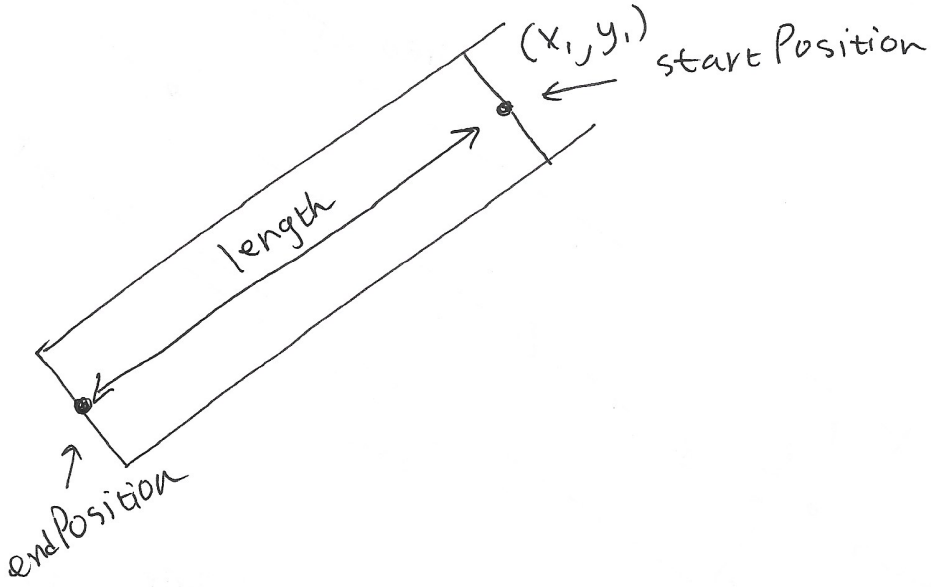
Calculation of Lane endPosition:

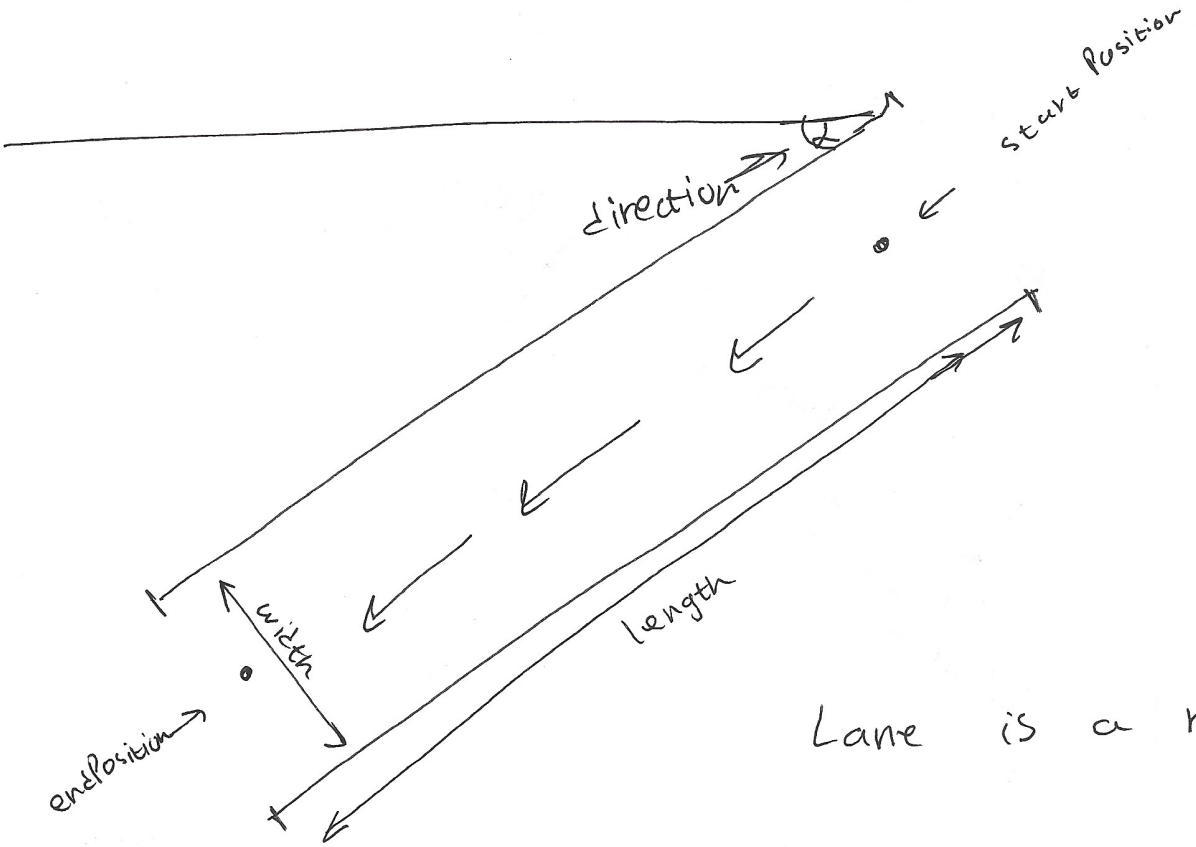


$$\text{endPosition} = \text{startPosition} + \text{lengthVector}$$

CPU: Haswell i5-4460

Network: Realtek RTL8168G/8111G

Lane Example



Lane is a rectangle Shape.

origin = start Position

rotation = direction

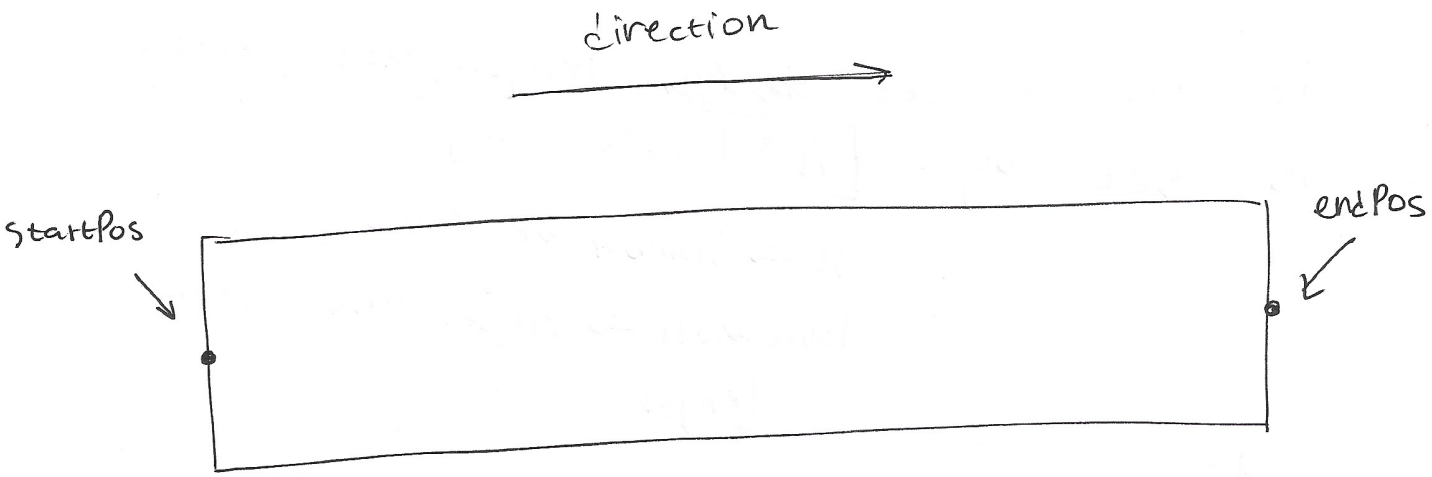
* Each lane is a queue of cars.

- every Time a cars enters it, it gets added
- when a car leaves the lane, it gets removed.

Lane Texturing:

Road color : RGB(44, 47, 50)

White color : RGB(



Road Class

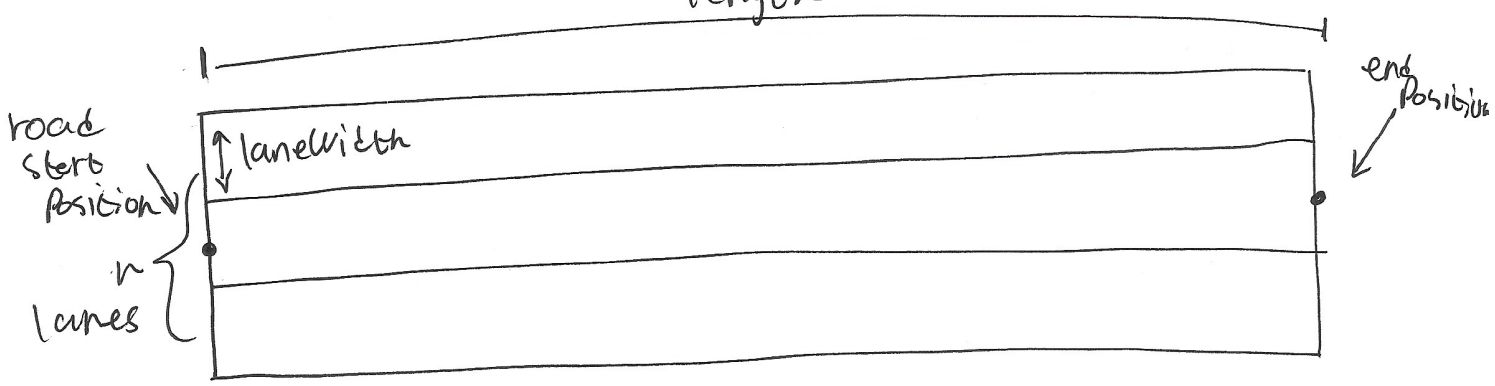
a road contains a few lanes.

from 1 lane, to n lanes.

it has a set ~~length~~ length, but width is set by: `n * laneWidth`

n → number of lanes

laneWidth → Preset lane width



every road has:

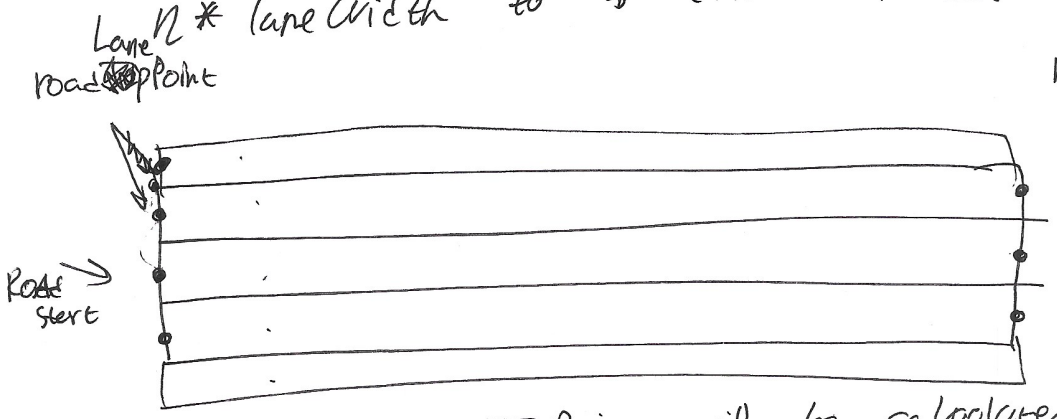
- length
- lane Count
- road Number
- direction (lane direction is inherited) / reversed
- start Position
- end Position
- an array of lanes

Adding a Lane into a road

when a lane is added into an existing road, all the other lanes' positions have to be recalculated.

for that purpose, a function AssignLanePosition will be called.

this function will divide the total width, which is n * laneWidth to n different lanes:

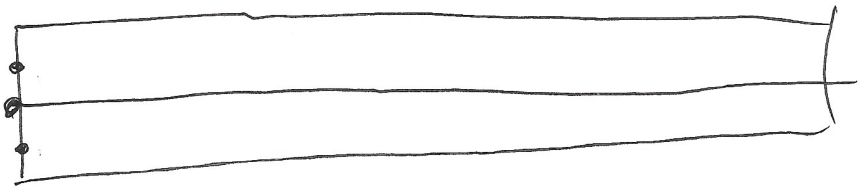


a Vector roadTopPoint will be calculated.

once calculated, the next lanes will start at:

i · laneDifference + firstLanePoint.

2/2 1 1+1.5 = 2.5



Cars

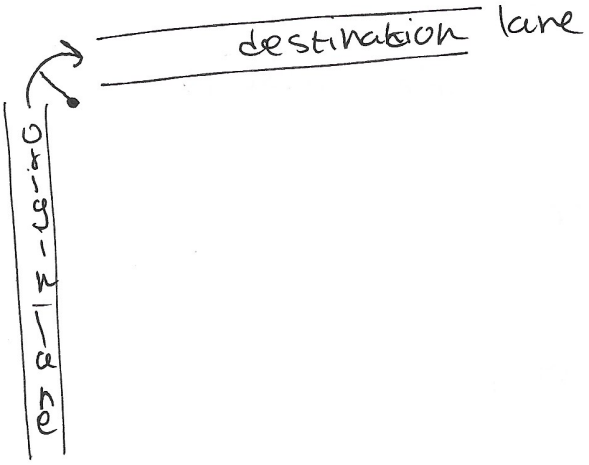
Cars will be created in a certain lane,

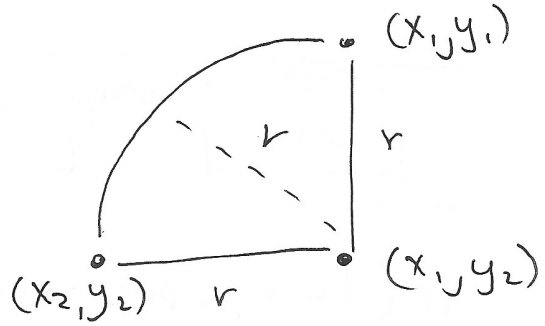
in a → { intersection → addCar (....) method. }

Cars will enter an intersection when they already have a destination-lane.

When a car will arrive to the intersection's center area, it will take a turn that is defined by the start position of the destination lane.

a calculation of the turning radius will be made;



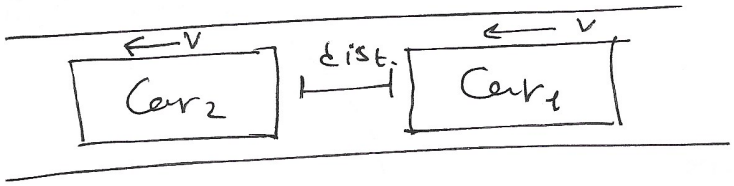


a turning amount will be calculated to that the car will arrive exactly to the dest. lane.

Cars will have multiple states:

```
switch (state){
    case turn: turn in a radius until target rotation is reached
    case drive:
        accelerate until a set max-speed is reached.
    case stop:
        decelerate until a min-speed (0) is reached.
}
```

a car will know when to accelerate and to brake according to the car in front of it:

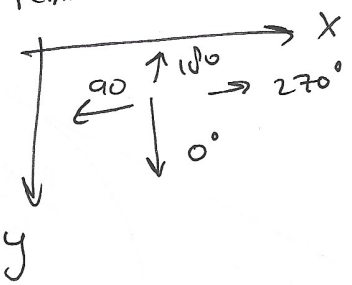


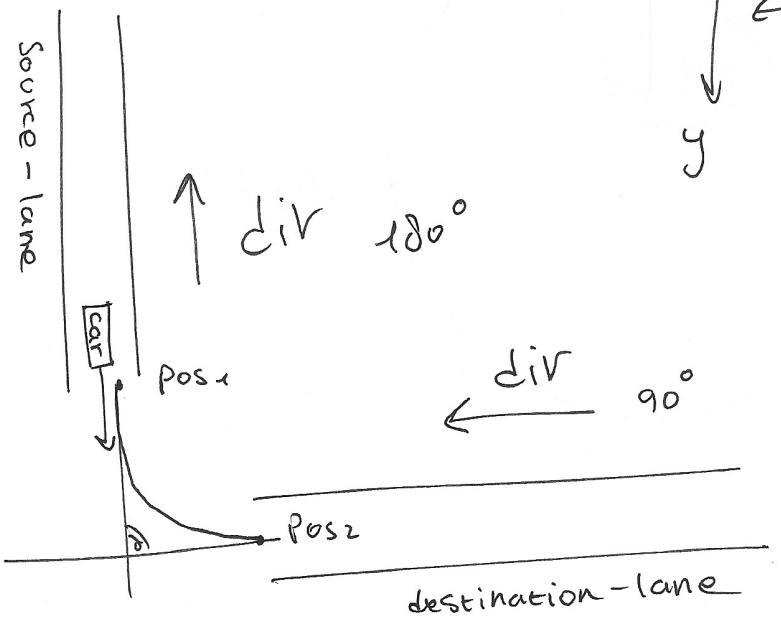

```
if(|Car1.pos - Car2.pos| < min dist){
        car1.state = stop;
}Else{
        car1.state = drive;
}
```

* each car has a pointer to the car in front of it.

Car Turning

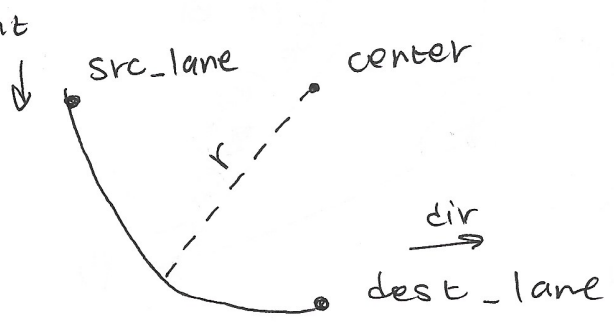
reminder:





target ~~deg~~ direction is | 360-(dest lane. direction)

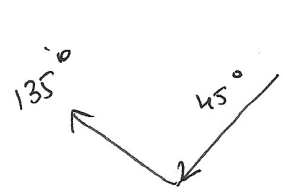
to do the turning, we will create an imagirary center point

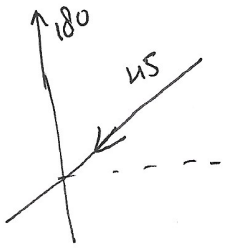


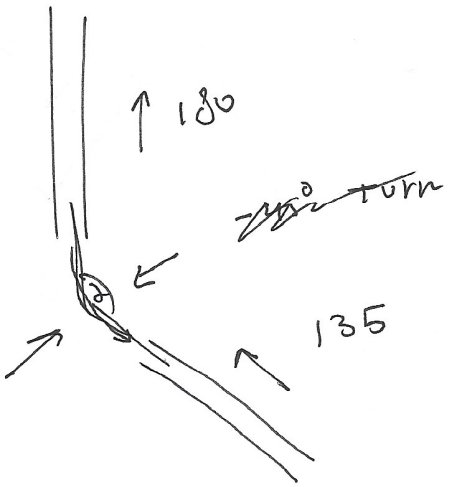
~~the travel distance~~

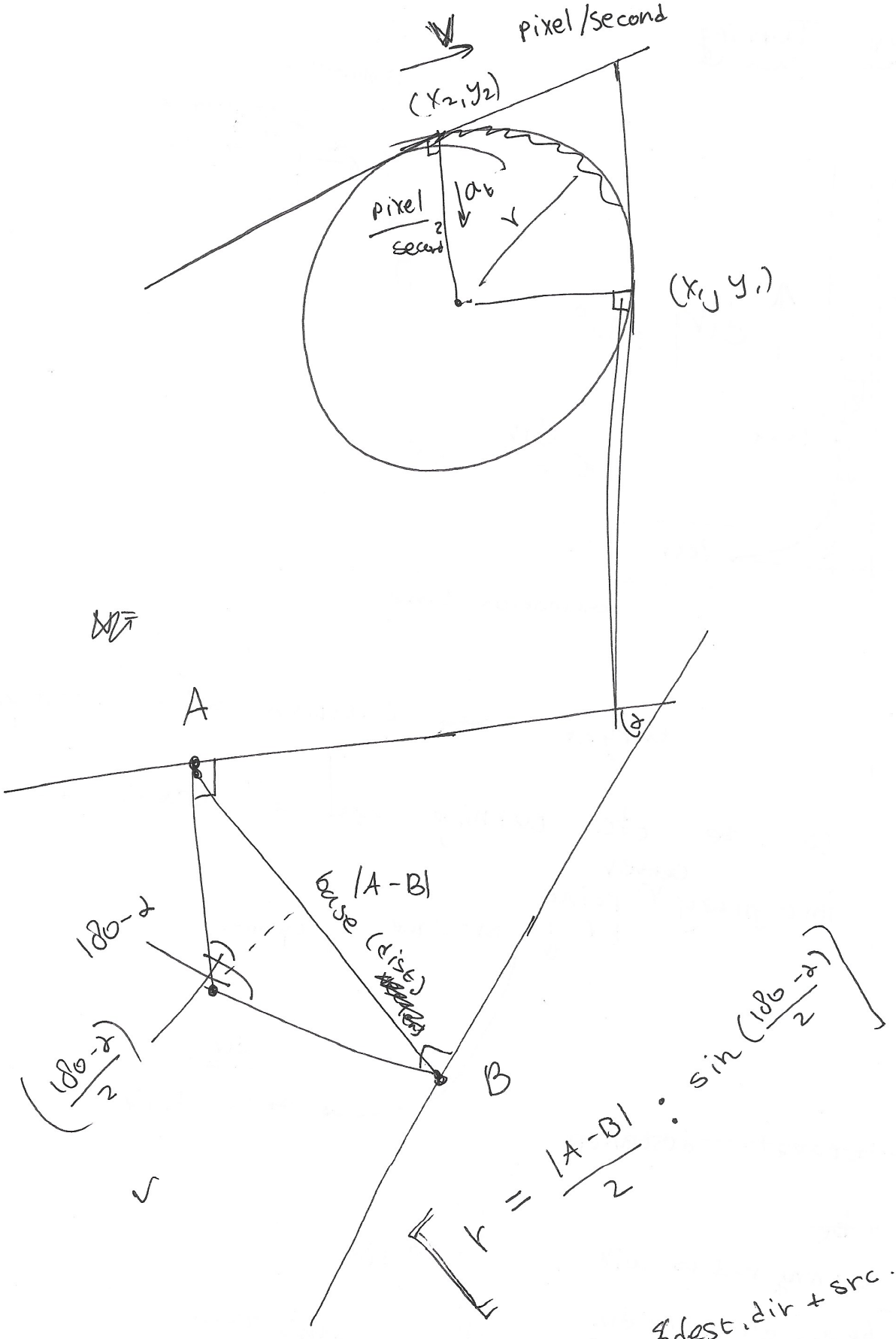
~~will be~~

the turning radius will be 180- dest. dir + src.dir








pixel/second
(x_2, y_2)
pixel / second²
(x_1, y_1)
A
B
α
$180-\alpha$
$\frac{180-\alpha}{2}$
base $|A-B|$
$r = \frac{|A-B|}{2} \cdot \sin\left(\frac{180-\alpha}{2}\right)$
$\alpha = 180 - \angle dest.dir + src.dir$

Braking Cars

First we need to calculate braking distance:

braking distance formula

$d = s^2/(250 * f) \rightarrow d = (s/3.6)^2/(250 f)$

d = distance in meters

s = speed in km/h

f = co-efficent of friction, normaly 0.8 on dry, 0.1 on ice

$$V^2 = V_0^2 + 2a(X - X_0)$$

$$0 = V_0^2 + 2a \cdot d$$

$$2a = -\frac{V_0^2}{d}$$

$$a = -\frac{V_0^2}{2d}$$

combine 2 formulas

$$a = \frac{-V_0^2}{\frac{2 \cdot (V_0/3.6)^2}{250 f}} = \frac{-V_0^2 \cdot 250 f}{2 \cdot (V_0/3.6)^2} = \frac{-V_0^2 \cdot 125 f}{V_0^2/12.96} = -1620 f$$

$$\boxed{a = -1620 \cdot f}$$

Cars have a set deceleration → a.

$$V^2 = V_0^2 + 2a(X - X_0)$$

$$2a(X - X_0) = -V_0^2$$

$$2a \cdot d = -V_0^2$$

$$d = \frac{-V_0^2}{2a}$$

$$\text{minA} = \frac{\text{minA} \cdot \text{weather}}{10}$$

$$\text{max\_speed} = 27 \text{ m/s}$$

$$a = 3 \text{ m/s}^2$$

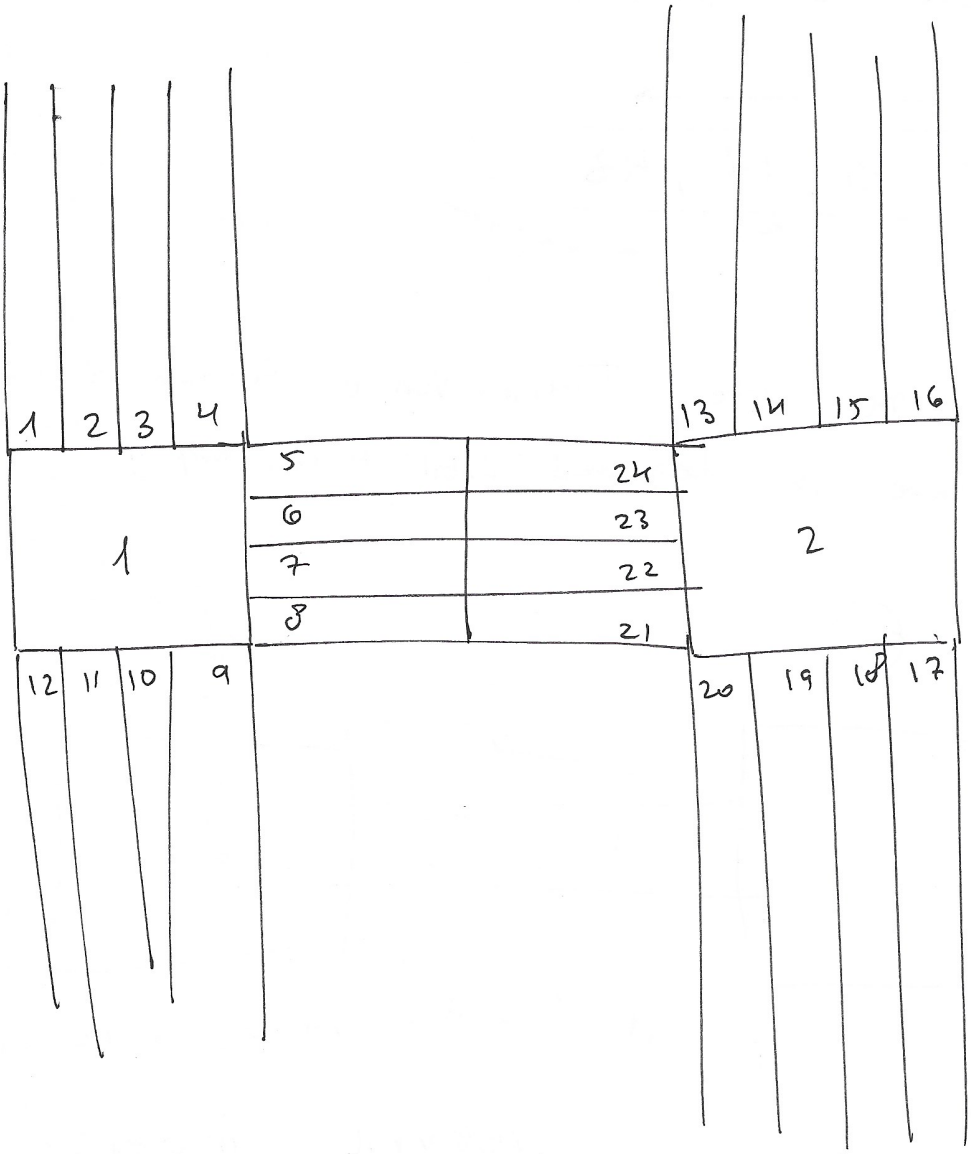
70 px = 1.8m

[1 px = 0.026 m]

12

Maps

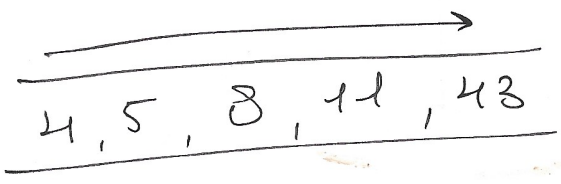
Maps are a birds-eye view of the whole simulation, that may include multiple intersections.



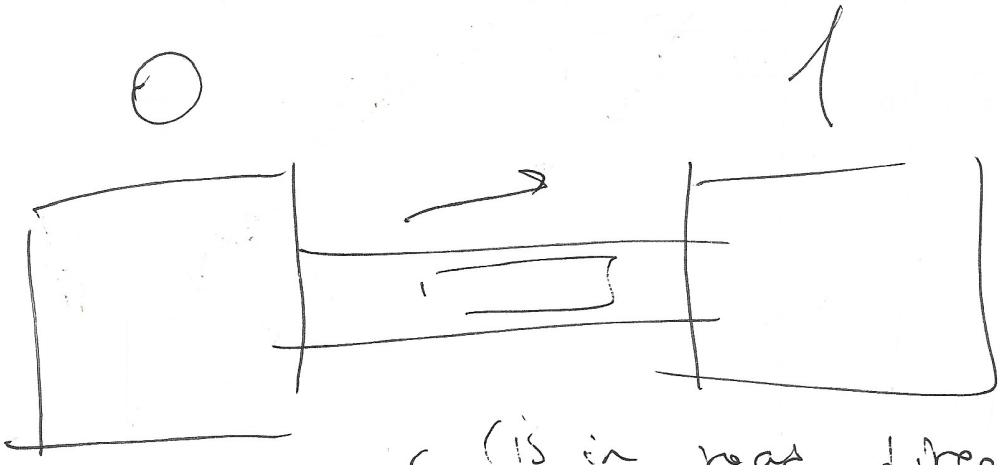
the connection will occur between 2 intersection using a connecting road.

Car's instruction set

in order for a car to do a full circuit in a map, it should have a instruction set: chronological a queue of lanes it should be in, in order:



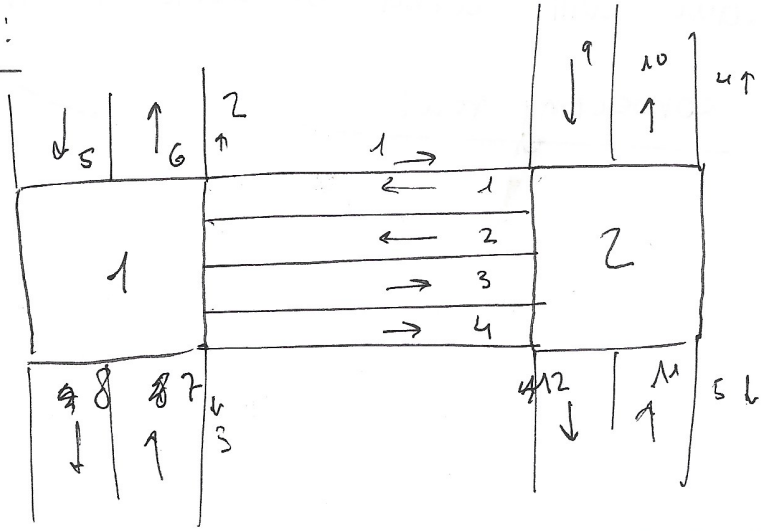
every time the TransferVehicle function is called, the queue is dequeued, until it is empty.



if (is in road direction

return inter[1]

example map:

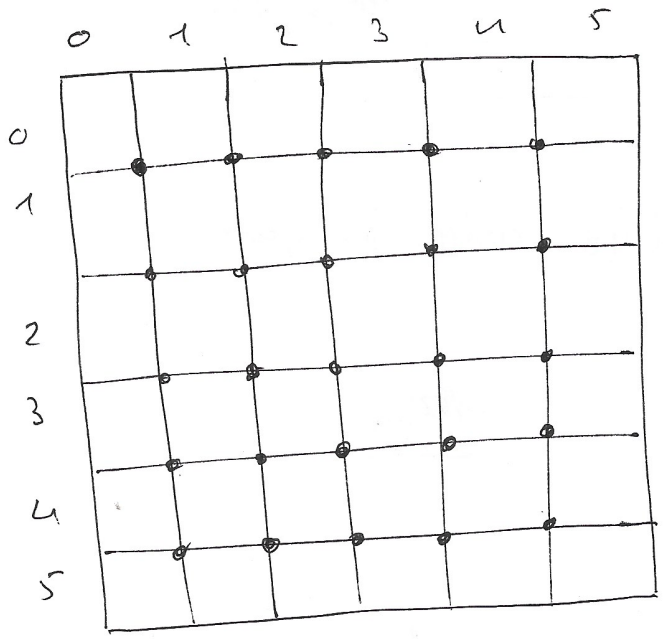


(670, 460)

(710, 580

Snap Grid

in the GUI, there a snap grid option will be available.
When checked, the simulator frame will be a shape-grid,
therefore intersection will be placed in preset positions.



when the frame is clicked on,
the snap grid point closest to
the mouse-pos at that moment
will be selected.

to calculate the closest grid point:

if $(x \,\%\, col\_width) > (col\_width/2)$ → use ceil function.

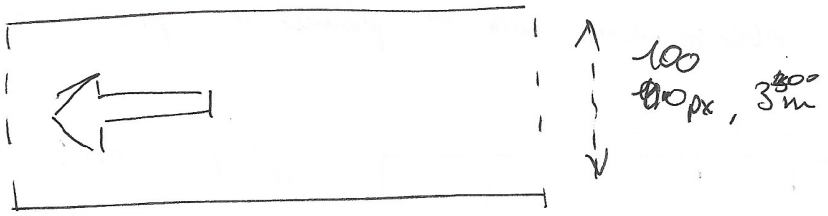
Else → use floor function.

same for the y-axis

Scale

average Lane width =

3 meters = 300 cm



lane width in game:

[1 px = 3 cm]

100 px

every pixel in the game represents 1cm in real life (in base scale)

Therfore, base scale is 3

m → 0.01
cm → 1
ft → 0.0328
inch → 0.3937
px