Design Overview for Traffic Controller

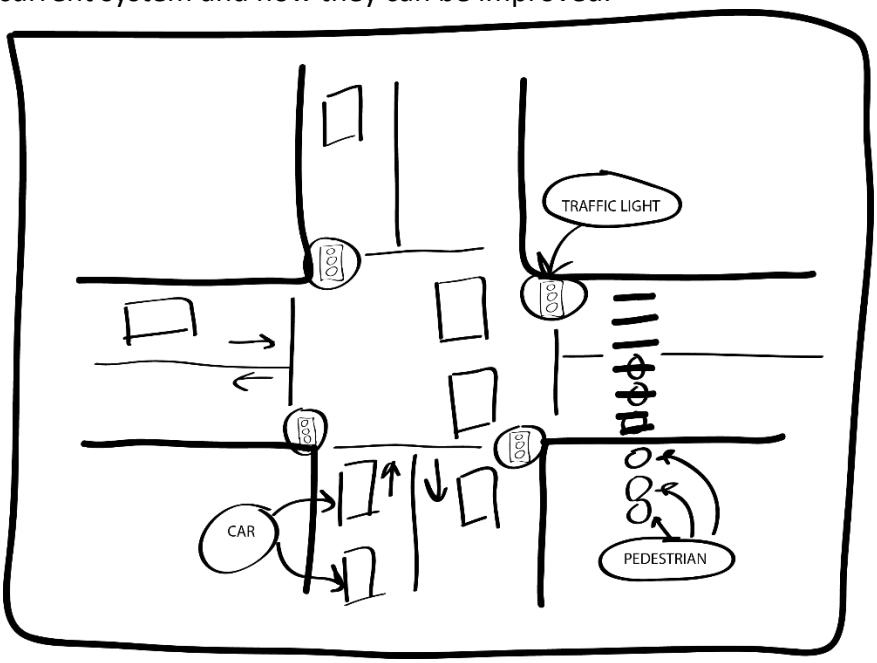
Name: Matthew Coulter
Student ID: 102573957

Summary of Program

My software will be a simulation of a traffic intersection. At this intersection, the player will control the traffic lights to let cars through, and stop other cars. The controlling is done by clicking on the traffic lights which will prompt cars to begin driving for the given lane/s. There will be a anger level for each vehicle, therefore if they ever need to wait for too long, this will cause a fail.
The overall strategy is to ensure the anger levels don't get too high by letting everyone go through. However, the player must ensure vehicles go through at the right time, otherwise this may cause a crash whereby the player will fail and need to start again.
Whilst this game will most likely be in its very basic form, there is lots of room for additional features to be implemented such as:

- Difficulties which can determine the size of the intersection
- Pedestrian crossing
- Alignment to a quick clock to simulate busy and non-busy traffic levels
- A point system people that wait less give more points when they go through
- Each car has its own wait time, the wait time goes down to be converted into points when they go through the intersection
- Use colours to represent wait time the driver's anger level: red = angry (~20 seconds), blue = willing to wait (~1 minute)
- Have a line where if the queue of moving object becomes too long, the game is over
- Have emergency service vehicles who only have 5 seconds to get through
- Buy lanes using the coins, allowing more cars to come through.
- traffic light has minimum time before changing state again

The overall aim of this game is to educate people about how traffic light intersections work, flaws of the current system and how they can be improved.



Required Roles

Table 1: GameObject details

Responsibility	Type Details
Parent class for all object within the game.	- color : Color - p1: Point2D + color : Color

Table 2: MovingObject details

Responsibility	Type Details
Child Class of GameObject, Parent Class of Objects that move such as Car, Pedestrian	- _waittime : int - _speed : int - _currentstate : bool + SpawnObject() : void <<abstract>> + KillObject() : void + Move() : void + UpdateState() : void

Table 3: Car details

Responsibility	Type Details
Child class of MovingObject. Follows the line created in Path and stops and starts according to the corresponding traffic light	-_width:int -_height:int + SpawnObject() : void <<override>> + SpeedUp() : void + SlowDown(): void + checkdistancebetweencar(): void + UpdateDirection() : void + InsideOfAnotherObject(List<List<MovingObject>> allobjects) : bool + Crash() : void

Table 4: Pedestrian details

Responsibility	Type Details
Child class of MovingObject. Function similar to a Car, but is situated on a footpath.	-_radius: int + SpawnObject() : void <<override>>

Required Roles

Table 5: TrafficLight details

Responsibility	Type Details
Child class of GameObject, clicking on it changes between red and green, causing the corresponding vehicles to stop or start respectively.	- _currentstate: bool - _correspondingpaths: List<Path> + UpdateLightColor() : void + UpdateMovingObjectState(): void

Table 6: Path details

Responsibility	Type Details
Is a line that determines the position and direction for MovingObject objects.	-_movingobjects : List<MovingObject> - _p1: point2d (where the car spawns) - _p2 : point2d (where the vehicle is deleted) - _spawnrate : int - _direction: double + CalculateAngle(): double + DrawLine(): void + AddMovingObject(): void