A Graphical Traffic Scenario editing and evaluation software
A Graphical Traffic Scenario editing and evaluation software

Master Thesis in Embedded and Intelligent Systems
Halmstad, May 2012

Author: Iulian Nicolae Carpatorea
Supervisor: Roland Philippsen
Examiner: Antanas Verikas

School of Information Science, Computer and Electrical Engineering
Halmstad University
PO Box 823, SE-301 18 HALMSTAD, Sweden
Preface

The project is the result of my Master Thesis at Halmstad University. It started out of the need of a fast, reliable tool that can assist human-machine interaction (HMI) researchers at Volvo Group Trucks Technology (GTT), Advance Technology & Research in creating, evaluating and explaining special cases of traffic scenarios. It proved to be challenging in some situations but with the help of my supervisor, Roland Philippsen, who has been there to guide me through it all, those challenges were overcome. I would also like to take the time to thank him for the valuable input and discussions.
Abstract

An interactive tool is developed for the purpose of rapid exploration of diverse traffic scenario. The focus is on rapidity of design and evaluation rather than on physical realism. Core aspects are the ability to define the essential elements for a traffic scenario such as a road network and vehicles. Cubic Bezier curves are used to design the roads and vehicle trajectory. A prediction algorithm is used to visualize vehicle future poses and collisions and thus provide means for evaluation of said scenario. Such a program was created using C++ with the help of Qt libraries.
List of figures

Figure 1: Linear Bezier.......................................................................................................................................................................... 4
Figure 2: Quadratic Bezier Curve ...................................................................................................................................................... 5
Figure 3: Cubic Bezier ............................................................................................................................................................................ 6
Figure 4: Program start state ........................................................................................................................................................... 11
Figure 5: Start point tangent line for a Bezier curve....................................................................................................................... 12
Figure 6: Tutorial step 2 ...................................................................................................................................................................... 13
Figure 7: A collection of road segments ....................................................................................................................................... 13
Figure 8: Cross intersection example ............................................................................................................................................ 14
Figure 9: Unconnected road network showing various components ............................................................................ 15
Figure 10: Connected road network example ........................................................................................................................... 16
Figure 11: Vehicle on road with start and end pose and trajectory ........................................................................................... 17
Figure 12: Two vehicles with paths and poses at time 00:00:00 ................................................................................................. 18
Figure 13: Two vehicles with paths, poses and prediction at time 00:00:02 ........................................................................... 18
Figure 14: Offset example of a Bezier curve including the offset control points .................................................................. 21
Figure 15: Example of a special case and the failed offset curve ............................................................................................... 21
Figure 16: Quick access options for the scenario ......................................................................................................................... 22
Figure 17: Road segments quick access options ......................................................................................................................... 22
Figure 18: Intersections quick access options ............................................................................................................................. 22
Figure 19: Edit first level quick access options ............................................................................................................................ 23
Figure 20: Road segments edit full quick access options ........................................................................................................ 23
Figure 21: Vehicles edit full quick access options ....................................................................................................................... 24
Figure 22: Intersection edit mode active .................................................................................................................................... 24
Figure 23: Scenario play controls .................................................................................................................................................. 24
Figure 24: Road network tree .......................................................................................................................................................... 25
Figure 25: Sample graph ................................................................................................................................................................. 26
Figure 26: Road curve ................................................................................................................................................................. 27
Figure 27: Control points

Figure 28: Tangent line at start position for a Bezier curve

Figure 29: Building first curve of a road segment

Figure 30: Radius design example

Figure 31: Edit sample

Figure 32: Circular intersection with exits

Figure 33: One exit circular intersection

Figure 34: Modified intersection based on figure 34

Figure 35: Radius of curvature and center of rotation for a vehicle

Figure 36: Reachable destination (start pose in red, end pose in blue)

Figure 37: Unreachable destination (start pose in red, end pose in blue)

Figure 38: Example of a deviation

Figure 39: Two time step prediction for a vehicle with 0 heading

Figure 40: Two time step prediction for a vehicle with not 0 heading

Figure 41: Now collision between two vehicles

Figure 42: Future collision example between two vehicles
1. Introduction

The purpose of this project is to create a tool that can let a user to create and assess, in short amount of time, special traffic scenarios in order to improve traffic safety and efficiency by means of automated systems. At same time to provide an easier way in communicating advantages for different systems in specific traffic situations.

In today's world, road traffic plays a major part of economy, infrastructure, culture and other. According to [1] in 1900, 4192 vehicles were produced in USA. After 110 years, in 2010, according to [2], the number of total active vehicles surpassed the 1 billion mark. That is roughly 1 vehicle per 7 persons if an equal distribution would exist. Looking again at [1] it can be seen that the number of vehicles will reach 1.2 billion in the next 20 years which is a 20% increase in the number of vehicles from 2010. When there is an increase in active vehicles, in order to maintain same level of road traffic, it is needed to increase road capacity by around same amount or build more roads. Building more roads can't always be a solution due to costs, limited space and other factors. One of the ways to increase road capacity is by means of automated or assisting systems incorporated in vehicles. A program that allows for quick construction of traffic scenarios in order to better communicate advantages and disadvantages of different automated or assisting systems was needed and gave birth to the idea for this project. Different automated systems are already present in cars, such as Lane Change Warning[3], Forward Collision Warning[4], Adaptive Cruise Control[5] and others. All of them contribute to increased safety on the roads. We also aim at improving some of the existing systems and set the basis for new ones.
2. Background

When talking about traffic scenarios, naturally we mainly think about roads and vehicles. As those two elements of traffic are the main focus of this project, methods for designing roads and vehicles were needed. To decide what to use for this we took a look at what was required from roads and what different countries use to construct their own roads. The final choice was between Cubic Bezier curves [8] and Clothoids [9]. Clothoids are used by law in many countries to design the roads, and their curvature properties make them more natural to describe vehicle trajectories. However, our aim here is to have an interactive rapid-prototyping tool, and Bezier curves are better suited for intuitive drawing and modification of curves. In this context, the loss of realism is deemed acceptable.

2.1. Roads

2.1.1. Introduction

In today’s world we can recognize four major ways of moving goods or people from one place to another. It can be named here that goods are transported via water, using boats, via land, using vehicles and trains and other transportation, and via air, using mostly airplanes. Out of these four major transportation systems, only two need constant maintenance and attention outside arrival and departure areas. The two would be the land transportation, via vehicles and trains. Vehicles need roads while trains need railroad in between point A and point B. Roads are integral part of today’s infrastructure and economy.

As roads are integral part of the infrastructure, it is important to design them in such a way that efficient transportation can take place. There are many meanings to what efficient might mean. We can refer to a road as efficient from the point of view of vehicle capacity, maximum number of vehicles that can pass a point in set amount of time. It can also be seen as space occupied by the road as the space available is limited and we must make the best use of what is available. There are many types of roads, like town roads, highways, motorways. Each type of road serves a purpose and this purpose determines the way a road is constructed and where. Because usually town roads construction constraints are local we will not delve into them but focus instead on highways or motorways. The constraints that we will talk about can apply to any road, including town roads.

2.1.2. Constraints

When designing roads many factors come into effect. We are, however only interested in vehicle motion. We will go through some of them in order to better understand the complex system of constructing roads.
An important factor is the road capacity. Road capacity is directly influenced by number of lanes of the road, stop places, such as intersections, traffic lights, and of course speed limit. Number of lanes, in turn, is influenced by, and not only, geographic location as well as economic factors, such as available funds. When the talk is about highways traffic lights and intersections that require a car to be stopped are usually absent, with some exceptions. For example, when a highway passes through town, one may encounter such signs or intersections. Speed is the one factor we are interested in because it directly affects road design.

European Union, further referred as EU, has adopted a set of rules for the E-Road network that passes through European countries and extends all the way to Asia. These rules are referred as standards and they can be seen here [7].

Roads that are straight basically support a cars’ maximum speed. However that is rarely the case, and curves are incorporated in highways. As it can be seen from [7], the radius of the curve of a road is directly influencing the speed on that road.

2.2. Bezier curves

Bezier curves are widely used in computer graphics to model smooth curves. Because a Bezier curve is defined by its control points all translations and rotations can be applied to those control points and as such they will also be applied to the curve itself making the process fast and easy. Most common Bezier curves used are quadratic and cubic, due to the fact that higher order curves become more computational expensive.

2.2.1. Introduction

Bezier curves can be seen as a series of linear interpolations. For example, a Bezier curve of degree 1, which is a line, is a linear interpolation between points P1 and P2 as it can be seen from the formula (1) and exemplified in fig. 1.

\[
B(t) = P_0 + t(P_1 - P_0) = (1-t)P_0 + tP_1, t \in [0,1]
\]

![Figure 1: Linear Bezier](image)

A quadratic Bezier curve can also be seen as a linear interpolation of a linear interpolation. This approach is also known as DeCasteljau construction algorithm
and can be extended to any degree. Formula (2) could be split in two parts to show how one might interpret those curves as recursive linear interpolations. From that we get (3) and it is shown in figure 2 an example of quadratic Bezier curve.

\[ B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2, t \in [0,1] \] (2)

![Figure 2: Quadratic Bezier Curve](223x504 to 373x653)

The next higher order for Bezier curves is the cubic one. The cubic Bezier curve is usually the highest order of Bezier curves that is commonly used and like the previous ones it can be seen as a series of linear interpolations. Cubic Bezier curves have two control points. In the case of a cubic, the curve starts at P0, goes towards P1 and arrives at P3 from the direction of P2. Generally, the curve does not intersect P1 and P2, but of course there are special cases in which it does. One such case is the case of a straight line, in which all the points are collinear and in sequence. Following same logic as in the case of quadratic Bezier curves it can be shown that the cubic ones are a series of linear interpolations. In (4) is the polynomial formula for a cubic curve, while in (5) it is rewritten to show it as a series of linear interpolations.

\[ B(t) = (1-t)^3 P_0 + 3(1-t)^2 tP_1 + (1-t)t^2 P_2 + t^3 P_3, t \in [0,1] \] (4)

\[ B(t) = (1-t)((1-t)[(1-t)P_0+tP_1]+t[(1-t)P_1+tP_2]+t[(1-t)P_2+tP_3]) \] (5)

The general formula for Bezier curves can be written as:

\[ B(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i P_i, t \in [0,1] \] (6)

As it can be seen from (6) the curve, or line in special cases, is guaranteed to pass
through the first and last points in the polynomial. Figure 3 shows one example of a cubic Bezier curve.

![Cubic Bezier](image)

**Figure 3: Cubic Bezier**

### 2.2.2. Properties

Below we present a list of properties that are relevant to the project; list is taken from [6]:

- One of the special cases is when all points are collinear and in sequence. In this case the curve is actually a straight line and it will pass through all points. It is important to note that the points have to be in sequence. Remembering how a Bezier curve can be described in words it can be seen why they have to be in sequence in order to be included in the line segment. If one of the control points, say $P_1$, lies outside the line segment formed by $P_0$ and $P_n$ but on same line, the curve, or line in this case, will move towards it but it won’t reach it. This property is important because drawing lines using a Bezier description will hold the properties of the Bezier and apply to the line.

- Another property is that the curve, at start and end, is tangent to the start and respectively end lines of the Bezier polygon. Bezier polygon is the polygon formed by all the points that define the curve, in the case of a cubic curve, by $P_0, P_1, P_2, P_3$. This gives the possibility to define the tangent at start and end of the curve. Parallel curves to Bezier curves are not Bezier curves. This is false in some special cases, such as when Bezier curve is a line.

- A Bezier curve can be split in two curves, which are also Bezier curves using the DeCasteljau algorithm. By applying this recursively we can split any Bezier curve into any number of other Bezier curves.

These properties helped us in solving some of the challenges encountered during
the project, such as creating an offset for a curve.

2.2.3. Bezier curve joining

Two cubic Bezier curves can be joined in one of the three ways.

A simple join ensures positional continuity and the condition for this is that the start or end points of either curve are at same position. For simplicity we will consider that a second curve start point is connected to the end point of a first curve. First curve is defined as

\[ B_1(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + (1-t) t^2 P_2 + t^3 P_3, \quad t \in [0,1] \] (7)

and second curve is defined as

\[ B_2(t) = (1-t)^3 Q_0 + 3(1-t)^2 t Q_1 + (1-t) t^2 Q_2 + t^3 Q_3, \quad t \in [0,1] \] (8)

We will further denote it as C0 and we can express the curve continuity as

\[ P_3 = Q_0 \] (9).

The next level of join continuity is the tangent continuity which can be achieved by having the first derivative of each curve at the join point equal and will be further denoted as C1. This can also be expressed in terms of control points as

\[ P_3 - P_2 = Q_1 - Q_0 \] (10)

It should be noticed from (10) that there is no constrain on the distance between \( P_3 \) and \( P_2 \).

The last important join property is the curvature continuity and this is achieved by having the second derivative equal at the join point and will be further denoted as C2. This can also be expressed in terms of control points as

\[ P_3 - 2P_2 + P_1 = Q_2 - 2Q_1 + Q_0 \] (11).

The reason why we are only interested in the join continuity is because Bezier curves are continuous to the degree \( n \), \( n \) being the degree of the curve.
3. Software

The software used to write the program was Qt. Qt is an application framework that supports cross-platform development of software. It is largely used in GUI (graphical user interface) application but not limited. The language used by Qt is C++ but extends an already very good programming language by using a special code generator called “Meta object compiler” or “moc” along with macros. On a side note, Qt is free and open-source software. Visit [10] for more information regarding license and other details. For people familiar with C++, Qt offers new possibilities with minimum time spent on learning. For people new to programming, Qt can be learned rather fast due to the extensive support that it provides and very good documentation. The visual interface can be considered friendly and it helps a lot especially for GUI applications. Many of the elements needed for GUI applications are already implemented and the user just has to make use of the graphic interface to access them. As a starting stone they can be used without reimplementation to get familiar with its functionality and what it can do.
4. Introductory tutorial

Before moving into technical aspects we present a step by step tutorial of how a scenario can be created from scratch with the program that was written in C++ using Qt SDK. First off whenever the program is started a choice can be made between loading an already designed scenario and starting new. It should also be stated that the canvas supports zoom in and out to a 10x factor. Rotating the view is not available at the time this tutorial was made. Also it is important to note that the editor in play mode plays only in real time.

When the program is started it looks as in figure 4.

![Figure 4: Program start state](image)

At this point roads or intersections can be designed freely. Vehicles however cannot be added due to the fact they are directly linked to roads in terms of starting pose and end pose.

To start with the “Add Road” option has to be enabled which also enables a choice of curve joining characteristics in the forms of C0, C1 and C2, representing the different degrees of knot join. Adding a road needs four points provided by the user in order to draw the Bezier curve. The four points are assigned by user using mouse clicks and they are paired into two lines, the two lines that are tangent to the start and end point of the curve. Mouse left click press assigns the start point of the curve while releasing it assigns the end point of current line and implicitly the control point that is represented by this line. As seen in figure 5 the “Add Road” and “C1” active mode was selected. This lets us design a road segment. C1 is there for the case when more
than one curve is present in the current road segment. In figure 5 we draw the tangent line to the start point.

![Figure 5: Start point tangent line for a Bezier curve](image)

Important to note that if we change the active mode now, the line will get deleted as the curve is not yet complete. We continue by providing the tangent line for the end point of the current curve, which is the first curve for this road segment. Figure 6 shows the representation of the first road segment containing one curve. As it can also be seen the start tangent line for the next curve has also been added due to our choice in curve continuity. Now there is a valid road segment in the scene which means active modes can be changed and the work so far will persist. “Add vehicle” mode can be now activated but we will continue to build a more complex network. Next steps in building road segments are same as the one presented here. However in case that “C0” is selected the user must provide all the lines in order to create the road.

Whenever the user desires to complete the current road segment, two choices are available. First choice is to use the default command for that which is “Right click” on the canvas. Second choice is to change the active mode in which case the program will automatically assume completing the current road segment is also required.
Following this procedure we can design any number or road segments. In this tutorial we will have initially three road segments presented in figure 7.

For cross intersection, after a position has been assigned, the user must provide number of possible connecting roads to that intersections and initial position. The position for this type of intersection is represented by a point. This is done by left clicking where the initial position is desired. All connectors(exits) are straight lines and their length can vary from 0 to any length supported by the program.
Intersection arms can be added by left clicking on initial position and dragging the mouse for the length and orientation desired. When the left mouse button is released the intersection arm is added. As with road segments, right click completes the intersection which has same effect as changing active mode. Figure 8 shows an example of cross intersection.

![Figure 8: Cross intersection example](image)

Circular intersections are created in two stages. First we draw the inner area represented by a circle. This is done by providing the location and the radius for the circle. As with road segments and cross intersections the location is provided on left mouse button press. The radius is provided by holding the left mouse button pressed, after specifying the position, and dragging the mouse. Releasing the mouse button will set the radius. After, connectors can be added, one by one. They are drawn as a line. Left mouse button is used once again to add the connectors. They can only be added once we have a valid circular area. Mouse button press defines the point belonging to circular area of the intersections while releasing it provides the second points, thus providing us with the line needed to add the connector. Once we are satisfied with the intersection, it can be completed by changing active mode or pressing right click button. In the edit mode, the radius of the intersection can be modified graphically or numerically by changing the radius value in the property window. Figure 9 depicts an example of a circular intersection alongside road segments.

After we have the road segments and intersections we require, next step is to connect them the way we want. It is important to remember that each member of the road network is designed individually.
For translation of items the “drag and drop” approach has been selected. To move an item the user must only select it and drag it to the desired position. Many components are translatable, all being explained in their respective section. In order to connect two road network items their respective loose ends have to be moved close. The interaction with the object on the canvas is done via the left mouse button, unless otherwise specified, with the possibility of combining them with keyboard buttons. Once the item is released it gets connected to the other item. The reverse is also possible and it is done by holding down “Shift” key while dragging the desired item away. All connections between road type items are stored in an undirected graph. When one item is connected to others all changes made to that item might affect the other as well. Figure 10 shows a simple road network containing two circular intersections and four road segments. The road segments can also be identified in the figure by the black rectangles. Now we can do a rudimentary example of vehicle motion and prediction.
First we need to add one or more vehicle. Vehicles are added in three stages. The three stages are subsequent and can’t be skipped. First stage determines the position of the vehicle. Like previously mentioned vehicles can only be added on existing road. Vehicles are added using the mouse. Left click determines the initial placement while still giving the possibility to drag it around. Release of the mouse click sets the position. Next stage gives the user the ability to set the initial orientation for the vehicle. An orientation is valid if the vehicle is able to determine the lane and direction. By giving this possibility we can have vehicles that drive on the wrong lane thus direct collisions can be studied as well. Third stage sets the destination for the vehicle. The destination is set as a pose. Same as starting position, vehicle orientation has to be valid in order for the vehicle path planner to be able to generate a path. At same time if start and goal position are not connected, and until they are, the vehicle will remain stationary. Having in mind the start and goal poses we distinguish four possible combinations. We denote for simplicity 1 as correct pose and 0 for reversed. When starting pose is 1 and goal pose is 1 the outcome is that the vehicle will drive correctly from start to finish. Start pose is 0 and end pose is 0 results in an England type of driving, on the left side. Whenever the start and goal poses are reversed relative to each other, ie start 0 and goal 1, a change in side of driving is required. One of the intuitive ways is to do it while passing an intersection. In the intersection the side of driving is changed to match the end pose side. Figure 11 shows a 0-1 path, right side lane for start and left side lane for end. The starting pose is shown in red while the end pose is shown in blue. Purple denotes initial path. As it can be seen from the figure 11, the vehicle enters the intersection from the wrong lane. This is due to the initial pose placement. It can also be seen that the continuation is correct, from a right side driving point of view. Path planning only considers optimal path from a distance point of view and it is
discussed in chapter 5.6.1. After we have setup a vehicle as explained before we can add as many as we like. Another important note is that vehicles don’t occupy space exclusively. Therefore it is possible to have two or more vehicle stacked on top of each other.

![Figure 11: Vehicle on road with start and end pose and trajectory](image)

Each vehicle is created with a default vehicle speed. The tool allows for modification of said speed at any time during the scenario. However the graphical user interface (GUI) is yet to be implemented due to limited time. Time related properties for the vehicles are determined by the current time of placement. This time can later be changed via the property window for each respective vehicle. The initial end time for it is set to the maximum duration of the scenario, editable via the property window. The end time comes into effect in the case that the vehicle requires a longer duration for it to complete the current path. As a result we have two possible stop situations for a vehicle: first, when the goal is reached and second, when it runs out of fuel or time in our case.

Figure 12 and 13 show two vehicles, in red for starting poses, and their end poses in different colors. The trajectory is also colored in purple and blue.

Once we have a road network and vehicles the scenario can be played and observed by the user. At any point it can also be paused and modified. Jumps at different time are also possible.
Figure 12: Two vehicles with paths and poses at time 00:00:00

Figure 13: Two vehicles with paths, poses and prediction at time 00:00:02
5. Project

5.1. Theoretical considerations

The program offers the possibility to design roads using Bezier curves as previously specified. When implementing path for vehicles a specific challenge has been encountered, one that is an active topic in the area of use for the curves. The challenge was offsetting the curve while maintaining the properties of the curve. Since Bezier curves don't have parallel counterparts except for the special cases the task of figuring an offset for it proved one of the challenges for this project. There are several ways to achieve parallelism with different accuracy. In [11], is presented comparison between different methods used for offsetting. Each of those methods has advantages and disadvantages. For this project a computational light and accurate as possible was needed. The final solution was partly based on [12].

The way we implemented the calculation of offset was by browsing through the whole curve and split it into smaller curves that can be offset easier and more accurate. There are several split conditions that we will enumerate and explain. The curve itself is traversed by percentage with a step of 0.5%.

- Through experimentation we found that whenever more than 16% of the curve was travelled since the last split, a new one was needed.

- Again through experimentation a change of 20 degrees in curvature was deemed as maximal allowed in order to get a good offset curve.

- By the previous condition we also say we want a fairly close approximation to an arc of circle which is easier to offset. Because of that we also need a split at all inflection points. On a cubic Bezier curve there is a maximum one inflection point.

After we have split our curve we can then offset it to both sides. The way this is done is by translating the control points needed to create a Bezier curve ensuring that the parallel curve is still a Bezier. We offset the control points obtained by splitting the initial curve and adjust the non-end or start points with a parameter that varies based on the offset value. After translating the control points we simply use them to generate new Bezier curves that are used for vehicle path planning. Figure 14 shows an example of a Bezier curve with original and offset control points. This method fails in special cases where the curvature change is too high in relation with the length traversed as exemplified in figure 15. Figures 14 and 15 were created using the program for this project.
5.2. Graphical User Interface (GUI)

The GUI is the connection between the program and the user. As such it is imperative that it provides quick access so that the user spends a minimum time navigating and most of the time making use of what the program is supposed to do. GUI is comprised of elements that can be changed and elements that show different aspects of the scenario. Elements that can be changed are, for example, the current active mode, while elements that display status are, for example, the time slider or the scenario. Some of these elements can be changed while at same time they display status.
5.2.1. Components

When building a GUI we must look at the most common actions the user will make in order to facilitate the best and fastest access to those actions.

In the project we distinguish several actions that are the core of the application. When creating a traffic scenario, having the possibility to add road segments or intersections is essential. Another essential feature is the possibility to add vehicles. Also, to make the application more versatile it is also important to allow quick edit mode activation. As seen in figure 16 the user has quick access to all those features.

![Figure 16: Quick access options for the scenario](image)

Further customization for each of these options is required to better describe some of the properties that define them. When we add roads, we add road segments. As previously mentioned, the road segments are cubic Bezier curves. One important property that needs to be provided is the type of knot continuity we would like to have between the curves. The GUI provides quick access to this type making the process simple and fast, as seen in figure 17.

![Figure 17: Road segments quick access options](image)

In the case of intersections, we distinguish two types of intersections that are most common, the circular and crossed. Is it therefore important to give a easy way to change between the two types when designing a road network. We exemplify this in figure 18.

![Figure 18: Intersections quick access options](image)

In the case of vehicles there are no special options that require quick access and as such there are none.
When editing, the user can select what items he wants to edit. The items that can be edited are the same as the ones that can be added, road segments, intersections and vehicles, as seen in figure 19. Having each of the items editable separately makes the process less complex and at same time not losing depth. At same time, selecting an active mode tells the user exactly what he can or can't do.

![Figure 19: Edit first level quick access options](image1)

When editing road segments, one of the options we have is to unite them. As such it is important to provide the user with an option to choose what type of knot join will use. As seen in figure 20, the same levels of continuity are provided as when building the roads. In the case of C1 and C2, one or more elements that are united will be modified. The first item that will be modified in order to satisfy the continuity conditions will be the item that the user itself is modifying. If other items are to be modified due to that, the order in which it will happen is a cascade from the current item. Figure 20 shows the GUI for edit mode for road segments.

![Figure 20: Road segments edit full quick access options](image2)

Vehicles have two elements visible, their position at the current time and the path which includes the start and end poses. By changing the position of the vehicle we introduce a deviation in the path. These deviations are not visually editable but they can be changed via the property panel. The graphical interaction has not been activated due to addition work that needed to be done in order to ensure a working program, work that wasn’t possible to include in the “to do” list. The elements of the path that can be changed are, like previously said, the start and end poses. When changing these elements we directly change the path and at same time the deviations. After such an operation, a revision of the deviation list is recommended to ensure that they still provide what the user first intended. Figure 21 shows the quick access options for vehicles.
For intersections, all elements that are part of them can be edited at same time so there is no need to split them when entering edit mode. To enter edit mode for intersections it is enough to press the associate option for it as displayed in figure 22.

Figure 22: Intersection edit mode active

Figure 23 denotes the area which contains the elements for playing the scenario.

Figure 23: Scenario play controls

The area contains a slider, for ease of navigation forward and backwards in time, a time edit, which contains the current time for the current state, in Hour:Minute:Seconds format. Present are also standard buttons for playing, such as “Play”, “Pause” and “Reset”.

The two time indicators provide a good way to estimate the percentage of the current state of the scenario being played and at same time a specific time for that. Each scenario can have a time span that can vary from 1 second to 24 hours, as this represents the maximum amount of time that can be represented in the time edit widget.
5.3. Road network

The road network is the root of all road type objects. It is what the name suggests, a road network. Considering the road network as the root, we can look at the road segments or intersections as children of the network. Here, we store the information regarding children. The road network is also responsible for storing information regarding relations between its children. In figure 24 we present a basic tree for the road network and possible children. We can also look at the road network object as a container for the other components and a way to manipulate all of them at once.

![Figure 24: Road network tree](image)

The road network holds this information from an object oriented point of view. Another purpose, like previously mentioned, is to hold the information regarding the relations between its children. The information is contained in an undirected graph. One of the purposes for this is to be able to generate a path for vehicles.

First, each time a road segment or an intersection is created by the user it is automatically added to the list of children of the road network. At same time the object is added to the graph, with no connections. In figure 25 it is presented a sample graph of how items could be connected.

When an item, road segment or intersection, is created it is always disconnected from other items. It should also be noted that in the graph structure, the items comprising the road network are the nodes, while the connections between them are the edges.

Analyzing the sample graph it should be noted that road segments can have at most two connecting edges. Intersections on the other hand can have any number of connections. In the realm of possible connections we have all possible combinations. In other words, a road segment can be connected to another road segment or to an intersection. At same time an intersection can be connected to a road segment or to another intersection. This gives flexibility when constructing a
road network. With those rules and items almost any type of road network can be constructed in a fast, efficient way.

![Sample graph](image)

**Figure 25: Sample graph**

5.4. Road segments

In the project, road segments are a piece of a road that does not intersect others or itself. Road segments are designed using cubic Bezier curves. A road segment is formed by one or more curves. It is important to note that from here after lines are also referred as curves, when referring to the roads. Lines are curves with infinite radius.

5.4.1. Components

All the items that form a road segment are created by the user using the GUI. Each road segment is comprised of at least one curve. Curve, as defined in the project, is one cubic Bezier curve. In figure 26 the Bezier curve is represented by the white dashed line.
Figure 26: Road curve

The road curve is a result of the user choosing the control points. The control points define the shape of the road. In the figure 27, the control points are represented with the blue rectangle. In figure 27 it can also be seen the tangents at the start and end of the curve. The black lines represent the tangent.

Figure 27: Control points

The control points are the most basic objects that are used to design the roads. Because Bezier curves are defined by those points it is not needed to directly draw the curve, specifying those points is a faster, more reliable process.

From a hierarchic point of view, the components that form a road could be looked at as a tree. The root in this case would be the road segment. Each road segment can have any number of road curves, but at least one. Each road curve, or Bezier curve, has exactly two lines, with the end points of the lines forming the control polygon for the

On the next level for the road segments, we have the tangential lines defined
by two points, such as start point and first control point or the second control point and the end point. For this level there is also allowed only moving. The other edit possibility, in the works of rotation, can be achieved by moving one of the points that form the respective line around the other point. There is only one issue with this approach that was deemed unnoticeable and therefore ignored. While moving a point around another point can modify the length of the line, and thus modifying the line. This could have been avoided by allowing the line to be rotated.

Moving up the ladder of item levels we reach the curve itself. The curve is the lowest item that is allowed to be rotated. Moving, or displacing an item, is allowed, generally, for all items that are present inside a scene. The rotation centre coincides with the scene position for the item in question.

The top most level item is the road segment. This item, much like the road curve can be translated or rotated.

5.4.2. Design

When the user interacts with the canvas, at any point there are three things happening. First there is a press of mouse button, a move while a mouse button is pressed and a release. This sequence triggers the program to construct the appropriate component of the current road segment. As such we distinguish two states that can exist at different times. First state occurs when user starts a new road segment. At this point the road segment will not be saved in the road network until it contains at least one curve. In this state the user decides the start position of the curve, and implicitly of the road segment, as well as the first control point, that defines the tangent at the start point. The second state occurs after releasing the mouse button. In this state the user already provided a start point and a control point. In order to be able to generate a curve a second control point and an end point are needed. In this state pressing the mouse button confirms the end position for the current curve and at same time the start position for the eventual next curve that is part of this road segment. Releasing the mouse button confirms the second control point for the current curve, which determines the direction from which the curve arrives at the end point and at same time it determines the position of the first control point for the eventual next curve. The join of the curves is done using C1, which is tangent continuity. This continuity can later be changed in the edit mode. These states can be seen in figure 28 and 29 post completion. In figure 28 we have the canvas post first state. The blue square represent the starting point and the first control point for the first curve as previously mentioned. In figure 29 it can be seen how it looks post first second state.
From hereafter the program will be in the defined second state until the user changes active mode, at which point it will automatically finish the road segment, or decides to complete the current road segment and construct a new one.

5.4.3. Properties

Each road segment has a set of properties that can be editable. Those properties are seen as the minimum required in order of having a flexibility that is deemed relevant to the design of the roads.

One of the defining elements is the start and end points. These elements are very important for the fact they determine the position of the road segment as well as they are the points which connect to other points, belonging to other road segments. We can see them as connectors. We can also refer to them as SP (start point) and EP (end point). Those points are defined by the first and the last curve that belong to this road segment. The types of connections that can be made between road segments are unrestricted. Connections such as SP – SP or EP – EP are possible.
Another element is the lanes, as well as lane width. This element is directly linked to the curves. Each curve will have the same number of lanes on either side as set for the whole road segment. The lane width determines the placement of each lane in relation to the middle of the road, which is the Bezier curve drawn by the user. In figure 30 we can depict one road segment containing road curves. The green in figure 30 represents the middle of the lane.

An important note is the lack of restrictions. Due to the fact that the curve and the road are presented visually was decided to leave out constraints such as minimum radius of curvature. This presents itself as a problem when offsetting the centre of the road, as it is required when drawing the lanes. In figure 30 it can be seen that lanes overlap that lead to discrepancies in the visuals of the road. These discrepancies are further propagated to the vehicle motion. Because of the visual representation we chose to let the user design them as they pleased. Such errors, or special cases, can easily be recognized without any type of measurement taking place.

![Figure 30: Radius design example](image)

5.4.4. Editing

It is imperative to have quick, efficient and visible ways to edit the components of the scenario. As such, for this task, special attention has been devoted. From an item point of view, there are several levels of editing.

Lowest level is considered the point. Any point for that matter. The points can be moved around and they directly affect whatever item is directly connected to them. In turn this propagates according to the rules discussed previously, and updates all the necessary elements. Editing them, which only includes moving them, is simple, straightforward and adds enough depth not to go further with that part.

On the next level for the road segments, we have the tangential lines defined
by two points, such as start point and first control point or the second control point and the end point. For this level there is also allowed only moving. The other edit possibility, in the works of rotation, can be achieved by moving one of the points that form the respective line around the other point. There is only one issue with this approach that was deemed unnoticeable and therefore ignored. While moving a point around another point can modify the length of the line, and thus modifying the line. Moving up the ladder of item levels we reach the curve itself. The curve is the lowest item that is allowed to be rotated. Moving, or displacing an item, is allowed, generally, for all items that are present inside a scene. The rotation centre coincides with the scene position for the item in question. The top most level item is the road segment. This item, much like the road curve can be translated or rotated.

The translation or rotation begins when pressing the mouse button. If an item gets selected while the user holds the mouse, it will move the same distance as the distance traveled by the mouse cursor. When the mouse button is released the effected changes take place.

In figure 31, on the left, there is a road segment that contains two road curves in a certain position. On the right it is the same segment after a line item has been translated.

The translation or rotation begins when pressing the mouse button. If an item gets selected while the user holds the mouse, it will move the same distance as the distance traveled by the mouse cursor. When the mouse button is released the effected changes take place.

5.5. Intersections

In traffic, we distinguish two types of intersections commonly. The crossing where two or more road segments meet and the circular which only requires one road segment to be connected to it. The main difference is that in a crossing type
intersection, a vehicle changes road segments directly. As such, this type of intersection can be looked at from a design point of view as a point. On the other hand, a circular intersection can be looked at as an ellipse (circles are special cases of ellipse) and the transition from one road segment to another includes traversing a portion of the intersection. Crossing intersection can also be looked at as a circular intersection, with radius 0. For that reason the main focus will be describing the circular intersection.

5.5.1. Components

In the case of crossing type of intersection there is no real need for design. By just allowing three or more road segments, or other type of intersection to be connected, we can form a crossing. The join between the items, in this case, from a continuity point of view, will be C0. However, for circular intersection some elements can be defined that make them unique, or different from other such intersection. One of the components is the inner ellipse (circle). For simplicity it will be further referred as a circle. The circle can have different radiuses, giving the intersection one of the defining properties, as seen in figure 32. At same time each intersection can be connected to any number of road segments, provided the radius of the circle is large enough to support them. An example can be seen in figure 32. The arms, or exits, seen in figure 33 can be connected to other road segments or intersections.

![Figure 32: Inner Circle of a circular intersection](image-url)
5.5.2. Properties

We distinguish three properties for the cross intersection: position, slope and length of exits.

Circular intersections properties are defined by the radius, number of exits. In turn each exit has properties that define their state. Those properties are the position on the circle, angle and length.

5.5.3. Editing

Like with road segments drag and drop using left mouse button allows us to translate the intersection. Holding down “Alt” key and left mouse button at same time allows us rotate them around their center.

Each exits’ properties can be edited in same manner as other items. In figure 34 there is an intersection with one exit.

It can be seen, in figure 34, that the exit is at around 35° on the circle, has a 30° inclination and a certain length. In figure 35, all of the above are modified. For translation we use left mouse button and when we want to rotate we hold down “Alt” key. The possibility to edit it, gives much freedom when designing intersections. At same time, giving the user liberty in adding or removing exits increases the speed
for the design process of a road network.

Figure 34: Modified intersection based on figure 34

5.6. Vehicles

Vehicles are the main subject of this project. The purpose is to study vehicle interaction. Vehicles can be added to the scenario on any road. The constraint is in the fact that they can only be placed on roads and this limits the freedom to play out all types of scenarios, such as off road driving.

5.6.1. Properties

The properties that should be expected are the ones expected of a two dimension animated graphic object. It has a position, a rotation, which form a pose. It also has a destination and a path. Other properties include front and back axis position in relation to the size of the vehicle. This property is one of the two factors that determine the center of rotation for the vehicle. They are also one of the factors that determine how fast the vehicle can turn. Smaller distance between the axis results in a smaller minimum radius of curvature for the vehicle path, given that the maximum steering value remains unchanged. Radius of curvature for a vehicle is depicted in figure 36.

Figure 35: Radius of curvature and center of rotation for a vehicle
5.6.2. Path planning

When dealing with vehicles in the editor, there is an implemented algorithm that calculates a vehicle's path. A vehicle trajectory is calculated based on the start pose, end pose and the starting time and respectively ending time.

First step in calculating trajectory is to determine if the destination is reachable, or in other words if the road containing the initial pose is connected to the road containing the end pose. If the answer is no, the outcome will be that the vehicle will remain stationary until the roads that contain those poses become connected. Figure 37 shows an example of reachable destination while figure 38 shows an example of unreachable destination.

If the destination is reachable, a graph search will return the possible paths that the vehicle can take in order to reach the destination. Following this, the shortest path in terms of time needed is selected and assigned to the vehicle. It is important to note that if at some point after a path has been calculated for a vehicle, the shortest path changes in length, such as stretching one of the road segments, the calculated path will not change. Four distinct behaviors were discovered that affect the calculation of the path. Those behaviors are determined by the combination of start and end pose in relation with the driving direction for the respective lanes. We have two states with two possible values which lead to $2^2 = 4$ possible combinations. The path planner takes into consideration those states.

![Figure 36: Reachable destination (start pose in red, end pose in blue)](image)
For the state where the start and end pose correspond to the normal driving direction for the respective lanes, the vehicle follows the path driving normally. In the case where both vehicle poses are reversed, the vehicle will drive UK style, left side driving.

For the two cases where the poses are in not in same relation with the road, i.e. start pose is normal while end pose is reversed, the path planner will first look if a correction can be made. By correction it is meant that the vehicle will change the relation between itself and the lane it is on at a certain point in time such that it matches the relation at destination. In the case that this is possible, it will do so; otherwise the vehicle will not move. The impossibility to move is determined whether the vehicle is required to do a U-turn in order to reach the destination.

The user can visualize the reasons for vehicles not moving and correct them if required.

5.6.3. Path deviations

Allowing the user to modify aspects of the scenario is the core of this project. As such, when inspecting possible scenario, small deviations may be required in a vehicle at a certain time. By having the feature to just stop the scenario playback and modify items we ensure a greater capacity to analyze higher amounts of possible scenarios in shorter time frames. From a design point of view, deviations only so often can be allowed otherwise the risk of undefined behavior will increase. Introducing such deviations is a straight forward and simple process. In order to be able to modify items in the scenario the user first needs to pause it and jump at the desired position in time, if other then the current one. Once there he then can turn on the edit mode which will allow him to make modifications, including deviating a vehicle. The deviation will carry on even if the original path that was created for
changes due to introducing new roads, modifying old ones. The deviations are stored as relative positions on the original path and are applied each time the path suffers modifications. A list of deviations can be seen by the user and they can be removed if so desired reverting the changes they affected onto the vehicles path. Figure 39 illustrates an example of a deviation.

![Figure 38: Example of a deviation](image)

5.6.4. Predictions

Vehicle predictions are updated at every time step. They are based on current pose and speed. A vehicle predictor calculates all the possible poses for an amount of time steps set by the user and displays the result visually. In figure 40, it can be seen a sample of how predictions will look. The red rectangle represents the vehicle; the darker grey represents the predictions for the next time step and lighter grey for the next step after that. As it can be seen the predictor calculates all possible poses while taking into account constraints. One such constraint would be maximum speed at a specific pose. At same time it is important to determine the maximum turn for a vehicle at a given pose. As seen in figure 40 the prediction is symmetric for the case where the relative heading is 0. When the heading is different from 0, the prediction changes shape and it can be seen in figure 41.

![Figure 39: Two time step prediction for a vehicle with 0 heading](image)
Collisions are split into two types, collisions that are occurring or now collisions, and prediction collisions or possible collisions.

The now collisions are detected using the vehicles position and shape. In other words it looks just at special type of collision since the only time that is possible is “now”. In the case of prediction collision, time has to be taken into account as well since while visually they might look like they will collide, it is not necessary so. It can be said that this is spatial and temporal collision detection since it has to be detected if the vehicles occupy same space at same time.

Collisions are highlighted by changing the colour of the vehicles colliding. Figure 42 illustrates a “now” collision while figure 43 a future possible collision. Collisions are color coded to differentiate between types.
5.6.6. Editing

The editable properties of a vehicle are as follows.

The starting pose can be changed by moving the indicator for it. This will also result in a recalculation of the path for the vehicle in question as well for the current position.

The end pose has same behavior as the starting pose from an editing point of view and as such can be changed by relocating the visual indicator for it. Similar to changing the start pose, a recalculation in vehicle path and current pose is to be expected.

Small deviations can be introduced to the normal path of a vehicle by stopping the scenario playback and moving the vehicle around the current position. It is important to note that introducing more than one deviation in same area can
lead to sporadic behavior for the vehicle. These deviations allow the user to directly affect a possible collision and at same time to observe how changing the current pose of the vehicle affect the prediction.
6. Conclusions and future work suggestions

We have managed to achieve an editor that allows a user to create a traffic scenario that includes designing the road network, adding vehicles and destination for such vehicles. This can be done in a very short amount of time leading to an efficient tool that can assist HMI to assess dangerous traffic scenario and at same time come up with solutions that can reduce the number of accidents, traffics jams and other. It also provides an interactive and fast way of presenting the potential of new solutions in road safety and related. Almost any type of road network can be created and as such an immeasurable number of scenarios can be played. Having the possibility to save and load the work leads to the ability to replay scenarios created by other people, meaning that different people can cooperate in reproducing or creating specific scenarios. The software requires no additional training giving the possibility to anybody to use it.

The program can be expanded in numerous ways. The immediate additions should be related to the vehicles themselves. As such velocity profiles seem a natural next step in improving the utility of the program. We can further increase the number of possible scenarios by adding a “smart” component to the vehicles, like the possibility to choose whether an obstacle, or car, should be avoided and a new path generated. Continuing in this direction one could implement a platoon driving function for vehicles where they just follow a lead vehicle. Having added those before it is then fairly fast to implement different types of vehicles such as trucks or motorbikes. Moving from the vehicle improvements onto road ones, traffic rules can be added and non-vehicle obstacle, i.e. fallen trees, pedestrians. While such additions are possible the key aspects of this program should be kept in mind when adding them, rapidity and flexibility. When building a road network we can also think of adding an engine that can create a road network from an image. This can then make possible interaction between this application and others that deal with mapping of roads.
7. References


[3] Bing-Fei Wu; Wei-Hsin Chen; Chai-Wei Chang; Chao-Jung Chen; Ming-Wei Chung; , "A New Vehicle Detection with Distance Estimation for Lane Change Warning Systems," Intelligent Vehicles Symposium, 2007 IEEE , vol., no., pp.698-703, 13-15 June 2007


