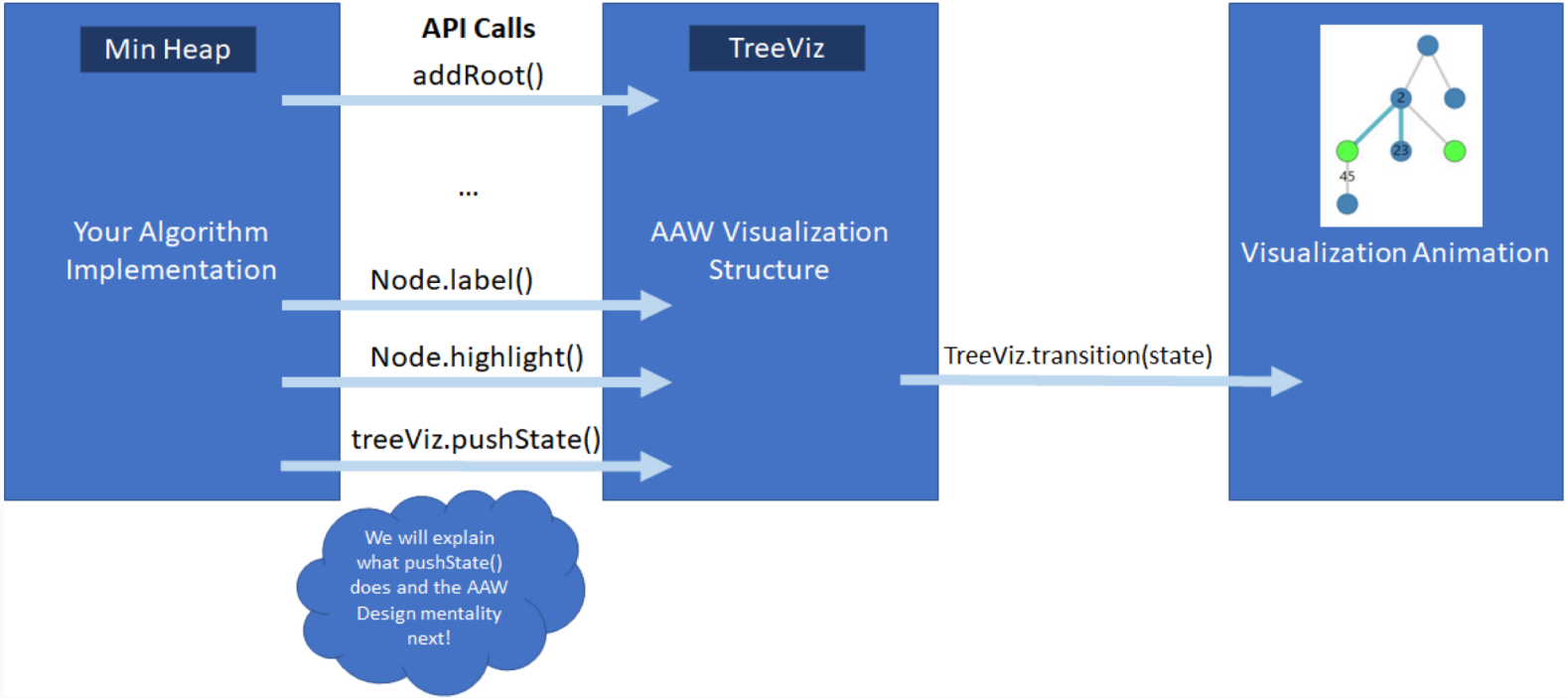
Demo


Toggle Menu
Home
About
Algorithmics
Animation
Workshop
What are you waiting for? Toggle Menu and pick a visualization!

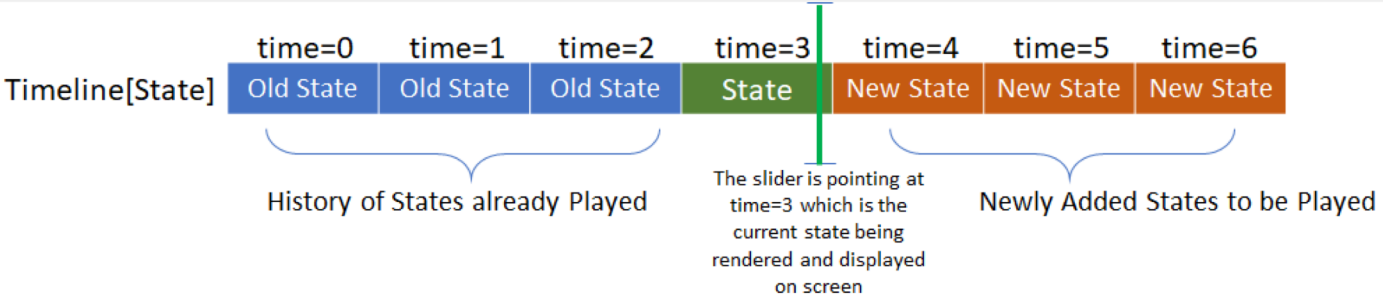
Under the hood

- Written in TypeScript
- D3.js for rendering
- Webpack for bundling & compilation
- Hosted on Github for version control & pull requests
- Basic data structures & algorithms from Loiane Groner (with permission)

AAW Design - Separation of concerns



AAW Design - Visualization Persistence




```
NodeViz.highlight();
tree.pseudo.setCurrentLine(1);
```

These API calls make up the changes to the new State to be pushed to the timeline

```
tree.pushState(); - appends state to the end of the timeline
```

```
controller.play(ms);  - Moves the slider forwards until
                        reaching the end every ms milliseconds.
```

Contributing - Easier than ever!

- Based around Github
- Documentation
- Guides

3. Your TS Visualization File

This section will break down a typical TS (TypeScript) visualization file into multiple components to help you understand how to implement one for yourself. In this file, all of your visualization code will be written. Create it in `src/algs` and name it something like `myVizTitle.ts`.

> Note: Your file will not compile unless it is included in `visualizations.config.ts`. Follow the example template at the top of the config file to include your visualization.

All components mentioned below should be included together in your file. For this tutorial we will be using the Tree Traversal visualization. See the full file in `treeTraverals.ts`.

Import Statements

Where classes and methods are imported from other files.

```
import {EdgeViz, NodeViz} from "../lib/components";
import {MultiTreeViz} from "../lib/structures";
import * as d3 from "d3";
import {Controller} from "../lib/gui";
import Queue from "../base/data-structures/queue";
import * as _ from "lodash";
```

A good IDE like IntelliJ IDEA will take care of this part automatically. So you can generally just forget about it.

AAW Url:
https://www.eecs.yorku.ca/~aaw/