Design Rationale

Our design choices were guided by a general solution to the problem.

This solution requires the successful completion of three main parts:

1. Navigating the map, and accurately storing all map information, particularly the location of lava and health traps, and of the keys.
2. With this information, implementing a path-finding algorithm that would determine a path allowing us to safely traverse the map and accrue the keys in the correct order (from highest to lowest), and inevitably reach the ‘finish’ line.
3. And finally, ‘control’ the car so that it would accurately follow this path.

Bearing this in mind, it was obvious that implementing all this functionality in one MyAIController class would lead to a very incohesive system. Particularly if all the logic were done in the update() method.

We noticed straight away that the update() method did two main jobs: firstly, determine where the car should go and secondly, how the car should get there. This made the update() method highly unreadable as a sequence of conditional clauses.

This was our first abstraction and major break-through: that we needed to separate the responsibility of determining where to go with the actual act of getting there. These are very two distinct actions and functionalities.