# CSCI862 Assignment 3

This document is just my musings on what we’re supposed to do and some thoughts on how to do it. Please offer suggestions and corrections! – Sean

**Diagrams not updated since document ver 1.0**

# What does this program do?

The following is my reading of what the program is supposed to do. I’ve also added a couple of extra things that I think will be useful.

## Activity Engine

Based on two sample data files, Vehicles.txt and Stats.txt, the program creates a set of data that simulates “normal” behaviour of traffic on a road.

Output is a human-readable log file (one for each day of the simulation).   
Log filename: BaseLogDay(number)

## Analysis Engine

Then an analysis is performed on the data set and statistics are compiled. Desired stats include:

* Totals for each even for each day
* Means and Standard Deviations (per vehicle type and overall?)
* Anything else that looks interesting

The stats produced here are the baseline stats. That is, normal activity. Anomaly score 0.

## Alert Engine

The user is prompted for a file containing new stats (same format as Stats.txt).

The user is prompted to give a name to the new dataset (no default, insist!) Run input validation on the name because we’ll use the name as a filename!

The activity engine should be run with the new stats file (but the same probabilities (as detailed below)). Log filename: (datasetname)LogDay(number)

The analysis engine then needs to be run again to produce a new set of stats.

## Comparison Engine

We then need to compare the user stats with the baseline stats, calculate anomaly scores and raise alerts if the threshold (2 x (sum of weights)) is exceeded.

A comparison log (datasetname)Comparison.log should be generated in addition to displaying to the screen.

Prompt user for another data file, or quit.

# Design Thoughts

## Probabilities

These will need to be adjusted to create “interesting but realistic” results

|  |  |
| --- | --- |
| **Probability *Per Minute* of** |  |
| Turning off the main road | 1% |
| Changing speed Up | 1% |
| Changing speed Down | 1% |
| Breaking speed limit | 0.1% |
| Parking | 1% |
| Resuming journey (Un-Parking) | 10% |
| Continuing driving | (default action if no event occurs) |

## Default Values

* Road entry speed – 55kph
* Vehicle speed adjustment size – 5kph per event (IE When a vehicle speeds up or slows down, it speeds up by 5kph in Minute 1, another 5kph in Minute 2, etc.)
* Resume speed – 55kph (assume they accelerate quickly back up to 55kph after resuming)

## Files

Vehicles.txt describes the functional characteristics of the vehicles.

* The name (informational only)
* Parking permission
* Format of registration (Used to create Unique ID for identifying vehicles in logs)
* Weighting for Frequency of Appearance
* Weighting for Impact of Speeding

## Potential anomalous behaviour:

* Unusually high or low frequency of parking
* Unusually high or low frequency of changes in speed
* Unusually high or low instances of turning
* Unusually high or low frequency of appearance of a particular vehicle type
* Unusually high or low instances of momentary speeding
* Unusually high or low *average* speed across the length of the road
* Vehicles parking when they’re not permitted to park

## Logging

Log to CSV for easy import into external apps and re-loading into app if necessary.

### Activities for log level NORMAL

* Arrive
* Turning off the main road
* Breaking speed limit
* Parking
* Resuming journey (Un-Parking)

### Activities for log level DEBUG

* Changing speed Up
* Changing speed Down
* Continuing driving

### Activities for log level ALERT

* Vehicle exceeded speed limit
* Vehicle parking illegally

### Sample log format / entries

* YYYYMMDD-HHmm, LogLevel, Rego, Type, Action, AvSpeed
* 20161015-0515, NORMAL, ABC123, Bus, Arrived
* 20161015-0518, NORMAL, ABC123, Bus, Turned
* 20161015-0521, NORMAL, AB44AC, Motorbike, Arrived
* 20161015-0521, ALERT, AB44AC, Motorbike, Parked
* 20161015-0525, NORMAL, Z9BC1J, Car, Arrived
* 20161015-0536, NORMAL, AB44AC, Depart, 58
* 20161015-0537, NORMAL, Z9BC1J, Car, Parked
* 20161015-0541, NORMAL, AB44AC, Motorbike, Resume
* 20161015-0545, NORMAL, AB44AC, Motorbike, Depart

# Objects needed: (List incomplete! Details incomplete!)

## Simulation

* + Road (road)
  + Vehicle [ ] (vehicle)

## VehicleType

* Name (string)
* Parkable (bool)
* Registration Format (string)
* VolumeWeight (int)
* SpeedWeight (int)
* turnProbability (float)
* parkProbability (float)
* speedChangeProbability (float)
* lawBreakerProbability (float)

## Vehicle instanceof VehicleType

* Registration (string)
* CurrentSpeed (int)
* CurrentPosition (int)
* Parked (bool)
* ThisAction (string)
* PreviousAction (string)

## Road

* Length (int)
* SpeedLimit (int)
* ParkingSpaces (int)
* FreeParkingSpaces (int)

## Stats

* Vehicles using road
* Total parkings
* Total turnings
* ?

## Utility

* Filenames
* Etc.

# Events

## Road Events:

1. Vehicle Arrival

## Vehicle Events:

1. Vehicle Turn (exit system via side road)
2. Vehicle Park / Un-park (starts driving again)
3. Vehicle Speed Adjust (up/down – randomly determine the direction, simulates having to slow down for other cars or hazards)
4. Vehicle Drive (no speed adjust)

# Program Logic

## Simulation

Following initial setup, the main program counts out the minutes of the day.

For each loop:

1. A new vehicle may be instantiated.
2. Each existing vehicle object is given a "tick".
3. Tick results are actioned (vehicles are removed, added, speed adjusted, etc.)
4. Loop counter is incremented.

When the counter has reached the end of the specified simulation period, analysis is performed.

#### Tick Action Processing (Loop phase 4)

1. Iterate through each vehicle:
   1. If current Action == Turn, log it and destroy vehicle object
   2. If current Action == Parked && PreviousAction != Parked && FreeParkingSpaces != 0, decrement FreeParkingSpaces else set Action = Previous Action
   3. Check if vehicle is permitted to park. If not, lot it, raise an anomaly counter
   4. If (CurrentAction == Drive || Speed) && PreviousAction == Parked, increment FreeParkingSpaces
   5. Calculate new road position using vehicle.getCurrentSpeed()
   6. Compare current speed to speed limit. If speed > limit && type != emergency, log it, raise an anomaly counter
   7. If new position > road.length, log Depart and destroy vehicle object
   8. If new position !> road.length, set new position

## Vehicle

On receiving a tick, each vehicle may perform a single Action. An Action is a ***potential*** Event

Generation of events is a calculated by a random number (RN) compared to a probability value (P). If the probability factor exceeds the random number, the event occurs. Random numbers are generated once per object for each "tick".

E.g.

RN = 0.034, P = 0.02 RN < P so Event occurs.

RN = 0.783, P = 0.02 RN > P so Event does not occur.

1. Vehicle chooses which Action to attempt
   1. (simple 1-4 random, drive/turn/park/speed)
2. RN is generated
3. Comparison to Probability is made
4. If successful, the Action is realised as an Event and values are set
   1. In the case that a non-parkable vehicle wants to park, another test against lawBreakerProbability must be made. Otherwise vehicle just continues driving.
   2. If an increase of speed would exceed the speed limit for this road, and the vehicle’s WeightSpeed > 0, another test against lawBreakerProbability must be made. Otherwise speed caps at speed limit.
5. If unsuccessful, the Action is ***not*** realised and values are set using the previous Action

*Probabilities are listed in table on page 2.*

# Things to Do

We should:

* After coding to work with the provided vehicle and stat files, we should create our own files and ensure that the results have the same feeling. I promise that the Professor will not be using files identical to those in the assignment brief when marking the assignment!

## Sample Vehicle attribute values

Vehicle attributes signal ***intention to change***. Main simulation loop does the actual changing. Thus at the end of a vehicle’s tick, the position has not changed.

|  |  |  |
| --- | --- | --- |
| **Vehicle Attribs on Event** | **Values at Tick\_Start** | **Values at Tick\_End** |
| **Registration** | ABC123 | ABC123 |
| **Current Speed** | 40 | 45 |
| **Current Position** | 558 | 558 |
| **Parked** | False | False |
| **Previous Action** | Drive | Drive |

|  |  |  |
| --- | --- | --- |
| **Vehicle Attribs on No Event** | **Values at Tick\_Start** | **Values at Tick\_End** |
| **Registration** | ABC123 | ABC123 |
| **Current Speed** | 40 | 40 |
| **Current Position** | 558 | 558 |
| **Parked** | False | False |
| **Previous Action** | Drive | Drive |

After all vehicles update their intended actions in loop Phase 2, the main loop processes the Events. Thus a vehicle’s position is only updated during loop Phase 3.

|  |  |  |
| --- | --- | --- |
| **Vehicle Attribs after Loop** | **Values at Loop\_Start** | **Values at Loop\_End** |
| **Registration** | ABC123 | ABC123 |
| **Current Speed** | 40 | 45 |
| **Current Position** | 558 | 1308 |
| **Parked** | False | False |
| **Previous Action** | Drive | Drive |



