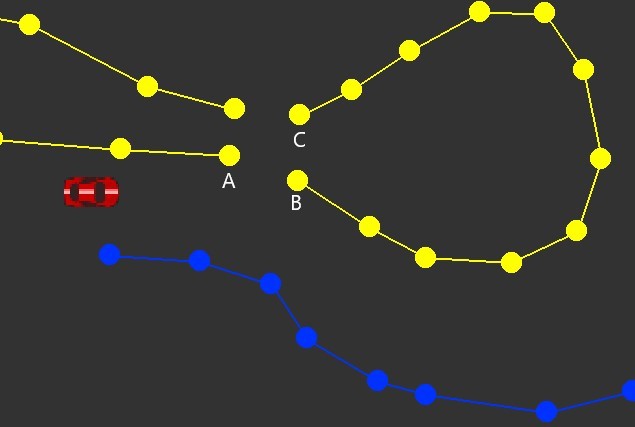
coneConnecting.py is a function class (to be used on Map class objects) for connecting Cone objects to form a track boundary.

The coneConnecter class contains 2 functions: connectCone() and connectConeSuperSimple(). First, connectCone() retrieves a list of nearby cones, filtered to exclude far-away cones and ones that are already doubly-connected. It then calculates a ‘strength’ value for every valid candidate in the list of nearby cones. The strength value is based on a number of parameters: distance between cones, connection-chain length of candidate, the difference between the existing connection angle (if any) and the angle to the candidate and the restrictiveness of the angle (see definition in most-restrictive-angle paragraph). The highest strength candidate is picked unless a candidate with a more restrictive angle exists.

The most-restrictive-angle check is based on the idea that, cones that narrow the track are more likely to be the right cone. For example, if a left-sided cone finds 2 candidates (of similar strength), then it is possible that 1 of those candidate cones is intended to be part of a much later stage of the track. In the example shown below, cone A should connect to cone B, not C, even if C has a higher strength.

connectConeSuperSimple() only considers squared distance and connection-chain-length. This could save a significant amount of CPU resources, as calculating the squared distance is far simpler than calculating distance and angle. At time of writing, this function is not used in any code (as far as I know), because there has been no incentive to it (instead of connectCone()) yet.