A Simualtion Study of Left Turning Movement

at an Unsignalized Intersection

by

Wuyen Wayne Ni

Project and report submitted to the Faculty of the

Virginia polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Civil Engineering

APPROVED:

A.A. Trani, Chairman

R.D. Walker

L.P. Rees

March, 1992

Blacksburg, Virginia

Acknowledgment

I would like to express my sincerest thanks to Dr. A.A. Trani for his guidance to my work in this report. Also, without his support and patience , I would not have an opportunity to stay in school and finish this project.

In addition, I would also like to thank Dr. R.D. Walker and Dr. L.P. Rees for serving on the Committee and for their encouragement and suggestion which helped me in the successful completion of this project.

I should also give my deep appreciation to Mr. Wei Tang, Mr. Fu-Sheng Ho, Mr. Kuang-Chao Yu, and Mr. Chung-Shy Liu for their encouragement and provision of computer equipments and knowledge. I also got support from friends all over the places. The list is too long to name all of them, but they can never be forgotten from my heart.

Finally, I would like to thank my parents for their support in every which way. This report is my dedication to them for their ultimate love.

Table of Contents

List of Figures

List of Tables

Chapter I Introduction

1.1 Introduction

Delay is a measure of driver discomfort, frustration, fuel consumption, and lost travel time. The left turning movement at an unsignalized intersection has always been a contributing factor to traffic delays because the turning vehicles have to yield to the through traffic. Clear sight then is a major factor for the turning vehicles to determine when and whether to slow down or move on.

In many instances, the drivers of the turning vehicles do not get the necessary clear view to judge whether cars on the opposite lanes are potential conflicts. Even for the very first vehicle waiting to execute a turn, the driver may have his view blocked by opposing left turning vehicles. The problem may be aggravated when the opposing vehicles are very close and also turning. For example, when two opposing left turning vehicles are waiting at the intersection, both drivers will experience some difficulty to see a section in the through lane twenty-five to one hundred feet back from the intersection. However, when vehicles are spaced farther apart, the drivers will have a better view by looking through the gaps left between adjacent vehicles. Not having a clear view to some critical section in the through lane , the drivers are in a more dangerous situation when vehicles are moving than they are idling.

After a turning vehicle has passed a critical point in its lane, it may not be able to have enough time to come to a full stop to avoid a collision with the vehicles in the through lane unless there is enough gap for the vehicle to cross the through lane. There is a dilemma zone for the turning vehicle. This zone begins at the point where the turning vehicle will have a collision with the through traffic due to insufficient safe braking distance and the zone ends at the point where the turning vehicle can safely cross the through lane. Within the zone, the turning vehicle neither can have a full stop nor is it able to cross the through lane without a collision. The critical point mentioned earlier then must be the beginning point of the dilemma zone. We define this point as the "point of no return" and is abbreviated as PNR in this study.

When a vehicle arrives at the PNR, the driver should have a clear view to be able to see a potential "danger zone" in the through lane and make judgement whether to slow down or move on. If the driver has a clear view, and there is no vehicle in the through lane or there is enough gap for his vehicle to cross the through lane safely, he moves on. However, when there are vehicles in the through lane, if the driver has a clear view, but there is not enough gap to cross the through lane, he will brake and come to a full stop to avoid a collision. On the other hand, if the driver's view to the "danger zone" is blocked, he will have to decelerate just in case there are vehicles in that critical zone, and he will come to a full stop as if he could see vehicles in the critical zone. Even there are no vehicles in the critical zone at this time, it is safer to slow down than to move on. The driver can re-assess the possibility to cross the through lane at a later time. If the driver gets a clear view at a later time, he can make a second judgement using the same rules and either move on or slow down. Following the same rules, the driver will either complete the left turning movement safely or stop completely at a safe spot.

When the vehicle stops at the safe spot and waits for a gap to cross the through lane, the driver will look at a different "danger zone" since the vehicle will start up and move at a lower speed; this, in general, requires more attention to avoid collision. Both definitions of the "danger zones" are further described mathematically in Chapter III. Under both conditions, there will be a time loss due to the acceleration lag of the vehicle. There will be also a time loss due to deceleration. At most signalized intersections, there would also be time losses for clearance but this type of time loss will not be discussed here since this study aims at unsignalized intersections.

1.2 Assumptions

This study will use a simulation model to find the time loss due to deceleration and acceleration. Some conditions are simplified in the study. The following paragraphs discuss these conditions.

1. The configuration of the intersection is idealized by having four straight lanes in each approach, with two lanes in each direction of the traffic flow.

2. The crossing lanes are perpendicular. If there is a curved leg in any approach, the situation is more complex and it may help increase or reduce the blocking of view time.

3. Equal number of lanes in each approach simplifies the computation.

4. The lanes are of equal widths.

5. The left lane is for left turning only but it does not have a turning bay.

6. No vehicles will change to its neighboring lane to increase the queue length nor vehicle will pull out to decrease the queue length.

7. In the through lane, all vehicles stay in the same lane. Also assumed is that there are no vehicles turning right at the intersection.

8. The turning vehicle comes in to the scenario at a reasonable speed. The driver will keep the same speed unless he decides to slow down, because he has a blocked view to the "danger zone", or he sees no opportunity to cross the through lane, or he has to avoid a collision with the vehicle immediately in front of his. Since the vehicle is prepared to make a left turn, it is assumed that there is no acceleration for the turning vehicle.

9. Usually there is a slight braking in the turning movement for comfort reasons. It is omitted since the turning vehicle is usually moving at a fairly low speed.

10. Since some oversized vehicles are prohibited during a certain period of time of day, the study concentrate in regular-sized vehicles only; trucks, campers, or vehicles carrying oversized loads are excluded.

11. The road has a grade of zero in each approach so as not to have vertical alignment problems.

12. Each driver has normal ability to make operate vehicle and to make traffic judgement. No atypical drivers are present.

13. Each driver has some variance to the gaps tolerated between vehicles.

14. There is no signalization at the intersection. There will be stop signs on the minor streets into which the left turning vehicles are turning. Full green time is given so that no clearance time will be lost. No vehicles will have to brake shortly after they start to move. No vehicle will increase speed after "second judgements" when the driver's view has been temporarily blocked but only maintain the same speed. Even though it is safe to cross the through lane, the vehicle does not increase speed.

15. The downstream of the left turning movement is clear so that there will be no bottleneck near the intersection.

16. There is no lateral sight obstruction for the driver.

The above assumptions are made for the purposes to simplify the model and to meet the prevailing traffic conditions.

Chapter II Literature Review

2.1 Definitions of Delay

Delay can be defined in a number of ways as follows [1].

1. Stopped delay: the amount of time of a vehicle spends stopped while traversing a given segment of highway facility.
2. Approach delay: stopped delay plus the time lost in decelerating before stopping and in accelerating after stopping.
3. Travel time delay: the difference between some predetermined optimum desired travel time for a given segment of highway and the actual time. It includes stopped delay and delay due to a slower speed than desired. Note that it covers the time lost due to a slower speed and it does not necessarily imply the vehicle will stop.
4. Time in-the-queue delay: the time lost in the waiting line while the vehicle is idling. In the Sagi-Campbell method [2], queue lengths are observed at specified times in each cycle.

5. Percentage of vehicle's stopping: the average percent of vehicles in the queue due to the inability to pass.

2.2 Delay Measurements

Methods for field measurement of travel time and delay are varied. The travel-time study, when adapted to intersection delay, measures the travel time from a point in advance of the intersection to a point in or beyond the intersection one or more approaches. The direct methods used to obtain the data are as follows [3]:

1. Test vehicles operated between key points.
2. License plate numbers and times recorded at key points.
3. Time-lapse photograph taken from a vantage point to permit the timing of each vehicle shown on the film.
4. Strip-chart records actuated by road tubes or observers operating switches.
5. Observers stationed at an advance point tracing individual vehicles through an intersection and recording time at critical points.
6. Modeling method. One or more theoretical assumptions are used to create the formulas used in place of, or as an element of, field observations.

All of these methods may require extensive personnel or time for the collection and analysis of the data. For example, the Berry-Van Til method [4] is one to measure stopped delay, which is periodically sampled.
In the point sampling method, some parameters such as stopped vehicles are observed at regular intervals. The volume density method can be used to determine travel time delay.

In a report by Reilly and Gardner [5], analysis of intersections was made resulting in a number of observations and recommendations, including:

1. The estimate of stopped delay by a "point sample" method was recommended as the basic field procedure, due to relative ease of use and precision.
2. For field studies of stopped delay by a "point sample" method, the field value should be multiplied by 0.92 to provide a more accurate estimate of true stopped delay, due to a skew in the measurement methodology.
3. The estimate of percentage of vehicles stopping was recommended as a useful distinct observation, with a corrective factor of 0.96.
4. The "path trace" method--while accurate--needs very high sample sizes to achieve reasonable results (1200 to 2700 samples) , precluding it as a practical technique.

5. The entire approach should be studied as an entity in most cases, rather than lane-by-lane, due to "numerous complicating factors that increase manpower requirements and reduce the reliability of the study results."

The modeling method is not treated in this report due to its "esoteric and difficult" nature.

2.3 Delay at Intersections

Some research has been done on the estimate of the delay at intersections. However, most of the existing methods attempt to calibrate their models based on the model in the Highway Capacity Manual (HCM) [6], which was first published in 1958 and later revised in 1965 and 1985.

According to the HCM, 1985, level of service at intersections is measured based on the average delay per vehicle. For the delay at signalized intersections, it is found to be

$$\text{(Eq. 1)}\quad d = 0.38 * C * (1 - g/C)^2 / [1 - (g/C) * X\} + 173 * X^2 * [(X - 1) + (X - 1)^2 + (16 * X/c)]^{0.5}$$

where

d: delay(sec/veh),

X = v/c ,

C: cycle length(sec),

g/C = green ratio,

v: flow rate(veh/hr), and

c: capacity(veh/hr).

Unlike at signalized intersections, the level of service at unsignalized intersections is measured based on the reserve or unused capacity of the approach. Unfortunately, there is no formula available for the estimate of delays at unsignalized intersections. This report attempts to find the delay value of the first vehicle in a left turning lane at an unsignalized intersection.

Chapter III Methodology

3.1 The Configuration of the Intersection

The configuration of the intersection is described as follows. Refer to Figure 1. The origin is set at the center of the intersection. The division lines in each approach form the X-axis and the Y-axis; with the east-west bound line as the X-axis, and the north-south bound line as Y-axis. The width of each lane, L, is assumed to be twelve feet.

Lane A and Lane D are the through lanes. Lane B and Lane C are the left turn lanes. Vehicles in the Lanes A, B, C, and D are referred as to Cars A, B, C, and D, respectively. The left turning movement of Car C follows a circular path with a radius same as the turning radius, R, assumed to be 33 feet, and is turning left to Lane F. The center of the circular path is located at (-c , -c), where

(Eq. 2) $c = R - 0.5 * L$.

Similarly, Car B's are turning left from Lane B to Lane G following a circular path with the center located at (c ,

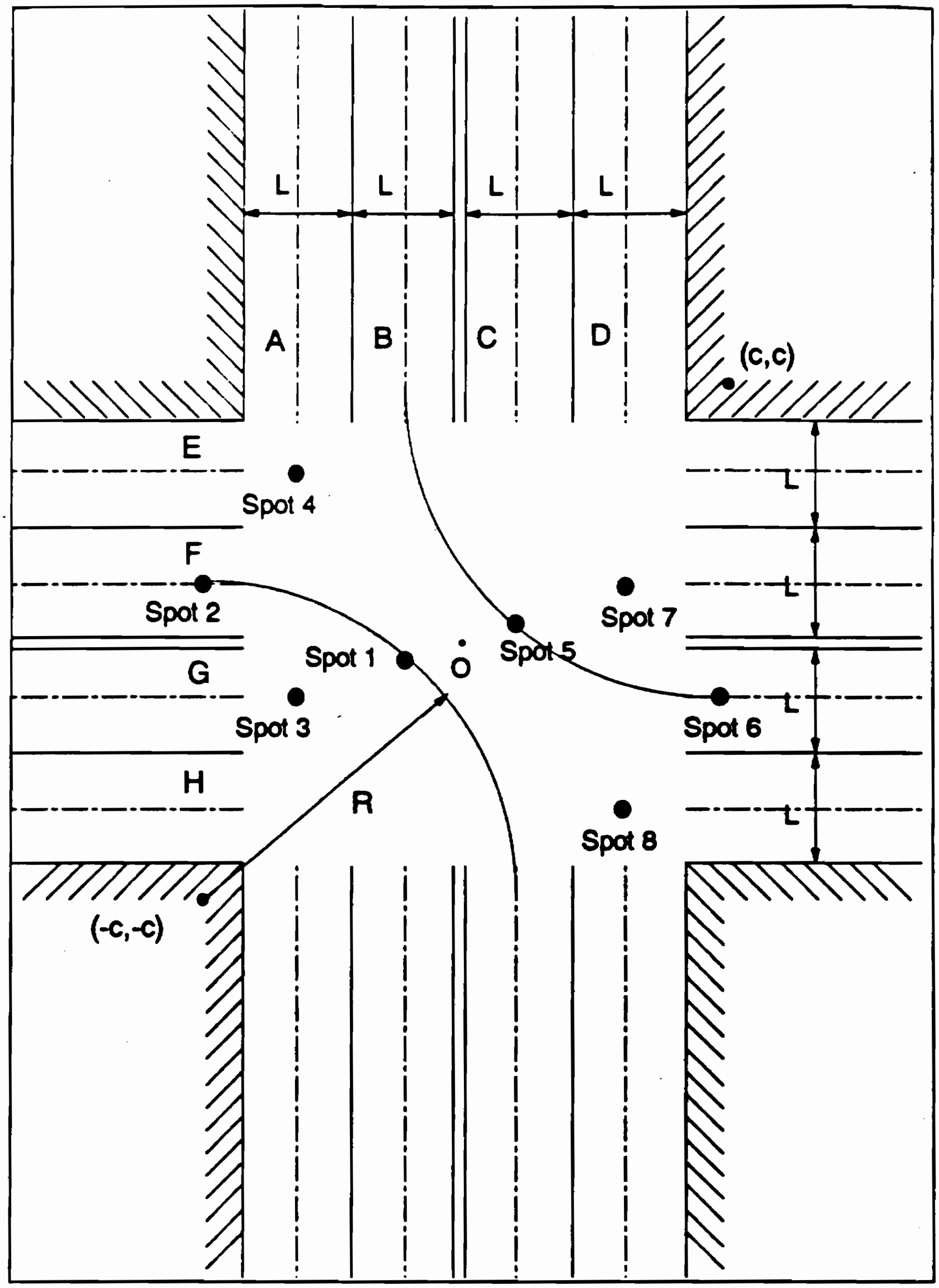

Figure 1. The Configuration of the Intersection

c). Car A's are moving straight south bound. The vehicles in Lane A and Lane B at random arrivals are ejected from 250 feet back off the X-axis and so is Car C.

3.2 The Definitions of Safe Spots

As described in Chapter I, and also referred to Figure 1, Car C should be able to have a full stop at some safe spot in order to avoid a collision with Car A. When the center of Car C is at the intersection of the circular path and the central line of Lane B, it is safe to say that no vehicles in Lane A will have a collision with Car C. By a definition as such, there will be enough margins for the vehicles in Lane A not to have a collision with Car C. This safe spot is called Spot 1.

There is a second safe spot to be defined. After Car C has completed its movement, the driver wants to make sure that there will be enough margin from his car to Car A. When Car C has just finished its left turning movement, it will be at some point in Lane F. Let that point be called Spot 2. It is the end of the circular path that the vehicle follows in the left turning movement and also is on the center line of Lane F.

For the through traffic, there are two more safe spots to be defined. Spot 3 is the point where an Car A has just passed the conflict point which is the intersection of the circular path and the center line of Lane A. Mathematically, Spot 3 is defined as the intersection of the center line of Lane A and the center line of Lane G. Having Car C being at Spot 1 and Car A being at Spot 3, the front of Car C and the tail of Car A type of collision can be avoided.

When Car C is at Spot 1, there is a safe Spot 4 for Car A to occupy so as to avoid the front of Car A and the front of Car C causing a collision. Also, when Car C is at Spot 2, the front of Car A and the tail of Car C type of collision can be avoided. This point is defined as the intersection of the center line of Lane A and the central line of Lane E.

(Eq. 3) Spot 1 $(x, y) = (-0.5 * L, q)$,

(Eq. 4) Spot 2 $(x, y) = (-c, 0.5 * L)$,

(Eq. 5) Spot 3 $(x, y) = (-1.5 * L, -0.5 * L)$,

(Eq. 6) Spot 4 $(x, y) = (-1.5 * L, 1.5 * L)$,

where

(Eq. 7) $q = [R^2 - (0.5 * L - c)^2]^{0.5} - c$.

3.3 The Definition of Point of No Return

As mentioned in Chapter I, the PNR is a critical point. Upon arriving at the PNR, the driver of Car C should make a judgement whether there is enough gap for his vehicle to cross Lane A and complete the left turning movement; that is, to pass Spot 2. If the gap is not sufficient for the vehicle to do so, Car C needs to travel a minimum braking distance before coming to a full stop at Spot 1. Assuming that Car C brakes at a constant deceleration, the time for braking is proportional to the speed of the vehicle. In other words, at a higher speed, the vehicle will travel a longer distance in longer time. In addition, there is a minimum PIJR[1] time for the driver to make judgement. The total braking distance for Car C to reduce from a speed of V_c to zero is calculated by the following equation:

(Eq. 8) $d_{Braking} = V_C{}^2 / 30 * (f \pm g)$,

where

---

[1]The PIJR time is the minimum time of reaction to external stimuli. P stands for perception, I for identification, J for judgement, and R for reaction. Also called PIEV (Perception, Intellection, Emotion, Volition) time.

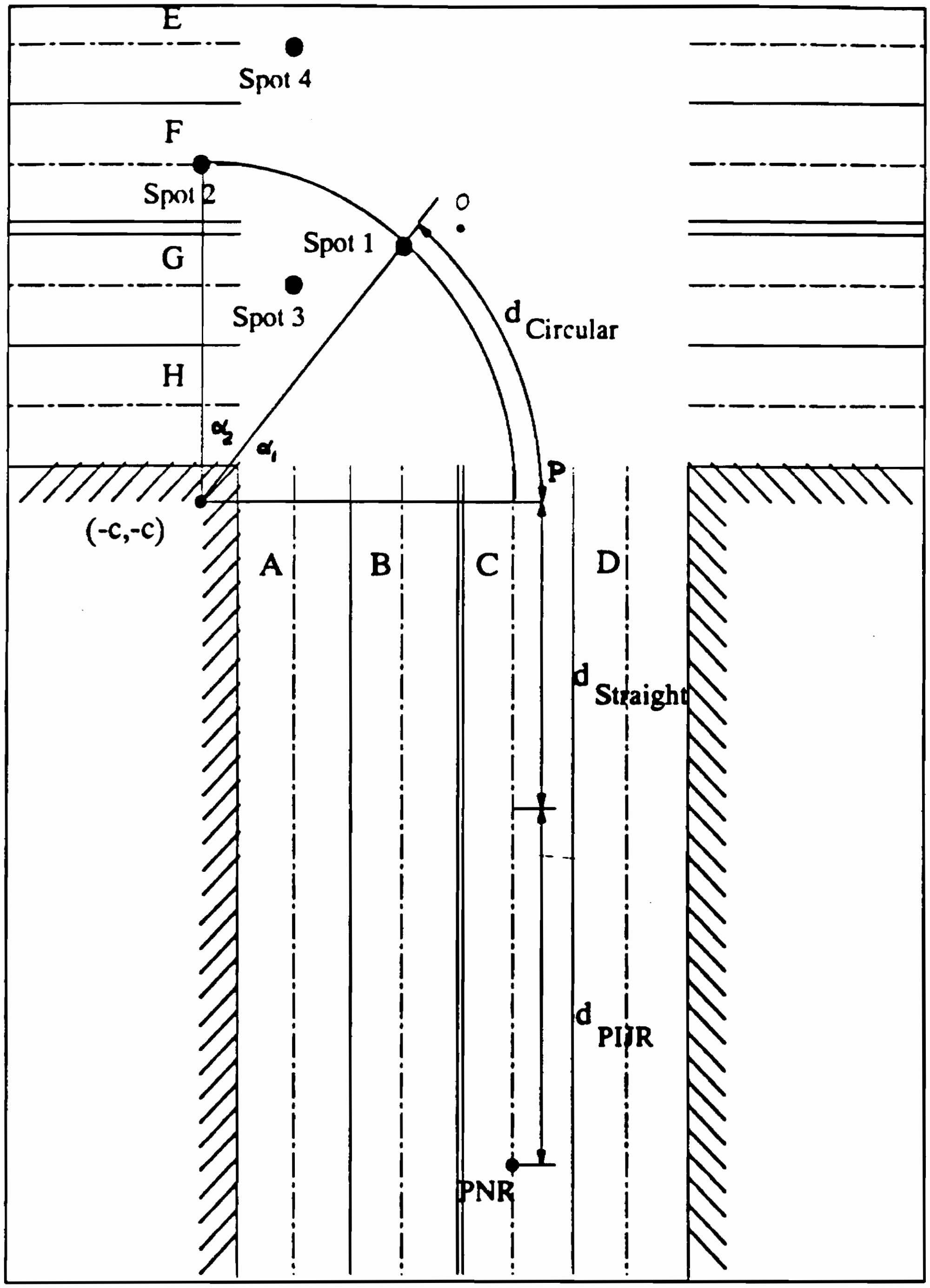

Figure 2. The Illustration of Point of No Return

V_C is the running speed of Car C, (mph),

f is the friction between the vehicle and the road surface, as in the AASHTO book [7],

and g is the grade of the road.

The distance that Car C traveled during t_{PIJR} is

(Eq. 9) $d_{PIJR} = 1.47 * V_C * t_{PIJR}$.

The sum of the terms on the left-hand side of the above two equations is the total distance the vehicle traveled during the time starting from making judgement to the time the vehicle has a fully stop at Spot 1. Note that part of the breaking distance is on the circular path.Also note two seconds of tPIJR is used[7]. Refer to Figure 2.

Let the starting point of the circular path be P. The circular distance from Spot 1 to P is

(Eq. 10) $d_{Circular} = R * \alpha_1$.

Therefore, the distance the vehicle traveled in the straight Lane C is

(Eq. 11) $d_{Straight} = (d_{Braking} + d_{PIJR}) - d_{Circular}$.

Thus, the point measured at a distance of($d_{Straight}$ + d_{PIJR} back from P will be the PNR.

3.4 The Definition of Danger Zone 1

The PNR is the ultimate point to start making a judgement and braking, if necessary. Nevertheless, the driver does not have to brake as long as he finds appropriate gap to cross Lane A. If there is any car within a zone of Lane A such that it will collide with Car C a few seconds later, Car C has to brake. On the other hand, if there is no Car A in this zone, Car C will move on. That is to say that Car C will have sufficient gap to cross Lane A $t_{Turning}$ seconds later, where

(Eq. 12) $t_{Turning} = (R * (\pi / 2) + d_{Straight} + d_{PIJR}) / (1.47 * V_{C)}.$

On the other hand, if there are vehicles in between the beginning and the ending points of the critical zone, Car C should make a judgement as to stop at Spot 1 in $t_{Braking}$ seconds. Thus, referred to Figure 3, the section of Lane A between the beginning and the ending point of the critical zone is defined as Danger Zone 1.

The beginning point is actually the position that a Car A1 is currently occupying for to arrive at Spot 3 $t_{Breaking}$

seconds later. The distance Car A1, moving at V_{A1}, traveling back from Spot 3 for $t_{Breaking}$ seconds, and is

(Eq. 13) $d_{Spot3} = 1.47 * V_{A1} * t_{Breaking}$.

The ending point is actually the position that a Car A2 is currently occupying for to arrive at Spot 4 $t_{Turning}$ seconds later. The distance Car A2, moving at V_{A2}, traveling back from Spot 4 for $t_{Turning}$ seconds is

(Eq. 14) $d_{Spot4} = 1.47 * V_{A2} * t_{Turning}$.

The position of Danger Zone 1 is relevant to the running speeds of Cars A1 and A2. Driver C will check if there are vehicles in this zone.

When the driver's view is blocked by the opposing left turning vehicles, sometimes he can not see anything in Danger Zone 1; he can not tell the beginning point of the zone--- which is calculated based on the speed of Car A. We shall use an average operating speed, V_{Oper}, instead of V_{A1} and V_{A2}, to define the location of Danger Zone 1 since it is

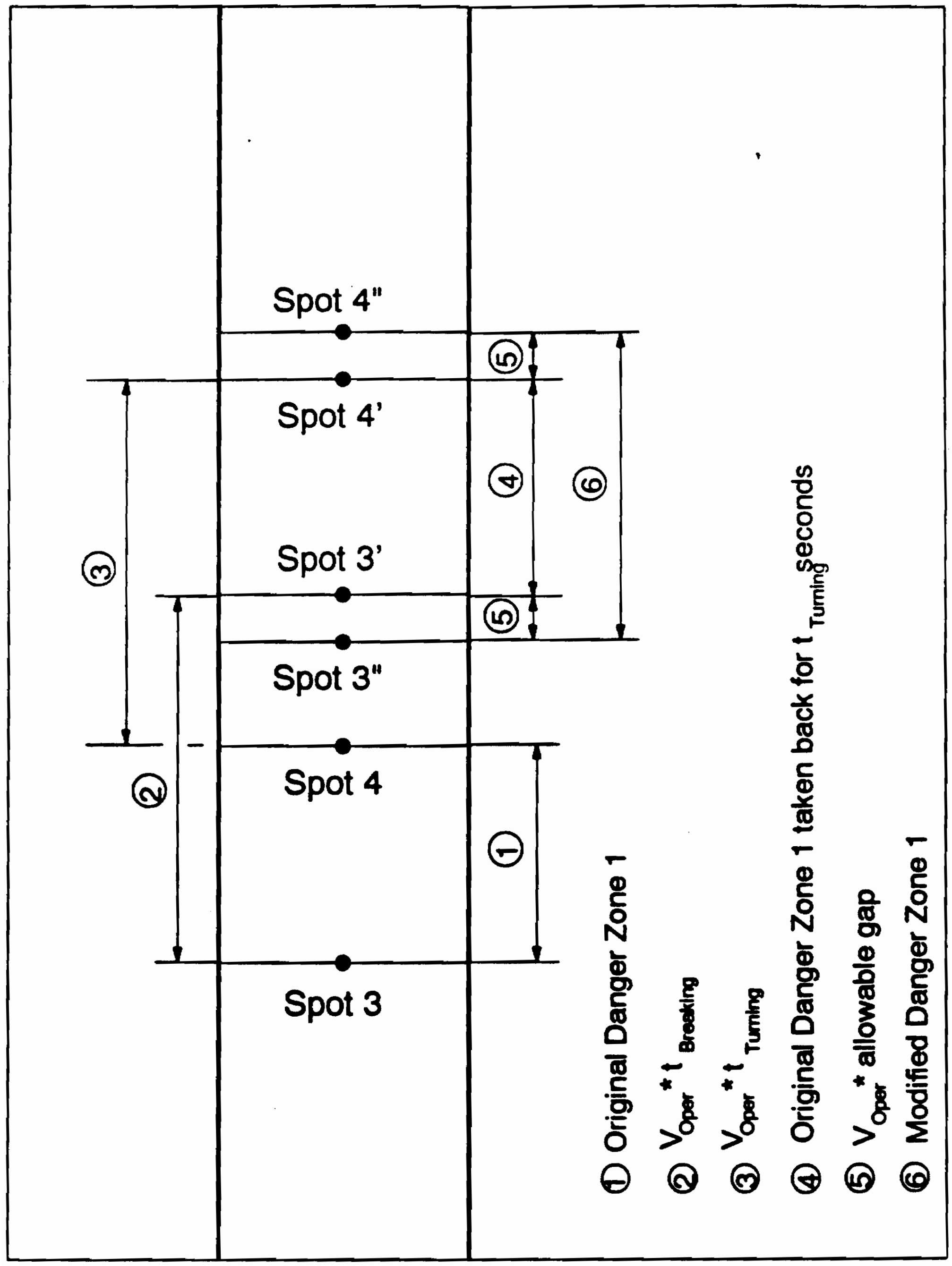

Figure 3. The Illustration of Danger Zone 1

difficult for Driver C to determine the speed of Car A. Now, the beginning and ending points of Danger Zone 1 are

(Eq. 15) $Spot3'_1(x\ ,\ y) = (-1.5 * L\ ,\ -0.5 * L + V_{Oper} * t_{Breaking}),$

(Eq. 16) $Spot4'_1(x\ ,\ y) = (-1.5 * L\ ,\ 1.5 * L + V_{Oper} * t_{Turning}).$

3.5 The Definition of Danger Zone 2

When Car C has stopped at Spot 1, waiting to cross Lane A, there is a different zone for Driver C to watch for; we shall call it Danger Zone 2. Refer to Figure 4. Since Car C will start to move the vehicle from a zero speed, there will be a start time loss. Besides, it takes more time to cross Lane A from starting up the vehicle than it is moving at an operating speed. We shall call this zone Danger Zone 2.

Define $t_{Crossing}$ as the time for Car C to cross Lane A. The distance that Car C traveled from Spot 1 to Spot 2 is

(Eq. 17) $d_{Crossing} = R * \alpha_2.$

Since a constant acceleration rate of Car C, $Accel_C$ is assumed, $t_{Crossing}$ can be calculated as

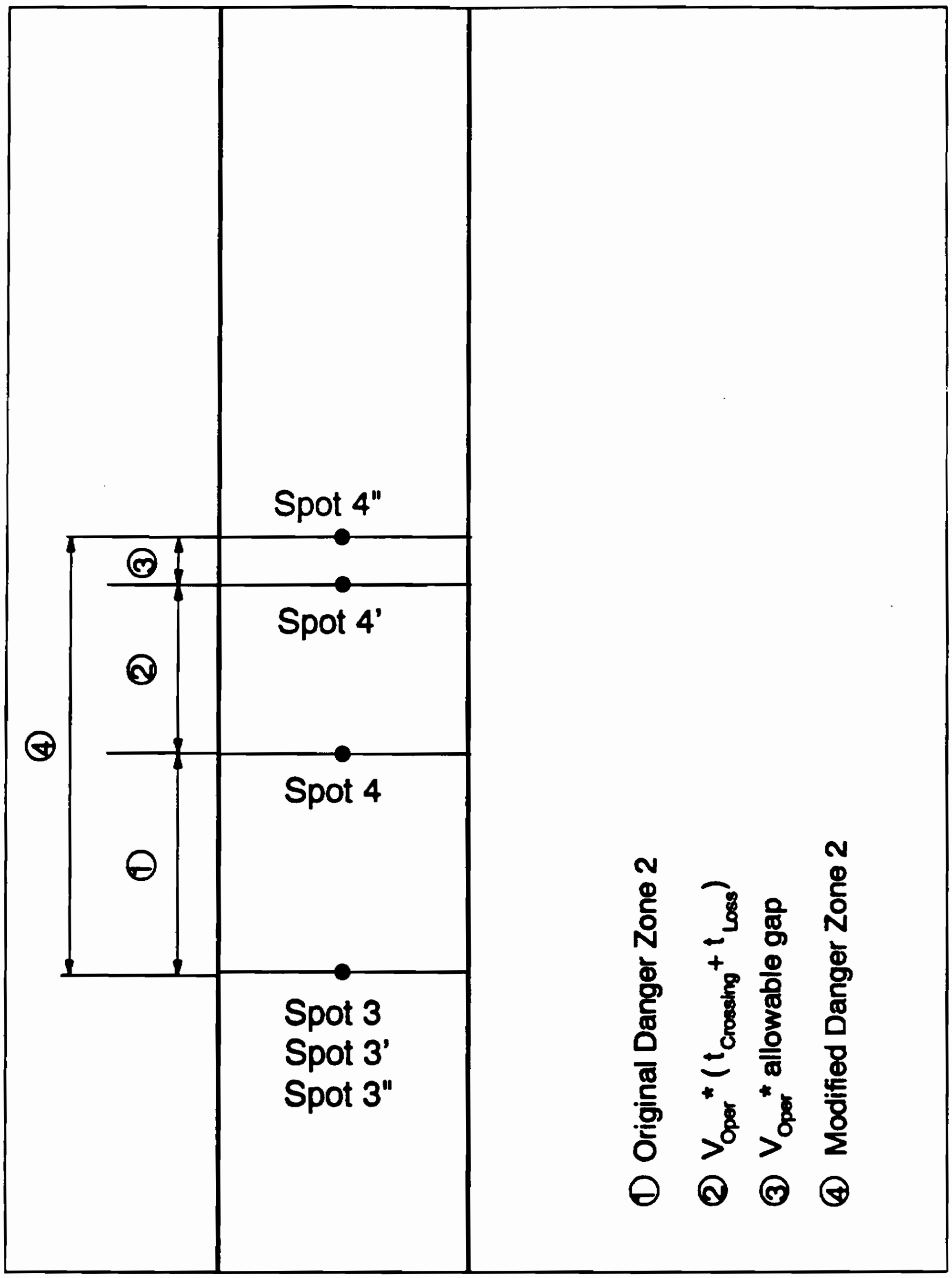

Figure 4. The Illustration of Danger Zone 2

(Eq. 18) $t_{Crossing} = [(2 * d_{Crossing}) / Accel_C]^{0.5}$.

As illustrated in Figure 4, the beginning point is Spot 3; the ending point of Danger Zone 2 is Spot 4 measured at a distance for Car C to travel back in a time of ($t_{Crossing} + t_{Loss}$).

(Eq. 19) $Spot3'_2(x , y) = (-1.5 * L , -0.5 * L)$.

(Eq. 20) $Spot4'_2(x , y) = (-1.5 * L , 1.5 * L + V_{Oper} * (t_{Crossing} + t_{Loss}))$.

3.6 Modification of the Location of Danger Zones 1 & 2

The calculation of the location of the two danger zones are according to the coordinates of Spot 1, Spot 2, Spot 3, and Spot 4. These points are the positions of the centers of the vehicles.argins between vehicles are not considered yet and therefore should be modified. The factor added is the drivers' tolerance of margins measured in the time units as allowable gaps.

There are regular drivers, aggressive drivers, and conservative drivers. We shall define these different types of drivers based on the tolerance of the margins between vehicles. The four types of drivers are classified as:

allowable gap = 0 seconds for limiting case,

allowable gap = 1 second for aggressive drivers,

allowable gap = 2 seconds for conservative drivers, and

allowable gap = 3 seconds for very conservative drivers.

Referring to Figure 3 and Figure 4 after the tolerance factor is added, the location of the modified Danger Zones 1 and 2 are re-defined as according to the following equations.

(Eq. 21) $Spot3''_1(x, y) = (-1.5 * L, -0.5 * L + V_{Oper} * (_{tBreaking} - \text{allowable gap})),$

(Eq. 22) $Spot4''_1(x, y) = (-1.5 * L, 1.5 * L + V_{Oper} * (t_{Turning} + \text{allowable gap})),$

(Eq. 23) $Spot3''_2(x, y) = (-1.5 * L, -0.5 * L - V_{Oper} * \text{allowable gap})),$

(Eq. 24) $Spot4''_2(x, y) = (-1.5 * L, 1.5 * L + V_{Oper} * (t_{Crossing} + t_{Loss} + \text{allowable gap}))$

3.7 Danger Zones Checks

If there are no vehicles in Lane B, Car C will have a clear view to the critical zone after it arrives at the PNR.

Here, the critical zone is referred to Danger Zone 1 when Driver C makes his judgement before Car C arrives at Spot 1, or it is referred to Danger Zone 2 while it is waiting at Spot 1. In the "clear view" case, it is very easy for Driver C to determine whether there are vehicles in Danger Zone 1. However, if there also are left turning vehicles in the opposing lane, Driver C's view to the critical zone may be blocked, especially when Car C and Car B are very close.

The dimension of all vehicles is assigned nine feet in width and seventeen feet in length as that of a design vehicle as in the AASHTO book [7]. Vehicles depicted in the rectangular shape is far from reality, but actual sizes of vehicles may be used. The driver's seat is assigned not based on the original data. Original data may be used, but the result will not be significant.

When the abscissa of the right front corner of Car B is equal to or greater than that of Driver C's seat, as shown in Figure 5, Car B will not cause a blocking problem. We might even say Car B has "passed" Car C. Otherwise, there may be a blocking problem unless Car B is near to its ejection point. Moreover, even Car A may block Driver C's view to Danger Zone 1.

To have a better understanding how the view of Driver C to the critical zone is clear or blocked, Figures 6 through 9 will give simple explanations for four possible cases.

Case 1. If $\theta_1 < \theta_2$, as in Figure 6, and there are no Car A in front of the critical zone, Driver C can make his judgement easily.

Case 2. If $\theta_1 < \theta_2$, as in Figure 7, and there are Car A's, and $\theta_4 > \theta_3$, the view of Driver C to the critical zone is also clear.

Case 3. If $\theta_1 > \theta_2$, as in Figure 8, it is clear that the view of driver C to the critical zone is partially or completely blocked.

Case 4. When $\theta_1 < \theta_2$, but $\theta_4 < \theta_3$, as in Figure 9, the view of Driver C to the critical zone is also partially or completely blocked.

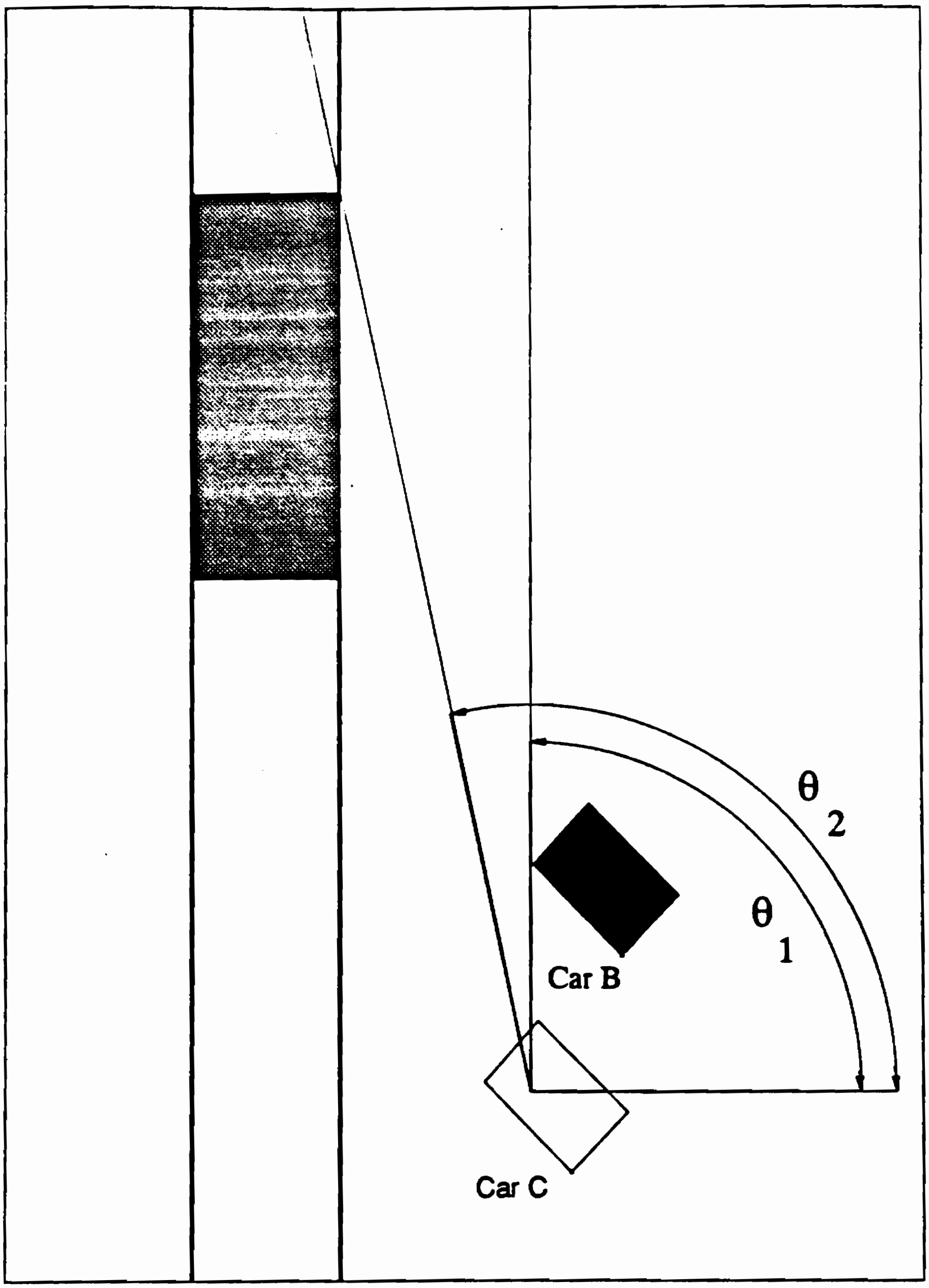

Figure 5. The Illustration of "Car B Passing Car C"

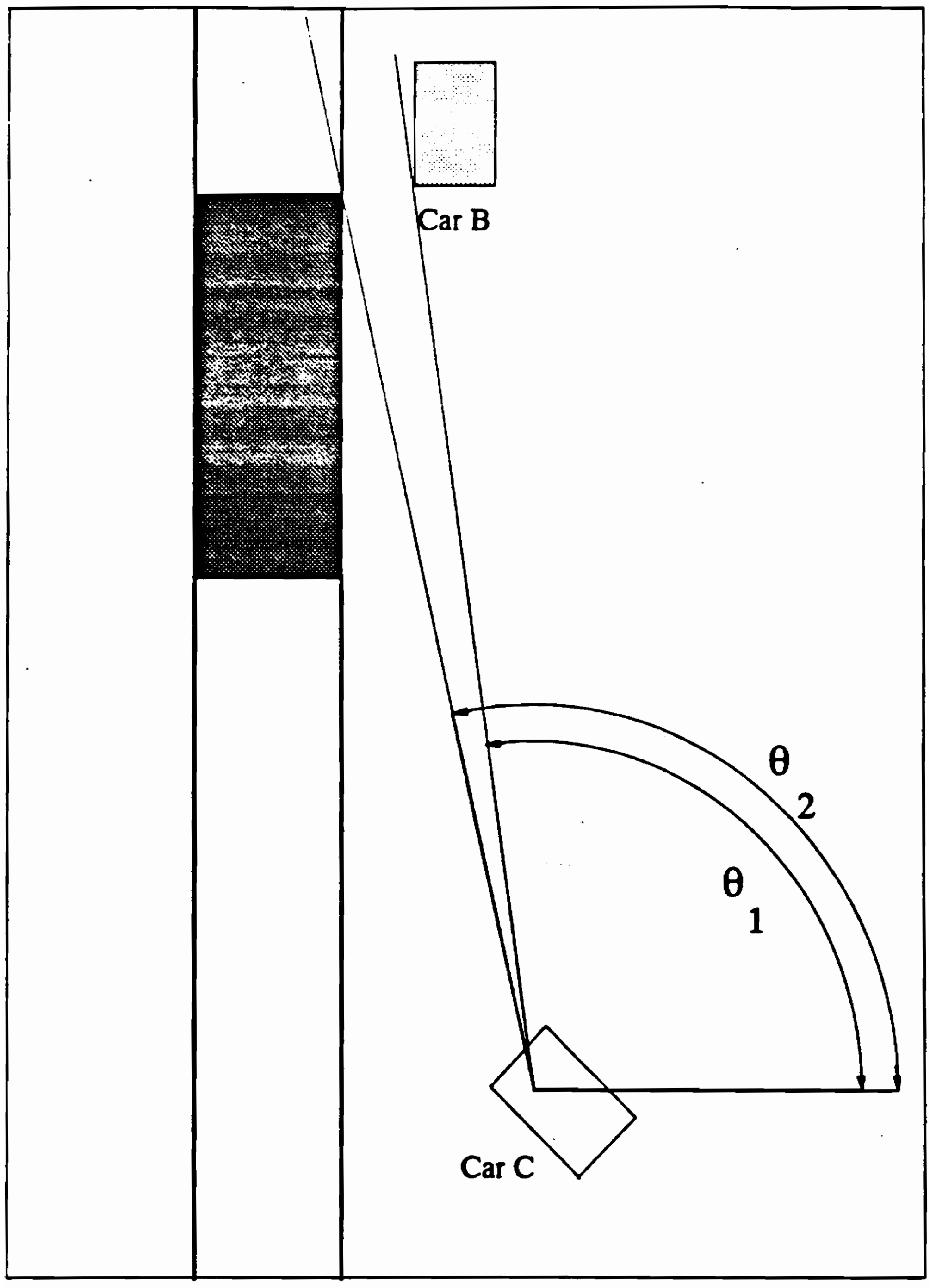

Figure 6. Case 1: Clear View

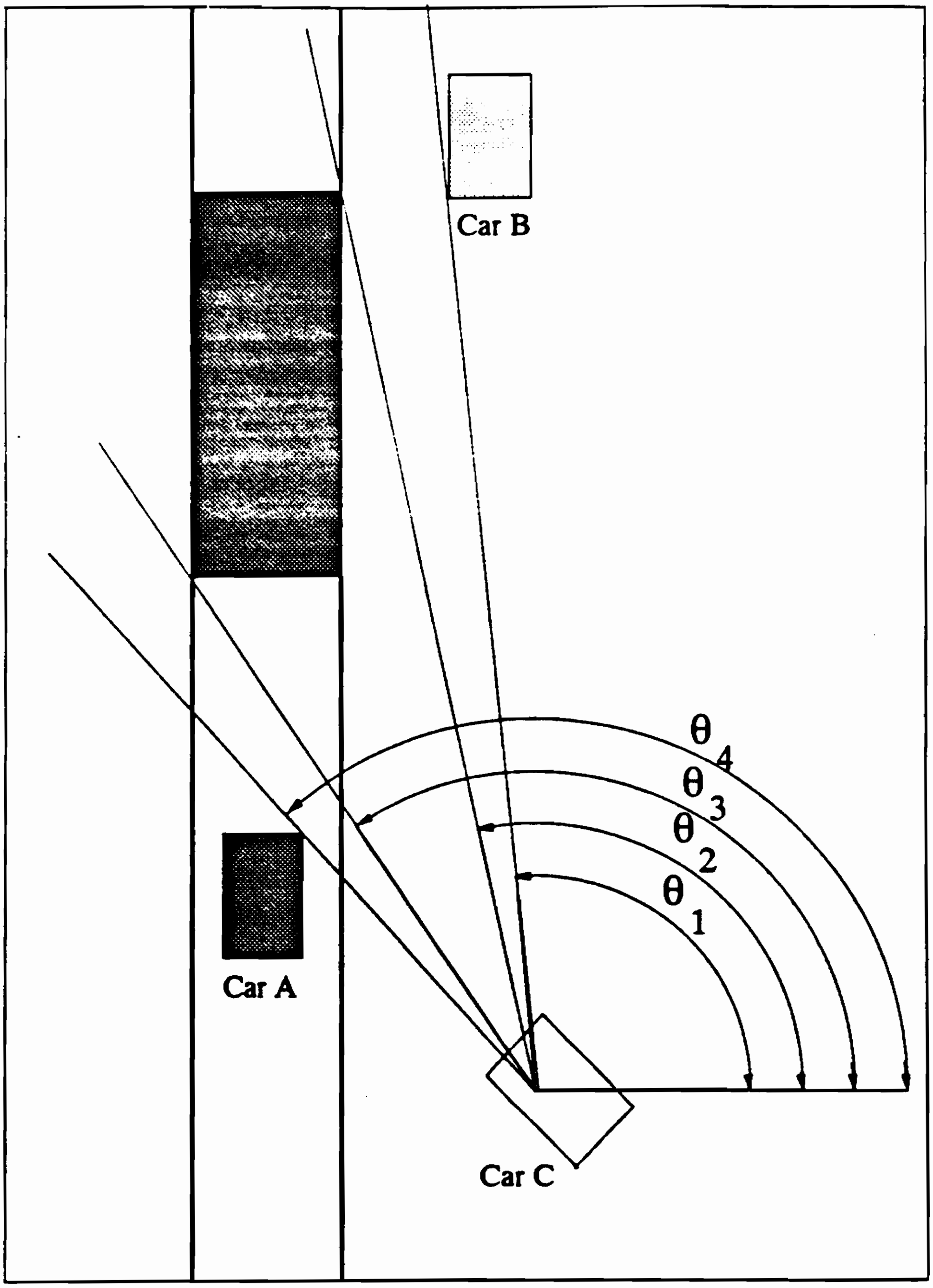

Figure 7. Case 2: Clear View 2

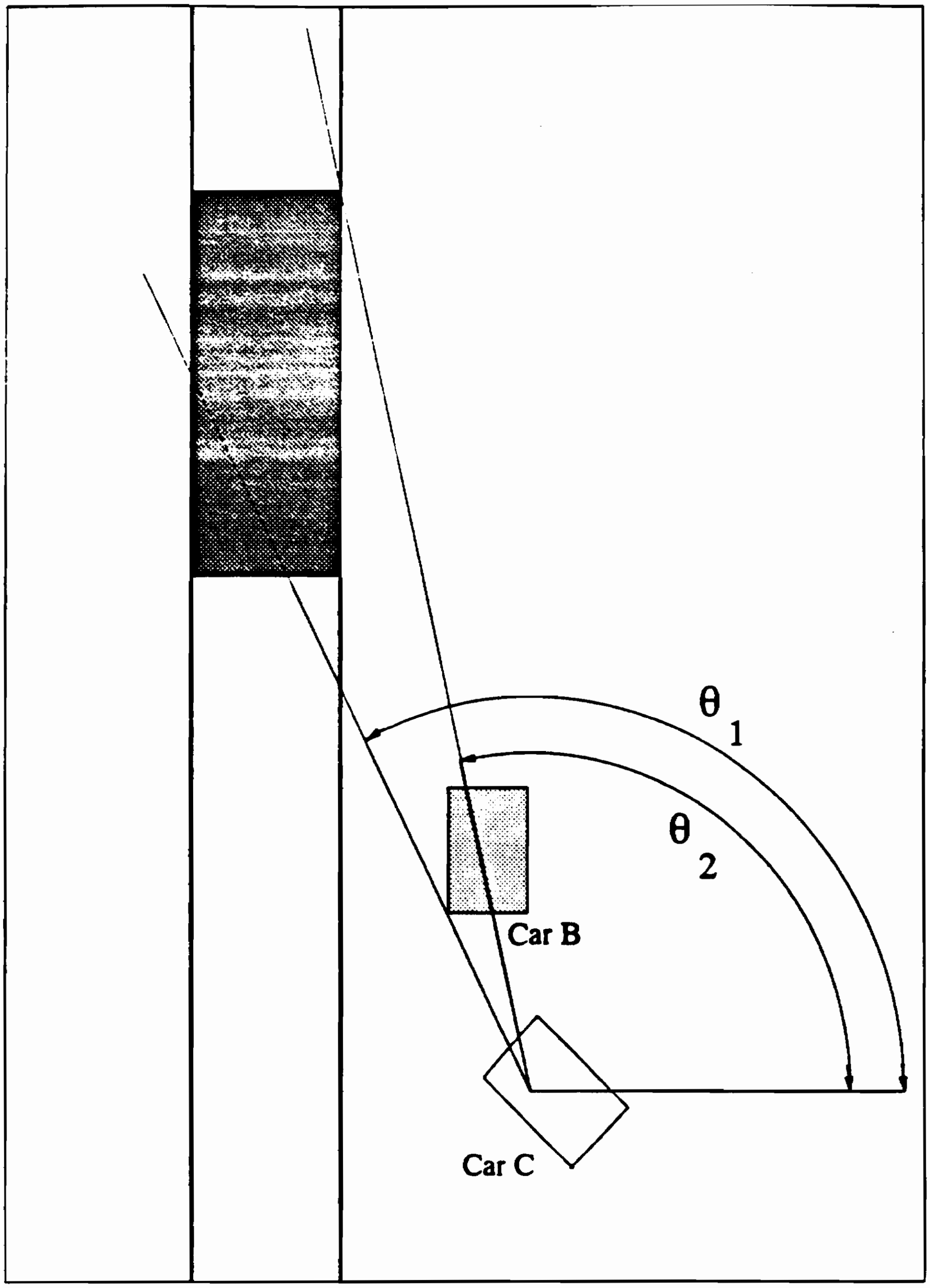

Figure 8. Case 3: Blocked View 1

As long as the view of Driver C to Danger Zone 1 is blocked, Driver C has to decelerate. However, it is not necessary to reduce the speed all the way to zero. For, at a later time, if the view is clear and there are no vehicles in Lane A, or Driver C finds appropriate gap to cross Lane A, he may maintain the already reduced speed until Car C completes the left turning movement. When the view of Driver C keeps being blocked, Car C may eventually come to a full stop at Spot 1. In either case, Driver C can assure himself not to have a collision with Car A. If Driver C has a clear view and sees vehicles in Danger Zone 1, he will be able to come to a full stop at Spot 1. If there is no vehicle in Danger Zone 1, Car C will certainly move on until it completes the left-turn movement.

When Car C is waiting at Spot 1, Driver C keeps looking for appropriate gap to cross Lane A safely by checking Danger Zone 2.

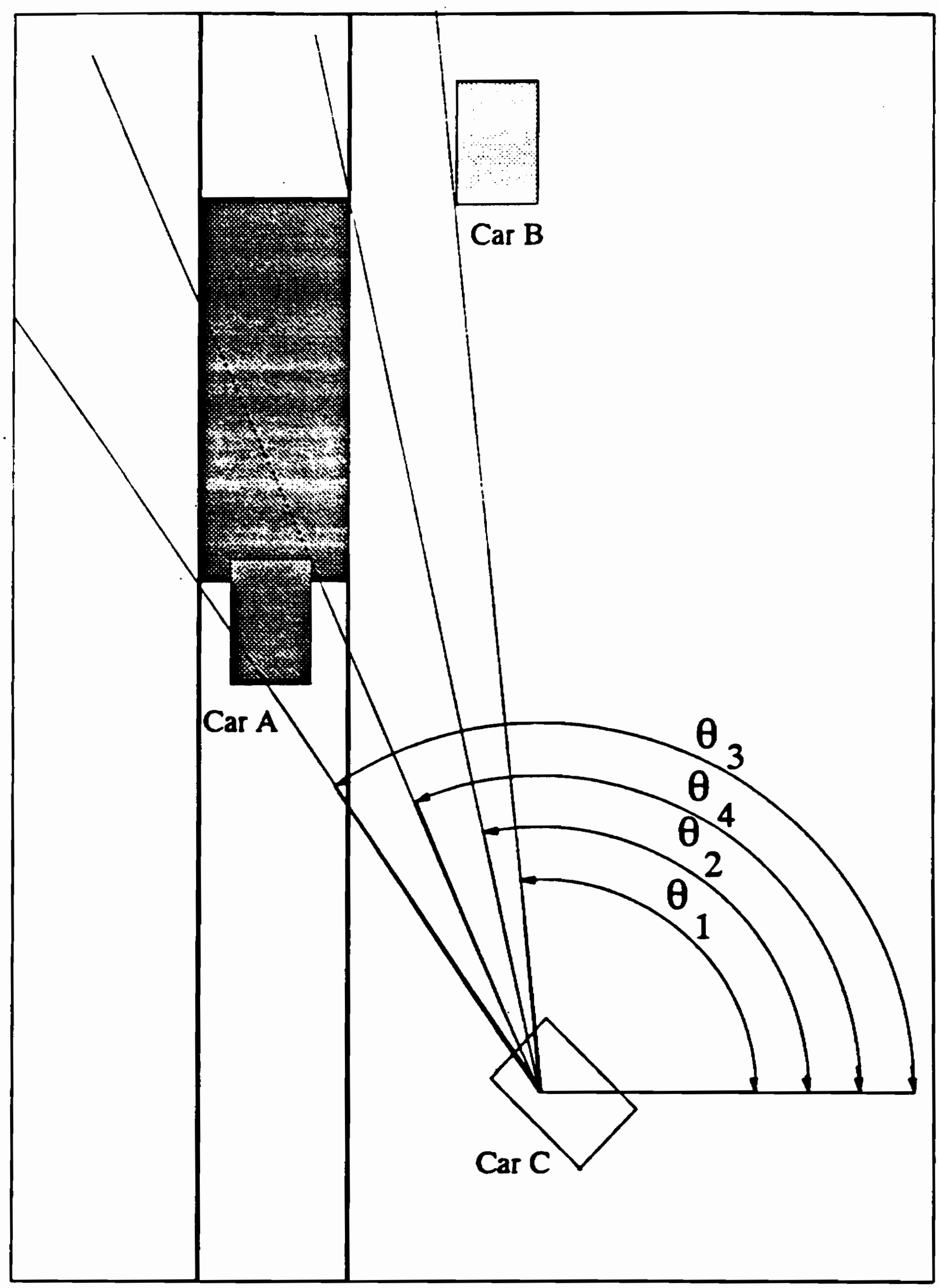

Figure 9. Case 4: Blocked View

3.8 Simulation Information Feedback

After Driver C makes judgement upon arriving at the PNR, he will give himself a signal of what to do at the next second.

There are seven signals and are described as follows.

1. **"Move on"**. This signal is given when Driver C's judgement is safe to keep going without speed change, or Car C has not arrived at and will not pass the PNR. Driver C does not have to make another judgement.

2. **"Decelerate"**. This signal is given when Driver C's judgement is not to cross Lane A because of an unsafe condition after checking Danger Zone 1. Car C keeps decelerating until it stop completely at Spot 1. Driver C does not have to make another judgement.

3. **"Accelerate"**. This signal is given when Car C has already started moving after waiting at zero speed at Spot 1. Car C keeps accelerating until it has reached its original speed, which was assigned at the ejection of Car C. No more judgement is to be made.

4. **"Wait"**. This signal is given when Car C is waiting at Spot 1, checking Danger Zone 2 for appropriate gap to cross Lane A, but it is still not safe to start moving. Driver C should keeps checking for the appropriate gap. When it is ready to start moving, Driver C gives himself a signal of **"Start"**.

5. **"Start"**. This signal is given when Lane A is clear for crossing. Driver C will give himself a signal of **"Accelerate"** after Car C has started moving.

6. **"Retry"**. The view of Driver C to Danger Zone 1 is temporarily blocked. He gives himself this signal so that he will try to check the safety for the crossing. That is, before Driver C decides either to move on or to decelerate, he has to make a judgement at every time step.

7. " ". The null signal is given only when Car C will be arriving before the PNR at the next second.

In the **"Wait"** mode, even Driver C decides to start moving, he still needs a couple more seconds to move the vehicle from zero speed to a slower speed. This is the start up time loss, t_{Loss}. Therefore, Driver C sends a pre-signal to tell himself to start moving t_{Loss} seconds later, and also sends himself a signal of "**Wait**" for the next second. Here, t_{Loss} is about three seconds.

3.9 Calculations of Next Speed and Position

After the judgement is made, the speed of Car C can be determined, and its position at the next time step can be calculated based on the speed determined from the judgement. Car A and Car B keep the same speed as long as there will be no head to tail collision in its own lane.

3.10 Delay Estimation

The delay is caused by the reduction of speed, while the reduction of speed is totally caused by the blocking of the view to the Danger Zones by the opposing traffic.

After a test run is done, total delay, $t_{TotalDelay}$, can be found easily. Suppose Car C can complete the left turning movement without reducing speed and takes time t_{Normal}. Also suppose Car C has reduced speed during the left turning movement, which is due to blocked view to Danger Zone 1, and takes t_{Actual} seconds to complete the left turning movement. It is obvious that the delay is the difference of these time as in the following equation:

(Eq. 25) $t_{TotalDelay} = t_{Actual} - t_{Normal}$.

Chapter IV Test Results and Discussions

4.1 Test Procedures and Results

A program written in Microsoft QuickBasic was used to develop a simulation model. The program first generates interarrival times for Car A (in the opposing through lane) and Car B (in the opposing left turning lane) from two Poisson distributions with user defined values (mean interarrival times). The algorithm of random interarrival time generation follows the method by Law & Kelton [8]. The program used the time radomizer provided by QuickBasic to get the seed. The model can be modified easily to test any combinations of speeds for Cars A, B, and C.

After interarrival times are known, the ejection (arrival) time for Car A or Car B can be obtained. Either Car A or Car B is ejected 250 feet back from the intersection. The initial speeds for Cars A and B are assigned ranging from 20 mph to 35 mph, and subsequent speeds for Cars A and B are constant. Their subsequent positions at each time step can be easily calculated. When Car A has passed Spot 3 or Car B has passed its counterpart of Spot 3, it is considered to be out of the scene. Thus, the queue length for the opposing through lane or the opposing left turn lane can be found. When Driver C later checks for the safety, he will check for cars in the current queues.

The ejection point of Car C is also 250 feet back from the intersection. The reason that the ejection point is chosen at 250 feet back from the intersection is to provide a minimum safe sight distance [7]. However, Car C is ejected at a later time such that the simulated traffic is established during the previous time period. After its speed is assigned, Car C is going to make a left turn.

Once Car C reaches the PNR, Driver C starts to make a judgement and he has three choices:

1. If Driver C has a clear view and it is safe to cross Lane A, he will move on until the left turn is completed. Driver C does not make another judgement.
2. If Driver C has a clear view but it is not safe to cross Lane A, he will brake and come to a full stop eventually at Spot 1. Driver C does not have to make a second judgement either.
3. If Driver C has a blocked view, he will decelerate, retry and make another judgement at the next time step.

Consequently, Car C will pass Spot 2, or it will stop at Spot 1 and wait to cross Lane A.

After Driver C makes a judgement, Car C will move on at the same speed, decelerate, wait at Spot 1 at zero speed, or accelerate after it starts up from Spot 1. Its subsequent positions can be calculated based on its speed. The acceleration rate is 5 ft/sec^2 [9];The deceleration rate is set at an uniform rate of 8 ft/sec^2 which is the normal value according to Khisty [10]. A higher value may reduce some delay but it may not be comfortable for the driver.

After Car C has reached Spot 2, the left turning movement is completed. The departure time is recorded and compared with its arrival time to obtain the actual time spent in the left turning movement. A normal turning time is calculated based on the speed assigned at the ejection. The delay will be the difference between the actual turning time and the ideal normal turning time. If Car C is still waiting at Spot 1 or is almost finishing the left turning movement when the simulation clock runs out of time, it is treated as if it had completed the left turning movement. The delay thus obtained is the extra time spent in the system.

4.2 Discussions

Each simulation run lasts thirty seconds, which is about the same as the green time of a regular signal cycle. For one simulation test for a particular set of λ_A and λ_B and a particular set of vehicle speeds, thirty runs were done to obtain an average value of delay. Various combinations of λ's and speeds for the particular sets are reasonably selected, in which The λ's range from two to eight, whereas the speeds range from 20 mph to 35 mph.

The test results are shown in the tables on the following pages.

After examining the results, some general findings are:

1. For drivers with smaller gap tolerance, which reflects the driver type, the delay is smaller.
2. When left turning vehicle has a higher speed, its delay is higher.
3. If the vehicle in the opposing through lane has a higher speed, the delay increases.
4. When the opposing through lane has a lower average interarrival time, the delay increases.

Table 1. Simulation Test Results. Test Set No. 1

	delay (sec)			
	gap = 0 (sec)	gap = 1 (sec)	gap = 2 (sec)	gap = 3 (sec)
λ_A = 2 sec				
λ_B = 2 sec	5.79	5.99	6.64	6.64
λ_B = 3 sec	5.99	6.42	6.64	6.64
λ_B = 4 sec	5.56	5.79	6.64	6.64
λ_B = 5 sec	5.56	6.42	6.42	6.64
λ_B = 6 sec	5.56	6.64	6.64	6.64
λ_A = 3 sec				
λ_B = 2 sec	4.26	5.13	6.21	6.42
λ_B = 3 sec	3.40	5.56	5.78	6.64
λ_B = 4 sec	4.70	5.13	5.78	6.64
λ_B = 5 sec	3.83	5.56	6.42	6.42
λ_B = 6 sec	3.40	5.34	5.99	6.21
λ_A = 4 sec				
λ_B = 2 sec	3.18	4.05	5.78	6.64
λ_B = 3 sec	2.97	3.62	5.78	5.99
λ_B = 4 sec	3.83	5.34	5.99	6.42
λ_B = 6 sec	2.54	4.26	5.56	5.78
λ_B = 8 sec	2.97	4.70	5.56	6.21

V_A = 20 mph, V_B = 20 mph, V_C = 20 mph

Table 2. Simulation Test Results. Test Set No. 2

	delay (sec)			
	gap = 0 (sec)	gap = 1 (sec)	gap = 2 (sec)	gap = 3 (sec)
λ_A = 2 sec				
λ_B = 2 sec	4.70	6.21	6.64	6.42
λ_B = 3 sec	3.83	5.99	6.64	6.64
λ_B = 4 sec	3.83	6.21	6.42	6.42
λ_B = 5 sec	4.70	5.78	6.42	6.64
λ_B = 6 sec	4.70	6.42	6.42	6.64
λ_A = 3 sec				
λ_B = 2 sec	3.40	5.99	6.21	6.42
λ_B = 3 sec	3.40	5.99	6.42	6.21
λ_B = 4 sec	3.18	5.78	6.21	6.42
λ_B = 5 sec	3.18	5.56	6.42	6.42
λ_B = 6 sec	2.75	4.91	6.64	6.64
λ_A = 4 sec				
λ_B = 2 sec	1.45	3.62	6.21	6.42
λ_B = 3 sec	2.34	5.13	5.13	6.21
λ_B = 4 sec	2.75	4.70	5.99	5.56
λ_B = 6 sec	4.26	3.83	5.99	6.21
λ_B = 8 sec	2.54	4.05	5.13	6.64

V_A = 25 mph, V_B = 20 mph, V_C = 20 mph

Table 3. Simulation Test Results. Test Set No. 3

	delay (sec)			
	gap = 0 (sec)	gap = 1 (sec)	gap = 2 (sec)	gap = 3 (sec)
λ_A = 2 sec				
λ_B = 2 sec	5.78	6.75	7.24	7.48
λ_B = 3 sec	5.30	6.75	7.24	7.48
λ_B = 4 sec	6.03	6.03	7.48	7.48
λ_B = 5 sec	5.06	5.06	7.48	7.48
λ_B = 6 sec	6.03	6.27	7.24	7.48
λ_A = 3 sec				
λ_B = 2 sec	2.88	3.36	7.00	7.48
λ_B = 3 sec	3.12	4.33	6.75	7.48
λ_B = 4 sec	3.12	5.06	7.00	7.48
λ_B = 5 sec	4.09	5.28	7.00	7.48
λ_B = 6 sec	3.12	6.51	7.24	7.48
λ_A = 4 sec				
λ_B = 2 sec	1.67	4.33	6.51	7.48
λ_B = 3 sec	0.94	2.88	6.27	7.48
λ_B = 4 sec	0.94	4.82	6.03	7.48
λ_B = 6 sec	2.15	3.61	6.51	7.48
λ_B = 8 sec	1.43	2.64	6.03	7.48

V_A = 25 mph, V_B = 25 mph, V_C = 25 mph

Table 4. Simulation Test Results. Test Set No. 4

	dealy (sec)			
	gap = 0 (sec)	gap = 1 (sec)	gap = 2 (sec)	gap =3 (sec)
λ_A = 2 sec				
λ_B = 2 sec	4.48	6.64	6.42	6.64
λ_B = 3 sec	4.91	6.21	6.64	6.64
λ_B = 4 sec	4.70	6.42	6.64	6.64
λ_B = 5 sec	3.62	6.42	6.64	6.64
λ_B = 6 sec	4.48	5.99	6.42	6.64
λ_A = 3 sec				
λ_B = 2 sec	3.83	5.56	6.21	6.64
λ_B = 3 sec	2.97	5.56	6.42	6.42
λ_B = 4 sec	2.97	6.00	6.42	6.64
λ_B = 5 sec	3.40	6.42	6.64	6.64
λ_B = 6 sec	3.83	5.34	6.42	6.42
λ_A = 4 sec				
λ_B = 2 sec	2.32	4.91	5.99	6.21
λ_B = 3 sec	2.97	5.56	6.42	5.79
λ_B = 4 sec	3.18	5.13	5.99	6.42
λ_B = 6 sec	2.75	4.05	6.42	6.42
λ_B = 8 sec	2.54	4.48	6.21	6.42

V_A = 30 mph, V_B = 20 mph, V_C = 20 mph

Table 5. Simulation Test Results. Test Set No. 5

	delay (sec)			
	gap = 0 (sec)	gap = 1 (sec)	gap = 2 (sec)	gap = 3 (sec)
λ_A = 2 sec				
λ_B = 2 sec	6.03	6.75	7.48	7.48
λ_B = 3 sec	5.06	6.51	7.48	7.48
λ_B = 4 sec	4.33	6.27	7.48	7.48
λ_B = 5 sec	4.33	6.51	7.00	7.48
λ_B = 6 sec	4.09	6.03	6.75	7.48
λ_A = 3 sec				
λ_B = 2 sec	3.85	4.57	6.27	7.48
λ_B = 3 sec	2.63	5.54	6.75	7.48
λ_B = 4 sec	3.12	4.09	6.51	7.48
λ_B = 5 sec	3.85	4.09	7.00	7.48
λ_B = 6 sec	2.40	4.82	6.75	7.48
λ_A = 4 sec				
λ_B = 2 sec	1.19	3.01	6.75	7.48
λ_B = 3 sec	2.15	2.88	6.75	7.48
λ_B = 4 sec	2.40	2.88	6.51	7.48
λ_B = 6 sec	1.91	3.12	5.78	7.48
λ_B = 8 sec	1.91	4.33	6.27	7.48

V_A = 30 mph, V_B = 25 mph, V_C = 25

Table 6. Simulation Test Results. Test Set No. 6

	delay (sec)			
	gap = 0 (sec)	gap = 1 (sec)	gap = 2 (sec)	gap = 3 (sec)
λ_A = 2 sec				
λ_B = 2 sec	4.26	4.48	6.64	6.64
λ_B = 3 sec	4.05	4.91	6.64	6.64
λ_B = 4 sec	4.70	5.99	6.64	6.64
λ_B = 5 sec	3.83	5.78	6.64	6.42
λ_B = 6 sec	4.91	5.13	6.64	6.42
λ_A = 3 sec				
λ_B = 2 sec	3.62	4.91	6.42	6.21
λ_B = 3 sec	4.05	3.40	6.21	6.21
λ_B = 4 sec	4.26	4.70	6.21	6.21
λ_B = 5 sec	4.26	4.91	6.64	6.42
λ_B = 6 sec	3.18	5.56	5.78	6.21
λ_A = 4 sec				
λ_B = 2 sec	2.75	4.05	6.21	6.21
λ_B = 3 sec	2.75	4.70	6.21	6.42
λ_B = 4 sec	3.40	4.48	6.64	6.64
λ_B = 6 sec	2.10	3.83	6.21	5.99
λ_B = 8 sec	2.97	4.48	6.42	5.78

V_A = 35 mph, V_B = 20 mph, V_C = 20 mph

Table 7. Simulation Test Results. Test Set No. 7

	delay (sec)			
	gap = 0 (sec)	gap = 1 (sec)	gap =2 (sec)	gap =3 (sec)
λ_A = 2 sec				
λ_B = 2 sec	3.12	6.51	7.24	7.48
λ_B = 3 sec	2.64	6.27	7.48	7.48
λ_B = 4 sec	2.40	6.27	7.00	7.48
λ_B = 5 sec	3.61	6.51	7.00	7.48
λ_B = 6 sec	1.67	6.75	7.00	7.48
λ_A = 3 sec				
λ_B = 2 sec	2.88	6.03	6.75	7.48
λ_B == 3 sec	1.43	5.54	6.51	7.48
λ_B = 4 sec	2.15	4.33	7.24	7.48
λ_B = 5 sec	1.67	3.85	6.51	7.48
λ_B = 6 sec	2.40	5.06	7.24	7.48
λ_A = 4 sec				
λ_B = 2 sec	0.94	3.61	7.00	7.48
λ_B = 3 sec	1.91	4.09	5.30	7.48
λ_B = 4 sec	1.43	2.40	6.27	7.48
λ_B = 5 sec	1.19	3.61	6.75	7.48
λ_B = 6 sec	1.43	3.85	5.78	7.48

V_A = 35 mph, V_B = 25 mph, V_C = 25 mph

Table 8. Simulation Test Results. No. 8

	delay (sec)			
	gap = 0 (sec)	gap = 1 (sec)	gap = 2 (sec)	gap = 3 (sec)
λ_A = 2 sec				
λ_B = 2 sec	3.15	5.69	6.45	7.72
λ_B = 3 sec	4.17	4.93	6.45	7.72
λ_B = 4 sec	3.40	5.69	7.21	7.72
λ_B = 5 sec	4.17	6.20	6.20	7.72
λ_B = 6 sec	2.39	4.17	6.96	7.72
λ_A = 3 sec				
λ_B = 2 sec	1.63	3.40	4.42	7.72
λ_B = 3 sec	2.14	3.40	4.17	7.72
λ_B = 4 sec	1.37	3.15	4.17	7.72
λ_B = 5 sec	2.14	3.40	4.93	7.72
λ_B = 6 sec	1.88	3.40	4.68	7.72
λ_A = 4 sec				
λ_B = 2 sec	0.87	1.88	2.14	7.72
λ_B = 3 sec	0.61	1.37	3.40	7.72
λ_B = 4 sec	0.61	1.37	3.91	7.72
λ_B = 6 sec	1.12	2.13	2.64	7.72
λ_B = 8 sec	0.61	2.91	3.15	7.72

V_A = 30 mph, V_B = 30 mph, V_C = 30 mph

5. When the opposing left turning lane has a higher average interarrival time, the delay decreases unless the average interarrival time in the opposing through lane is very low.

Other points about model flexibility, restriction, or expandability are briefly discussed below.

1. This is only a 2-D model. The vehicle height factor is totally ignored. A 3-D model could be developed when Driver C has his view to the critical zone blocked but can see the top part of the body of Car A.

2. The angle to a corner of the critical zone instead of the corner of Car A or B and the allowable gap provide extra safety factors.

3. This is a continuous simulation; therefore, nothing happens in a split second. Probably no vehicle will arrive at the exact position of the PNR. However, when the time step is smaller, it will give a more accurate estimate of delay.

4. The study does not include larger-sized vehicles for the purpose of simplicity. Even at truck prohibited intersections in a large city, large-sized, articulated buses

are allowed. Therefore, special consideration should be given for buses because of it irregular turning path and longer start-up time loss.

5. Just like trucks, most vehicles do not follow a circular path in the turning movement. In this case, the trajectory of the turning path may be approximated to a circular one.

6. The location of the ejection point has to be properly chosen. It should be long enough to provide sufficient turning capability but not too long to let vehicles cut in from the through lane. It is recommended to be 50 to 100 feet away from the PNR and larger than the minimum safe sight distance.

7. In reality, vehicles in the opposing through lane may lose several vehicles to the right turning movement. But it is not easy to control since some turning maneuver is not quite predictable.

8. The comfort factor is ignored. Drivers tend to decelerate to some degree just for the purpose of comfort. However, in most studies, this factor is also ignored although it could cause small delays.

9. The center of the vehicle is used to determine its position. Even when the centers of vehicles are not very close, the corners of the vehicles may perhaps provide minimal margins. The allowable gap will pick up the drawback.

10. We assume that Car A and Car B are always moving at constants speed; whereas they sometimes accelerate for aggressive drivers or decelerate for conservative drivers when they see a left turning vehicle approaching. If the speeds of Car A and Car B are randomly chosen, a pre-caution should be taken to avoid a head to tail type of collision in the same lane. Car C may not yield to the through traffic and may accelerate to cross Lane A instead.

11. Deceleration is actually not a constant because the driver may have not made a correct judgement and have to decelerate more at the next time step to compensate for misjudgments.

12. The data used for the start-up time loss and t_{PIJR} are not in fractions of a second. This could cause a round-off error if we choose whole one second as the increment of the time in the computer run. An adjustment of smaller scale will reduce this round-off error.

13. Car B's are indeed affected by Car D's and should be considered if a more accurate model is desired.

14. If there are several left turning vehicles, the trailing vehicles are very likely just following the leading vehicles and are unaware of the opposing traffic unless they are far apart. The driver of a closely trailing vehicle is most likely to have his view towards the Danger Zones blocked by the vehicle immediately in front of his. Thus, view blocking will cause more delay by the leading vehicles than by the opposing left turning vehicles.

If some of the points mentioned above are all considered in the model, and after the methodology for the estimate of delay for the trailing vehicles is developed, the speeds for each of the opposing left turning vehicles can have different values, as well as each Car A is assigned a different speed, the model will better represent the traffic at the intersection.

Chapter V Conclusion and Recommendation

The 1985 Highway Capacity Manual does not provide the methodology for analytical values of the delay at unsignalized intersections. In order to find this delay value, a simulation model is developed. It simulates the judgement of the driver in the left turning lane by checking whether there are vehicles in the potential danger zones. This is similar to finding appropriate critical gaps between successive vehicles in the opposing through lane. The critical point for the driver to make judgement is at the PNR (Point of No Return).

The driver will: (1) move on to cross the opposing through lane or (2) brake and eventually come to a full stop at a safe spot, then looking for appropriate gap to cross, both under the condition if the driver has a clear view; or (3) decelerate and reassess the safety if he has the view to the danger zone blocked by the opposing left turning vehicles.

The model was to test different combinations of (1) vehicle speeds, (2) average interarrival times, and (3) allowable gaps (a description of driver types) as sensitivity variables. The results generally proved to be reasonable.

Possible future work should be aimed at:

1. Develop methodology for the drivers of trailing left turning vehicles to make judgement such that the drivers in the opposing left turning lane can use the same strategy to determine theirs speeds.
2. Other constant variables such as acceleration rate, deceleration rate, time step scale, total simulation time, total number of test runs, and so on, can be assigned different values for to get more accurate results.

This study does not intend to replace the conventional method (Highway Capacity Manual method) but instead to provide an analytical solution based on a computer simulation program. The original idea is to help design an electronic sensor-guided equipment to make judgement for drivers or to have an automatic navigation controlon the highway.

Appendix: Computer Program Listing

```
'
'000000000000000000000000000000000000000000000000000000000000000000000000
'
'A PROGRAM FOR "A SIMULATION STUDY OF LEFT TURNING MOVEMENT AT AN
'
'UNSIGNALIZED INTERSECTION"
'
'000000000000000000000000000000000000000000000000000000000000000000000000
'
DECLARE SUB CornerA (T%, CarA%, XA!(), YA!(), WidthA!(), LengthA!(), RFCXA!(),
        RFCYA!(), LFCXA!(), LFCYA!(), RRCXA!(), RRCYA!(), LRCXA!(), LRCYA!())
DECLARE SUB CornerB (T%, CarB%, XB!(), YB!(), WidthB!(), LengthB!(), RFCXB!(),
        RFCYB!(), LFCXB!(), LFCYB!(), RRCXB!(), RRCYB!(), LRCXB!(), LRCYB!())
DECLARE SUB DriverSeatC (T%, CarC%, XC!(), YC!(), WidthC!(), LengthC!(),
                                  DriverSeatXC!(), DriverSeatYC!())
DECLARE SUB NextPosAccelC (T%, CarC%, SpeedC!(), AccelC!(), XC!(), YC!(), R!,
                                                              L!, deltaT%)
DECLARE SUB NextPosDecelC (T%, CarC%, SpeedC!(), DecelC!(), XC!(), YC!(), R!,
                                                              L!, deltaT%)
DECLARE SUB NextPosSameC (T%, CarC%, SpeedC!(), XC!(), YC!(), R!, L!, deltaT%)
TimeStep% = 32
CarNo% = 30
deltaT% = 1
DIM SHARED HeadwayA%(CarNo%), HeadwayB%(CarNo%), HeadwayC%(CarNo%)
DIM SHARED InTimeA%(CarNo%), IntimeB%(CarNo%), IntimeC%(CarNo%)
DIM SHARED OutTimeA%(CarNo%), OutTimeB%(CarNo%), OutTimeC%(CarNo%)
DIM SHARED ArrivalA%(TimeStep%), ArrivalB%(TimeStep%), ArrivalC%(TimeStep%)
DIM SHARED CountA%(TimeStep%), CountB%(TimeStep%), CountC%(TimeStep%)
DIM SHARED QueueA%(TimeStep%), QueueB%(TimeStep%), QueueC%(TimeStep%)
DIM SHARED WidthA!(CarNo%), WidthB!(CarNo%), WidthC!(CarNo%)
DIM SHARED LengthA!(CarNo%), LengthB!(CarNo%), LengthC!(CarNo%)
DIM SHARED SignalC$(TimeStep%, CarNo%), PreSignalC$(TimeStep%, CarNo%)
DIM SHARED XA!(TimeStep%, CarNo%), XB!(TimeStep%, CarNo%),
                                  XC!(TimeStep%, CarNo%)
DIM SHARED YA!(TimeStep%, CarNo%), YB!(TimeStep%, CarNo%),
                                  YC!(TimeStep%, CarNo%)
DIM SHARED PNRXC!(CarNo%), PNRYC!(CarNo%)
DIM SHARED SmallestA%(TimeStep%), SmallestB%(TimeStep%), SmallestC%(TimeStep%)
DIM SHARED LargestA%(TimeStep%), LargestB%(TimeStep%), LargestC%(TimeStep%)
DIM SHARED RFCXA!(TimeStep%, CarNo%), RFCYA!(TimeStep%, CarNo%)
DIM SHARED RFCXB!(TimeStep%, CarNo%), RFCYB!(TimeStep%, CarNo%)
DIM SHARED RFCXC!(TimeStep%, CarNo%), RFCYC!(TimeStep%, CarNo%)
DIM SHARED LFCXA!(TimeStep%, CarNo%), LFCYA!(TimeStep%, CarNo%)
DIM SHARED LFCXB!(TimeStep%, CarNo%), LFCYB!(TimeStep%, CarNo%)
DIM SHARED LFCXC!(TimeStep%, CarNo%), LFCYC!(TimeStep%, CarNo%)
DIM SHARED RRCXA!(TimeStep%, CarNo%), RRCYA!(TimeStep%, CarNo%)
DIM SHARED RRCXB!(TimeStep%, CarNo%), RRCYB!(TimeStep%, CarNo%)
DIM SHARED RRCXC!(TimeStep%, CarNo%), RRCYC!(TimeStep%, CarNo%)
DIM SHARED LRCXA!(TimeStep%, CarNo%), LRCYA!(TimeStep%, CarNo%)
DIM SHARED LRCXB!(TimeStep%, CarNo%), LRCYB!(TimeStep%, CarNo%)
DIM SHARED LRCXC!(TimeStep%, CarNo%), LRCYC!(TimeStep%, CarNo%)
DIM SHARED DriverSeatXC!(TimeStep%, CarNo%), DriverSeatYC!(TimeStep%, CarNo%)
DIM SHARED InitSpeedA!(CarNo%), InitSpeedB!(CarNo%), InitSpeedC!(CarNo%)
DIM SHARED SpeedA!(TimeStep%, CarNo%), SpeedB!(TimeStep%, CarNo%),
                                      SpeedC!(TimeStep%, CarNo%)
DIM SHARED AccelC!(CarNo%), DecelC!(CarNo%), FrictionC!(CarNo%)
DIM SHARED InA%(TimeStep%), InB%(TimeStep%), InC%(TimeStep%)
DIM SHARED OutA%(TimeStep%), OutB%(TimeStep%), OutC%(TimeStep%)
DIM SHARED AngleARight!(CarNo%), AngleBLeft!(CarNo%)
DIM SHARED JudgementA1$(TimeStep%), JudgementA2$(TimeStep%)
```

```
DIM SHARED SignalZoneA1$(TimeStep%), SignalZoneA2$(TimeStep%)
DIM UA!(50), UB!(50)
DIM DelayC!(50)
'
'000000000000000000000000000000000000000000000000000000000000000000000000000
'
'111111111111111111111111111111111111111111111111111111111111111111111111111
'
'THE FOLLOWING BLOCK DEFINES THE CONFIGURATION OF THE INTERSECTION
'
'AND SAFE SPOTS.
'
'111111111111111111111111111111111111111111111111111111111111111111111111111
'
L! = 12!
R! = 33!
OperSpeed! = 35!
g! = 0!
Pi! = 4! * ATN(1!)
C! = (R! - .5 * L!)
Spot1X! = -.5 * L!
Spot1Y! = SQR(R! ^ 2 - (C! - L! / 2!) ^ 2) - C!
Spot2X! = -C!
Spot2Y! = .5 * L!
Spot3X! = -1.5 * L!
Spot3Y! = -.5 * L!
Spot4X! = -1.5 * L!
Spot4Y! = 1.5 * L!
P1X! = .5 * L!
P1Y! = -C!
Alpha1! = ATN((Spot1Y! + C!) / (Spot1X! + C!))
Alpha2! = (Pi! / 2!) - Alpha1!
'
'111111111111111111111111111111111111111111111111111111111111111111111111111
'
'222222222222222222222222222222222222222222222222222222222222222222222222222
'
'THE PROGRAM STARTS SIMULATIONS FROM HERE!
'
'RANDOM INTERARRIVAL TIME GENERATION FOR CAR A'S AND CAR B'S.
'
'CALCULATE THEIR POSITIONS AT EACH TIME STEP.
'
'222222222222222222222222222222222222222222222222222222222222222222222222222
'
TotalNumberOfRun% = 30
NumberOfRun% = 1
TotalDelayC! = 0!
Simulation:
tStep% = 30
lambdaA% = 4
iA% = 1
TTT% = 0
A:
kA% = 1
cprod! = 1
RANDOMIZE TIMER
RepeatA:
UA!(kA%) = RND
cprod! = UA!(kA%) * cprod!
IF cprod! < EXP(-lambdaA%) THEN
  HeadwayA%(iA%) = kA%
ELSE
   kA% = kA% + 1
   GOTO RepeatA
```

```
END IF
TTT% = TTT% + HeadwayA%(iA%)
IF TTT% < tStep% THEN
iA% = iA% + 1
GOTO A
ELSEIF TTT% = tStep% THEN
   GOTO WriteA
ELSE
  iA% = iA% - 1
END IF
WriteA:
OPEN "b:headwaya.dat" FOR OUTPUT AS #101
   FOR iiA% = 1 TO iA%
      WRITE #101, HeadwayA%(iiA%)
   NEXT iiA%
CLOSE #101
lambdaB% = 8
iB% = 1
TTT% = 0
B:
kB% = 1
cprod! = 1
RANDOMIZE TIMER
RepeatB:
UB!(kB%) = RND
cprod! = UB!(kB%) * cprod!
IF cprod! < EXP(-lambdaB%) THEN
  HeadwayB%(iB%) = kB%
ELSE
   kB% = kB% + 1
   GOTO RepeatB
END IF
TTT% = TTT% + HeadwayB%(iB%)
IF TTT% < tStep% THEN
iB% = iB% + 1
GOTO B
ELSEIF TTT% = tStep% THEN
   GOTO WriteB
ELSE
  iB% = iB% - 1
END IF
WriteB:
OPEN "b:headwayB.dat" FOR OUTPUT AS #102
   FOR iiB% = 1 TO iB%
      WRITE #102, HeadwayB%(iiB%)
   NEXT iiB%
CLOSE #102
OPEN "b:widtha.dat" FOR OUTPUT AS #103
OPEN "b:lengtha.dat" FOR OUTPUT AS #104
OPEN "b:speeda.dat" FOR OUTPUT AS #105
FOR jA% = 1 TO iA%
   WidthA!(jA%) = 7!
   LengthA!(jA%) = 19!
   InitSpeedA!(jA%) = 25!
   WRITE #103, WidthA!(jA%)
   WRITE #104, LengthA!(jA%)
   WRITE #105, InitSpeedA!(jA%)
NEXT jA%
CLOSE #103
CLOSE #104
OPEN "b:widthb.dat" FOR OUTPUT AS #106
OPEN "b:lengthb.dat" FOR OUTPUT AS #107
OPEN "b:speedb.dat" FOR OUTPUT AS #108
FOR jB% = 1 TO iB%
   WidthB!(jB%) = 7!
```

```
   LengthB!(jB%) = 19!
   InitSpeedB!(jB%) = 20!
   WRITE #106, WidthB!(jB%)
   WRITE #107, LengthB!(jB%)
   WRITE #108, InitSpeedB!(jB%)
NEXT jB%
CLOSE #106
CLOSE #107
CLOSE #108
'
'2222222222222222222222222222222222222222222222222222222222222222222222222222
'
'3333333333333333333333333333333333333333333333333333333333333333333333333333
'
'INPUT HEADWAYS BETWEEN CARS, CAR SPEEDS, CAR WIDTHS, CAR LENGTHS,
'
'CAR C'S ACCELERATION AND DECELERATION.
'
'CALCULATE ARRIVAL TIMES OF CAR A'S AND CAR B'S.
'
'RECORD COUNT AT ALL TIME STEPS.
'
'DEFINE THE EJECTION POINT.
'
'3333333333333333333333333333333333333333333333333333333333333333333333333333
'
DriverTypeC%(1) = 0
InitSpeedC!(1) = 20!
IF InitSpeedC!(1) < 20! THEN
   FrictionC!(1) = .43
ELSEIF InitSpeedC!(1) = 20! THEN
   FrictionC!(1) = .4
ELSEIF InitSpeedC!(1) = 25! THEN
   FrictionC!(1) = .38
ELSEIF InitSpeedC!(1) = 30! THEN
   FrictionC!(1) = .35
ELSEIF InitSpeedC!(1) = 35! THEN
   FrictionC!(1) = .34
ELSE
END IF
OPEN "b:HeadwayA.dat " FOR INPUT AS #1
CntA% = 0
DO UNTIL EOF(1)
   CntA% = CntA% + 1
   INPUT #1, HeadwayA%(CntA%)
   ArrivalA%(CntA%) = ArrivalA%(CntA% - 1) + HeadwayA%(CntA%)
   FOR ttaa% = 0 TO (ArrivalA%(CntA%) - 1)
      XA!(ttaa%, CntA%) = 0
      YA!(ttaa%, CntA%) = 0
      SpeedA!(ttaa%, CntA%) = 0!
   NEXT ttaa%
      XA!(ArrivalA%(CntA%), CntA%) = -1.5 * L!
      YA!(ArrivalA%(CntA%), CntA%) = 250!
      SpeedA!(ArrivalA%(CntA%), CntA%) = InitSpeedA!(CntA%)
   FOR ttaaa% = (ArrivalA%(CntA%) + 1) TO (TimeStep% - 1)
      SpeedA!(ttaaa%, CntA%) = InitSpeedA!(CntA%)
      XA!(ttaaa%, CntA%) = -1.5 * L!
      YA!(ttaaa%, CntA%) = YA!(ttaaa% - 1, CntA%)
                           - 1.47 * SpeedA!(ttaaa%, CntA%) * deltaT%
   NEXT ttaaa%
LOOP
FOR NumberA% = 1 TO CntA%
   FOR TA% = ArrivalA%(NumberA% - 1) TO (ArrivalA%(NumberA%) - 1)
      CountA%(TA%) = NumberA% - 1
   NEXT TA%
```

```
NEXT NumberA%
FOR TAQ% = (ArrivalA%(NumberA% - 1)) TO TimeStep%
     CountA%(TAQ%) = NumberA% - 1
NEXT TAQ%
CLOSE #1
OPEN "b:HeadwayB.dat" FOR INPUT AS #2
CntB% = 0
DO UNTIL EOF(2)
   CntB% = CntB% + 1
   INPUT #2, HeadwayB%(CntB%)
   ArrivalB%(CntB%) = ArrivalB%(CntB% - 1) + HeadwayB%(CntB%)
   FOR ttbb% = 0 TO (ArrivalB%(CntB%) - 1)
      XB!(ttbb%, CntB%) = 0
      YB!(ttbb%, CntB%) = 0
      SpeedB!(ttbb%, CntB%) = 0!
   NEXT ttbb%
      XB!(ArrivalB%(CntB%), CntB%) = -.5 * L!
      YB!(ArrivalB%(CntB%), CntB%) = 250!
      SpeedB!(ArrivalB%(CntB%), CntB%) = InitSpeedB!(CntB%)
   FOR ttbbb% = (ArrivalB%(CntB%) + 1) TO (TimeStep% - 1)
      SpeedB!(ttbbb%, CntB%) = InitSpeedB!(CntB%)
      DistanceTraveled! = 1.47 * SpeedB!(ttbbb% - 1, CntB%) * deltaT%
      SELECT CASE XB!(ttbbb% - 1, CntB%)
      CASE IS = -.5 * L!
         YB!(ttbbb%, CntB%) = YB!(ttbbb% - 1, CntB%) - DistanceTraveled!
         IF YB!(ttbbb%, CntB%) < C! THEN
            ArcLength! = DistanceTraveled! - (YB!(ttbbb% - 1, CntB%) - C!)
            thetaB1! = ArcLength! / R!
            IF thetaB1! > (Pi! / 2!) THEN
               XB!(ttbbb%, CntB%) = C! + (ArcLength! - R! * (thetaB1!
                                                   - Pi! / 2!))
               YB!(ttbbb%, CntB%) = -.5 * L!
            ELSE
               XB!(ttbbb%, CntB%) = XB!(ttbbb% - 1, CntB%)
                                  + (R! - R! * COS(thetaB1!))
               YB!(ttbbb%, CntB%) = C! - R! * SIN(thetaB1!)
            END IF
         ELSE
               XB!(ttbbb%, CntB%) = -.5 * L!
         END IF
      CASE IS >= C!
          XB!(ttbbb%, CntB%) = XB!(ttbbb% - 1, CntB%) + DistanceTraveled!
          YB!(ttbbb%, CntB%) = -.5 * L!
      CASE ELSE
          ArcLength! = DistanceTraveled!
          thetaB1! = ATN((C! - YB!(ttbbb% - 1, CntB%)) / (C! -XB!(ttbbb% - 1,
                                                              CntB%)))
          NewThetaB1! = thetaB1! + (ArcLength / R!)
          IF NewThetaB1! < (Pi! / 2!) THEN
             XB!(ttbbb%, CntB%) = XB!(ttbbb% - 1, CntB%)
                                + (R! - R! * COS(NewThetaB1!))
             YB!(ttbbb%, CntB%) = C! - R! * SIN(NewThetaB1!)
          ELSE
             XB!(ttbbb%, CntB%) = C! + R! * (NewThetaB1! - (Pi! / 2!))
             YB!(ttbbb%, CntB%) = -.5 * L!
          END IF
      END SELECT
   NEXT ttbbb%
LOOP
FOR NumberB% = 1 TO CntB%
   FOR TB% = ArrivalB%(NumberB% - 1) TO (ArrivalB%(NumberB%) - 1)
      CountB%(TB%) = NumberB% - 1
   NEXT TB%
NEXT NumberB%
FOR TBQ% = (ArrivalB%(NumberB% - 1)) TO TimeStep%
```

```
      CountB%(TBQ%) = NumberB% - 1
NEXT TBQ%
CLOSE #2
OPEN "b:HeadwayC.dat" FOR INPUT AS #3
CntC% = 0
DO UNTIL EOF(3)
   CntC% = CntC% + 1
   INPUT #3, HeadwayC%(CntC%)
   ArrivalC%(CntC%) = ArrivalC%(CntC% - 1) + HeadwayC%(CntC%)
LOOP
FOR ttcc% = 0 TO (ArrivalC%(1) - 1)
   SpeedC!(ttcc%, 1) = 0!
NEXT ttcc%
CLOSE #3
OPEN "b:WidthA.dat" FOR INPUT AS #5
CntA% = 0
DO UNTIL EOF(5)
   CntA% = CntA% + 1
   INPUT #5, WidthA!(CntA%)
LOOP
CLOSE #5
OPEN "b:WidthB.dat" FOR INPUT AS #6
CntB% = 0
DO UNTIL EOF(6)
   CntB% = CntB% + 1
   INPUT #6, WidthB!(CntB%)
LOOP
CLOSE #6
OPEN "b:WidthC.dat" FOR INPUT AS #7
CntC% = 0
DO UNTIL EOF(7)
   CntC% = CntC% + 1
   INPUT #7, WidthC!(CntC%)
LOOP
CLOSE #7
OPEN "b:LengthA.dat" FOR INPUT AS #9
CntA% = 0
DO UNTIL EOF(9)
   CntA% = CntA% + 1
   INPUT #9, LengthA!(CntA%)
LOOP
CLOSE #9
OPEN "b:LengthB.dat" FOR INPUT AS #10
CntB% = 0
DO UNTIL EOF(10)
   CntB% = CntB% + 1
   INPUT #10, LengthB!(CntB%)
LOOP
CLOSE #10
OPEN "b:LengthC.dat" FOR INPUT AS #11
CntC% = 0
DO UNTIL EOF(11)
   CntC% = CntC% + 1
   INPUT #11, LengthC!(CntC%)
LOOP
CLOSE #11
OPEN "b:SpeedA.dat" FOR INPUT AS #13
CntA% = 0
DO UNTIL EOF(13)
   CntA% = CntA% + 1
   INPUT #13, InitSpeedA!(CntA%)
LOOP
CLOSE #13
OPEN "b:SpeedB.dat" FOR INPUT AS #14
CntB% = 0
```

```
DO UNTIL EOF(14)
   CntB% = CntB% + 1
   INPUT #14, InitSpeedB!(CntB%)
LOOP
CLOSE #14
OPEN "b:DecelerC.dat" FOR INPUT AS #19
CntC% = 0
DO UNTIL EOF(19)
   CntC% = CntC% + 1
   INPUT #19, DecelC!(CntC%)
LOOP
CLOSE #19
OPEN "b:AccelerC.dat" FOR INPUT AS #23
CntC% = 0
DO UNTIL EOF(23)
   CntC% = CntC% + 1
   INPUT #23, AccelC!(CntC%)
LOOP
CLOSE #23
'
'333333333333333333333333333333333333333333333333333333333333333333333333333
'
'444444444444444444444444444444444444444444444444444444444444444444444444444
'
'CALCULATE THE QUEUE LENGTH AT ALL TIME STEPS.
'
'CHECK LANES FOR (1): IF THE LANE IS EMPTY?
'                (2): IF A VEHICLE COMES IN AT THIS TICK?
'                (3): IF A VEHICLE WENT OUT AT LAST TICK?
'                (4): WHAT IS THE QUEUE LENGTH AT THIS TICK?
'
'444444444444444444444444444444444444444444444444444444444444444444444444444
'
CheckForLaneA:
FOR Thyme = 1 to 30
IF CountA%(Thyme%) = 0 THEN
   QueueA%(Thyme%) = 0
   InA%(Thyme%) = 0
   OutA%(Thyme% + 1) = 0
   GOTO CheckForLaneB
ELSEIF CountA%(Thyme%) = CountA%(Thyme% - 1) THEN
   InA%(Thyme%) = 0
   GOTO Step3A
ELSE
   InA%(Thyme%) = 1
END IF
Step3A:
QueueA%(Thyme%) = QueueA%(Thyme% - 1) + InA%(Thyme%) - OutA%(Thyme%)
DefineSmallestA:
IF QueueA%(Thyme%) = 0 THEN
   SmallestA%(Thyme%) = 0
   GOTO DefineLargestA
ELSEIF SmallestA%(Thyme% - 1) = 0 AND InA%(Thyme%) = 1 THEN
   SmallestA%(Thyme%) = CountA%(Thyme%)
   GOTO DefineLargestA
ELSEIF OutA%(Thyme%) = 0 THEN
  SmallestA%(Thyme%) = SmallestA%(Thyme% - 1)
  GOTO DefineLargestA
ELSE
   SmallestA%(Thyme%) = SmallestA%(Thyme% - 1) + 1
END IF
DefineLargestA:
IF QueueA%(Thyme%) = 0 THEN
   LargestA%(Thyme%) = 0
   GOTO CheckForLaneB
```

```
ELSE
   LargestA%(Thyme%) = CountA%(Thyme%)
END IF
NextPositionsForCarsInLaneA:
IF QueueA%(Thyme%) = 0 THEN
   GOTO CheckForLaneB:
ELSE
END IF
CheckForPassingSafetySpot3:
IF YA!(Thyme% + 1, SmallestA%(Thyme%)) <= -L! THEN
   OutA%(Thyme% + 1) = 1
   GOTO CheckForLaneB
ELSE
   OutA%(Thyme% + 1) = 0
END IF
CheckForLaneB:
IF CountB%(Thyme%) = 0 THEN
   QueueB%(Thyme%) = 0
   InB%(Thyme%) = 0
   OutB%(Thyme%) = 0
   GOTO Step3B
ELSEIF CountB%(Thyme%) = CountB%(Thyme% - 1) THEN
   InB%(Thyme%) = 0
   GOTO Step3B
ELSE
   InB%(Thyme%) = 1
END IF
Step3B:
QueueB%(Thyme%) = QueueB%(Thyme% - 1) + InB%(Thyme%) - OutB%(Thyme%)
DefineSmallestB:
IF QueueB%(Thyme%) = 0 THEN
   SmallestB%(Thyme%) = 0
   GOTO DefineLargestB
ELSEIF SmallestB%(Thyme% - 1) = 0 AND InB%(Thyme%) = 1 THEN
   SmallestB%(Thyme%) = CountB%(Thyme%)
   GOTO DefineLargestB
ELSEIF OutB%(Thyme%) = 0 THEN
   SmallestB%(Thyme%) = SmallestB%(Thyme% - 1)
   GOTO DefineLargestB
ELSE
   SmallestB%(Thyme%) = SmallestB%(Thyme% - 1) + 1
END IF
DefineLargestB:
IF QueueB%(Thyme%) = 0 THEN
   LargestB%(Thyme%) = 0
   GOTO CheckForOutB
ELSE
   LargestB%(Thyme%) = CountB%(Thyme%)
END IF
CheckForOutB:
IF QueueB%(Thyme%) = 0 THEN
   GOTO NextThymeStep
ELSEIF XB!(Thyme%, SmallestB%(Thyme%)) < 1.5 * L! THEN
   OutB%(Thyme% + 1) = 0
   GOTO NextThymeStep
ELSE
   OutB%(Thyme% + 1) = 1
END IF
NextThymeStep:
NEXT Thyme%
'
'444444444444444444444444444444444444444444444444444444444444444444444444444
'
'555555555555555555555555555555555555555555555555555555555555555555555555555
'
```

```
'DEFINE POINT OF NO RETURN.
'
'555555555555555555555555555555555555555555555555555555555555555555555555
'
DefinePNRinLaneC:
pijrTime! = 1.7
BreakingDistanceC! = InitSpeedC!(1) ^ 2 / (30! * (FrictionC!(1) + g!))
dC1! = R! * Alpha1!
dC2! = BreakingDistanceC! - dC1!
dC3! = 1.47 * InitSpeedC!(1) * pijrTime!
PNRYC!(1) = -C! - dC2! - dC3!
'
'555555555555555555555555555555555555555555555555555555555555555555555555
'
'666666666666666666666666666666666666666666666666666666666666666666666666
'
'START A SINGLE SIMULATION RUN FOR A TOTAL RUN TIME OF 30 SECONDS.
'
'666666666666666666666666666666666666666666666666666666666666666666666666
'
TotalRunTime% = 30
T% = 1
AdvanceClock:
CheckForLaneC:
SmallestC%(T%) = 1
CarC% = 1
IF T% < ArrivalC%(1) THEN
   GOTO CheckForClock
ELSEIF T% = ArrivalC%(1) THEN
   SpeedC!(T%, CarC%) = InitSpeedC!(1)
   XC!(T%, CarC%) = .5 * L!
   YC!(T%, CarC%) = -250!
   GOTO CheckForPNR
ELSE
END IF
'
'666666666666666666666666666666666666666666666666666666666666666666666666
'
'777777777777777777777777777777777777777777777777777777777777777777777777
'
'CHECK WHETHER CAR C HAS PASSED PNR: (1) NO; GO TO NEXT TIME STEP.
'                                    (2) YES; CHECK FEEDBACK SIGNAL
'                                        AND ASSESS SAFETY.
'
'SIGNAL FEEDBACKS:
'           (1): SIGNAL$ = "WAIT"-----Check PreSignal from the last tick.
'                                -----Make Judgement2.
'                                -----Find next position.
'                                -----Send Signal and PreSignal.
'           (2): SIGNAL$ = "START"----Add one acceleration.
'                                 ----Skip Judgement2.
'                                 ----Send Signal.
'                                   --Check if left turning is completed.
'           (3): SIGNAL$ = "ACCEL"----Add anotherc acceleration until
'                                 ----initial speed is reached.
'                                 ----Skip Judgement2.
'                                 ----Send Signal.
'                                 ----Check if left turning is completed.
'           (4): SIGNAL$ = "MOVE ON"--Keep same speed.
'                                   --Skip Judgement1.
'                                   --Send Signal.
'                                   --Check if left turning is completed.
'           (5): SIGNAL$ = "DECEL"----Subtract one deceleration until
'                                 ----zero speed.
'                                 ----Skip Judgement1.
```

```
'                                ----Send Signal.
'           (6): SIGNAL$ = "RETRY"----Subtract one deceleration until
'                                ----zero speed.
'                                ----Make Judgement1.
'                                ----Send Signal.
'           (7): SIGNAL$ = ""---------PNR just reached.
'                           ---------=Retry.
'                           ---------Make Judgement1.
'                           ---------Send Signal.
'
'MAKING JUDGEMENT
'           (1) CLEAR VIEW: (A) IF SAFE, MOVE ON AND CROSS LANE A.
'                           (B) IF UNSAFE, DECELERATE AND COME TO A FULL STOP.
'           (2) BLOCKED VIEW: DECELERATE, REASSESS SAFETY AT NEXT TIME STEP.
'
'77777777777777777777777777777777777777777777777777777777777777777777777777
'
CheckForPNR:
IF YC!(T%, CarC%) >= PNRYC!(CarC%) THEN GOTO CheckSignalC
   SpeedC!(T% + 1, CarC%) = SpeedC!(T%, CarC%)
   XC!(T% + 1, CarC%) = XC!(T%, CarC%)
   YC!(T% + 1, CarC%) = YC!(T%, CarC%) + 1.47 * SpeedC!(T% + 1, CarC%) * deltaT%
IF YC!(T% + 1, CarC%) < PNRYC!(CarC%) THEN GOTO CheckForClock
   SignalC$(T%, CarC%) = ""
CheckSignalC:
'
SELECT CASE SignalC$(T%, CarC%)
   CASE "Start"
      GOTO StartC
   CASE "Accelerate"
      GOTO AccelerateC
   CASE "Move on"
      GOTO MoveOnC
   CASE "Decelerate"
      GOTO DecelerateC
   CASE "Retry"
      GOTO RetryC
   CASE "Wait"
      GOTO WaitC
   CASE ELSE
      GOTO RetryC
END SELECT
StartC:
SpeedC!(T% + 1, CarC%) = SpeedC!(T%, CarC%) + .8618 * AccelC!(CarC%) * deltaT%
CALL NextPosAccelC(T% + 1, CarC%, SpeedC!(), AccelC!(), XC!(), YC!(), R!,
                                                           L!, deltaT%)
SignalC$(T% + 1, CarC%) = "Accelerate"
   IF XC!(T% + 1, CarC%) > Spot2X! THEN
      OutC%(T% + 1) = 0
      GOTO CheckForClock
   ELSE
      ExtraAngle! = Alpha2!
      ExtraArc! = R! * ExtraAngle!
      ExtraTime! = (-2! * 1.47 * SpeedC!(T% + 1, CarC%)
                  + SQR((2! * 1.47 * SpeedC!(T% + 1, CarC%)) ^ 2
                  + 8! * AccelC!(CarC%) * ExtraArc!)) / (2! * AccelC!(CarC%))
      DepartureTime! = T% + ExtraTime!
      GOTO NextRun
   END IF
AccelerateC:
SpeedC!(T% + 1, CarC%) = SpeedC!(T%, CarC%) + .8618 * AccelC!(CarC%) * deltaT%
IF SpeedC!(T% + 1, CarC%) < InitSpeedC!(CarC%) THEN
   CALL NextPosAccelC(T% + 1, CarC%, SpeedC!(), AccelC!(), XC!(), YC!(), R!,
                                                              L!, deltaT%)
   SignalC$(T% + 1, CarC%) = "Accelerate"
```

```
      IF XC!(T% + 1, CarC%) > Spot2X! THEN
         OutC%(T% + 1) = 0
         GOTO CheckForClock
      ELSE
         ExtraAngle! = (Pi! / 2!) - ATN((YC!(T%, CarC%) + C!)
                                    / (XC!(T%, CarC%) + C!))
         ExtraArc! = R! * ExtraAngle!
         ExtraTime! = (-2! * 1.47 * SpeedC!(T% + 1, CarC%)
                     + SQR((2! * 1.47 * SpeedC!(T% + 1, CarC%)) ^ 2
                     + 8! * AccelC!(CarC%) * ExtraArc!)) / (2! * AccelC!(CarC%))
         DepartureTime! = T% + ExtraTime!
         GOTO NextRun
      END IF
ELSE
   SpeedC!(T% + 1, CarC%) = InitSpeedC!(CarC%)
   CALL NextPosSameC(T% + 1, CarC%, SpeedC!(), XC!(), YC!(), R!, L!, deltaT%)
   SignalC$(T% + 1, CarC%) = "Move on"
      IF XC!(T% + 1, CarC%) > Spot2X! THEN
        OutC%(T% + 1) = 0
        GOTO CheckForClock
      ELSE
         ExtraAngle! = (Pi! / 2!) - ATN((YC!(T%, CarC%) + C!)
                                    / (XC!(T%, CarC%) + C!))
         ExtraArc! = R! * ExtraAngle!
         ExtraTime! = (-2! * 1.47 * SpeedC!(T% + 1, CarC%)
                     + SQR((2! * 1.47 * SpeedC!(T% + 1, CarC%)) ^ 2
                     + 8! * AccelC!(CarC%) * ExtraArc!)) / (2! * AccelC!(CarC%))
         DepartureTime! = T% + ExtraTime!
         GOTO NextRun
      END IF
END IF
MoveOnC:
SpeedC!(T% + 1, CarC%) = SpeedC!(T%, CarC%)
CALL NextPosSameC(T% + 1, CarC%, SpeedC!(), XC!(), YC!(), R!, L!, deltaT%)
SignalC$(T% + 1, CarC%) = "Move on"
   IF XC!(T% + 1, CarC%) > Spot2X! THEN
      OutC%(T% + 1) = 0
      GOTO CheckForClock
   ELSE
      ExtraAngle! = (Pi! / 2!) - ATN((YC!(T%, CarC%) + C!) / (XC!(T%, CarC%)
                                                                   + C!))
      ExtraArc! = R! * ExtraAngle!
      ExtraTime! = ExtraArc! / SpeedC!(T% + 1, CarC%)
      DepartureTime! = T% + ExtraTime!
      GOTO NextRun
   END IF
DecelerateC:
SpeedC!(T% + 1, CarC%) = SpeedC!(T%, CarC%) - .6818 * DecelC!(CarC%) * deltaT%
IF SpeedC!(T% + 1, CarC%) <= 0! THEN
   SpeedC!(T% + 1, CarC%) = 0!
   XC!(T% + 1, CarC%) = Spot1X!
   YC!(T% + 1, CarC%) = Spot1Y!
   SignalC$(T% + 1, CarC%) = "Wait"
   PreSignal% = 999
   GOTO CheckForClock
ELSE
   DistanceTraveled! = 1.47 * SpeedC!(T% + 1, CarC%) * deltaT%
                     - .5 * DecelC!(CarC%) * deltaT% ^ 2
   SELECT CASE XC!(T%, CarC%)
      CASE IS = .5 * L!
         YC!(T% + 1, CarC%) = YC!(T%, CarC%) + DistanceTraveled!
         IF YC!(T% + 1, CarC%) > -C! THEN
            ArcLength! = DistanceTraveled! - (-C! - YC!(T%, CarC%))
            ThetaC1! = ArcLength! / R!
            XC!(T% + 1, CarC%) = R! * COS(ThetaC1!) - C!
```

```
            YC!(T% + 1, CarC%) = R! * SIN(ThetaC1!) - C!
            SignalC$(T% + 1, CarC%) = "Decelerate"
         ELSE
            XC!(T% + 1, CarC%) = .5 * L!
            SignalC$(T% + 1, CarC%) = "Decelerate"
         END IF
         CASE ELSE
            ArcLength! = DistanceTraveled!
            ThetaC1! = ATN((YC!(T%, CarC%) + C!) / (XC!(T%, CarC%) + C!))
            ThetaC2! = ArcLength! / R!
            IF (ThetaC1! + ThetaC2!) >= Alpha1! THEN
               XC!(T% + 1, CarC%) = Spot1X!
               YC!(T% + 1, CarC%) = Spot1Y!
               SignalC$(T% + 1, CarC%) = "Wait"
            ELSE
               XC!(T% + 1, CarC%) = R! * COS(ThetaC1! + ThetaC2!) - C!
               YC!(T% + 1, CarC%) = R! * SIN(ThetaC1! + ThetaC2!) - C!
               SignalC$(T% + 1, CarC%) = "Decelerate"
            END IF
      END SELECT
      GOTO CheckForClock
   END IF
RetryC:
IF XC!(T%, CarC%) >= (.5 * L!) THEN
   TurnDistance1! = R! * Alpha1! + (-C! - YC!(T%, CarC%))
   TurnTime1! = (-2! * 1.47 * SpeedC!(T%, CarC%)
            + SQR((2! * 1.47 * SpeedC!(T%, CarC%)) ^ 2
            + 8! * DecelC!(CarC%) * TurnDistance1!)) / (2! * DecelC!(CarC%))
   TurnDistance2! = R! * (Pi! / 2!) + (-C! - YC!(T%, CarC%))
   TurnTime2! = TurnDistance2! / SpeedC!(T%, CarC%)
ELSE
   TurnDistance1! = R! * (Alpha1! - ATN((YC!(T%, CarC%) + C!)
                                 / (XC!(T%, CarC%) + C!)))
   TurnTime1! = (-2! * 1.47 * SpeedC!(T%, CarC%)
            + SQR((2! * 1.47 * SpeedC!(T%, CarC%)) ^ 2
            + 8! * DecelC!(CarC%) * TurnDistance1!)) / (2! * DecelC!(CarC%))
   TurnDistance2! = R! * ((Pi! / 2!) - ATN((YC!(T%, CarC%) + C!)
                                    / (XC!(T%, CarC%) + C!)))
   TurnTime2! = TurnDistance2! / SpeedC!(T%, CarC%)
END IF
Define Danger Zone A1
DZ1X1Left! = -2! * L!
DZ1Y1! = Spot3Y! + OperSpeed! * (TurnTime1! - DriverTypeC%(CarC%))
DZ1X2Right! = -1! * L!
DZ1Y2! = Spot4Y! + OperSpeed! * (TurnTime2! + DriverTypeC%(CarC%))
CALL DriverSeatC(T%, CarC%, XC!(), YC!(), WidthC!(), LengthC!(),
                           DriverSeatXC!(), DriverSeatYC!())
Define Angle DZ1Right
AngleDZ1Right! = Pi! - ATN((DZ1Y2! - DriverSeatYC!(T%, CarC%))
                  / (DriverSeatXC!(T%, CarC%) - DZ1X2Right!))
Define Angle DZ1Left:
AngleDZ1Left! = Pi! - ATN((DZ1Y1! - DriverSeatYC!(T%, CarC%))
                  / (DriverSeatXC!(T%, CarC%) - DZ1X1Left!))
CheckForBlockedViewC1:
   IF QueueB%(T%) = 0 THEN
     JudgementA1$(T%) = "Clear View"
     GOTO MakeJudgementA1
   ELSE
   END IF
FOR CarB% = SmallestB%(T%) TO LargestB%(T%)
   CALL CornerB(T%, CarB%, XB!(), YB!(), WidthB!(), LengthB!(), RFCXB!(),
    RFCYB!(), LFCXB!(), LFCYB!(), RRCXB!(), RRCYB!(), LRCXB!(), LRCYB!())
   IF DriverSeatXC!(T%, CarC%) >= RRCXB!(T%, CarB%) THEN
      AngleBRR! = Pi! - ATN((RRCYB!(T%, CarB%) - DriverSeatYC!(T%, CarC%))
                        / (DriverSeatXC!(T%, CarC%) - RRCXB!(T%, CarB%)))
```

```
    ELSE
       AngleBRR! = ATN((RRCYB!(T%, CarB%) - DriverSeatYC!(T%, CarC%))
                   / (RRCXB!(T%, CarB%) - DriverSeatXC!(T%, CarC%)))
    END IF
    IF DriverSeatXC!(T%, CarC) >= RFCXB!(T%, CarB%) THEN
       AngleBRF! = Pi! - ATN((RFCYB!(T%, CarB%) - DriverSeatYC!(T%, CarC%))
                         / (DriverSeatXC!(T%, CarC%) - RFCXB!(T%, CarB%)))
    ELSE
       AngleBRF! = ATN((RFCYB!(T%, CarB%) - DriverSeatYC!(T%, CarC%))
                   / (RFCXB!(T%, CarB%) - DriverSeatXC!(T%, CarC%)))
    END IF
    IF AngleBRF! > AngleBRR! THEN
       AngleBLeft!(CarB%) = AngleBRF!
    ELSE
       AngleBLeft!(CarB%) = AngleBRR!
    END IF
 NEXT CarB%
 SELECT CASE QueueA%(T%)
    CASE IS = 0
       FOR CarB% = SmallestB%(T%) TO LargestB%(T%)
          IF RRCXB!(T%, CarB%) >= DriverSeatXC!(T%, CarC%) THEN
            JudgementA1$(T%) = "Clear View"
          ELSEIF AngleBLeft! <= AngleDZ1Right! THEN
             JudgementA1$(T%) = "Clear View"
          ELSE
             JudgementA1$(T%) = "Blocked View"
          GOTO MakeJudgementA1
          END IF
       NEXT CarB%
          GOTO MakeJudgementA1
    CASE ELSE
       FOR CarA% = SmallestA%(T%) TO LargestA%(T%)
          CALL CornerA(T%, CarA%, XA!(), YA!(), WidthA!(), LengthA!(), RFCXA!(),
           RFCYA!(), LFCXA!(), LFCYA!(), RRCXA!(), RRCYA!(), LRCXA!(), LRCYA!())
          AngleARight!(CarA%) = Pi! - ATN((LRCYA!(T%, CarA%)
                                - DriverSeatYC!(T%, CarC%))
                                / (DriverSeatXC!(T%, CarC%) - LRCXA!(T%, CarA%)))
       NEXT CarA%
       FOR CarA% = SmallestA%(T%) TO LargestA%(T%)
          IF AngleARight!(CarA%) >= AngleDZ1Left! THEN
             FOR CarB% = SmallestB%(T%) TO LargestB%(T%)
                IF RRCXB!(T%, CarB%) >= DriverSeatXC!(T%, CarC%) THEN
                   JudgementA1$(T%) = "Clear View"
                ELSEIF AngleBLeft! <= AngleDZ1Right! THEN
                   JudgementA1$(T%) = "Clear View"
                ELSE
                   JudgementA1$(T%) = "Blocked View"
                   GOTO MakeJudgementA1
                END IF
             NEXT CarB%
             GOTO MakeJudgementA1
          ELSE
             FOR CarB% = SmallestB%(T%) TO LargestB%(T%)
                IF AngleBLeft! > AngleDZ1Right! THEN
                   JudgementA1$(T%) = "Blocked View"
                   GOTO MakeJudgementA1
                ELSE
                   JudgementA1$(T%) = "Clear View"
                END IF
             NEXT CarB%
          END IF
       NEXT CarA%
 END SELECT
 MakeJudgementA1:
 SELECT CASE JudgementA1$(T%)
```

```
CASE "Blocked View"
   SpeedC!(T% + 1, CarC%) = SpeedC!(T%, CarC%)
                          - 1.47 * DecelC!(CarC%) * deltaT%
   IF SpeedC!(T% + 1, CarC%) <= 0! THEN
      SpeedC!(T% + 1, CarC%) = 0!
      XC!(T% + 1, CarC%) = Spot1X!
      YC!(T% + 1, CarC%) = Spot1Y!
      SignalC$(T% + 1, CarC%) = "Wait"
      PreSignal% = 999
   ELSE
      CALL NextPosDecelC(T% + 1, CarC%, SpeedC!(), DecelC!(), XC!(), YC!(),
                                                           R!, L!, deltaT%)
      SignalC$(T% + 1, CarC%) = "Retry"
   END IF
CASE "Clear View"
   FOR CarA% = SmallestA%(T%) TO LargestA%(T%)
      IF YA!(T%, CarA%) >= DZ1Y1! AND YA!(T%, CarA%) <= DZ1Y2! THEN
         SignalZoneA1$(T%) = "Dangerous"
         EXIT FOR
      ELSE
         SignalZoneA1$(T%) = "Safe"
      END IF
   NEXT CarA%
   SELECT CASE SignalZoneA1$(T%)
      CASE "Dangerous"
        SpeedC!(T% + 1, CarC%) = SpeedC!(T%, CarC%)
                               - .6818 * DecelC!(CarC%) * deltaT%
         IF SpeedC!(T% + 1, CarC%) <= 0! THEN
            SpeedC!(T% + 1, CarC%) = 0!
            XC!(T% + 1, CarC%) = Spot1X!
            YC!(T% + 1, CarC%) = Spot1Y!
            SignalC$(T% + 1, CarC%) = "Wait"
            PreSignal% = 999
         ELSE
            DistanceTraveled! = 1.47 * SpeedC!(T% + 1, CarC%) * deltaT%
                              - .5 * DecelC!(CarC%) * deltaT% ^ 2
            SELECT CASE XC!(T%, CarC%)
               CASE IS = .5 * L!
                  YC!(T% + 1, CarC%) = YC!(T%, CarC%) + DistanceTraveled!
                  IF YC!(T% + 1, CarC%) > -C! THEN
                     ArcLength! = DistanceTraveled! - (-C! - YC!(T%, CarC%))
                     ThetaC1! = ArcLength! / R!
                     XC!(T% + 1, CarC%) = R! * COS(ThetaC1!) - C!
                     YC!(T% + 1, CarC%) = R! * SIN(ThetaC1!) - C!
                  ELSE
                     XC!(T% + 1, CarC%) = .5 * L!
                  END IF
               CASE ELSE
                  ArcLength! = DistanceTraveled!
                  ThetaC1! = ATN((YC!(T%, CarC%) + C!)
                           / (XC!(T%, CarC%) + C!))
                  ThetaC2! = ArcLength! / R!
                  IF (ThetaC1! + ThetaC2!) >= Alpha1! THEN
                     XC!(T% + 1, CarC%) = Spot1X!
                     YC!(T% + 1, CarC%) = Spot1Y!
                  ELSE
                     XC!(T% + 1, CarC%) = R! * COS(ThetaC1! + ThetaC2!) - C!
                     YC!(T% + 1, CarC%) = R! * SIN(ThetaC1! + ThetaC2!) - C!
                  END IF
            END SELECT
            SignalC$(T% + 1, CarC%) = "Decelerate"
         END IF
      CASE "Safe"
         SpeedC!(T% + 1, CarC%) = SpeedC!(T%, CarC%)
         CALL NextPosSameC(T% + 1, CarC%, SpeedC!(), XC!(), YC!(), R!,
```

```
                                                        L!, deltaT%)
            SignalC$(T% + 1, CarC%) = "Move on"
            IF XC!(T% + 1, CarC%) > Spot2X! THEN
               OutC%(T% + 1) = 0
            ELSE
               ExtraAngle! = (Pi! / 2!) - ATN((YC!(T%, CarC%) + C!)
                                        / (XC!(T%, CarC%) + C!))
               ExtraArc! = R! * ExtraAngle!
               ExtraTime! = ExtraArc! / SpeedC!(T% + 1, CarC%)
               GOTO NextRun
            END IF
         CASE ELSE
      END SELECT
   CASE ELSE
END SELECT
GOTO CheckForClock
WaitC:
PreSignal% = PreSignal% - 1
IF PreSignal% = 0 THEN
   SpeedC!(T% + 1, CarC%) = 1.47 * AccelC!(CarC%) * deltaT%
   DistanceTraveled! = 1.47 * SpeedC!(T%, CarC%) * deltaT%
                     + .5 * AccelC!(CarC%) * deltaT% ^ 2
   ArcLength! = DistanceTraveled!
   ThetaC1! = ATN((YC!(T%, CarC%) + C!) / (XC!(T%, CarC%) + C!))
   ThetaC2! = ArcLength! / R!
   IF (ThetaC1! + ThetaC2!) >= (Pi! / 2) THEN
      XC!(T% + 1, CarC%) = -C! - R! * (ThetaC1! + ThetaC2! - (Pi! / 2))
      YC!(T% + 1, CarC%) = .5 * L!
   ELSE
      XC!(T% + 1, CarC%) = R! * COS(ThetaC1! + ThetaC2!) - C!
      YC!(T% + 1, CarC%) = R! * SIN(ThetaC1! + ThetaC2!) - C!
   END IF
      SignalC$(T% + 1, CarC%) = "Start"
      GOTO CheckForClock
ELSEIF PreSignal% = 1 THEN
      SpeedC!(T% + 1, CarC%) = 0!
      SignalC$(T% + 1, CarC%) = "Wait"
      XC!(T% + 1, CarC%) = Spot1X!
      YC!(T% + 1, CarC%) = Spot1Y!
      GOTO CheckForClock
ELSE
END IF
Define Danger Zone A2
StartUpTimeLoss! = 2!
CrossATime! = SQR(8! * AccelC!(CarC%) * (R! * Alpha2!))
            / (2! * DecelC!(CarC%)) + StartUpTimeLoss!
DZ2X1Left! = -2! * L!
DZ2Y1! = Spot1Y!
DZ2X2Right! = -1! * L!
DZ2Y2! = Spot4Y! - OperSpeed! * (CrossATime! + StartUpTimeLoss!
                                 + DriverTypeC%(CarC%))
CALL DriverSeatC(T%, CarC%, XC!(), YC!(), WidthC!(), LengthC!(),
                            DriverSeatXC!(), DriverSeatYC!())
Define AngleDZ2Right
AngleDZ2Right! = Pi! - ATN((DZ2Y2! - DriverSeatYC!(T%, CarC%))
               / (DriverSeatXC!(T%, CarC%) - DZ2X2Right!))
Define AngleDZ2Left
AngleDZ2Left! = Pi! - ATN((DZ2Y2! - DriverSeatYC!(T%, CarC%))
              / (DriverSeatXC!(T%, CarC%) - DZ2X1Left!))
CheckForBlockedViewC2:
IF QueueB%(T%) = 0 THEN
  JudgementA1$(T%) = "Clear View"
  GOTO MakeJudgementA2
ELSE
END IF
```

```
FOR CarB% = SmallestB%(T%) TO LargestB%(T%)
   CALL CornerB(T%, CarB%, XB!(), YB!(), WidthB!(), LengthB!(), RFCXB!(),
    RFCYB!(), LFCXB!(), LFCYB!(), RRCXB!(), RRCYB!(), LRCXB!(), LRCYB!())
   IF DriverSeatXC!(T%, CarC%) >= RRCXB!(T%, CarB%) THEN
      AngleBRR! = Pi! - ATN((RRCYB!(T%, CarB%) - DriverSeatYC!(T%, CarC%))
                        / (DriverSeatXC!(T%, CarC%) - RRCXB!(T%, CarB%)))
   ELSE
      AngleBRR! = ATN((RRCYB!(T%, CarB%) - DriverSeatYC!(T%, CarC%))
                  / (RRCXB!(T%, CarB%) - DriverSeatXC!(T%, CarC%)))
   END IF
   IF DriverSeatXC!(T%, CarC) >= RFCXB!(T%, CarB%) THEN
      AngleBRF! = Pi! - ATN((RFCYB!(T%, CarB%) - DriverSeatYC!(T%, CarC%))
                        / (DriverSeatXC!(T%, CarC%) - RFCXB!(T%, CarB%)))
   ELSE
      AngleBRF! = ATN((RFCYB!(T%, CarB%) - DriverSeatYC!(T%, CarC%))
                  / (RFCXB!(T%, CarB%) - DriverSeatXC!(T%, CarC%)))
   END IF
   IF AngleBRF! > AngleBRR! THEN
      AngleBLeft!(CarB%) = AngleBRF!
   ELSE
      AngleBLeft!(CarB%) = AngleBRR!
   END IF
NEXT CarB%
SELECT CASE QueueA%(T%)
   CASE IS = 0
      FOR CarB% = SmallestB%(T%) TO LargestB%(T%)
         IF RRCXB!(T%, CarB%) >= DriverSeatXC!(T%, CarC%) THEN
            JudgementA1$(T%) = "Clear View"
         ELSEIF AngleBLeft! <= AngleDZ2Right! THEN
            JudgementA1$(T%) = "Clear View"
         ELSE
            JudgementA1$(T%) = "Blocked View"
            GOTO MakeJudgementA2
         END IF
      NEXT CarB%
      GOTO MakeJudgementA2
   CASE ELSE
      FOR CarA% = SmallestA%(T%) TO LargestA%(T%)
         CALL CornerA(T%, CarA%, XA!(), YA!(), WidthA!(), LengthA!(), RFCXA!(),
          RFCYA!(), LFCXA!(), LFCYA!(), RRCXA!(), RRCYA!(), LRCXA!(), LRCYA!())
         AngleARight!(CarA%) = Pi! - ATN((LRCYA!(T%, CarA%)
                               - DriverSeatYC!(T%, CarC%))
                               / (DriverSeatXC!(T%, CarC%) - LRCXA!(T%, CarA%)))
      NEXT CarA%
      FOR CarA% = SmallestA%(T%) TO LargestA%(T%)
         IF AngleARight!(CarA%) >= AngleDZ2Left! THEN
            FOR CarB% = SmallestB%(T%) TO LargestB%(T%)
               IF RRCXB!(T%, CarB%) >= DriverSeatXC!(T%, CarC%) THEN
                  JudgementA1$(T%) = "Clear View"
               ELSEIF AngleBLeft! <= AngleDZ2Right! THEN
                  JudgementA1$(T%) = "Clear View"
               ELSE
                  JudgementA1$(T%) = "Blocked View"
                  GOTO MakeJudgementA2
               END IF
            NEXT CarB%
         ELSE
            FOR CarB% = SmallestB%(T%) TO LargestB%(T%)
               IF AngleBLeft! > AngleDZ2Right! THEN
                  JudgementA1$(T%) = "Blocked View"
                  GOTO MakeJudgementA2
               ELSE
                  JudgementA1$(T%) = "Clear View"
               END IF
            NEXT CarB%
```

```
         END IF
      NEXT CarA%
END SELECT
MakeJudgementA2:
SELECT CASE JudgementA1$(T%)
   CASE "Blocked View"
      SpeedC!(T% + 1, CarC%) = 0!
      XC!(T% + 1, CarC%) = Spot1X!
      YC!(T% + 1, CarC%) = Spot1Y!
      SignalC$(T% + 1, CarC%) = "Wait"
      PreSignal% = 999
   CASE "Clear View"
      FOR CarA% = SmallestA%(T%) TO LargestA%(T%)
         IF YA!(T%, CarA%) >= DZ2Y1! AND YA!(T%, CarA%) <= DZ2Y2! THEN
            SignalZoneA2$(T%) = "Not Enough Gap"
            EXIT FOR
         ELSE
            SignalZoneA2$(T%) = "Start 2 Seconds Later"
         END IF
      NEXT CarA%
      SELECT CASE SignalZoneA2$(T%)
         CASE "Not Enough Gap"
            SpeedC!(T% + 1, CarC%) = 0!
            XC!(T% + 1, CarC%) = Spot1X!
            YC!(T% + 1, CarC%) = Spot1Y!
            SignalC$(T% + 1, CarC%) = "Wait"
            PreSignal% = 999
         CASE "Start 2 Seconds Later"
            SpeedC!(T% + 1, CarC%) = 0
            XC!(T% + 1, CarC%) = Spot1X!
            YC!(T% + 1, CarC%) = Spot1Y!
            SignalC$(T% + 1, CarC%) = "Wait"
            PreSignal% = 2
         CASE ELSE
      END SELECT
   CASE ELSE
END SELECT
CheckForClock:
T% = T% + deltaT%
IF T% <= TotalRunTime% THEN GOTO AdvanceClock
   DepartureTime! = T%
'
'8888888888888888888888888888888888888888888888888888888888888888888888888
'
'9999999999999999999999999999999999999999999999999999999999999999999999999
'
'CALCULATE DELAY AND AFTER MULTIPLE RUNS FIND AVERAGE DELAY.
'
'PRINT RESULTS AND RELATED PARAMETERS.
'
'9999999999999999999999999999999999999999999999999999999999999999999999999
'
NextRun:
   NormalTurnTime! = (R! * (Pi! / 2!) + (P1Y! + 250!))
                                 / (1.47 * InitSpeedC!(1))
   ActualTurnTime! = DepartureTime! - ArrivalC%(CarC%)
   DelayC!(NumberOfRun%) = ActualTurnTime! - NormalTurnTime!
   TotalDelayC! = TotalDelayC! + DelayC!(NumberOfRun%)
   NumberOfRun% = NumberOfRun% + 1
IF NumberOfRun% <= TotalNumberOfRun% THEN GOTO Simulation
StopRun:
   AverageDelayC! = TotalDelayC! / NumberOfRun%
   PRINT "Total number of runs = "; TotalNumberOfRun%
   PRINT "AverageDelayC! = "; AverageDelayC!
   PRINT "lambdaA = "; lambdaA%, "lambdaB = "; lambdaB%
```

```
   PRINT "Allowable Gap (reflecting driver type) = "; DriverTypeC%(1)
   PRINT "SpeedA = "; InitSpeedA!(CarA%), "SpeedB = "; InitSpeedB!(CarB%),
                                                "SpeedC = "; InitSpeedC!(1)
END
'
'9999999999999999999999999999999999999999999999999999999999999999999999999999
'
SUB CornerA (T%, CarA%, XA!(), YA!(), WidthA!(), LengthA!(), RFCXA!(), RFCYA!(),
                     LFCXA!(), LFCYA!(), RRCXA!(), RRCYA!(), LRCXA!(), LRCYA!())
RFCXA!(T%, CarA%) = XA!(T%, CarA%) - .5 * WidthA!(CarA%)
RFCYA!(T%, CarA%) = YA!(T%, CarA%) - .5 * LengthA!(CarA%)
LFCXA!(T%, CarA%) = XA!(T%, CarA%) + .5 * WidthA!(CarA%)
LFCYA!(T%, CarA%) = YA!(T%, CarA%) - .5 * LengthA!(CarA%)
RRCXA!(T%, CarA%) = XA!(T%, CarA%) - .5 * WidthA!(CarA%)
RRCYA!(T%, CarA%) = YA!(T%, CarA%) + .5 * LengthA!(CarA%)
LRCXA!(T%, CarA%) = XA!(T%, CarA%) + .5 * WidthA!(CarA%)
LRCYA!(T%, CarA%) = YA!(T%, CarA%) + .5 * LengthA!(CarA%)
END SUB

SUB CornerB (T%, CarB%, XB!(), YB!(), WidthB!(), LengthB!(), RFCXB!(), RFCYB!(),
                     LFCXB!(), LFCYB!(), RRCXB!(), RRCYB!(), LRCXB!(), LRCYB!())
IF (YB!(T%, CarB%) >= C!) THEN
   RFCXB!(T%, CarB%) = XB!(T%, CarB%) - .5 * WidthB!(CarB%)
   RFCYB!(T%, CarB%) = YB!(T%, CarB%) - .5 * LengthB!(CarB%)
   LFCXB!(T%, CarB%) = XB!(T%, CarB%) + .5 * WidthB!(CarB%)
   LFCYB!(T%, CarB%) = YB!(T%, CarB%) - .5 * LengthB!(CarB%)
   RRCXB!(T%, CarB%) = XB!(T%, CarB%) - .5 * WidthB!(CarB%)
   RRCYB!(T%, CarB%) = YB!(T%, CarB%) + .5 * LengthB!(CarB%)
   LRCXB!(T%, CarB%) = XB!(T%, CarB%) + .5 * WidthB!(CarB%)
   LRCYB!(T%, CarB%) = YB!(T%, CarB%) + .5 * LengthB!(CarB%)
ELSEIF (YB!(T%, CarB%) = -.5 * L!) THEN
   RFCXB!(T%, CarB%) = XB!(T%, CarB%) + .5 * LengthB!(CarB%)
   RFCYB!(T%, CarB%) = YB!(T%, CarB%) - .5 * WidthB!(CarB%)
   LFCXB!(T%, CarB%) = XB!(T%, CarB%) + .5 * LengthB!(CarB%)
   LFCYB!(T%, CarB%) = YB!(T%, CarB%) + .5 * WidthB!(CarB%)
   RRCXB!(T%, CarB%) = XB!(T%, CarB%) - .5 * LengthB!(CarB%)
   RRCYB!(T%, CarB%) = YB!(T%, CarB%) - .5 * WidthB!(CarB%)
   LRCXB!(T%, CarB%) = XB!(T%, CarB%) - .5 * LengthB!(CarB%)
   LRCYB!(T%, CarB%) = YB!(T%, CarB%) + .5 * WidthB!(CarB%)
ELSE
   ThetaB! = Pi! + ATN((YB!(T%, CarB%) - C!) / (XB!(T%, CarB%) - C!))
   RFCXB!(T%, CarB%) = XB!(T%, CarB%) + .5 * WidthB!(CarB%) * COS(ThetaB!)
                                    - .5 * LengthB!(CarB%) * SIN(ThetaB!)
   RFCYB!(T%, CarB%) = YB!(T%, CarB%) + .5 * WidthB!(CarB%) * SIN(ThetaB!)
                                    + .5 * LengthB!(CarB%) * COS(ThetaB!)
   LFCXB!(T%, CarB%) = XB!(T%, CarB%) - .5 * WidthB!(CarB%) * COS(ThetaB!)
                                    - .5 * LengthB!(CarB%) * SIN(ThetaB!)
   LFCYB!(T%, CarB%) = YB!(T%, CarB%) - .5 * WidthB!(CarB%) * SIN(ThetaB!)
                                    + .5 * LengthB!(CarB%) * COS(ThetaB!)
   RRCXB!(T%, CarB%) = XB!(T%, CarB%) + .5 * WidthB!(CarB%) * COS(ThetaB!)
                                    + .5 * LengthB!(CarB%) * SIN(ThetaB!)
   RRCYB!(T%, CarB%) = YB!(T%, CarB%) + .5 * WidthB!(CarB%) * SIN(ThetaB!)
                                    - .5 * LengthB!(CarB%) * COS(ThetaB!)
   LRCXB!(T%, CarB%) = XB!(T%, CarB%) - .5 * WidthB!(CarB%) * COS(ThetaB!)
                                    + .5 * LengthB!(CarB%) * SIN(ThetaB!)
   LRCYB!(T%, CarB%) = YB!(T%, CarB%) - .5 * WidthB!(CarB%) * SIN(ThetaB!)
                                    - .5 * LengthB!(CarB%) * COS(ThetaB!)
END IF
END SUB

SUB CornerC (T%, CarC%, XC!(), YC!(), WidthC!(), LengthC!(), RFCXC!(), RFCYC!(),
                     LFCXC!(), LFCYC!(), RRCXC!(), RRCYC!(), LRCXC!(), LRCYC!())
IF YC!(T%, CarC%) <= -C! THEN
   RFCXC!(T%, CarC%) = XC!(T%, CarC%) + .5 * WidthC!(CarC%)
   RFCYC!(T%, CarC%) = YC!(T%, CarC%) + .5 * LengthC!(CarC%)
```

```
   LFCXC!(T%, CarC%) = XC!(T%, CarC%) - .5 * WidthC!(CarC%)
   LFCYC!(T%, CarC%) = YC!(T%, CarC%) + .5 * LengthC!(CarC%)
   RRCXC!(T%, CarC%) = XC!(T%, CarC%) + .5 * WidthC!(CarC%)
   RRCYC!(T%, CarC%) = YC!(T%, CarC%) - .5 * LengthC!(CarC%)
   LRCXC!(T%, CarC%) = XC!(T%, CarC%) - .5 * WidthC!(CarC%)
   LRCYC!(T%, CarC%) = YC!(T%, CarC%) - .5 * LengthC!(CarC%)
ELSE
   ThetaC! = ATN((YC!(T%, CarC%) + C!) / (XC!(T%, CarC%) + C!))
   RFCXC!(T%, CarC%) = XC!(T%, CarC%) + .5 * WidthC!(CarC%) * COS(ThetaC!)
                                     - .5 * LengthC!(CarC%) * SIN(ThetaC!)
   RFCYC!(T%, CarC%) = YC!(T%, CarC%) + .5 * WidthC!(CarC%) * SIN(ThetaC!)
                                     + .5 * LengthC!(CarC%) * COS(ThetaC!)
   LFCXC!(T%, CarC%) = XC!(T%, CarC%) - .5 * WidthC!(CarC%) * COS(ThetaC!)
                                     - .5 * LengthC!(CarC%) * SIN(ThetaC!)
   LFCYC!(T%, CarC%) = YC!(T%, CarC%) - .5 * WidthC!(CarC%) * SIN(ThetaC!)
                                     + .5 * LengthC!(CarC%) * COS(ThetaC!)
   RRCXC!(T%, CarC%) = XC!(T%, CarC%) + .5 * WidthC!(CarC%) * COS(ThetaC!)
                                     + .5 * LengthC!(CarC%) * SIN(ThetaC!)
   RRCYC!(T%, CarC%) = YC!(T%, CarC%) + .5 * WidthC!(CarC%) * SIN(ThetaC!)
                                     - .5 * LengthC!(CarC%) * COS(ThetaC!)
   LRCXC!(T%, CarC%) = XC!(T%, CarC%) - .5 * WidthC!(CarC%) * COS(ThetaC!)
                                     + .5 * LengthC!(CarC%) * SIN(ThetaC!)
   LRCYC!(T%, CarC%) = YC!(T%, CarC%) - .5 * WidthC!(CarC%) * SIN(ThetaC!)
                                     - .5 * LengthC!(CarC%) * COS(ThetaC!)
END IF
END SUB

SUB DriverSeatC (T%, CarC%, XC!(), YC!(), WidthC!(), LengthC!(),
                              DriverSeatXC!(), DriverSeatYC!())
IF (YC!(T%, CarC%) <= (-R! + .5 * L!)) THEN
   DriverSeatXC!(T%, CarC%) = XC!(T%, CarC%) - .3 * WidthC!(CarC%)
   DriverSeatYC!(T%, CarC%) = YC!(T%, CarC%) + .1 * LengthC!(CarC%)
ELSE
   ThetaC! = ATN(YC!(T%, CarC%) + C! / XC!(T%, CarC%) + C!)
   DriverSeatXC!(T%, CarC%) = XC!(T%, CarC%)
                            - .3 * WidthC!(CarC%) * COS(ThetaC!)
                            - .1 * LengthC!(CarC%) * SIN(ThetaC!)
   DriverSeatYC!(T%, CarC%) = YC!(T%, CarC%)
                            - .3 * WidthC!(CarC%) * SIN(ThetaC!)
                            + .1 * LengthC!(CarC%) * COS(ThetaC!)
END IF
END SUB

SUB NextPosAccelC (T%, CarC%, SpeedC!(), AccelC!(), XC!(), YC!(), R!,
                                                         L!, deltaT%)
   Pi! = 4 * ATN(1!)
   C! = R! - .5 * L!
   DistanceTraveled! = 1.47 * SpeedC!(T%, CarC%) * deltaT%
                     + .5 * AccelC!(CarC%) * deltaT% ^ 2
ArcLength! = DistanceTraveled!
ThetaC1! = ATN((YC!(T%, CarC%) + C!) / (XC!(T%, CarC%) + C!))
ThetaC2! = ArcLength! / R!
IF (ThetaC1! + ThetaC2!) >= (Pi! / 2) THEN
   XC!(T% + 1, CarC%) = -C! - R! * (ThetaC1! + ThetaC2! - (Pi! / 2))
   YC!(T% + 1, CarC%) = .5 * L!
ELSE
   XC!(T% + 1, CarC%) = R! * COS(ThetaC1! + ThetaC2!) - C!
   YC!(T% + 1, CarC%) = R! * SIN(ThetaC1! + ThetaC2!) - C!
END IF
END SUB

SUB NextPosDecelC (T%, CarC%, SpeedC!(), DecelC!(), XC!(), YC!(), R!,
                                                         L!, deltaT%)
   Pi! = 4 * ATN(1!)
   C! = R! - .5 * L!
```

```
   DistanceTraveled! = 1.47 * SpeedC!(T% + 1, CarC%) * deltaT%
                            - .5 * DecelC!(CarC%) * deltaT% ^ 2
SELECT CASE XC!(T%, CarC%)
   CASE IS = .5 * L!
      YC!(T% + 1, CarC%) = YC!(T%, CarC%) + DistanceTraveled!
      IF YC!(T% + 1, CarC%) > -C! THEN
         ArcLength! = DistanceTraveled! - (-C! - YC!(T%, CarC%))
         ThetaC1! = ArcLength! / R!
        XC!(T% + 1, CarC%) = R! * COS(ThetaC1!) - C!
        YC!(T% + 1, CarC%) = R! * SIN(ThetaC1!) - C!
      ELSE
        XC!(T% + 1, CarC%) = .5 * L!
      END IF
   CASE ELSE
      ArcLength! = DistanceTraveled!
      ThetaC1! = ATN((YC!(T%, CarC%) + C!) / (XC!(T%, CarC%) + C!))
      ThetaC2! = ArcLength! / R!
      IF (ThetaC1! + ThetaC2!) >= Alpha1! THEN
         XC!(T% + 1, CarC%) = Spot1X!
         YC!(T% + 1, CarC%) = Spot1Y!
      ELSE
         XC!(T% + 1, CarC%) = R! * COS(ThetaC1! + ThetaC2!) - C!
         YC!(T% + 1, CarC%) = R! * SIN(ThetaC1! + ThetaC2!) - C!
      END IF
END SELECT
END SUB

SUB NextPosSameC (T%, CarC%, SpeedC!(), XC!(), YC!(), R!, L!, deltaT%)
   Pi! = 4 * ATN(1!)
   C! = R! - .5 * L!
   DistanceTraveled! = 1.47 * SpeedC!(T%, CarC%) * deltaT%
SELECT CASE XC!(T%, CarC%)
   CASE IS = .5 * L!
      YC!(T% + 1, CarC%) = YC!(T%, CarC%) + DistanceTraveled!
      IF YC!(T% + 1, CarC%) > -C! THEN
         ArcLength! = DistanceTraveled! - (-C! - YC!(T%, CarC%))
         ThetaC1! = ArcLength! / R!
        XC!(T% + 1, CarC%) = R! * COS(ThetaC1!) - C!
        YC!(T% + 1, CarC%) = R! * SIN(ThetaC1!) - C!
      ELSE
        XC!(T% + 1, CarC%) = .5 * L!
      END IF
   CASE ELSE
      ArcLength! = DistanceTraveled!
      ThetaC1! = ATN((YC!(T%, CarC%) + C!) / (XC!(T%, CarC%) + C!))
      ThetaC2! = ArcLength! / R!
      IF (ThetaC1! + ThetaC2!) >= (Pi! / 2) THEN
         XC!(T% + 1, CarC%) = -C! - R! * (ThetaC1! + ThetaC2! - (Pi! / 2))
         YC!(T% + 1, CarC%) = .5 * L!
      ELSE
         XC!(T% + 1, CarC%) = R! * COS(ThetaC1! + ThetaC2!) - C!
         YC!(T% + 1, CarC%) = R! * SIN(ThetaC1! + ThetaC2!) - C!
      END IF
END SELECT
END SUB
```

Bibliography

1. McShane, W.R. and R.P. Roess, "Traffic Engineering", 1990

2. Sagi, G.S. and L.R. Campbell, "Vehicle Delay at Signalized Intersection, Theory and Practice", Traffic Engineering, 1969

3. Homburger, W.S., editor, "Transportation and Traffic Engineering Handbook", 2nd ed., 1982

4. Berry, D.S. and C.J. Van Til, "A Comparison of Three Methods for Measuring Delay at Intersections", Traffic Engineering, 1954

5. Reilley, W.R. and C.C. Gardner, "Technique or Measuring Delay at Intersections", TRB 644, 1977

6. "Highway Capacity Manual", TRB Special Report 209, 1985

7. "A Policy on Geometric Design of Highway and Streets", AASHTO, 1984

8. Law, A.M. and W.D. Kelton, "Simulation Modeling and Analysis", 1982

9. Pinataro, L.J., "Traffic Engineering Theory and Practice", 1972

10. Khisty, C.J., "Transportation Engineering", 1990