Case Study – Train Signaling System

ecobee interview – Kendall Auel

This case study highlights the design decisions involved in creating a train signaling system. It will provide an interactive system for creating train track segments, connecting them together, adding signals at specified locations, adding trains to the tracks, and providing destination points for the trains. The train traffic is then protected from collisions by use of a signaling system on each segment of track.

Trains continue to travel until they arrive at a terminator (their destination).
The system will have automatic route planning to find optimal routes for the train to reach its destination.

Technical Decisions

The case study:

- Will be implemented on a desktop PC running Ubuntu Linux (20.04).
- Will be written in C++ (g++ 9.4.0).
- Development using an Eclipse C++ CMake project.
- Source code will be committed to a github repository.

The track network is essentially a special case of a non-directed graph, with the graph edges representing track segments, the edge weights are the track lengths, and the nodes are one of three flavors:

- Terminator (only one edge connected)
- Continuation (two edges are connected)
- Junction (three edges are connected)

The junction isn't a true graph node, since it has a left/right switch from the common edge to one of the fork edges. Also, there is no path possible from one fork edge to another.

This will make route optimization a little tricky, since a straightforward adjacency list and breadth-first search must take into account the special junction rules.

Basic Operation

Signals

As an embedded system case study, the main part of this project that represents a device under control is the signal device. To more accurately represent distributed embedded microprocessors, the signal

objects could run as separate threads, with simulated train sensing devices and a communication protocol.

Track Segment

A track segment has a length and two endpoints.

An endpoint is attached to a node. Each node can have up to three endpoints. Each endpoint can also support a signaling device.

If the endpoint has:

- One connection – it is a terminator. The train stops at the endpoint.
- Two connections – it is a continuation. The train stops only if a signal is red.
- Three connections – it is a junction. The junction can be set to left or right. The train stops only if the switched connection signal is red, or if incoming from a fork that is not switched.

Train

A train has a length, a direction, and a destination. It may also have a velocity, acceleration, and maximum velocity – although that can be skipped for simplicity.

- We can ignore the length by assuming it is always less than one track segment.
- Direction is relative to the track segment that the train is currently on (toward endpoint A or endpoint B).
- The destination is an endpoint of a specific track segment, and a route to get there.

Route

The only control a train has over its route is the setting of the junction switches. At its simplest, a route is an ordered list of junction switch settings, and only for junctions we approach from the common edge.

Simulation

There are two elements of the simulation.

1. The movement of trains from one segment to the next.
2. The update of signals to reflect safety of passage into each segment.

Generally speaking, the importance of the signals is at junction points. We do not want a train entering a section of continuous track only to stop partway due an oncoming train. Both trains will be required to stop, resulting in a deadlock.

Instead, we want trains to proceed through a junction only if we can protect the train’s route through to its next junction passage.

If a train is following another train going in the same direction, the signal at each continuous connection point serves to prevent the lead train from being rear-ended. But these signals won't cause a deadlock as both trains can continue moving.

Each simulation step will transfer a moving train from the segment it is currently on to the next segment in its direction of motion, provided the signal in front of it is green.

Then, the safety signals are re-evaluated based on the new position of the train.

If no train can continue moving (at a terminator, at its destination, or its signal stays red) then the simulation is complete.

Implementation

The main objects for the implementation for the track layout are:

- Graph edges (the track segments)
  - Optional signal at one or both endpoints.
- Graph nodes (the connection points)
  - Junction switch setting (for junctions)
- Collections of edges and nodes for iterating as needed.

Each train consists of

- The train attributes (length, position, direction)
- The route
  - Current route status

In addition we will have interactive functions:

- The track layout builder
- The train placer / route builder
- The train motion simulator

Track Segment (Edge)

Data:

- Identifier
- Length
- Connection (node) 1 and node connection type
- Connection (node) 2 and node connection type
- Pointer to occupying train (or null)

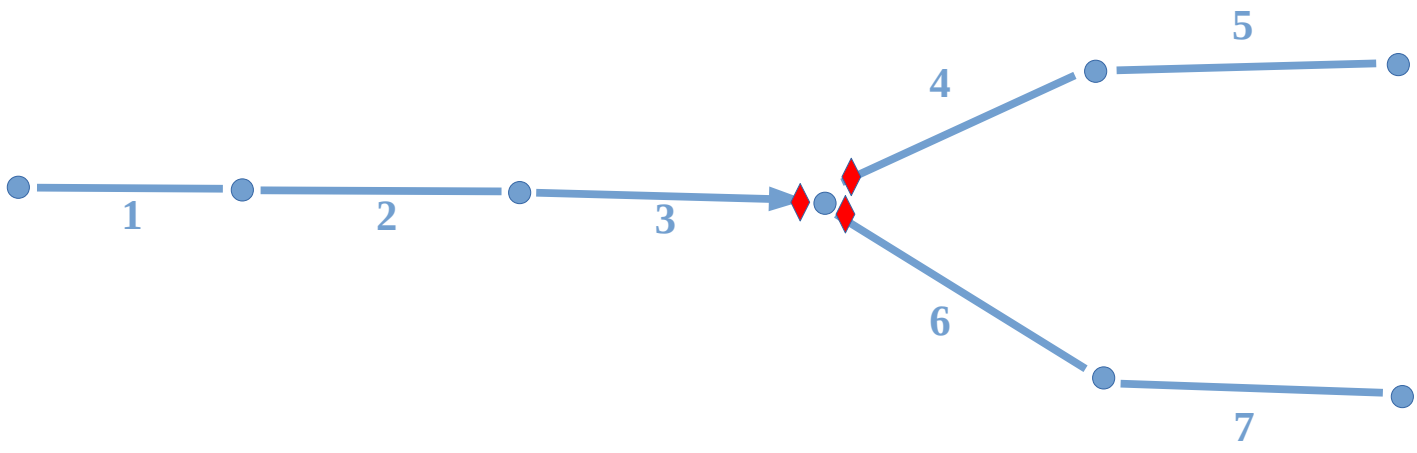
Methods:

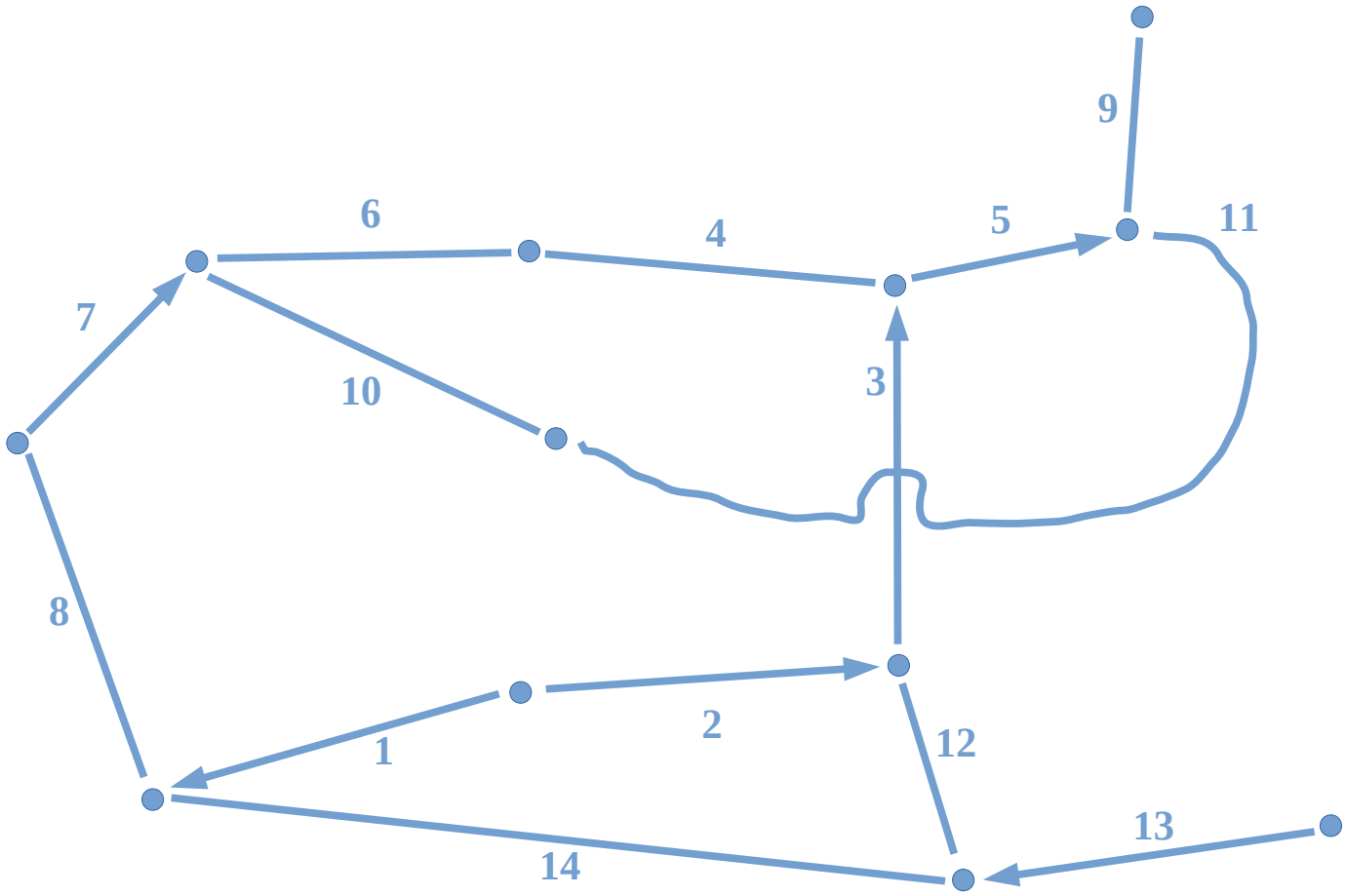
- Connect track segment
- Get signal status for an endpoint
- Get node for an endpoint
- Set occupying train

Track Connection (Node)

Data:

- Identifier
- Track (edge) 1 and endpoint – this is a terminator, continuation, or junction common
- Track (edge) 2 and endpoint – this is a continuation, or junction left fork
- Track (edge) 3 and endpoint – this is a junction right fork
- Junction switch setting




9
6
4
5
11
7
10
3
8
2
1
12
13
14