

California State University, San Bernardino

**CSUSB ScholarWorks**

---

Theses Digitization Project

John M. Pfau Library

---

2002

## Animated vehicle turning path simulation system on an Internet/ Intranet browser

Yuwen Deng

Follow this and additional works at: <https://scholarworks.lib.csusb.edu/etd-project>



Part of the [Software Engineering Commons](#)

---

### Recommended Citation

Deng, Yuwen, "Animated vehicle turning path simulation system on an Internet/Intranet browser" (2002).  
*Theses Digitization Project*. 2091.  
<https://scholarworks.lib.csusb.edu/etd-project/2091>

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact [scholarworks@csusb.edu](mailto:scholarworks@csusb.edu).

ANIMATED VEHICLE TURNING PATH SIMULATION SYSTEM

ON AN INTERNET/INTRANET BROWSER

---

A Project  
Presented to the  
Faculty of  
California State University,  
San Bernardino

---

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science  
in  
Computer Science

---

by  
Yuwen Deng  
December 2002

ANIMATED VEHICLE TURNING PATH SIMULATION SYSTEM

ON AN INTERNET/INTRANET BROWSER

---

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by


Yuwen Deng


December 2002

Approved by:

  
Dr. Richard J. Botting, Chair, Computer Science

*Oct 21/02*  
Date

  
Dr. Kerstin Voigt, Computer Science

  
Dr. George M. Georgiou, Computer Science

## ABSTRACT

Animated simulation is a fast and easy way to test many engineering designs' accuracy. By combining the accessibility of internet/intranet browser and Java graphic programming, the project achieved the goal of providing civil engineers a tool to help with the road design without any paperwork going around the company. It helps engineers predict vehicles movement without having to test it on the road.

First, the system loads a map as a background. Then, the user will be able to draw a road on the map for a vehicle to follow. After the road has been selected the user will then select a vehicle to be simulated driving the road. The user will see if the road is wide enough for the vehicle or where the road needs to be expanded.

## ACKNOWLEDGMENTS

The accomplishment of this project was due to the helpful advice of Dr. Richard J. Botting and Dr. Owen Murphy on the design of drawing of the road. The insightful comments and suggestions were really the inspiration of the design.

Also thanks to Mr. YauHuei Deng at China Engineering Consultants, Inc. providing mathematical formula of calculation on the behavior of vehicle turning movement. Without the formula the simulation will not be as close to reality as one wished to achieve.

Also thanks to my family and friends' support and encouragements. The project would not have been so smooth without that.

Last but not least, I thank GE Transportation Systems for their tolerance of my divided efforts on both working and studying at the same time.

## TABLE OF CONTENTS

ABSTRACT .....	iii
ACKNOWLEDGMENTS .....	iv
LIST OF TABLES .....	vii
LIST OF FIGURES .....	viii
CHAPTER ONE: INTRODUCTION	
Purpose of the Project .....	1
Scope of the Project .....	2
Significance of the Project .....	3
Limitation of the Project .....	5
CHAPTER TWO: REVIEW OF RELATED WORK	
Curve Design .....	6
Vehicle Movement Simulation .....	11
CHAPTER THREE: METHODOLOGY	
System Architecture and Design .....	14
System Deployment .....	14
User Case and Scenarios .....	15
Sequence Diagram .....	17
Class Diagram .....	17
Curve Design .....	20
Vehicle Simulation Algorithm .....	23
CHAPTER FOUR: SUMMARY AND CONCLUDING REMARKS .....	26

APPENDIX A: GLOSSARY .....	28
APPENDIX B: SOURCE CODE .....	31
REFERENCES .....	53

## LIST OF TABLES

Table 1. System Requirements .....	4
------------------------------------	---



## LIST OF FIGURES

Figure 1.	Bezier Curve .....	7
Figure 2.	Smoothness of Bezier Curve Segment .....	9
Figure 3.	Trailer Truck Turning Path .....	10
Figure 4.	Example of Vehicle Movement .....	12
Figure 5.	Deployment Diagram .....	14
Figure 6.	User Case Diagram .....	15
Figure 7.	Sequence Diagram .....	18
Figure 8.	Class Diagram .....	19
Figure 9.	Composite Bezier Curve .....	20
Figure 10.	Java Code for Second Degree Bezier Curve .	22

## CHAPTER ONE

### INTRODUCTION

#### Purpose of the Project

When a vehicle takes a sharp turn at a narrow intersection or ramp it can leave the road and cause an accident. Roads must be designed to allow vehicles that make wide turns safely. This is not easy to do by hand and paper design. A simulation system can help engineers reduce accidents. Many conditions will affect the width that a vehicle needs; such as the size of the vehicle, the turning radius, the number of trailers of the vehicle, and distance between hitch and axis, etc. This proposed simulation system would be a good tool for engineers to simulate the turning path of a vehicle. Hopefully, by using this simulation system, the road will be more precisely planned, and will be easier for drivers to use.

In general, the current vehicle types could be divided into the following categories [3]:

Passenger Car (P): Such as a sedan, sports utility vehicle, or minivan.

Single Unit Truck or Bus (SU): Trucks with no trailer or hitch that moves as a single unit. Example would be a school bus, rental truck, etc.

Trailer Trucks (WB-XX): cars with trailers or cars that have hitches to connect two or more parts in the vehicle. Each part will have ability to move in different direction, example will be semi-truck, RV with a trailer of car, pick-up truck with a boat, etc.

In the simulation, a map will be loaded to the system first as a background. Then, the user will be able to draw his/her road on the map. After that, he/she will be able to select a vehicle to run the simulation test. During the simulation, the user could pause the animation at any time to check if there is any point that the vehicle hit the edge of the road. This could tell the user that if the road is wide enough for the vehicle or not.

#### Scope of the Project

The Project "Animated Vehicle Turning Path Simulation System on a Internet/Intranet Browser" deals with both the design, and implementation of this Simulation System. It was inspired by an older simulation system, which has been used for decades in China Engineering Consultants, Inc. as

a helpful tool when dealing with road intersection design. The original way to design the road is semi-computerized. The map had been scanned and digitized by AutoCAD. Then the engineer will draw the desired route on the AutoCAD file. A program call DCAD, gives the coordinate value when given the proper variables, then computes and gives the outermost tracking line of the vehicle's corners, this line will then be added into the AutoCAD file (.DWG). Finally the engineer can open the file in AutoCAD and see the result.

The original procedures are not user friendly and needs to run on different software: AutoCAD and DCAD. Besides, DCAD is a MSDOS application. This causes users many troubles when switching between AutoCAD and DCAD. Also the program is not interactive, the user can only see the result after finished the design. This deprived the user of the power of doing incremental design.

Base on this purpose, the project software system has the requirements list at table 1.

### Significance of the Project

The significant achievements of the project are:  
first achieve animated vehicle movement in Java 2D

graphical programming; second explore and find an easy way of computing the path of vehicle using Composite Bezier Curves.

Table 1. System Requirements

	Client side:	Server and Development side:
Operating System:	Red Hat Linux 7X, MS Windows 9X or later	Red Hat Linux 7X, MS Windows 9X or later
Memory:	64MB recommend or higher	64 MB recommend or higher
Free disk space:	200 MB recommended	200 MB recommended
Browser:	Netscape 4.5, Internet Explorer 5.0 or higher support java 1.4 script	Java Development Kit 1.4 Beta (SDK 1.4 Beta) or newer
Map files:		1:200 scale Kept in applet rectory as in (.GIF) form

## Limitation of the Project

The Project has several limitations as below:

First, because the networked design of the system, the software requires IE or Netscape browser to run. Also, since the language used is Java, the loading time for each execution takes a little more time than we expected.

Second, the mathematical formula base on traditional vehicle design, some newly designed vehicle may not be able to simulate in this system. An example will be the newly designed GMC truck, which gives the rear wheel the ability to change direction during turning. This kind of new design does not fit the traditional car design. It will require different kind of formula to simulate the behavior of the movement.

Third, the direction path that users draw was implemented with second-degree composite Bezier curves. Thus it is only first degree continuous. If the user accidentally clicks two points at the same position, it will still show a kink in the curve.

## CHAPTER TWO

### REVIEW OF RELATED WORK

#### Curve Design

Since most driving paths cannot have shapes that can be described by simple analytic functions, much research has found it is better to define curves in a piecewise manner. The continuity and smoothness across the joins between the pieces can be built into the parametrisation of the sections on either side.

Because the tangents, normals, curvatures are needed to be determined, polynomial functions of the parameters are an obvious choice. High degree polynomial can describe complex curves, but require a large number of coefficients whose physical significance is difficult to grasp. Thus they are inappropriate for a road designer. Moreover, the use of high degree polynomials may introduce unwanted oscillations in the curve (kinks in the curve). Thus this project uses three Point Bezier polynomial curves as the base and uses a composite way to join two lines or curve to form a smooth continuous curve.

a. The Bernstein-Bezier polynomial curve:

Bezier (1970) [5] describes a general polynomial curve given by

$$r = r(u) = \sum \left[ \binom{n}{i} u^i (1-u)^{n-i} r_i \right]$$

Where  $r_0, r_1, r_2, \dots, r_n$  are the position vectors of the  $n+1$  vertices  $p_0, p_1, p_2, \dots, p_n$  of a generalized characteristic 'polygon'. The Bezier cubic curve is an example of this general curve where  $n=3$ .

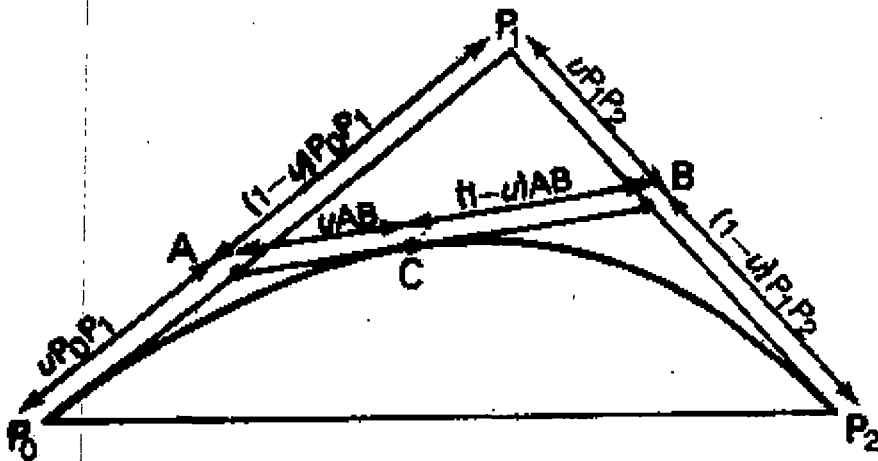


Figure 1. Bezier Curve



The point with parameter  $u$  on the curve can be constructed as follows. The point A which divides  $P_0 P_1$  in the ratio  $u: 1-u$  is jointed to the point B which divides  $P_1 P_2$  in the same ratio (see Figure 1). The segment AB is now divided in the same ratio at the point C.

Thus C is the point with parameter  $u$  on the parametric quadratic curve. Bezier shows that this method of construction can be extended to higher order curves.

#### b. Composite Bezier Curves approximation:

The cubic Bezier segment, introduced in section a is given by

$$r(u) = (1-u)^3 r_0 + 3u(1-u)^2 r_1 + 3u^2(1-u) r_2 + u^3 r_3, \quad 0 \leq u \leq 1$$

Here  $r_0$ ,  $r_1$ ,  $r_2$  and  $r_3$  are the vertices of the characteristic polygon to which the curve is an approximation. Bezier is less restrictive than Ferguson [5] in inter-segment continuity conditions. Whereas Ferguson's method is used primarily for curve fitting, Bezier's approach offers the extra freedom needed for curve design.

For example, to construct a curve that is determined by 7 vertices. Instead of a 6-degree Bezier curve, a

better idea is to join two 3-degree Bezier Segments together.

As showed in Figure 2,  $r_3^{(1)}$  is also  $r_0^{(2)}$ . Besides that, to maintain the continuity of tangent direction we require

$$3/a_1 (r_3^{(1)} - r_2^{(1)}) = 3/a_2 (r_1^{(2)} - r_0^{(2)}) = T$$

Where  $a_1$ ,  $a_2$  are the magnitudes of  $r'^{(1)}(1)$  and  $r'^{(2)}(0)$  respectively, which we now permit to be different. Thus the two segments are collinear.

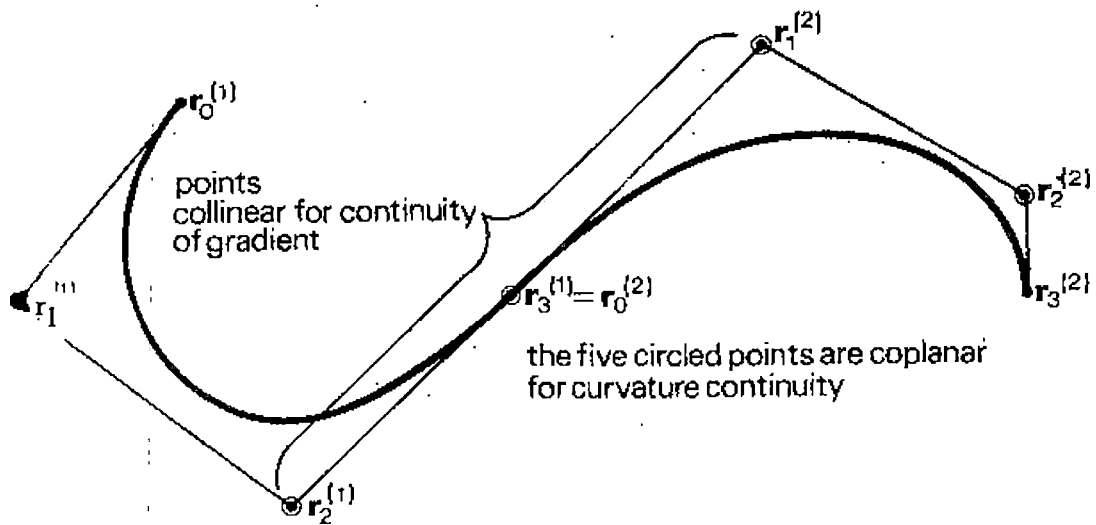


Figure 2. Smoothness of Bezier Curve Segment

Using this result it is possible to build a composite Bezier curve with positional, gradient and curvature continuity.

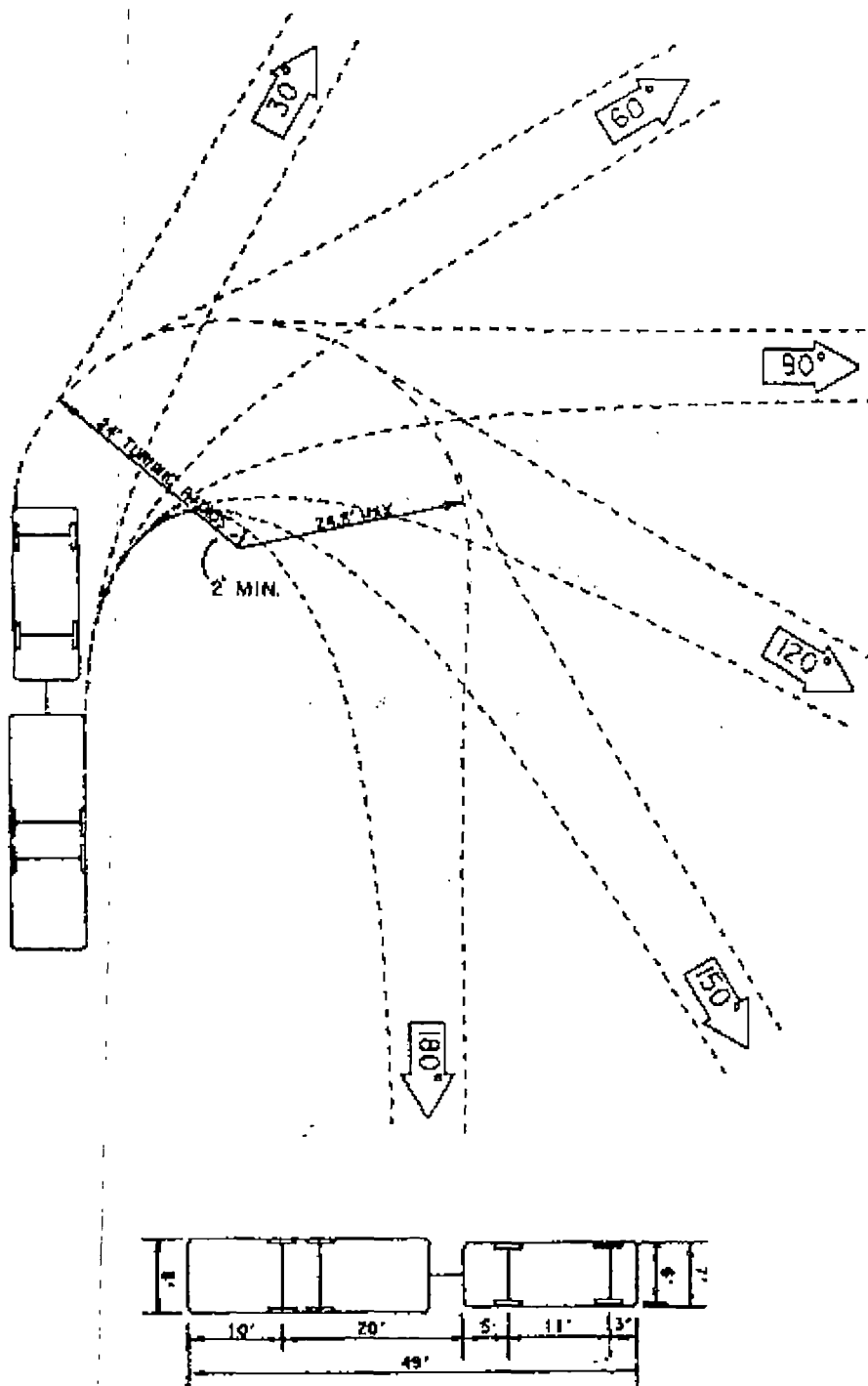


Figure 3. Trailer Truck Turning Path

## Vehicle Movement Simulation

In 1990, American Association of State Highway and Transportation Officials (AASHTO) [1] published "A Policy on Geometric Design of Highways and Streets" presented the minimum turning paths for 15 different kinds of vehicles. Example shows in figure 3.

The principal dimensions affecting design are the minimum turning radius, the tread width, the wheelbase, and the path of inner rear tire. These principal dimensions have then been set as a standard of highway and street designs.

In 1991, Mr. YauHuei Deng [2] purposed the following formula. Since the front tire goes almost as directed, it could be seen as a control point in the vehicle movement. Consider single unit vehicle, the rear tire's position determined the angel of the vehicle, and by related coordinate, the rest of the car body's position could also be determined.

So, to calculate the rear tire position becomes the key of the simulation. Suppose  $P_i=(XP_i, YP_i)$  is the control point before movement.  $P_j=(XP_j, YP_j)$  is the control point after a small time period of movement.  $Q_i=(XQ_i, YQ_i)$  is the

rear tire before movement, and  $Q_j = (XQ_j, YQ_j)$  is the rear tire after movement.

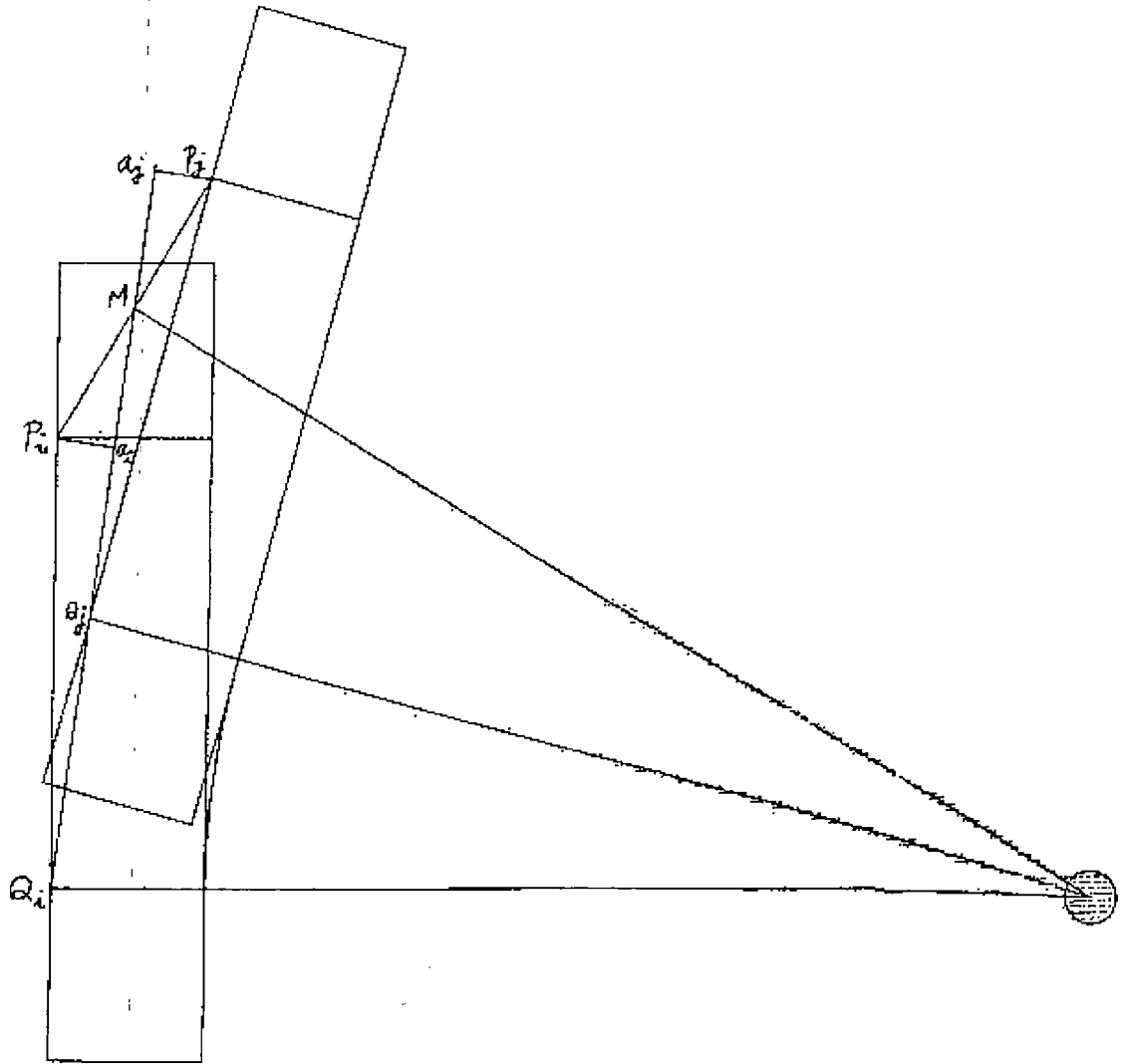


Figure 4. Example of Vehicle Movement

The X and Y coordinate of Q can then be determined as follows.

$$DX = (XP_i + XP_j) / 2 - XQ_i$$

$$DY = (YP_i + YP_j) / 2 - YQ_i$$

$$S = ((XP_j - XP_i) * DX + (YP_j - YP_i) * DY) / (DX * DX + DY * DY)$$

$$XQ_j = XQ_i + S * DX$$

$$YQ_j = YQ_i + S * DY$$

Once we have the first unit of the vehicle's simulation, the hitch point could be seen as the control point for second unit (Point P), then use the same formula could get the Q of second unit. Thus multiple unit vehicles could also be simulated.

## CHAPTER THREE

### METHODOLOGY

#### System Architecture and Design

##### System Deployment

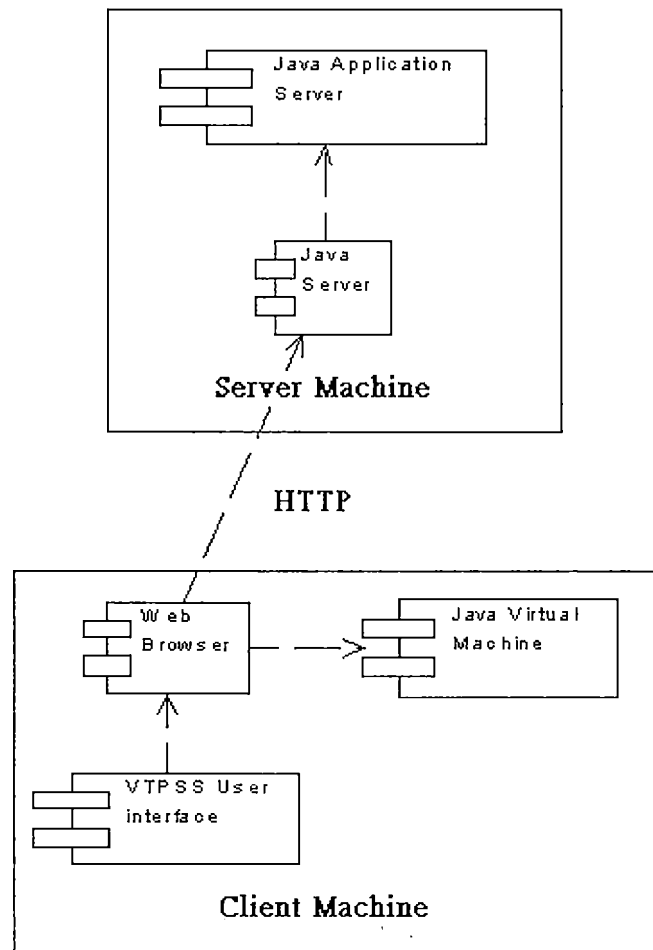


Figure 5. Deployment Diagram

This system has a client side and a server side.  
Connection between client and server is http and will go

through a web browser. The Deployment diagram [4] showed in Figure 5.

#### User Case and Scenarios

User Case Diagram [4] is as Figure 6.

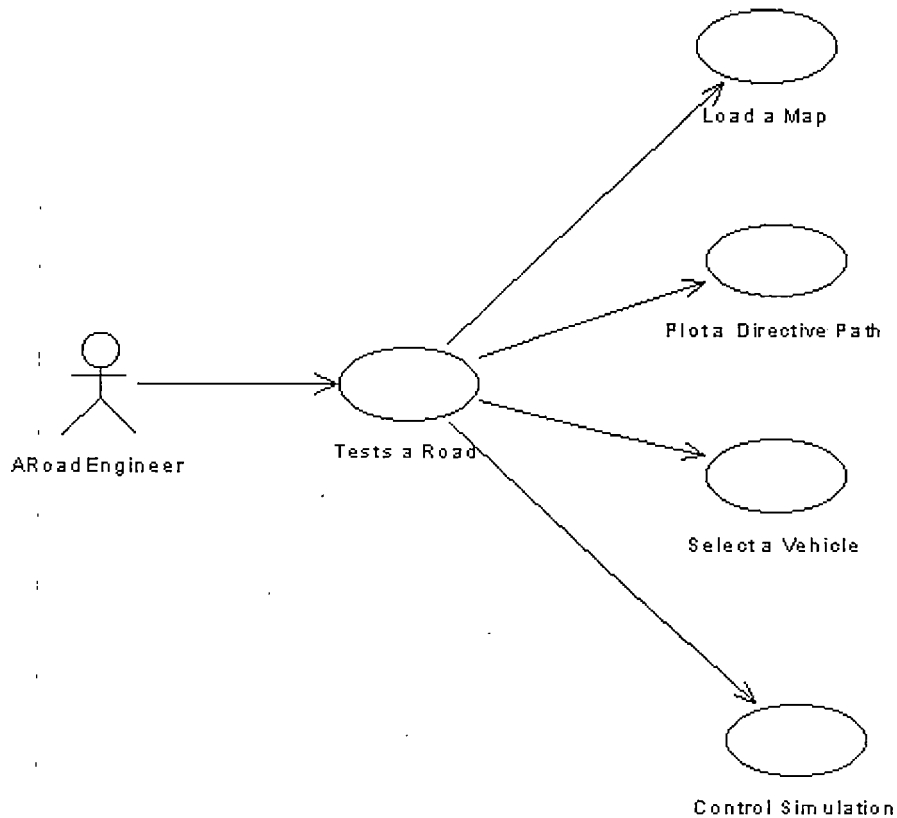


Figure 6. User Case Diagram

A typical scenario will be the following:

1. The user starts the system
2. The user requests to load a map into the system



3. The system loads the map on the browser
4. The user draws a desired control points on the map
5. The system plots the possible path according to the control points
6. The user moves the control points to adjust the path
7. The system re-plots the path to fit the new control points
8. The user selects a vehicle to run the simulation
9. The system draws the selected car in the show car panel
10. The user may wants to switch different kinds of vehicles
11. The system shows different kinds of vehicles according to user selection
12. The user starts the simulation by hitting the run button
13. The system starts the simulation
14. The user hits the pause button to freeze the simulation
15. The system stops the animation and shows the last frame of the simulation.

- 16.The user uses print button from the browser to print the frame
- 17.The system prints the frame through the web browser's default printer
- 18.The user hits run button again to resume the simulation
- 19.The system continues the animation simulation
- 20.The system ends simulation
- 21.The user closes the browser of click to different web page

#### Sequence Diagram

The Sequence Diagram showed at Figure 7. [4]

#### Class Diagram

This project is object oriented [7]. The three major classes are the VTPSS class, the ShowcarPanel class, and the Animatedpanel class. VTPSS class inherits from JApplet class [6]. ShowcarPanel and AnimatedPanel classes inherit from JPanel [6]. AnimatedPanel class also implements the

runable interface [6] to accomplish animation. Points use

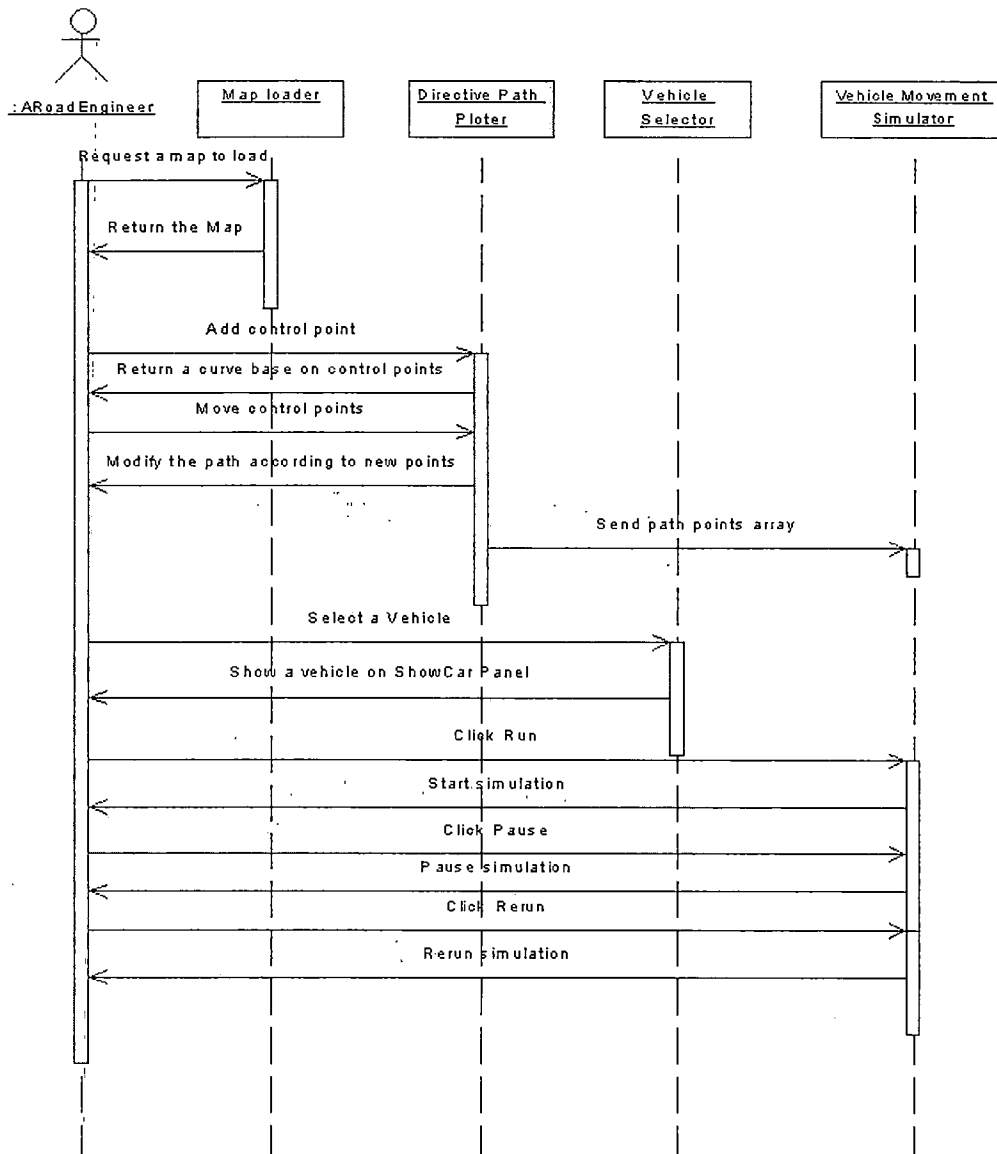


Figure 7. Sequence Diagram

to draw vehicles are kept in objects of arrays of doubles for the simplicity to access X and Y axis values also for

the fast computability when doing the computation of relative axis values versus absolute axis values. The class diagram showed at Figure 8. [4]

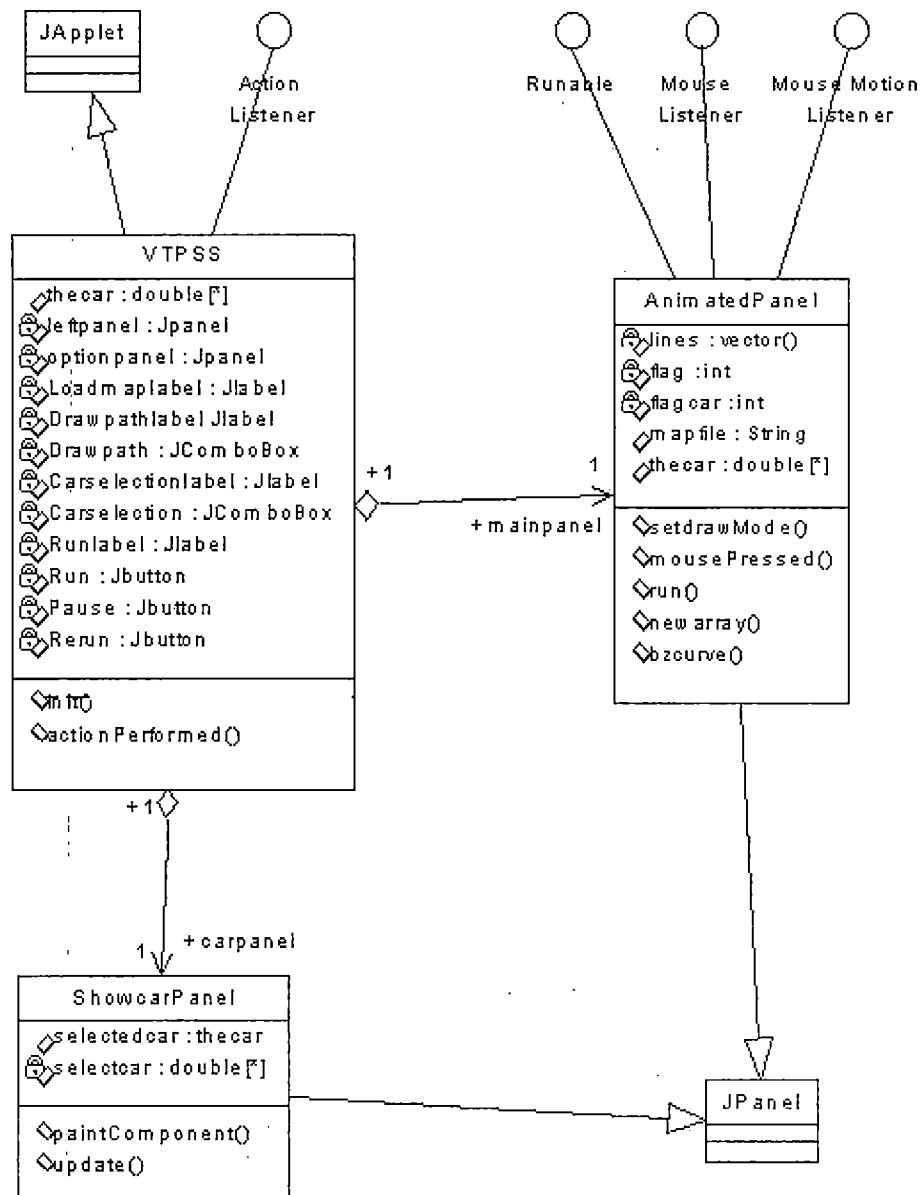


Figure 8. Class Diagram

## Curve Design

The curve used in this project is second-degree composite Bezier curve. The reasons to choose this algorithm are: First, it is the Bezier curve that goes closest to the control point. Second it is easier to compute and modify.

The actual way to do the second-degree composite Bezier curve is:

1. Suppose we have 5 control points  $P_0$ ,  $P_1$ ,  $P_2$ ,  $P_3$ , and  $P_4$ ; Shows as Figure9.

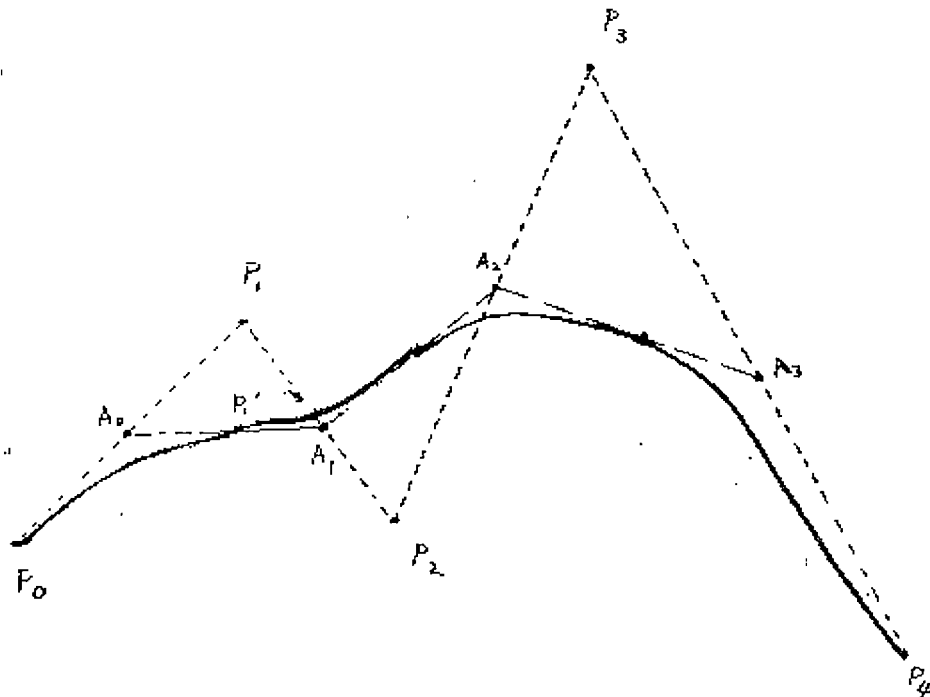


Figure 9. Composite Bezier Curve

2. Get the med-point of P0 and P1 as A0. Do the same for each two points to get A1, A2 and A3
3. Get the mid-point of A0 and A1 as P1'. Do the same for each two points to get P2' and P3'
4. Make a new array as {P0, A0, P1', A1, P2', A2, P3', A3, and P4}
5. Take each three point in the array with one point overlap to make a Bezier curve segment. {P0, A0, P1'}, {P1', A1, P2'}, ... {P3', A3, P4}
6. The Java to calculate the second-degree Bezier curve listed as Figure 10.

N is the number of points between P0 and P1. Once decided the number of point needed in between P0 and P1. Base on the distance between P0 and P1, This formula will give the proper X and Y coordinate. By connect very close points the smooth line could be plot use this Bezier curve function.

Since the each joint point P1', P2', and P3' are on the same line with it's previous and following point. (Example: A0, P1' and A1 are on the same line) So, the segments are first-degree continuous.

```

DD= (px0-px1)*(px0-px1) + (py0-py1)*(py0-py1);

D01 = Math.pow(DD, 0.5);

DD= (px1-px2)*(px1-px2) + (py1-py2)*(py1-py2);

D12 = Math.pow(DD, 0.5);

N = (int)((D01 + D12) / 5.0 +1.0);

for (int i = 1; i <= N; i++){

    t = (double)i/(double)N;

    x0 = px0 * (1.0 -t) + px1 * t;

    y0 = py0 * (1.0 -t) + py1 * t;

    x1 = px1 * (1.0 -t) + px2 * t;

    y1 = py1 * (1.0 -t) + py2 * t;

    x = x0 * (1.0 -t) + x1 * t;

    y = y0 * (1.0 -t) + y1 * t;

    bzcount +=2;

    afterbzarray[bzcount] = x;

    afterbzarray[bzcount+1] = y;

```

Figure 10. Java Code for Second Degree Bezier Curve

## Vehicle Simulation Algorithm

The algorithm is based on the previous work that has been proved working by Mr. YauHuei Deng at China Engineering Consultants, Inc. the steps are as follow:

1. Suppose the present control point  $P_i$  will move to next point at  $P_j$ .

2. Calculate  $Q_j$  by the following formula:

$$DX = (XP_i + XP_j) / 2 - XQ_i$$

$$DY = (YP_i + YP_j) / 2 - YQ_i$$

$$S = ((XP_j - XP_i) * DX + (YP_j - YP_i) * DY) / (DX * DX + DY * DY)$$

$$XQ_j = XQ_i + S * DX$$

$$YQ_j = YQ_i + S * DY$$

3. Use  $P_j$ ,  $Q_j$ , CarLength (distance between P and Q), and the relative coordinate (use P as 0,0 point) of each point to calculate the new position of each point (example as point A)

$$XA(\text{new}) = XP_j - \text{carcoordinate}(YA) / \text{CarLength} * (XQ_j - XP_j) - \text{carcoordinate}(XA) / \text{CarLength} * (YQ_j - YP_j);$$

$$YA(\text{new}) = YP_j - \text{carcoordinate}(YA) / \text{CarLength} * (YQ_j - YP_j) + \text{carcoordinate}(XA) / \text{CarLength} * (XQ_j - XP_j);$$



4. For a single unit vehicle, this is enough for all the points that was in the vehicle. Just run every point in the vehicle through the same formula as point A. the  $A_j$  could be found. Than plot all the new points of the vehicle at the window.

5. For a multiple unit vehicle, the hitch point's new position ( $H_j$ ) could be calculated from previous formula. This hitch point is also the control point of second unit. Base on the same formula the  $Q_j$  could be given.

$$DX = (XH_1 + XH_j) / 2 - XQ_1$$

$$DY = (YH_1 + YH_j) / 2 - YQ_1$$

$$S = ((XH_j - XH_1) * DX + (YH_j - YH_1) * DY) / (DX * DX + DY * DY)$$

$$XQ_j = XQ_1 + S * DX$$

$$YQ_j = YQ_1 + S * DY$$

6. Once got  $H_j$  and  $Q_j$ . Use the same formula could got the rest of the point in second unit of vehicle. Here the CarLength is distance between H and Q, and the relative coordinate is base on H (use H as 0,0 point) (example as point A)

$$XA(\text{new}) = XH_j - \text{carcoordinate}(YA) / \text{CarLength} * (XQ_j - XH_j) - \text{carcoordinate}(XA) / \text{CarLength} * (YQ_j - YH_j);$$

$$YA(\text{new}) = YH_j - \text{carcoordinate}(YA) / \text{CarLength} * (YQ_j - Yh_j) + \text{carcoordinate}(XA) / \text{CarLength} * (XQ_j - XH_j);$$

7. Thus for multiple unit vehicles could also be computed in the same formula

8. The way to give user the animation feeling is just a matter of keep re-flashing each position of the vehicle in a constant period of time so give the user a feeling of moving

## CHAPTER FOUR

### SUMMARY AND CONCLUDING REMARKS

The Animated Vehicle Turning Path Simulation System on Internet / Intranet Web Browser presented in this project is intended to provide civil engineers a easy-to-use all functional simulation system that could help them with the Highway and Street Design. The two major achievements in this project are: First, it successfully implements vehicle movement trace algorithm using Java graphic animation. Second, it implements an easy to calculate and useful way to plot a moving direction path base on several control points.

However, due to the limited time of the project, there are still many improvement could be done in this project.

1. Based on the variety of vehicles on the road, user should be allowed to create his/her own desired vehicle to run this simulation.
2. Maps could be in different scale, giving the system an ability to adjust the vehicle's size to fit the map's scale could do a further improvement.

3. A more advance thought about this system is that by given an existing road, a perfect system should be able to try to give a most likely fit driving path. Also, when running the simulation, system should be able to catch the point where the vehicle hit the edge of the road. So user could know this problem spot and fix it without run and pause the program to catch the problem spot by hand.
4. The users should be able to import and export files describing roads from other applications used by civil engineers such as AutoCAD.

Though, this project is not perfect, and needs many improvements, many technique has been learned and applied in this project. Especially the animation and 2D graphic programming ability has been a good learning experience for the student.

I wish to inspire more researches in this field and hope, by presenting this project to civil engineers will really help them in road design and make the design process easier and the future road more safe for drivers.

## APPENDIX A

### GLOSSARY

AVTPSS	Animated Vehicle Turing Path Simulation System
Bezier Curve	Bernstein-Bezier proposed a polynomial curve at 1970 to plot a smooth curve base on vertices
AASHTO	American Association of Street Highway and Transportation Officials
CECI	China Engineering Consultants, Inc.
JDK	Abbreviation for Java Development Kit provide java programmer built-in methods and classes
UML	Unified Modeling Language is a object oriented modeling language for specifying, visualizing, and documenting the artifacts of an object oriented system under development
GUI	Graphic User Interface, and interface that has image as well as words on the screen
Animation	Use paint and repaint in a very short period of time to give user a visual feeling of object movement on the screen

Vehicle	an automatic powered machine that transport person of cargos on Highway or Street
Applet	Program that was written in Java programming language and runs only on top of web browser like Netscape or Internet Explorer
Control Point	Point on each unit of a vehicle, which determines the moving direction and distance of that unit
Server	Networked computer which supports java applet and also where map files are located.

APPENDIX B

SOURCE CODE



1. The HTML file "VTPSS.html":

```
<html>
  <head>
    <title>Vehicle Turning Path Simulation System
(0.3)</title>
  </head>
  <body>
    <h2>Vehicle Turning Path Simulation System (0.3)<a
href="..\..\..\WINDOWS/Desktop/my%20project%20documentatio
n.doc">** See Project Documentation **</a></h2>
    <applet code=VTPSS.class width=960 height=550>
      alt="Your browser understands the <APPLET> tag
but isn't running the applet, for some reason."
      Your browser is completely ignoring the
      <APPLET> tag!
    </applet>
  </body>
</html>
```

2. VTPSS.java:

```
/*
*****

```

This is apart of master project for student: Yuwen Deng  
Name of the project is:

ANIMATED VEHICLE TURNING PATH SIMULATION SYSTEM  
ON AN INTERNET/INTRANET BROWSER

This Project is instructed by Dr. Richard J. Botting  
with committee Members: Dr. Kerstin Voigt  
Dr. George M. Georgiou  
Dr. Owen Murphy

at California State University, San Bernardino

```
*****
/

```

```
/*
*****

```

Description:

This is the main class for this program

It will load three major panels

1. mainpanel: an object of AnimatedPanel

2. carpanel: an object of ShowcarPanel

3. leftpanel: has all the function buttons and text area

Main program will call setdrawmode method to switch from  
different mode of the mainpanel; call update method to

update selected car coordinate array in carpanel

Action Listener is implemented in this class to have this  
object response to mouse and keyboard events.

```
*****
/

```

```
import java.awt.event.*;
import java.awt.*;
import java.awt.geom.*;
import java.applet.*;
```

```

import java.util.Vector;
import javax.swing.*;

//----- VTPSS class start-----
--
public class VTPSS extends JApplet implements
ActionListener{
    double rotateangel= 0; // constant to rotate the
coordinate
    double xdistent= 0; // X distance from origin
    double ydistent= 0; // Y distance from origin
    public double thecar[] ={}; // the array to keep selected
vehicle coordinate array
    // the following are standard vehicle coordinates array
that fit in 1:200 scale map
    public double testv[]={
        0,0,
        0,-25,
        0,0,
        0,-30,
        15,-30
    };// testv[] is a test draw for test vehicle movement
    public double vc[]={
        0,0,
        0,-25,
        0,0,
        8,-5,
        -8,-5,
        -8,-2,
        8,-2,
        8,-32,
        -8,-32,
        -8,-5,
        0,0
    };// vc[] is array for Passenger Car coordinate

    public double st[]={
        0,0,
        0,-40,
        0,0,
        8,-5,
        -8,-5,
        -8,-2,
        8,-2,
        8,-52,
        -8,-52,
        -8,-5,
        0,0
    };// st[] is array for Single Unit Truck coordinate

```

```

public double tt[]={
    0,0,
    0,-25,
    0,0,
    8,-5,
    -8,-5,
    -8,-2,
    8,-2,
    8,-30,
    -8,-30,
    -8,-5,
    0,0,

    0,-22,
    0,-82,
    0,-22,
    8,-40,
    -8,-40,
    -8,-32,
    8,-32,
    8,-87,
    -8,-87,
    -8,-40,
    0,-22
}; // tt[] is array for trailer truck coordinate

public double rv[]={
    0,0,
    0,-50,
    0,0,
    8,-5,
    -8,-5,
    -8,-2,
    8,-2,
    8,-55,
    -8,-55,
    -8,-5,
    0,0,

    0,-50,
    0,-82,
    0,-50,
    8,-62,
    -8,-62,
    -8,-60,
    8,-60,
    8,-87,
    -8,-87,
    -8,-62,

```

```

    0,-50
}; // rv[] is array for RV with a trailer coordinate

public double dt[]={
    0,0,
    0,-25,
    0,0,
    8,-5,
    -8,-5,
    -8,-2,
    8,-2,
    8,-30,
    -8,-30,
    -8,-5,
    0,0,

    0,-22,
    0,-82,
    0,-22,
    8,-40,
    -8,-40,
    -8,-32,
    8,-32,
    8,-87,
    -8,-87,
    -8,-40,
    0,-22,

    0,-82,
    0,-142,
    0,-82,
    8,-100,
    -8,-100,
    -8,-92,
    8,-92,
    8,-147,
    -8,-147,
    -8,-100,
    0,-82

}; // dt[] is array for Double Trailer Truck coordinate

// define and create a mainpanel which is an object of
// animatedpanel but has it's own method to load map files
Animatedpanel mainpanel = new Animatedpanel(0){
    public void paintComponent(Graphics showcargraphic){
        super.paintComponent(showcargraphic);
        Graphics2D showcargraphic2 =
        (Graphics2D) showcargraphic;

```

```

        Image mapImage = getImage(getDocumentBase(),
mapfile); // use getDocumentBase() method that provide by
JApplet to load map file
        showcargraphic2.drawImage(mapImage, 0, 0, this);
// use flag that was defined in AnimatedPanel to set
different drawing mode.
// flag=1 set the panel in run mode
// flag=2 set the panel in load map mode
// flag=3 set the panel in add or edit control point mode
// flag=4 set the panel in pause mode
// flag=5 set the panel in rerun mode
        if ( (mainpanel.flag == 1) ||
            (mainpanel.flag == 4) ||
            (mainpanel.flag == 5)) {
            showcargraphic2.setPaint(Color.red);
            for (int i = 0; i <= bzcount-2; i+=2) {

showcargraphic2.drawLine((int)pz[i], (int)pz[i+1], (int)pz[i
+2], (int)pz[i+3] );
            }
            showcargraphic2.setPaint(Color.blue);
            if (thecar.length != 0) {
                for (int i=0; i < thecar.length-3 ; i+=2) {
                    showcargraphic2.drawLine(
(int)thecar[i], (int)thecar[i+1],
(int)thecar[i+2], (int)thecar[i+3] );
                }
            }
            if (mainpanel.flag == 3) {
                showcargraphic2.setPaint(Color.green);
                for (int i=0; i < mylines.length - 3; i+=2) {
                    showcargraphic2.fillRect(mylines[i]-2,
mylines[i+1]-2, 4, 4);
                    showcargraphic2.fillRect(mylines[i+2]-2,
mylines[i+3]-2, 4, 4);
                }
                showcargraphic2.drawLine(mylines[i], mylines[i+1], mylines[i
+2], mylines[i+3]);
            }
            if ((oldx > 0) && (oldy > 0)) {
                showcargraphic2.setPaint(Color.yellow);
                showcargraphic2.fillRect((int)oldx,
(int)oldy, 8, 8);
            }

            showcargraphic2.setPaint(Color.red);
            for (int i = 0; i <= bzcount-2; i+=2) {

```

```

showcargraphic2.drawLine((int)pz[i],(int)pz[i+1],(int)pz[i
+2], (int)pz[i+3] );
    }
}
};

JPanel leftpanel = new JPanel();
ShowcarPanel carpanel = new ShowcarPanel( thecar );
JPanel optionpanel = new JPanel();
    // load map options
JLabel Loadmaplabel = new JLabel("Type the map to
load:", JLabel.LEFT);
JTextField Mapname = new JTextField("put file name
here");
JButton Load = new JButton("Load");
    // Draw path options
JLabel Drawpathlabel = new JLabel("Draw the Path:",
JLabel.LEFT);
JComboBox Drawpath = new JComboBox();
    // Car selection options
JLabel Carselectionlabel = new JLabel("Select
Vehicle:", JLabel.LEFT);
JComboBox Carselection = new JComboBox();
    // Run options
JLabel Runlabel = new JLabel("Animation control
buttons:", JLabel.LEFT);
JButton Run = new JButton("Run");
JButton Pause = new JButton("Pause");
JButton Rerun = new JButton("Rerun");

// Methods....

public void actionPerformed(ActionEvent event){
    String command = event.getActionCommand();
    if (command == "Load"){
        String filename = Mapname.getText();
        mainpanel.updatemap(filename);
        mainpanel.setflag(2);
    }
    if (command == "Run"){
        mainpanel.setflag(1);
    }
    if (command == "Pause"){
        mainpanel.setflag(4);
    }
    if (command == "Rerun"){
        mainpanel.setflag(5);
    }
}

```

```

JComboBox source = (JComboBox)event.getSource();
String item = (String)source.getSelectedItem();
if (item == "Passenger Car"){
    thecar = vc;
    mainpanel.selectcar(thecar, 1);
    carpanel.update(thecar);
}
if (item == "Single Unit Truck") {
    thecar = st;
    mainpanel.selectcar(thecar, 2);
    carpanel.update(thecar);
}
if (item == "Trailer Truck") {
    thecar = tt;
    mainpanel.selectcar(thecar, 3);
    carpanel.update(thecar);
}
if (item == "RV with trailer") {
    thecar = rv;
    mainpanel.selectcar(thecar, 4);
    carpanel.update(thecar);
}
if (item == "Double Trailer Truck") {
    thecar = dt;
    mainpanel.selectcar(thecar, 5);
    carpanel.update(thecar);
}

if (item == "EditPoint"){
    mainpanel.setflag(3);
    mainpanel.setDrawMode(0);
}
if (item == "AddPoint"){
    mainpanel.setflag(3);
    mainpanel.setDrawMode(1);
}
}
// Initiated the applet
public void init() {
    Container contentPane = getContentPane();
    contentPane.setSize(960, 550);
    contentPane.setLayout(new BorderLayout());
    optionpanel.setLayout(new GridLayout(11,1));
    optionpanel.add(Loadmaplabel);
    optionpanel.add(Mapname);
    Load.addActionListener(this);
    optionpanel.add(Load);
    optionpanel.add(Drawpathlabel);
    Drawpath.addItem("AddPoint");
    Drawpath.addItem("EditPoint");
}

```

```

Drawpath.addActionListener(this);
    optionpanel.add(Drawpath);
    optionpanel.add(Carselectionlabel);
    Carselection.addItem("Passenger Car");
    Carselection.addItem("Single Unit Truck");
    Carselection.addItem("Trailer Truck");
    Carselection.addItem("RV with trailer");
    Carselection.addItem("Double Trailer Truck ");
    Carselection.setBackground(Color.lightGray);
Carselection.addActionListener(this);
    optionpanel.add(Carselection);
    optionpanel.add(Runlabel);
    Run.setBackground(Color.cyan);
Run.addActionListener(this);
    Pause.setBackground(Color.cyan);
Pause.addActionListener(this);
    Rerun.setBackground(Color.cyan);
Rerun.addActionListener(this);
    optionpanel.add(Run);
    optionpanel.add(Pause);
    optionpanel.add(Rerun);
    leftpanel.setLayout(new GridLayout(2,1,0,0));
    leftpanel.add(carpanel);
    leftpanel.add(optionpanel);
    mainpanel.setBackground(Color.white);
    contentPane.add("Center", mainpanel);
    contentPane.add("West", leftpanel);
}
// in case the applet is not running
public String getAppletInfo() {
    return "The Vehicle Turing Path Simulation System
(1.0).";
}
}
//----- VTPSS class end-----

```

### 3. Animatedpanel.java:

/\*\*\*\*\*\*

This is apart of master project for student: Yuwen Deng  
Name of the project is:  
ANIMATED VEHICLE TURNING PATH SIMULATION SYSTEM  
ON AN INTERNET/INTRANET BROWSER

This Project is instructed by Dr. Richard J. Botting  
with committee Members: Dr. Kerstin Voigt  
Dr. George M. Georgiou  
Dr. Owen Murphy  
at California State University, San Bernardino

\*\*\*\*\*/

/\*\*\*\*\*\*



Description:

This class is used to create a panel for:

4. Load the selected map

5. Add / edit control points

6. Run / pause / rerun the animation simulation

Main program will call setdrawmode method to switch from different mode of the panel

The composite bezier curve calculation will be performed in bezcurve method

\*\*\*\*\*/

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.applet.*;
import java.awt.geom.*;
import java.util.Vector;
import java.lang.reflect.*;
```

```
//----- Animatedpanel class start-----
--
```

```
class Animatedpanel extends JPanel
implements Runnable, MouseListener, MouseMotionListener
{
//
    public static final int LINES = 0; // means edit point
    public static final int POINTS = 1; //means add point
    int mode = POINTS;

    int movedpoint = 0;

    public int L0;
    public double oldx=0;
    public double oldy=0;
    Vector lines = new Vector();
    public int mylines[] = {};
    public double px[] = {};
    public double py[] = {};
        double px2;
        double py2;
        double px0 ;
        double py0 ;

    double pz[] = new double[10000];
    int bzcount;
    int counter =0;
    int selectpointmode =0;
    Vector colors = new Vector();
    int x1,y1;
```

```

        int x2,y2;
        int x1, y1;
//
        int flag=0;
// flag=1 set the panel in run mode
// flag=2 set the panel in load map mode
// flag=3 set the panel in add or edit control point mode
// flag=4 set the panel in pause mode
// flag=5 set the panel in rerun mode

        int flagcar = 0;
// flagcar=1 set thecar[] to load vc[] Passenger Car
// flagcar=2 set thecar[] to load st[] Single Unit Truck
// flagcar=3 set thecar[] to load tt[] Trailer Truck
// flagcar=4 set thecar[] to load rv[] RV with Trailer
// flagcar=5 set thecar[] to load dt[] Double Trailer
Truck

        String mapfile = "Demomap0.gif";
        public double thecar[] ={};
        // the carcoordinates:
// the following are standard vehicle coordinates array
that fit in 1:200 scale map
        public double testv[]={
            0,0,
            0,-25,
            0,0,
            0,-30,
            15,-30
        };// testv[] is a test draw for test vehicle movement
        public double vc[]={
            0,0,
            0,-25,
            0,0,
            8,-5,
            -8,-5,
            -8,-2,
            8,-2,
            8,-32,
            -8,-32,
            -8,-5,
            0,0
        };// vc[] is array for Passenger Car coordinate

        public double st[]={
            0,0,
            0,-40,
            0,0,
            8,-5,
            -8,-5,

```

```

-8,-2,
 8,-2,
 8,-52,
-8,-52,
-8,-5,
 0,0
};// st[] is array for Single Unit Truck coordinate

public double tt[]={
 0,0,
 0,-25,
 0,0,
 8,-5,
-8,-5,
-8,-2,
 8,-2,
 8,-30,
-8,-30,
-8,-5,
 0,0,

 0,-22,
 0,-82,
 0,-22,
 8,-40,
-8,-40,
-8,-32,
 8,-32,
 8,-87,
-8,-87,
-8,-40,
 0,-22
};// tt[] is array for trailer truck coordinate

public double rv[]={
 0,0,
 0,-50,
 0,0,
 8,-5,
-8,-5,
-8,-2,
 8,-2,
 8,-55,
-8,-55,
-8,-5,
 0,0,

 0,-50,
 0,-82,
 0,-50,

```

```

        8,-62,
        -8,-62,
        -8,-60,
        8,-60,
        8,-87,
        -8,-87,
        -8,-62,
        0,-50
    };// rv[] is array for RV with a trailer coordinate

    public double dt[]={
        0,0,
        0,-25,
        0,0,
        8,-5,
        -8,-5,
        -8,-2,
        8,-2,
        8,-30,
        -8,-30,
        -8,-5,
        0,0,

        0,-22,
        0,-82,
        0,-22,
        8,-40,
        -8,-40,
        -8,-32,
        8,-32,
        8,-87,
        -8,-87,
        -8,-40,
        0,-22,

        0,-82,
        0,-142,
        0,-82,
        8,-100,
        -8,-100,
        -8,-92,
        8,-92,
        8,-147,
        -8,-147,
        -8,-100,
        0,-82
    };// dt[] is array for Double Trailer Truck coordinate

    public Animatedpanel(int flagin){

```

```

        flag = flagin;
        Thread a = new Thread(this);
    //
        addMouseMotionListener(this);
        addMouseListener(this);
    //
        a.start();
    }

    //
    public void setDrawMode(int mode) {
        switch (mode) {
            case LINES:
            case POINTS:
                this.mode = mode;
                break;
            default:
                throw new IllegalArgumentException();
        }
    }

    public void mouseDragged(MouseEvent e) {
    }

    public void mouseMoved(MouseEvent e) {
    }

    public void mousePressed(MouseEvent e) {
        e.consume();
        colors.addElement(getForeground());
        lines.addElement(new Rectangle(e.getX(), e.getY(),
-1, -1));
        bzcount=0;
        if (mode == 0) {
            x1 = e.getX();
            y1 = e.getY();
            if (selectpointmode == 0){
                double dd = 999999999; // set the number that
is practically infinity
                for (int i = 0; i < mylines.length; i+=2){
                    double dx = Math.abs(x1 - mylines[i]);
                    double dy = Math.abs(y1 - mylines[i+1]);
                    double d = dx* dx + dy*dy;
                    if (d < dd) {
                        dd=d;
                        movedpoint= i;
                        selectpointmode =1;
                    }
                }
            }
        }
    }

```

```

        oldx = px[movedpoint];
        oldy = py[movedpoint];
    }
    else if (selectpointmode ==1){
        mylines[movedpoint]= x1;
        px[movedpoint]=x1;
        mylines[movedpoint+1]=y1;
        py[movedpoint]=y1;
        selectpointmode =0;
    }
    repaint();
}
else if (mode ==1){
    mylines = (int[])goodArrayGrow(mylines);
    mylines[mylines.length -1] = e.getX();
    px = (double[])goodArrayGrow(px);
    px[px.length -1] = e.getX();
    px = (double[])goodArrayGrow(px);
    mylines = (int[])goodArrayGrow(mylines);
    mylines[mylines.length -1] = e.getY();
    py = (double[])goodArrayGrow(py);
    py[py.length -1] = e.getY();
    py = (double[])goodArrayGrow(py);
    x1 = e.getX();
    y1 = e.getY();
    repaint();
}
}

public void mouseReleased(MouseEvent e) {
    // this mouse event is not used

public void mouseEntered(MouseEvent e) {
    // this mouse event is not used

public void mouseExited(MouseEvent e) {
    // this mouse event is not used

public void mouseClicked(MouseEvent e) {
    // this mouse event is not used
//

public void run(){
    while (true){
        if (flag == 4){    // pause code do nothing
        }
        if (flag == 5){    // rerun code here
            counter = 0;
            if (counter < bzcount){
                try {Thread.sleep(40);}

```

```

        catch (InterruptedException e){}
        thecar = newarray(thecar);
        this.repaint();
    }
}

if (flag == 1){    // animation code here
    if (counter <= bzcount){
        try {Thread.sleep(40);}
        catch (InterruptedException e){}
        thecar = newarray(thecar);
        this.repaint();
    }
}
else if (flag ==2){    // change map code here
    try {Thread.sleep(100);}
    catch (InterruptedException e){}
    this.repaint();
}
else if (flag == 3){    // draw bezier curve here
    try {Thread.sleep(1);}
    catch (InterruptedException e){}
    // mode 0
    if((mode == 0) && ( bzcount < L0)){
        for (int i = 2; i <= px.length -2; i+=2){
            px[i-1]= (px[i-2]+px[i])/2;
            py[i-1]= (py[i-2]+py[i])/2;
        }
        if (px.length >= 2){
            pz[0]=px[0];
            pz[1]=py[0];
            px2 = px[0];
            py2 = py[0];
        }
        for (int i = 2; i <= px.length -2; i+=2){
            px0 = px2;
            py0 = py2;
            if (i == px.length -2){
                px2 = px[i];
                py2 = py[i];
            }
            else{
                px2 = (px[i-1] + px[i+1])/2;
                py2 = (py[i-1] + py[i+1])/2;
            }
        }
        bzcount =
            BZCURVE(px0,py0,px[i-1],py[i-1],
                px2,py2, bzcount, pz);
    }
}
}

```

```

        //end mode 0
        if ((L0 < px.length )){
            for (int i = 2; i <= px.length -2; i+=2){
                px[i-1]= (px[i-2]+px[i])/2;
                py[i-1]= (py[i-2]+py[i])/2;
            }
            if (px.length >= 2){
                pz[0]=px[0];
                pz[1]=py[0];
                px2 = px[0];
                py2 = py[0];
            }
            for (int i = 2; i <= px.length -2; i+=2){
                px0 = px2;
                py0 = py2;
                if (i == px.length -2){
                    px2 = px[i];
                    py2 = py[i];
                }
                else{
                    px2 = (px[i-1] + px[i+1])/2;
                    py2 = (py[i-1] + py[i+1])/2;
                }
                bzcount =
                    BZCURVE(px0,py0,px[i-1],
                        py[i-1],px2,py2, bzcount, pz);
            }
            L0 = px.length;
        } // end L0 if condition
    } // end flag 3
} //end while
} //end run
public void setflag(int set){
    flag = set;
}
public void updatemap(String Filename){
    mapfile = Filename;
}

public double[] newarray(double[] oldarray){
    // the function to get next point of the car
    // select a car coordinate:
    if (flagcar == 1)carcoordinate = vc;
    if (flagcar == 2)carcoordinate = st;
    if (flagcar == 3)carcoordinate = tt;
    if (flagcar == 4)carcoordinate = rv;
    if (flagcar == 5)carcoordinate = dt;
    double[] newarray = new double[ oldarray.length ];
    double XPi = oldarray[0];
    double YPi = oldarray[1];

```



```

double XQi = oldarray[2];
double YQi = oldarray[3];
//new position array:
double XPj = pz[counter];
double YPj = pz[counter+1];
counter = counter +2;
//test
double DX = (XPi + XPj)/2 - XQi;
double DY = (YPi + YPj)/2 - YQi;
double S = (
    (XPj - XPi) * DX + (YPj - YPi) * DY)
    / (DX * DX + DY * DY);
double XQj = XQi + S * DX;
double YQj = YQi + S * DY;
    newarray[0] = XPj;
    newarray[1] = YPj;
    newarray[2] = XQj;
    newarray[3] = YQj;
double CarLength=0;
if (flagcar==1) CarLength = 25;
if (flagcar==2) CarLength = 40;
if (flagcar==3) CarLength = 25;
if (flagcar==4) CarLength = 50;
if (flagcar==5) CarLength = 25;

if (flagcar == 3 || flagcar == 4){
    // save the oldarray value before change it
    double XPi2 = oldarray[22];
    double YPi2 = oldarray[23];
    double XQi2 = oldarray[24];
    double YQi2 = oldarray[25];
    // caculate the other points:
    for (int i = 4; i < 24; i+=2){
        newarray[i] = XPj - carcoordinate[i+1]
            /CarLength * (XQj -XPj) -
            carcoordinate[i]
            /CarLength * (YQj - YPj);
        newarray[i+1] = YPj - carcoordinate[i+1]
            /CarLength * (YQj - YPj) +
            carcoordinate[i]
            /CarLength * (XQj -XPj);
    }

    //new position array:
    double XPj2 = newarray[22];
    double YPj2 = newarray[23];
    double DX2 = (XPi2 + XPj2)/2 - XQi2;
    double DY2 = (YPi2 + YPj2)/2 - YQi2;
    double S2 = (

```

```

                                (XPj2 - XPi2) * DX2 + (YPj2 - YPi2)
* DY2)
                                / (DX2 * DX2 + DY2 * DY2);
    double XQj2 = XQi2 + S2 * DX2;
    double YQj2 = YQi2 + S2 * DY2;
    newarray[24] = XQj2;
    newarray[25] = YQj2;

    double CarLength2=0;
    if (flagcar ==3){
        CarLength2 = 120; //second car length
        for (int i = 26; i < carcoordinate.length -1;
i+=2){
            newarray[i] = ((XPj2) -
(carcoordinate[i+1]+45)
                                /CarLength2 * (XQj2 -XPj2)
                                - (carcoordinate[i]-
0)/CarLength2
                                * (YQj2 - YPj2));
            newarray[i+1] = ((YPj2) -
(carcoordinate[i+1]+45)
                                /CarLength2 * (YQj2 - YPj2)
                                + (carcoordinate[i]-
0)/CarLength2
                                * (XQj2 -XPj2));
        }
    }
    if (flagcar ==4){
        CarLength2 = 65; //second car length
        for (int i = 26; i < carcoordinate.length -1;
i+=2){
            newarray[i] = ((XPj2) -
(carcoordinate[i+1]+100)
                                /CarLength2 * (XQj2 -XPj2)
                                - (carcoordinate[i]-
0)/CarLength2
                                * (YQj2 - YPj2));
            newarray[i+1] = ((YPj2) -
(carcoordinate[i+1]+100)
                                /CarLength2 * (YQj2 - YPj2)
                                + (carcoordinate[i]-
0)/CarLength2
                                * (XQj2 -XPj2));
        }
    }

    if (flagcar != 3 && flagcar != 4) {
        for (int i = 4; i < carcoordinate.length -1;
i+=2){

```

```

        newarray[i] = XPj - carcoordinate[i+1]/CarLength
        * (XQj -XPj) - carcoordinate[i]/CarLength
* (YQj - YPj);
        newarray[i+1] = YPj -
carcoordinate[i+1]/CarLength
        * (YQj - YPj) + carcoordinate[i]/CarLength
* (XQj -XPj);
    }
}

    return newarray;
}
public void selectcar(double[] newcar, int flagcarin){
    thecar = newcar;
    flagcar = flagcarin;
}

// the path function
public double getYfromX( double x ){
    return (0.0005 * x * x - 0.0001 * x );
}

// utility method
static Object goodArrayGrow(Object a)
{
    Class cl = a.getClass();
    if (!cl.isArray()) return null;
    Class componentType =
a.getClass().getComponentType();
    int length = Array.getLength(a);
    int newLength = length +1;

    Object newArray = Array.newInstance(componentType,
newLength);
    System.arraycopy(a, 0, newArray, 0, length);
    return newArray;
}

// bzcurve method
// The composite bezier curve calculation will be
performed here
    public int BZCURVE(double px0,double py0,double px1,
double py1,double px2,double py2,
int bzcount, double
afterbzarray[]){
        double DD;
        double D01;
        double D12;
        int N;
        double t;
        double x0, y0, x1, y1, x, y;
        DD= (px0-px1)*(px0-px1) + (py0-py1)*(py0-py1);

```

```

        D01 = Math.pow(DD, 0.5);
        DD= (px1-px2)*(px1-px2) + (py1-py2)*(py1-py2);
        D12 = Math.pow(DD, 0.5);
        N = (int)((D01 + D12) / 5.0 +1.0);
        for (int i = 1; i <= N; i++){
            t = (double)i/(double)N;
            x0 = px0 * (1.0 -t) + px1 * t;
            y0 = py0 * (1.0 -t) + py1 * t;
            x1 = px1 * (1.0 -t) + px2 * t;
            y1 = py1 * (1.0 -t) + py2 * t;
            x = x0 * (1.0 -t) + x1 * t;
            y = y0 * (1.0 -t) + y1 * t;
            bzcount +=2;
            afterbzarray[bzcount] = x;
            afterbzarray[bzcount+1] = y;
        }
        return bzcount;
    }
}
// bzcurve method end
}
//----- Animatedpanel class end-----
--

```

#### 4. ShowcarPanel.java:

/\*\*\*\*\*\*

This is apart of master project for student: Yuwen Deng  
 Name of the project is:  
 ANIMATED VECHICLE TURNING PATH SIMULATION SYSTEM  
 ON AN INTERNET/INTRANET BROWSER

This Project is instructed by Dr. Richard J. Botting  
 with committee Members: Dr. Kerstin Voigt  
 Dr. George M. Georgiou  
 Dr. Owen Murphy  
 at California State University, San Bernardino

\*\*\*\*\*/  
 /\*\*\*\*\*

#### Description:

This class is used to create a panel to show the selected car in the panel.

Main program will call update method to update the selected car array and redraw it in the panel

\*\*\*\*\*/

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.applet.*;
import java.awt.geom.*;

```

```

//----- ShowcarPanel class Start-----
--

class ShowcarPanel extends JPanel
{
    private double selectedcar[];
    public ShowcarPanel(double array[])
    {
        selectedcar = array;
    }
    public void paintComponent(Graphics showcargraphic)
    {
        super.paintComponent(showcargraphic);
        Dimension d = getSize();
        double centerpointX= d.width/2;
        double centerpointY= d.height/2;
        Graphics2D showcargraphic2 =
            (Graphics2D) showcargraphic;
        AffineTransform at =
            AffineTransform.getTranslateInstance(
                centerpointX, centerpointY-70);
        at.rotate(-Math.PI);
        showcargraphic2.transform(at);
        showcargraphic2.setPaint(Color.blue);
        if (selectedcar.length != 0){
            for (int i=0; i < selectedcar.length-3 ; i+=2){
                showcargraphic2.drawLine(

(int)selectedcar[i], (int)selectedcar[i+1],
                    (int)selectedcar[i+2], (int)selectedcar[i+3]
);
            }
        }

        public void update(double newarray[] ){
            selectedcar = newarray;
            this.repaint();
        }
    }
}
//----- ShowcarPanel class end-----
--

```

## REFERENCES

- [1] James P. Pitz, Michigan et al. *A Policy on Geometric Design of Highways and Streets*. AASTO, 1990
- [2] YauHuei Deng. Vehicle Turning Path's Calculation and Application. China Technology Vol. 29 pp. 3 - 14, 1996
- [3] Highway and Street Design Standard. Department of Motor Vehicles, pp. 4 - 10 1987
- [4] Martin Flower. UML Distilled Second Edition. Addison Wesley Longman, Inc. 2000
- [5] D. Faux. And M. J. Pratt. Computational Geometry for Design and Manufacture. Ellis Horwood, 1979
- [6] Jonathan Knudsen. Java 2D Graphics. O'Reilly, May 1999
- [7] David M. Geary. Graphic Java2 Mastering the JFC 3<sup>Rd</sup> Edition. Sun Microsystems Press A Prentic Hall Title 1999