A description document that explains your implementation decisions and justifications. The comments you added in your code should serve as a good start point to generate this document. You should describe how to compile and run your code, as part of your explanation. Word processing

(MS Word) or PDF are acceptable file formats for this document.

For my code I have started with it being turn based and will later implement proper timing, doing it this way shows me the intersection at a slower pace to identify any errors as they occur. Each turn is set to be about 1 second so in future scenarios I can have loops to repeat for X amount of seconds

The intersection itself is by defaults a 12 x 12 2d array where every value of the array is a square on the board. The board itself can be either a road, grass/land, lights or intersection and will be. The board itself will have a text document be read if chosen depending on the different intersections, this way multiple different intersections can be used. All the board tells us is what kind of square to draw and each one corresponds to a color in the GUI

The vehicles themselves are handled as a sort of link list where every lane is its own list. This was when we traverse the list from bottom to top we can see if there is a vehicle in the next link, if yes the car doesn’t move. I found it a very simple way to deal with the lanes. It also makes it easier when changing lanes. We just have the desired link be set to the vehicle changing lanes/turning and then set its previous location to be null.

This also gives us an effective way to check the destination link to see if there is a car there.

The most useful thing to come out of this is when debugging since the lanes are all separate, when the car is drawn on the screen if 2 cars inhabit the same square both are drawn and it becomes very obvious with the deformed shape. This has helped me work out exceptions when the cars are driving and would crash.

Continuing my comment previously about turning, because they’re objects and in a link list the buttons that the user has access too become clickable only when the user is able to actually perform the action. Otherwise the button becomes unclickable and the only option left will be to end or go straight. The end turn is useful if they want to wait to turn since they current can not.

The lights themselves are acting as a looped link list. Every turn/second it moves to the next light in the array. The amount of green, yellow and red lights can be set depending on how many seconds/turns you want that light to be active for. It is noted that the seconds of the yellow and green light combined must be equal or less then the seconds/turns of the red light to avoid any accidents.

The user also has 3 options to add their own vehicle they can drive. 3 buttons that become unclickable if any of them are clicked just once. Each one of these is a subclass of vehicle that was defined for the assignment, car, bus and truck. Within their own individual classes, you can set their specific information such as color, size and speed. The color for now as been set to a specific color for each vehicle to help identify which vehicle is what more clearly.

For actually making turns we use a few shared commands, giving the left turn as an example the user advances one square, changes lanes into the new lane, basically turning the car and switches over to the new lane that its turning left into and then now continue straight normally. In order for the car to make a proper left turn without any accidents the left turn button will only be clickable if a specific square in the opposite lane is clear. To help simply this if the user is in the intersection it would take 4 turns to clear the intersection, meaning at the 3rd turn is the risk of intersection, so when the user is at the intersection it will check 3 spaces back from the opposite lane, if its open then the user can turn.

Now to simulate a real intersection every turn multiple random integers are generated, one for if a vehicle will be inserted, one for which type of vehicle it will be and one for which lane it will be inserted on. by being random it will give us different scenarios.

Every turn the CPU vehicles also have options, they can either change lanes or go straight be default until they reach an intersection at which point more options become available, here they can also go left and right. It is noted that to avoid any issues the cpu and driver cant change lanes while in the intersection as although it isn’t illegal it is a very dangerous move that should also be avoided