Universität Duisburg-Essen · Campus Duisburg

Fakultät für Ingenieurwissenschaften

Abteilung Maschinenbau

Lehrstuhl für Mechatronik

Bachelorarbeit/Masterarbeit

zur Erlangung des Grades eines

Bachelor/Master of Science Maschinenbau (Vertiefung Mechatronik)

Thema: **DEUTSCH**

ENGLISH

vorgelegt von: **Rui Sun**

geb. am 08.05.1993 in China

Matrikel-Nr.: 3057943

Erstgutachter: **Prof. Dr.-Ing. Dieter Schramm**

Zweitgutachter: **XXXX**

Betreuer: **Dipl.-Ing. Stephan Schweig**

Duisburg, XX.XX.XXXX

contents

List of Figures

Abstract

The driving simulation is becoming more and more important in the automotive field. The road network model is the basis of driving simulation. So there is a requirement to establish road network model which map the real road network. The traditional method is to create a road network model manually. But this way costs to much time and has high possibility to make the error. There are some companies offer some paid software though work well paid too much.

The goal of this thesis is to produce a plugin/program, which can generate geometrical road and relevant objects e.g signals, lane and elevation automatically precisely quickly and free according to openDRIVE which is a open file format describing physical properties of individual roads and road networks for driving simulation applications using Unity3D which is a game engine which can also be used as a simulator.

Keywords: driving simulator, geometrical road generation, OpenDRIVE, Unity3D

Chapter 1

Introduction

1.1 Motivation

There are many ways to logical description of a road network and road objects. For example, if you want to describe a straight road segment, simply assume it as a straight Line segment. Things become in a three-dimension coordinate system, determine a line segment. You can determine a line by start coordinate, and end coordinate or start coordinate, direction vector, and the length of the line. So there are several standardized formats developed to logical description of the road network and road objects. The major format is OpenDRIVE, developed by VIRES. Some companies provided their own commercial solution to generate road according to OpenDRIVE. But the way to implement still not too clearly and not under an open source lisence. So there is a requirement to analysis OpenDRIVE data format and visualization data to generate roads.

1.2 Objective of the work

This thesis work will be focusing on the development of plugins/program which can read specific OpenDRIVE format data and transfer it into a predefined data structure and visualization the data in a simulator(Unity3D). The openDRIVE already includes describing physical properties such as the length of the road and elevation, of individual roads and road networks, but it is in specific XML format, and it is difficult to fast read and interpret by a simulator, so it is necessary to interpret OpenDRIVE format data into a specific pattern that is better recognized by the simulator firstly.

The plugins/program should have the following features：

- **Easy to operate**
- **Efficient**
- **Be released under open source license**
- **At least visualization road structure, lane, elevation, signals, and crossing.**
- **Low code complexity**

1.3 Structure of the work

So building the application requires the following steps:

• data analysis and structure establish

read and interpret OpenDRIVE format data into a predefined data structure.

• road and texture generation

read data from the predefined data structure, analysis, and visualization of the data.

Chapter 2

Tools introduction

Since we already decided which kinds of tool to use, firstly introduce the tools.

And explain why we use the tools.

2.1 OpenDRIVE

OpenDRIVE® is a XLM format open file for the logical description of road networks[1].

The following figure shows the typical incorporation of an OpenDRIVE® file into a simulation tool-chain[1]：

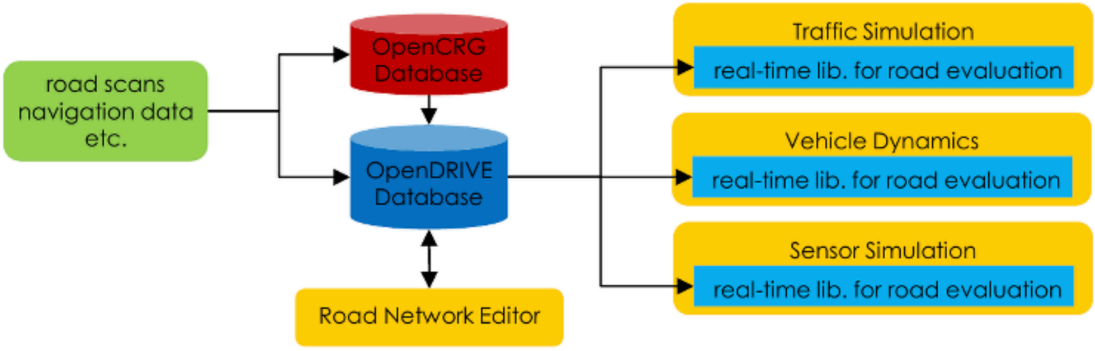


Figure 2. 1 simulation tool-chain[3]

From the figure we can see, mostly the OpenDRIVE database is produced by road scans or navigation data, and then the OpenDRIVE database will be used to e.g. vehicle dynamics, traffic simulation and sensor simulation[1]. In the plugins/program, it will be used to traffic simulation.

OpenDRIVE file uses a hierarchical structure which is established by XML nodes, to store the features that can describe the geometry of roads and the logic (e.g lanes, signs, signals).

Why use OpenDRIVE?

Official website of OpenDRIVE lists the following advantages：

OpenDRIVE® is under open source lisence [1].

OpenDRIVE® contains all important key features of real road networks[1].

OpenDRIVE® has established a format which has a broad international users[1].

2.2 Unity3D

Offical website write that Unity3D is a powerful cross-platform 3D engine and a user-friendly development environment[4]. 3D games and applications can easily established by it. *The engine can be used to create both three-dimensional and two-dimensional games as well as simulations for its many platforms*[5]. A simulator is a device that enables the operator to reproduce or represent under test conditions phenomena likely to occur in actual performance[6]. Our plugins/program is based on unity3D to performance driving-simulator.

Why we use Unity3D

1. In addition to its superior performance, the most important reason is that it can not only be used as a simulator, it is also a model making tool, so we can complete model generation and driving simulation in it at the same time.

2.Unity3D support using the script to create game objects. The script can help us efficient to analysis OpenDRIVE form data, generate geometry road and lane texture.

2.3 C#

According to the statement of the official website , C# (pronounced "See Sharp") is a simple, modern, object-oriented, and type-safe programming language. C# has its roots in the C family of languages and will be immediately familiar to C, C++, Java, and JavaScript programmers[8].

Why we use C#

Unity3D supported three kinds of languages for creating scripts: C#, unityscript and Boo, according to the official website, it recommends user to use C# in Unity3D script[8].

Chapter 3

Data analysis and structure establish

In this chapter, we will analysis OpenDRIVE data format, and establish a predefined data structure in order to simulator easy to read data and efficient to generate geometry road.

3.1 OpenDRIVE data format analysis

The OpenDRIVE data format is based on road nodes as a unit. An OpenDRIVE data format may include thousands of road node(figure 3.1 a). They can describe a road network. Each road node describes a road section and includes all record(e.g. length of the road, elevation of the road, the number of the lane in the road section (figure 3.1 b)) that can describe the geometry and logic of the road section.

After use the OpenDRIVE data format successfully to generate a road network, I found that it is not necessary(according to our goal of the thesis) to use all the record in each road node, in the following I will introduce some necessary record according to the official specification1.4[2] and how to use the record to generate a road section in each road node.

Before explaining the node record, firstly you should know some Co-ordinate Systems.

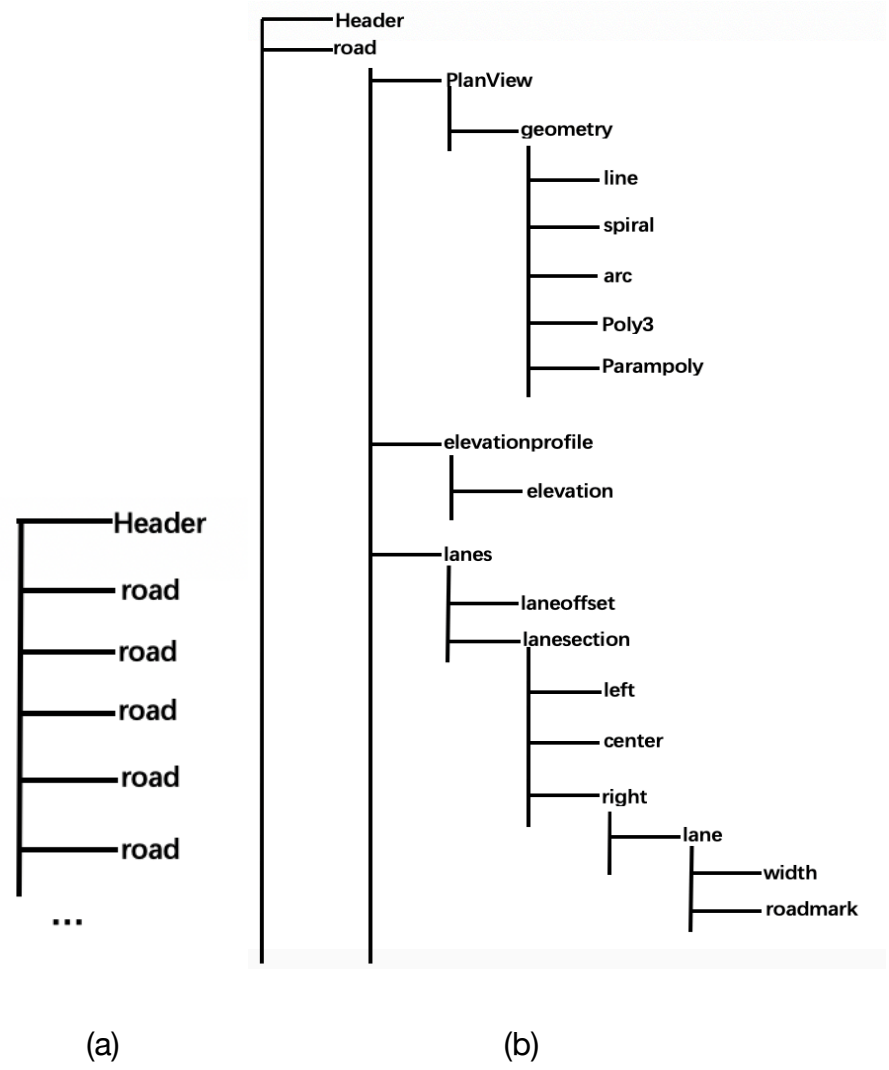


Figure 3. 1 OpenDRIVE Data Structure

- **Overview**

Track co-ordinates and inertial system are most frequently used in OpenDRIVE file.

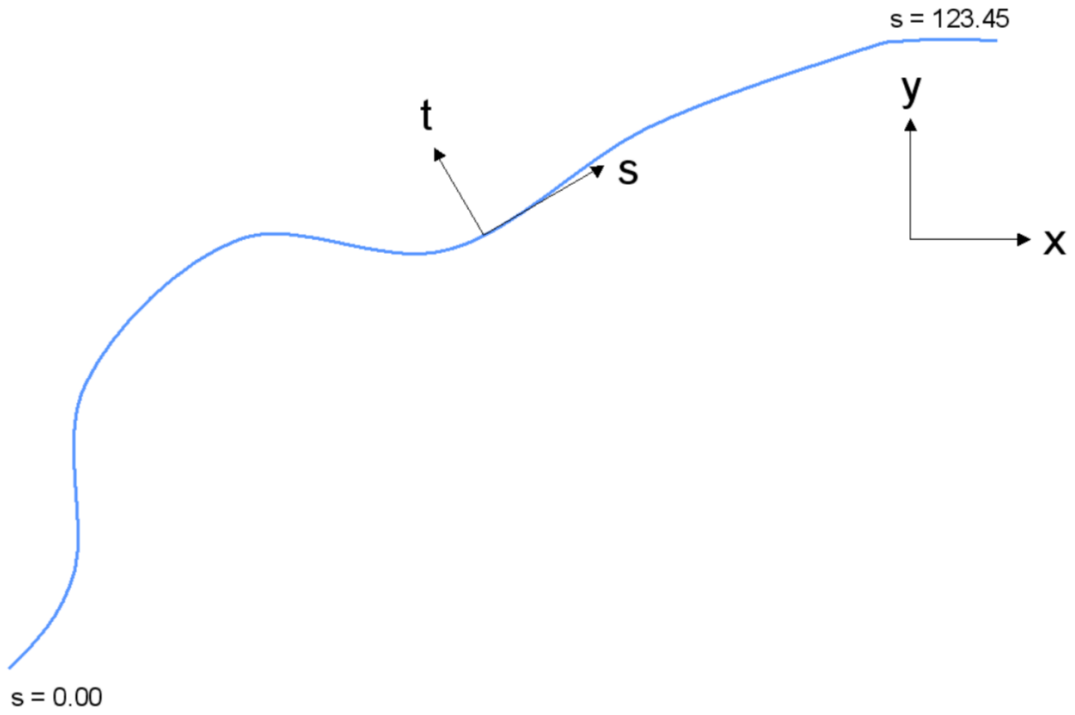


Figure 3. 2: co-ordinates System[3]

Figure 3.2 shows the two co-ordinates system. Each co-ordinates system in its dimension to describe the position vector. In the inertial system, assume the road track as a line track, vector (*x,y*) belongs to the inertial system. The track coordinate system applies along the line track. Vector (*t,s*) belong to track co-ordinate[3].

- **Intertial system**

You can consider an inertial system as a world co-ordinates system. Factor *x* means forward, factory *y* means left, factor *z* means up[3]. If you decide a position vector *(x,y,z)*, you can only decide on point in the inertial system. Points decide line, lines decide flat, so if you decide hundreds of thousands of points, you will decide any geometry road. For convenient, all geometry coordinate is based on the internal system.

- **Track system**

(a) (b)

Figure 3. 3: Track System 1[3]

According to the Format Specification, Rev. 1.4[3], factor *s* means position along the road track, measured in [m] from the beginning of the track, calculated in the XY-plane. Factor *t* means lateral position, left is positive. Factor *h* means up. The track system is most used to describe the logic of the road. For example, you can describe a new road lane line use two vectors *A* and *B* as shown in figure 3.4 compare to figure 3.3(a).

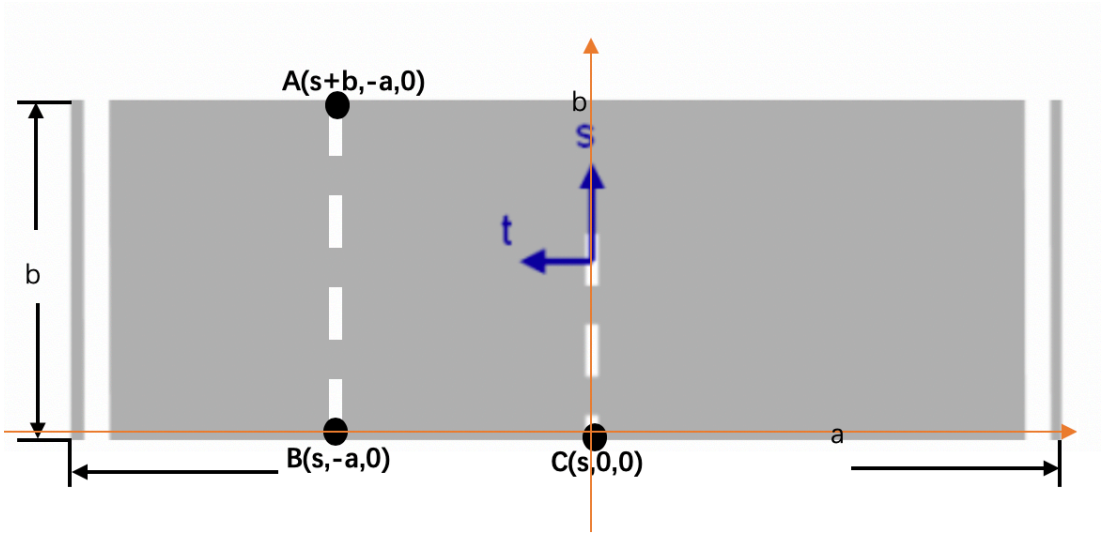


Figure 3. 4: Track System 2[3]

- **Road node**

The unit of the road network, road network may consist of hundreds of thousands of roads section. Each road node represents a road section, and include the describe record of the geometry and logic of the road section.it is mainly used to create the geometry of the road section[3].

- **Planview node**

Belong to road node. The plan view record contains a series of geometry node records which define the layout of the road's chord line in the x/y-plane (plan view)[3].

- **Geometry node**

Belong to planview node, there are five kinds of sub-nodes in geometry node line node, spiral node, arc node, poly3 node, and parampoly3 node[3].

```
<geometry s="0.0000000000000000e+00" x="-7.0710678119936246e+00" y="-7.0710678117566195e+00" hdg="7.8539816339160284e-01" length="4.8660000002386461e-01">
    <line/>
</geometry>
<geometry s="4.8660000002386461e-01" x="-6.7269896522493644e+00" y="-6.7269896520163819e+00" hdg="7.8539816339001800e-01" length="3.1746031746031744e+00">
    <spiral curvStart="-0.0000000000000000e+00" curvEnd="-1.2698412698412698e-01"/>
</geometry>
<geometry s="3.6612031746270390e+00" x="-4.3409250448547327e+00" y="-4.6416930098216129e+00" hdg="5.8383605706600694e-01" length="9.1954178989066371e+00">
    <arc curvature="-1.2698412698412698e-01"/>
</geometry>
<geometry s="1.2856621073533676e+01" x="4.3409256447834164e+00" y="-4.6416930099218154e+00" hdg="-5.8383605708086783e-01" length="3.1746031746031744e+00">
     <spiral curvStart="-1.2698412698412698e-01" curvEnd="-0.0000000000000000e+00"/>
</geometry>
<geometry s="1.6031224248136851e+01" x="6.7269902521255664e+00" y="-6.7269896521471884e+00" hdg="-7.8539816341104807e-01" length="4.8660000002379050e-01">
    <line/>
</geometry>
```

Figure 3. 5: Geometry Node

Figure 3.5 showed three main node—line node, arc node, and spiral node. Each kind of node can describe a specific geometry line, different geometries are connected end to end to form a geometric chord line[3]. In specification 1.5 the chord line is named road reference line.

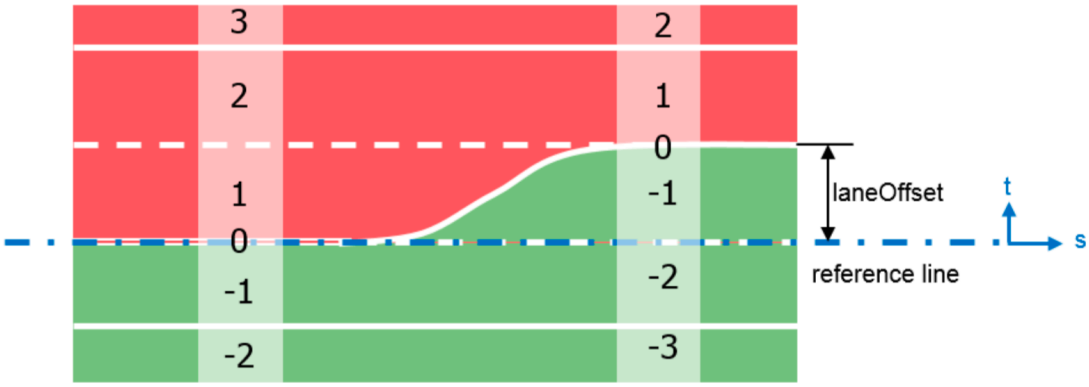
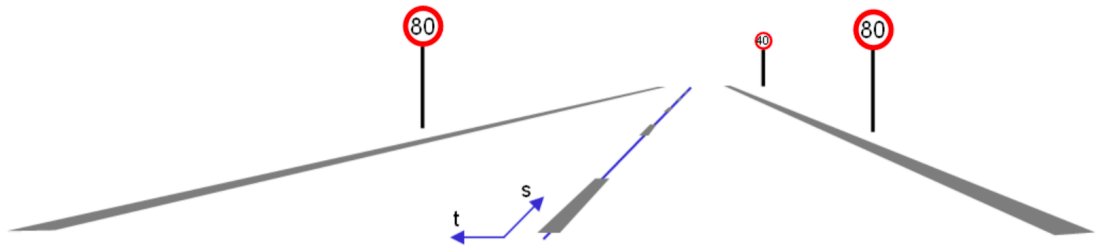


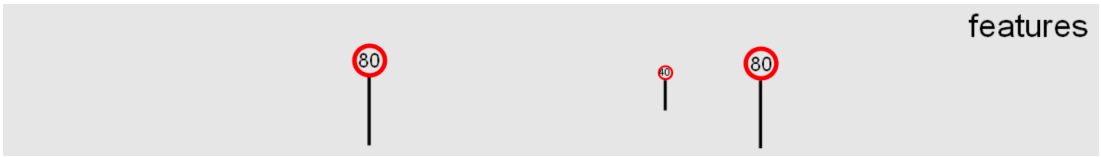
Figure 3. 6: Chord Line[2]

Some time the road reference (chord) line may have offset in order to easy to implement a logic road describe. So the laneoffset node is that to describe in which s position along the road reference line has offset, the new produced line named lane reference line. In figure 3.6 showed, the number 0 line is reference line.

As figure 3.7 showed some features (b) and lanes (c) are along the reference line (d) to describe a logic road.



(a) logic road



(b) features

(c) lanes

(d) reference line

Figure 3. 7: Reference Line[2]

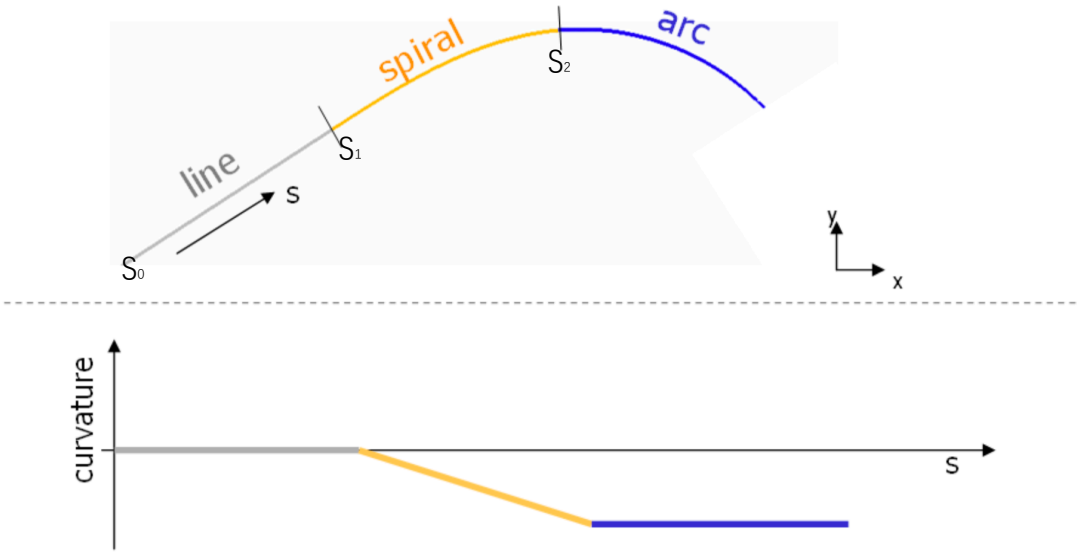


Figure 3. 8: Geometry Lines[2]

As the figure 3.8 showed, there are three kinds of geometry lines, first geometry is a line, the start position in the road section of the line is s0, the length of the line is s1-s0. The second geometry is spiral, the start of the spiral is s1 the length is s2-s1, third is an arc, the start position is s2 and the length of the arc is the length of the road section which the geometry belong to-s2.

So geometry nodes describe road reference(chord) line as figure 3.8 illustrate, laneoffset nodes according to the road reference(chord) line to describe lane reference line, When there is no laneoffset node, the road reference line is overlapped with the lane reference line. and other including logic record nodes (e.g features and lanes) along the road reference(chord) line to describe a road network.

- **Line node**

In the line node, there are five parameters--s, x, y, hdg, and length.

S being the absolute position (track coordinate system) in its road section, x/y is the coordinate of the point in Intertial system, hdg is heading angle of the line direction (Intertial system), and the length is the length of the line[3].

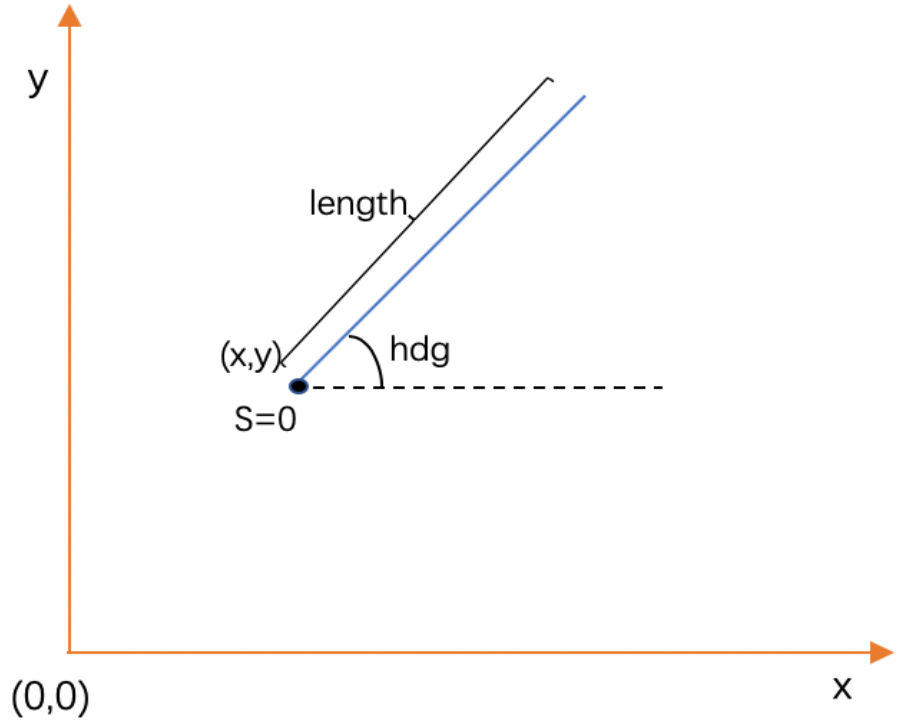


Figure 3. 9: Geometry Line

- **Spiral node**

Spiral node has two more parameters named curvStart and curvEnd.This record describes a spiral section from curvature=0 as part of the road reference line. For this type of spiral, the curvature change between the start and end of the element is linear[3]. The curvature change rate is cdot. The equation of calculating cdot is following:

$$cdot = (curvEnd - curvStart)/length_{spiral} \quad (3.1)$$

In figure 3.8, the yellow line section is the section of a spiral line.

Figure 3.10 Show curvature change rate is 1, according to different cdot, you can draw different kinds of spiral with different curvature change rate.

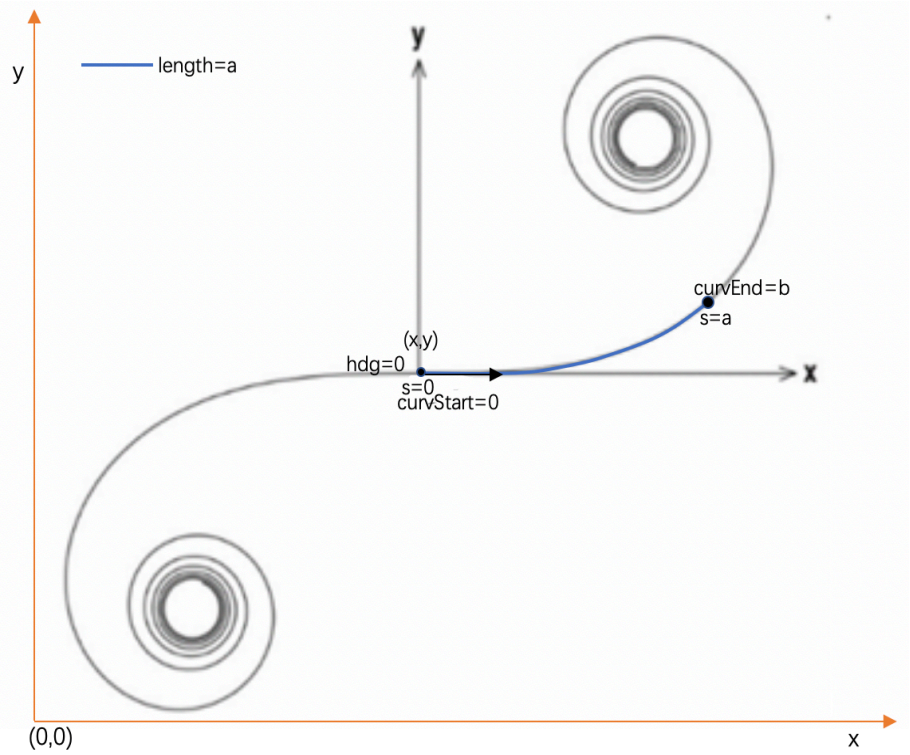


Figure 3. 10: Geometry Spiral

The cdot of figure 3.10 is calculated in following equation:

$$cdot = (b - 0)/a = 1 \tag{3.2}$$

- **Arc node**

Arc node has one more parameter curvature. It is enough to draw an arc using parameter curvature, s, x, y, hdg, and length illustrated in figure 3.11[3].

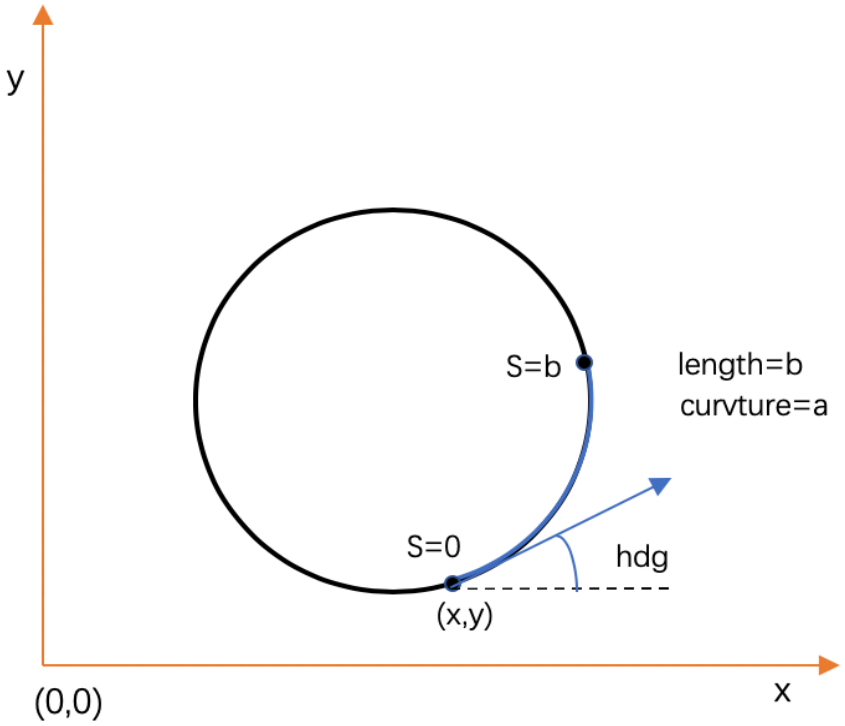


Figure 3. 11: Geometry Arc

- **Poly3 node**

This record describes a cubic polynomial as part of the road's chord line[3].

$$\text{vlocal} = \text{a} + \text{b} * \text{du} + \text{c} * du^2 + \text{d} * \text{du}^3 \quad (3.3)$$

According to the equation and the parameter poly3 node offered(figure 3.12 illustrated), you can easy to draw a geometry blue line showed in figure 3.12.

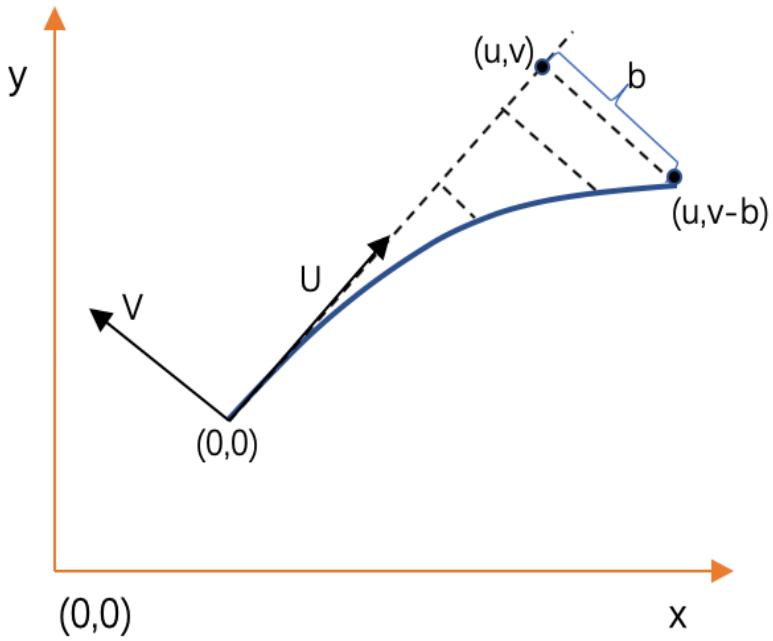


Figure 3. 12: Geometry Poly3

- **Parampoly3 node**

This record describes a parametric cubic curve as part of the road reference line[3].

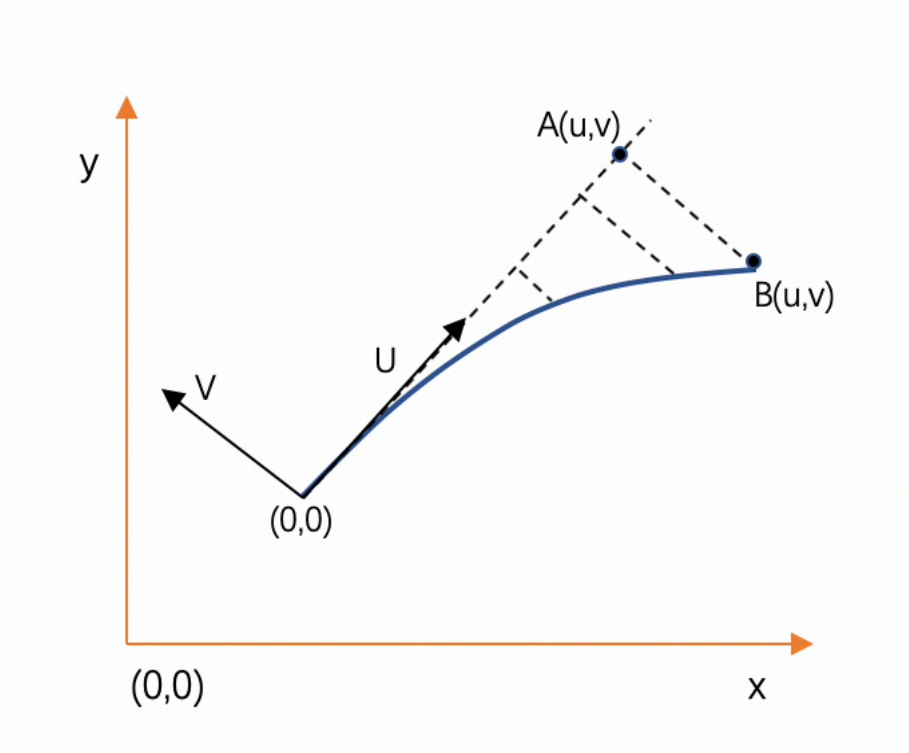


Figure 3. 13: Geometry Parampoly3

$$ulocal = au + bu * p + cu * p^2 + du * p^3 \quad (3.4)$$

$$vlocal = av + bv * p + cv * p^2 + dv * p^3 \quad (3.5)$$

According to the equation and the parameter Parampoly3 node (figure 3.13 illustrated), you can easily to draw a geometry blue line showed in figure 3.13.

- **Elevation node**

The elevation record defines an elevation entry at a given road reference line position. Entries must be defined in increasing order along the road reference line. The parameters of entry are valid until the next entry starts or the road's chord line ends[3]. Per default, the elevation of a road is zero.

The elevation is calculated by the equation:

$$Elev = a + b * ds + c * ds^2 + d * ds^3 \quad (3.6)$$

ds is the delta between the start position and the requested position.

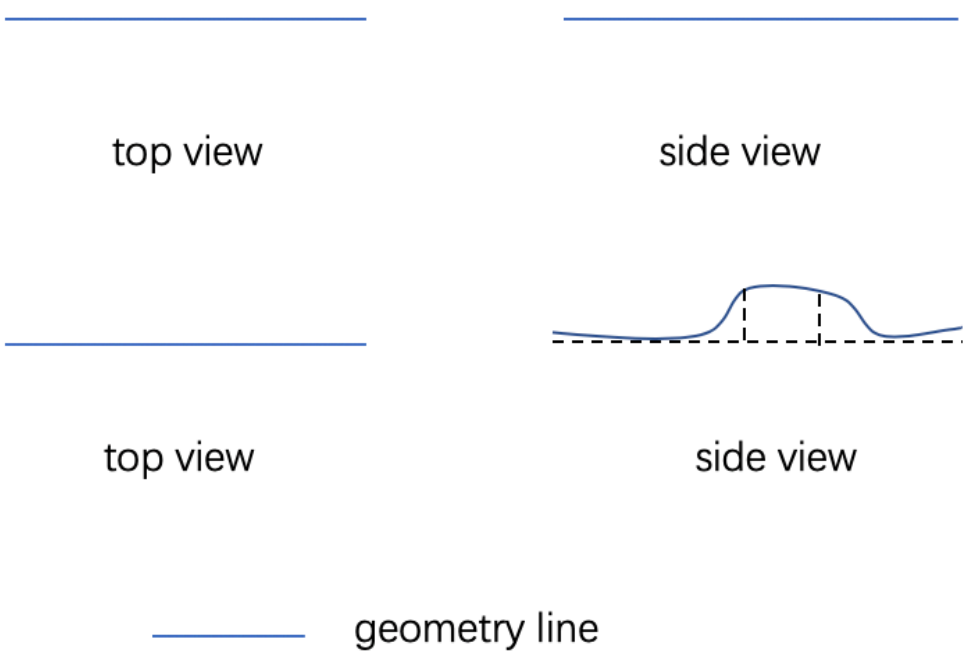


Figure 3. 14: Elevation

Figure 3.14 illustrates two geometry line. First two figure describes a line without elevation in two different views compares with the next two figure with elevation.

- **Lanes node**

Lanes node belong to road node. Lanes node is consist of by lanesection nodes. It describes the lane information in the road section[3].

It is mainly used to create the texture of the road.

- **Lane section node**

Lane section is the unit of the road section. Each road section may consist of some lanesections[3]. The way to calculating the length of the lanesecion is similar to the geometry node.

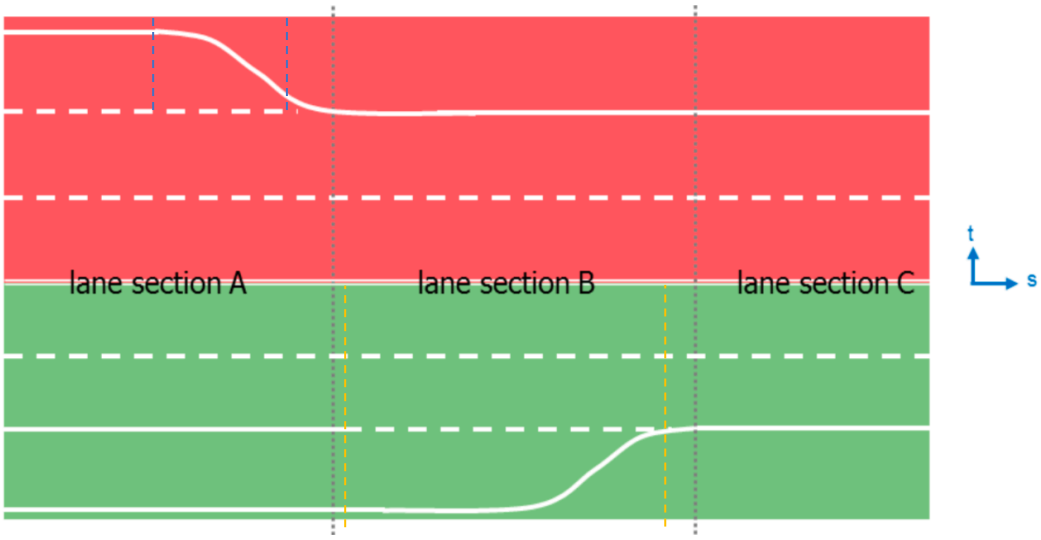


Figure 3. 15: Lanesection[2]

It is convenient to describe a road section using many lanesections when the number of lanes changed along the road section. As figure 3.15 illustrates, there are three lanesection. Lane section A has 6 lanes. Lane section B has 5 lanes. Lane section C has 4 lanes.

- **Lane node**

In each lanesection, there are three kinds of sub-nodes[3]. They are the left node, center node, and right node. Each sub-node has some lane node. Each lane node has a specific id in the lanesection node which lane node belongs to. as figure 3.15 illustrates, the red area described by the left node, the blue area described by the right node, and the number 0 line described the center node.

- **Width node**

In each lane node, there are some width nodes. It describes the track of the lane using formula. A new width node is required when the geometry formula is changed[3]. The way to calculating the length of the width node is similar to the geometry node. Each width node can describe a specific geometry of the lane section. As figure 3.15 showed, there are three width nodes in lanesection A red area divided by two blue broken lines. Some different width nodes can describe a complex geometry of the lane section.

- **roadMark node**

roadMark node is the sub-node of the lane node. It describes the type of lane[3]. Each lane may have not only one road mark type. They may be broken, soild none or others.

As figure 3.15 showed, in the lanesection B green area, there are three types of lane divided by the yellow broken line. The first section is soild, the second section is broken and the last section is none.

- **Signals**

The signal record is used to provide information about signs and signals along a road. Signals are signs that can change their state dynamically (e.g. traffic lights). The signal record consists of the main record and an optional lane validity record[3].

3.2 Establish predefine data structure

Each node in the OpenDRIVE data format has enough parameter to describe a specific geometry or logic relationship. So establish the following data structure to store the node from OpenDRIVE file.

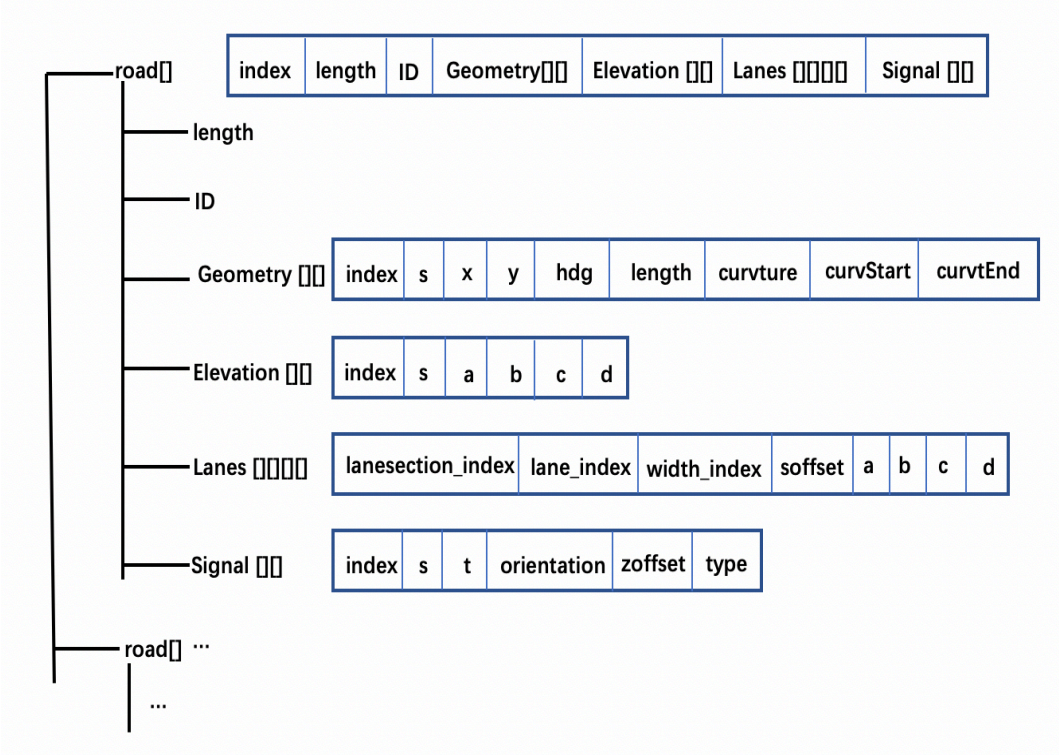


Figure 3. 16: Predefine Data Structure

As the figure 3.16 illustrate, in each road node has some sub-nodes which can describe the gemotry and logic relationship of the road section. So the outter loop is road[i] to read road node. in the road loop, there should be four array

1. geometry[index][parameter]
2. elevation[index][parameter]
3. lanes[lanesection_index][lane_index][width_index][parameter]
4. signals[index][parameter]

to store parameters.

- **Code implementation**

Algorithm: establish predefine data structure and load data from OpenDrive

```
for r in road do
    road[r].length <-read road section length
    road[r].id <-read road section id
    for g in geometry do
        for p in geometry_parameter do
            geometry[g][p] <-read parameter in geometry node
    for e in elevation do
        for p in elevation_parameter do
            elevation[e][p] <-read parameter in elevation node
    for ls in lanesection do
        for la in lane do
            for w in width do
                for p in width_parameter do
                    lanes[is][la][w][p] <-read parameter in lane node
    for s in signals do
        for p in signals_parameter do
            signals[s][p] <-read parameter in signals node
Return predefine data structure including data from OpenDRIVE
```

Chapter 4

Road and texture generation

You already establish a predefined structure to store the parameter which comes from OpenDRIVE data format. Using the parameter you can easily to establish a road network in Unity3D. In this chapter, I will explain:

1. How to generate the geometry of the road and road mesh in Unity3D.
2. How to generate lane texture according to the predefined structure data in Unity3D.

4.1Road generation

Analysis line arc spiral parameterpoly3 and elevation node and generate road mesh in Unity3D. Unity3D scene is a world coordinate system(x,y,z). You can assume the inertial system overlap the scene coordinate system. So according to the inertial system coordinate, you can only decide one specific road network in Unity3D scene.

- **line analysis**

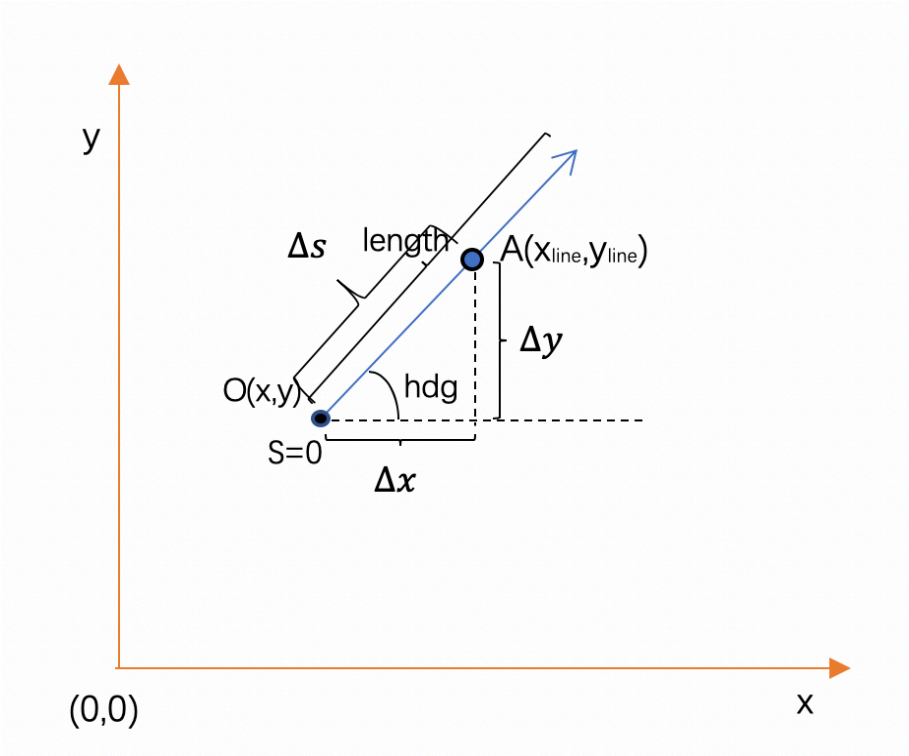


Figure 4. 1: Analysis Geometry Line

As figure 4.1 illustrates, the blue point $A(x_{line},y_{line})$ is a point in geometry line, it is calculated by the following equation:

$$x_{line} = x + \Delta s * \cos(hdg) \quad (4.1)$$

$$y_{line} = y + \Delta s * \sin(hdg) \quad (4.2)$$

Compare with figure 3.9, what is Δs means? As the figure 4.2 illustrates, a line consists of many points which have the same spacing named Δd which is defined by the user between each point.

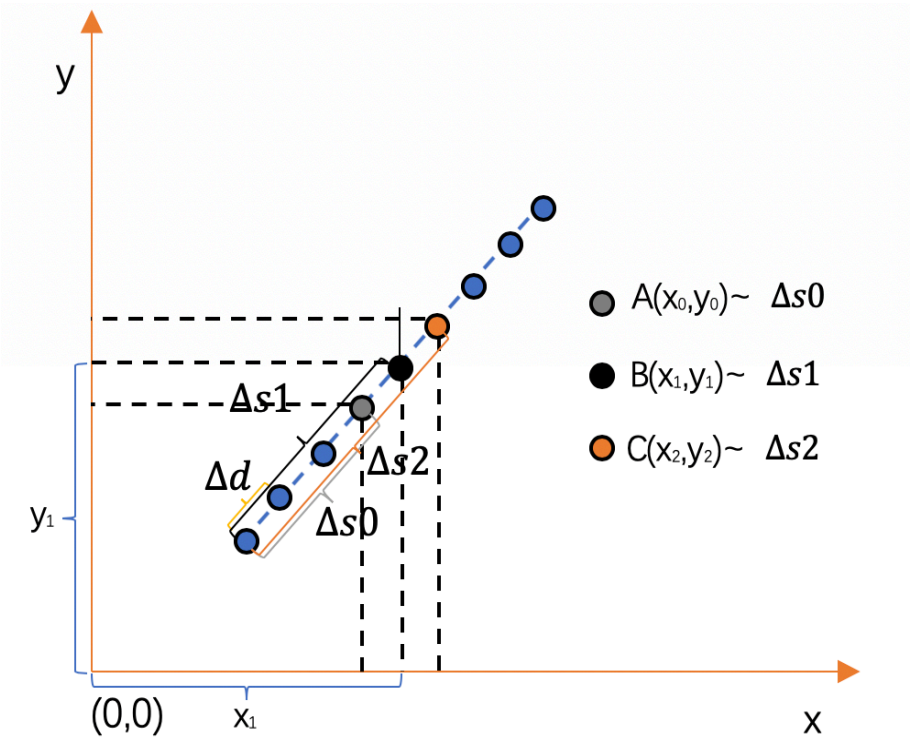


Figure 4. 2: Congruent Relationship

Δs consist of many Δd , for example $\Delta s0$ consists of three Δd and $\Delta s2$ consists of five Δd. Each Δs match specific a point in the geometry line according to Trigonometric function you can calcute point belonging to the line. So repeat calculate the points you will get a line. And this way also suits for next geometry analysis.

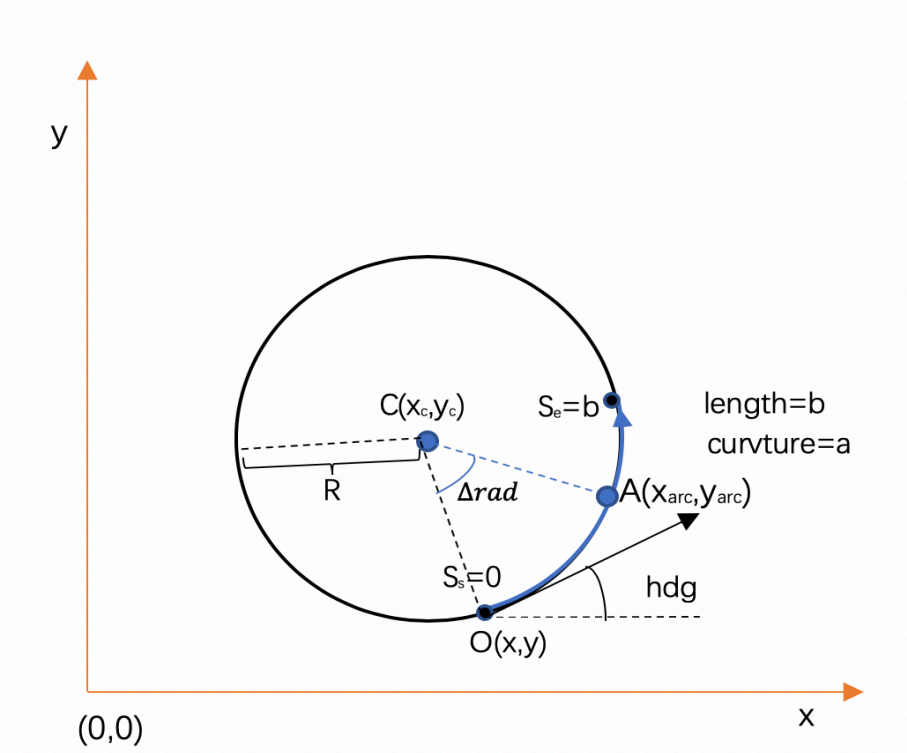
- **Code implementation**

Algorithm: calculate point set of geometry line
1: for Δs in range(0,line.length) do step +0.1
2: x_{line} *<- x+cos(hdg)* Δs*
3: y_{line} *<- y+sin(hdg)* Δs*

4: Return point set of geometry line

● Arc analysis

According to the specification1.5 convention[3], the curvature is the position when the arc is counter-clockwise motion, and negative when the arc is



clockwise motion.

Figure 4. 3: Analysis Geometry Arc

As figure 4.3 illustrate, first decide the radius of the Arc using the following equation:

$$R = |1/curvature| \tag{4.3}$$

Then according to the curvature symbol to decide the center coordinate $C(x_c,y_c)$ of the circle.

$$x_c = \cos(hdg \pm \Pi/2 - \Pi) * R \tag{4.4}$$

$$y_c = \sin(hdg \pm \Pi/2 - \Pi) * R \tag{4.5}$$

Then according to the center coordinate of the circle to calculate the point which on the arc.

$$x_{arc} = x_c + \cos(hdg \pm rad + -\Pi/2) * R \tag{4.6}$$

$$\mathrm{y}_{arc} = y_c + \sin(hdg \pm rad + -\Pi/2) * R \tag{4.7}$$

Lastly repeat above procedure using variable *rad*.

$$rad = \Delta \mathrm{s}/length_{arc} \tag{4.8}$$

- **Code implementation**

Algorithm: calculate point set of geometry arc
1: radius<- abs(1/curvature)
2: for Δ*s* in range(0,arc.length) step +0.1
3: if *curvature* <0
4: x_*center* <- *cos(hdg+PI/2-PI)*radius*
5: y_*center* <-*sin(hdg+PI/2-PI)*radius*
6: x_{arc} <- *x_center+cos(hdg-raidus+PI/2)** Δ*s*
7: y_{arc} <-*y_center+sin(hdg-radius+PI/2)** Δ*s*
8: if *curvature* >=0
*9: x_center <-cos(hdg-PI/2-PI)*radius*
*10: y_center <-sin(hdg-PI/2-PI)*radius*

11: x_{arc} *<-x_center+cos(hdg+radius-PI/2)*Δs
12: y_{arc} *<-y_center+sin(hdg+radius-PI/2)*Δs
13: returen point set of geometry arc

- **Spiral analysis**

According to official instruction, it recommends to use it offers function odrSpiral to analysis the spiral, so I transfer the official code which written by C to C#. the odrSpiral prerequisite is assumed geometry start point *curvature* is 0 as the figure 4.4 point $O(x_s, y_s)$ showed. The function needs two parameter *cdot* and Δs. It returns two value Δu and Δv in inner coordinate system illustrated in figure 4.4. the procedure of draw geometry spiral is described as the following steps:

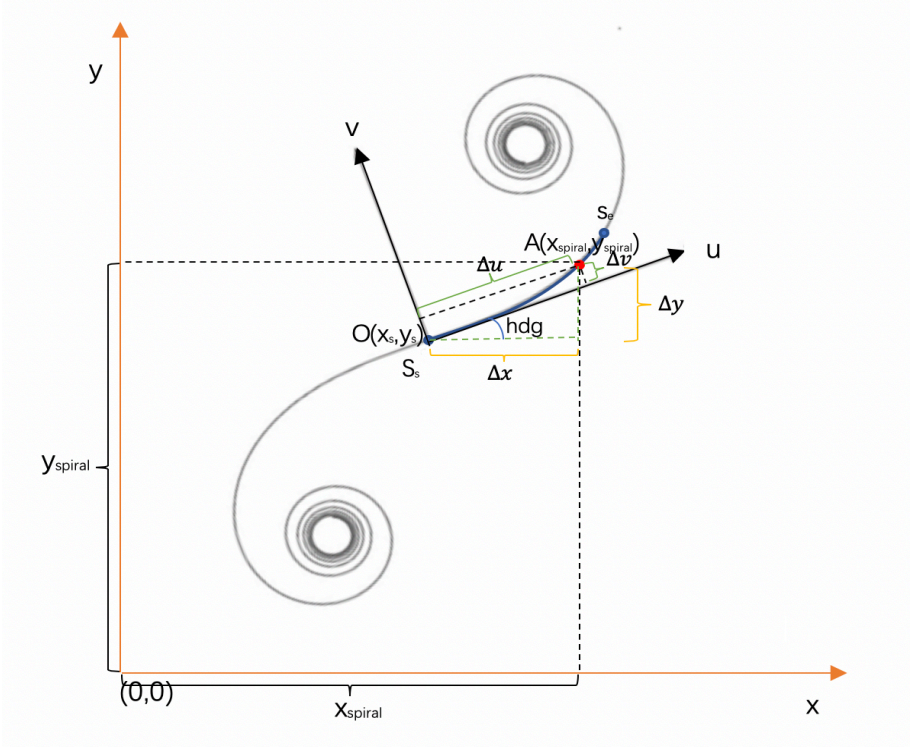


Figure 4. 4: Analysis Geometry Spiral

1.use parameter Δs and cdot and function *odrSpiral* to calculate Δu and Δv in the inner coordinate system. According to the following equation:

$$\Delta u = \text{odrSpiral}(\Delta s, cdot).\text{x} \tag{4.9}$$

$$\Delta v = \text{odrSpiral}(\Delta s, cdot).\text{y} \tag{4.10}$$

2.Transfer Δu to Δx and Δv to Δy using the following equation:

$$\Delta x = \Delta u * \cos(hdg) - \Delta v * \sin(hdg) \tag{4.11}$$

$$\Delta y = \Delta u * \sin(hdg) + \Delta v * \cos(hdg) \tag{4.12}$$

3.Calculate Point $A(x_{spiral}, y_{spiral})$ in inertial coordinate System according to point $O(x_s, y_s)$ Δx and Δy using the following equeation:

$$x_{spiral} = x_s + \Delta x \tag{4.13}$$

$$y_{spiral} = y_s + \Delta y \tag{4.14}$$

4. repeat procedure 1-3 according to variouble Δs increase progressively until $\Delta s = \text{length}_{spiral}$.

- **Small Trick**

In the real situation, the *curvStart* is not always being 0. For example, in figure 4.5 showed, obviously *curvStart* of point S is not equal to 0. How to draw geometry spiral in this situation when the prerequisite of function odrSpiral is assumed *curvStart*=0?

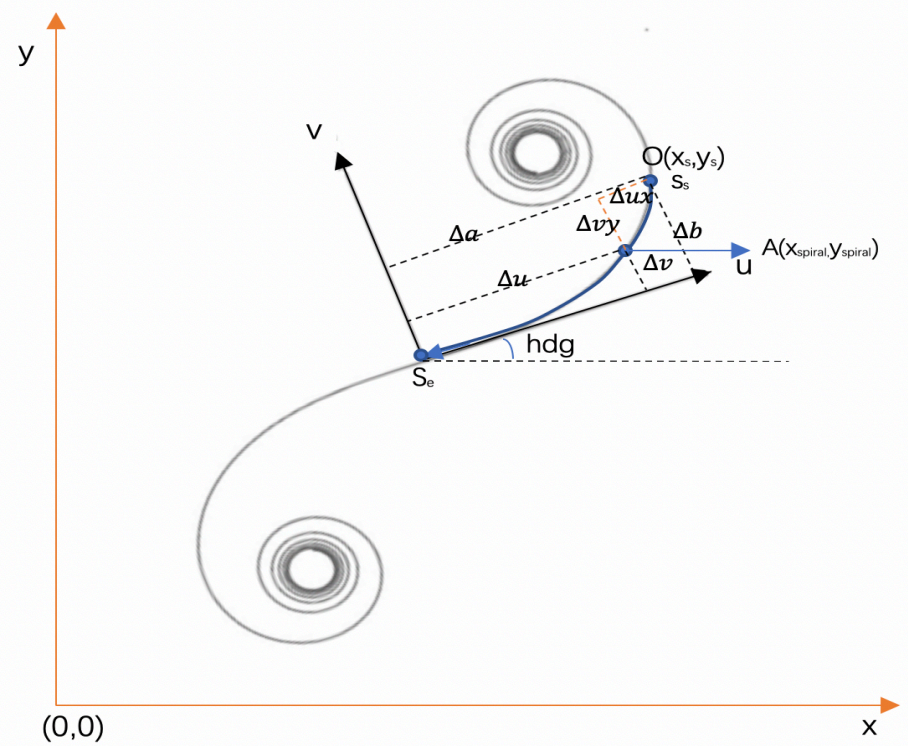


Figure 4. 5: Calculate Geometry Spiral

An alternative way is that through a series of calculations to get the point coordinate set which belongs to the geometry spiral in the inner coordinate system and rotate the point coordinate set to the normal situation (showed in figure 4.6), then rotate and translation the point coordinate set to inertial coordinate system as above already illustrated. The detail is illustrated in the following:

1.Assume the *curvEnd* is *curvStart* and *curvStart* is *curvEnd*, so that we can get two value Δa and Δb (showed in figure 4.5) using the following equation:

$$\Delta a = \text{odrSpiral}(\text{length}_{\text{spiral}}, \textit{cdot}).\text{x} \quad (4.15)$$

$$\Delta b = \text{odrSpiral}(\text{length}_{\text{spiral}}, \textit{cdot}).\text{y} \quad (4.16)$$

2.calculate the point coordinate in the geometry spiral using the following equation(showed in figure 4.5):

$$\Delta u = \text{odrSpiral}(\Delta s, \textit{cdot}).\text{x} \quad (4.17)$$

$$\Delta v = \text{odrSpiral}(\Delta s, \textit{cdot}).\text{y} \quad (4.18)$$

3.the point $A(x_{\text{spiral}}, y_{\text{spiral}})$ in inner coordinate system is calculate by following equation(showed in figure 4.5) and point $O(\Delta a, \Delta b)$ in inner coordinate coordinate system:

$$x_{\text{spiral}} = \Delta a - \Delta u = \Delta ux \quad (4.19)$$

$$y_{\text{spiral}} = \Delta b - \Delta v = \Delta vy \quad (4.20)$$

4.repeat step 2-3 according to variable Δs decrease progressively until Δs=0.

5. Rotate the point set to figure 4.6 showed using the following equation:

$$\Delta ux' = \Delta ux * \cos(\text{hdg}_0) - \Delta vy * \sin(\text{hdg}_0) \quad (4.21)$$

$$\Delta vy' = \Delta ux * \cos(\text{hdg}_0) + \Delta vy * \sin(\text{hdg}_0) \quad (4.22)$$

6.transfer inner coordinate point set to inertial coordinate system using the following equation:

$$x = x_s + \cos(hdg) * \Delta u x' - \sin(hdg) * \Delta v y' \quad (4.23)$$

$$y = y_s + \cos(hdg) * \Delta u x' + \sin(hdg) * \Delta v y' \quad (4.24)$$

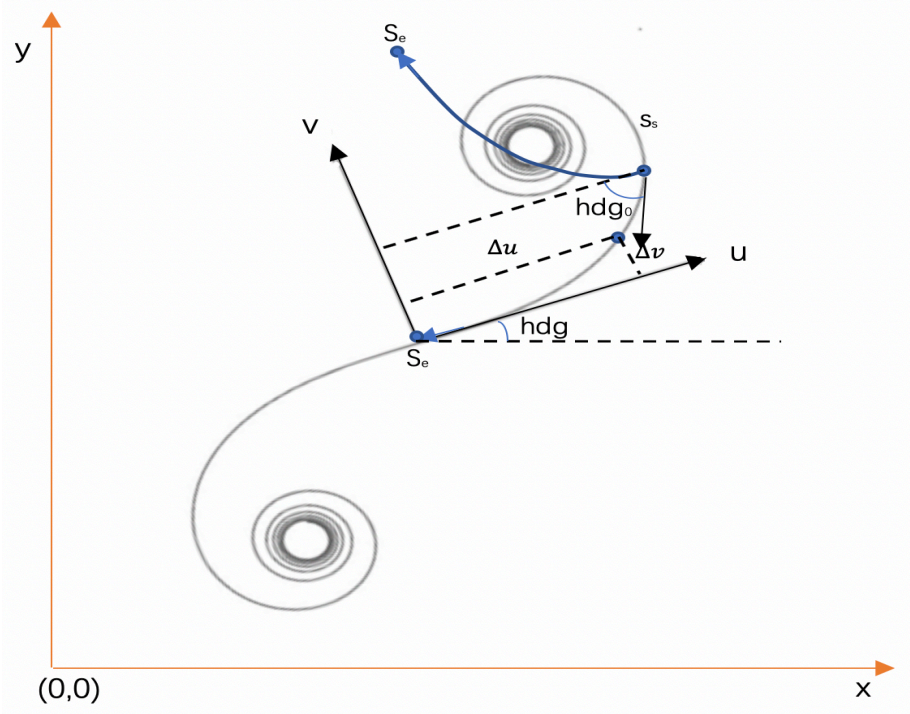


Figure 4. 6: roate geometry spiral

- **Code implementation**

Algorithm: calculate point set of geometry spiral

```
If curvstart = 0
    for l in range(0,spiral.length) step +0.1
        x_sample=odrspiral(l,codt).x
        y_sample=odrspiral(l,codt).y
        x_rotate=x_sample*cos(hdg)-y_sample*sin(hdg)
        x_roate=y_sample*sin(hdg)+y_sample*cos(hdg)
If curvstart != 0
    thdg=length/2*curvstart
    x_end=odrspiral(spiral.length,codt).x
    y_end=odrspiral(spiral.length,codt).y
    for l in range(0,spiral.length) step -0.1
        x_sample=x_end-odrspiral(l,codt).x
        y_sample=y_end-odrspiral(l,codt).y
        x_rotate=x_sample*cos(hdg+thdg)-y_sample*sin(hdg+thdg)
        y_rotate=y_sample*sin(hdg+thdg)+y_sample*cos(hdg+thdg)
returen point set of geometry spiral
```

- **Parameter Poly analysis**

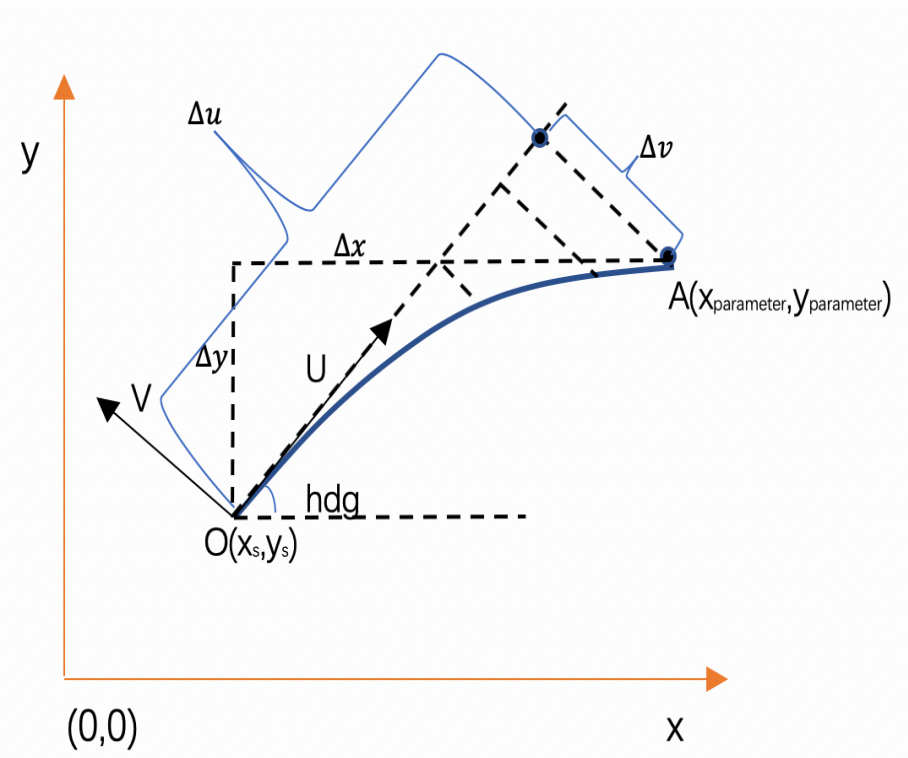


Figure 4. 7: Analysis parameter poly

1.calculate Δu and Δv as the figure 4.7 showed using following equation:

$$\Delta u = aU + bU * p + cU * p * p + dU * p * p * p \quad (4.25)$$

$$\Delta v = aV + bV * p + cV * p * p + dV * p * p * p \quad (4.26)$$

$$p = \Delta s / length_{parameter} \quad (4.27)$$

2.repeat step 1 according to variable Δs increase progressively until $\Delta s = length_{parameter}$

3.transfer inner coordinate system point set to inertial coordinate system using the following equation:

$$x_{parameter} = x_s + \Delta u * \cos(hdg) - \Delta v * \sin(hdg) \quad (4.28)$$

$$y_{parameter} = x_s + \Delta u * \sin(hdg) + \Delta v * \cos(hdg) \quad (4.29)$$

- **Code implementation**

Algorithm: calculate point set of geometry poly
1: for / in range(0,poly.length) step +0.1
2: P<- l/poly.length
3: Δu *<- aU + bU * p + cU * p * p + dU * p * p * p*
4: Δv *<- aV + bV * p + cV * p * p + dV* p * p * p*
5: Return point set of geometry poly

after analysis get a point set, according to the point you can visualization chord line in the Unity3D.

- **Elevation analysis**

After analysis geometry node, we can establish a geometry road in two dimensions, they consist of (*x*,*y*) point set. Elevation node gives the third dimension to let the geometry road become three dimensions (*x*,*y*,*z*).

According to the specification 1.4, there is a formula :

$$ele = a + b * ds + c * ds * ds + d * ds * ds \tag{4.30}$$

ds is the absolute distance along the road section[3].

Meanwhile, each *ds* map a point(*x*,*y*), so we can according to the *ds* to find which point(*x*,*y*) match which *z* (as figure4.8 illustrate).

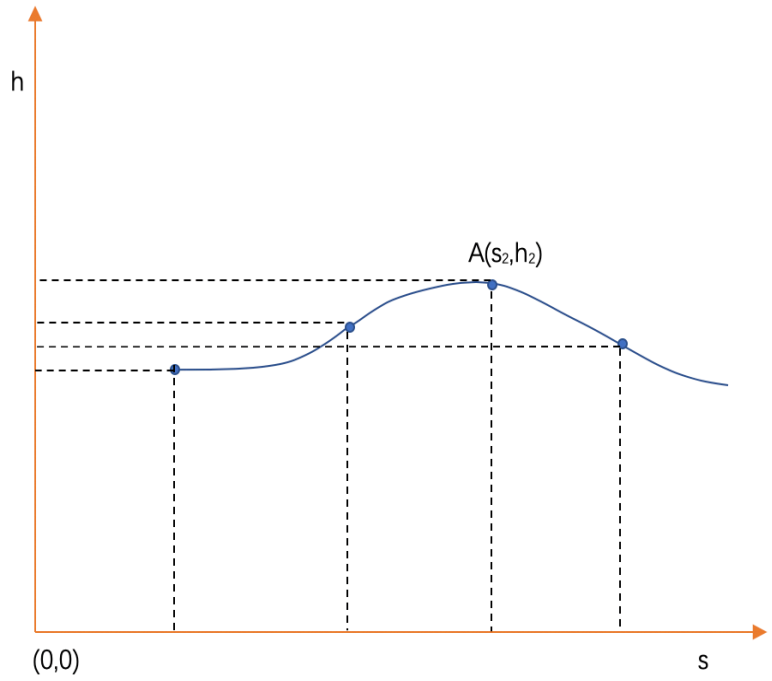


Figure 4. 8: Analysis Elevation

- **Code implementation**

Algorithm: calculate the elevation along the chord line
1: for I in range(0,elevation.number) step+1
2: for j in range(0,elevation.length) step+1
3: z_elevation <- $ele = a + b * ds + c * ds * ds + d * ds * ds$
4: Return z_elevation set

- **Max left right width**

A road is a face, not a line, so we should give each road reference line section a width to establish the road face. The width of each road section is determined by the Lane Width Record. Each road section has several lanesections, each lanesection has several lanes, each lane has several width ndoe.the width of the road section is determined by the sum of width node.

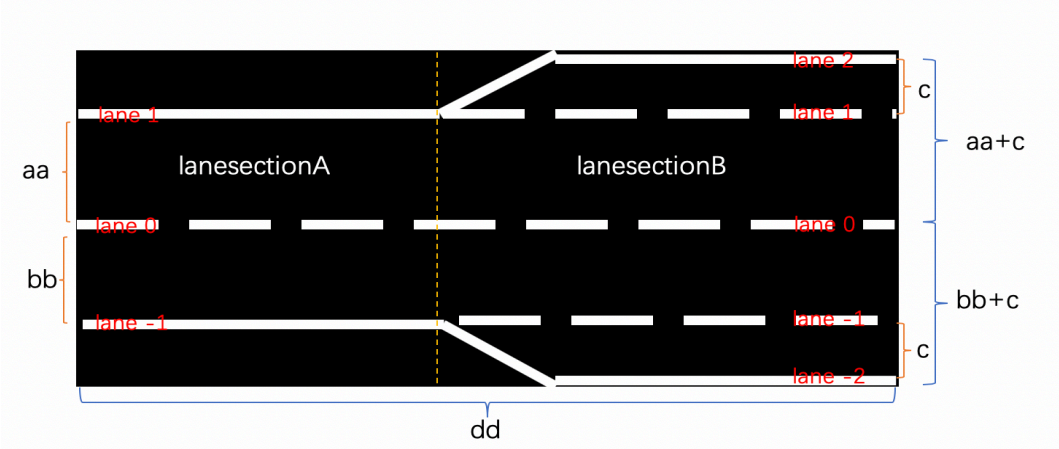


Figure 4. 9: Max Width

As the figure 4.9 showed, though road lanescetionA left width is *aa*, right width is *bb* but lanestionB left width is *aa*+*c*, right width is *bb*+*c*, the road section left width is *aa*+*c*, right width is *bb*+*c*.

- **Code implementation**

Algorithm: calculate the width of the road section
1: for k in range(0,lane.length) step+1 2: for p in range(0,lane[k].length) step+1 3: for m in range(lane[k][p].length) step+1 4: max_left <- sum(lane[k][p][m].left_width) 5: max_right <- sum(lane[k][p][m].right_width) 6: Return max_left and max_right

4.1.1 Establish road mesh in Unity3D

In unity3d, face is based on triangle.

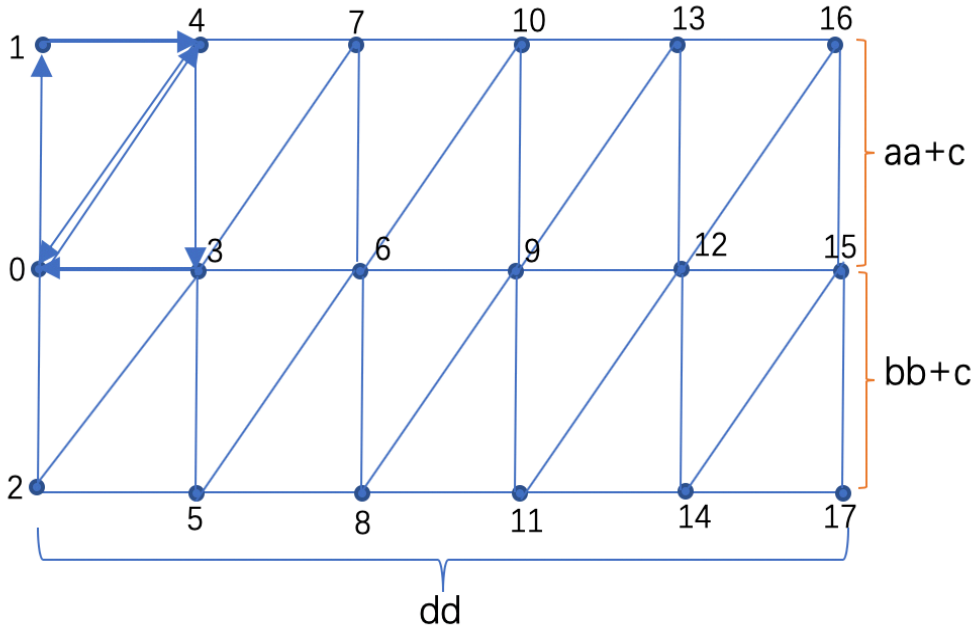


Figure 4. 10: Road Mesh Triangulation case

As figure 4.10 showed, it illustrates the way of build a road mesh in unity3d.

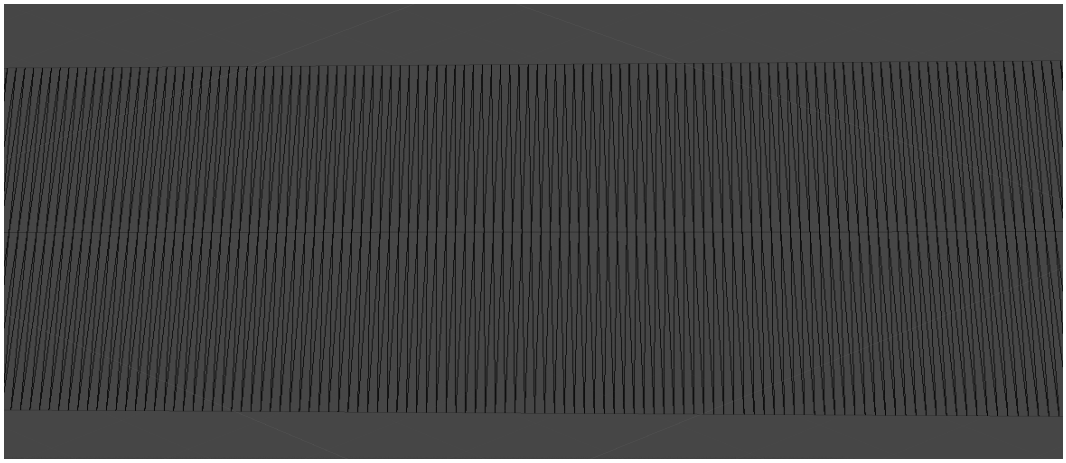


Figure 4. 11: road mesh in unity3d

As figure 4.11 showed, it illustrates the case of road mesh in Unity3D.

- **Signals analysis**

The most import is to put the signal in the right position and right direction.

- **Right position**

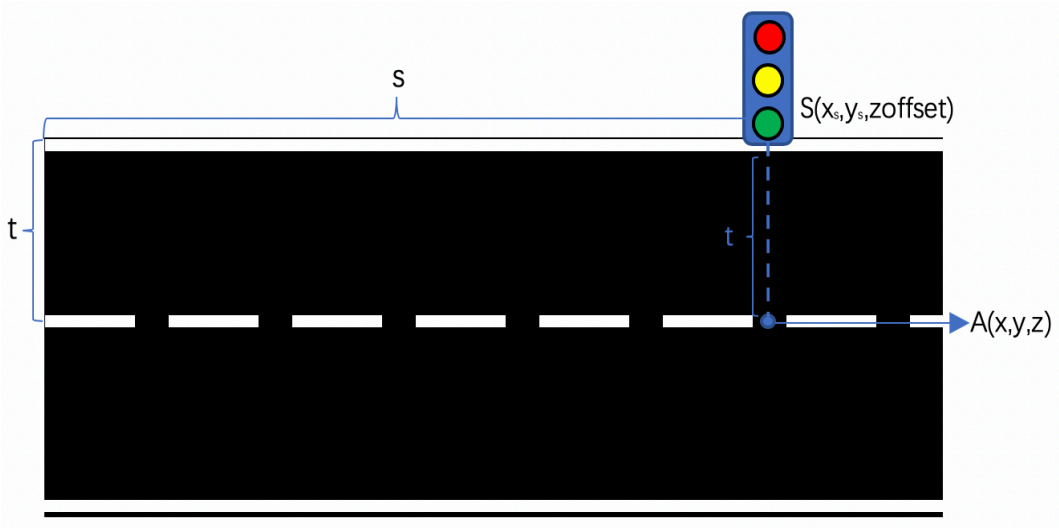


Figure 4. 12: signal position

According to the specification1.5[3], parameter *s* is the absolute position along the road section, we already knew that each *s* match a point(*x,y,z*) which in road reference line in world coordinate system, as figure 4.12 showed *s~A(x,y,z)*. The parameter *t* is the offset in the direction of vertical point A. so the way to find the right position is to illustrate in the following:

1. using parameter *s* to find point *A(x,y,z)*
2. using parameter *t* to calculate the point $s(x_s, y_s, z)$
3. replace *z* in point $s(x_s, y_s, z)$ using parameter zoffset and get new point $s(x_s, y_s, zoffset)$.

- **Right direction**

According to the specification1.5[3], orientation is the "+" = valid in positive track direction "-" = valid in negative track direction "none" = valid in both directions.

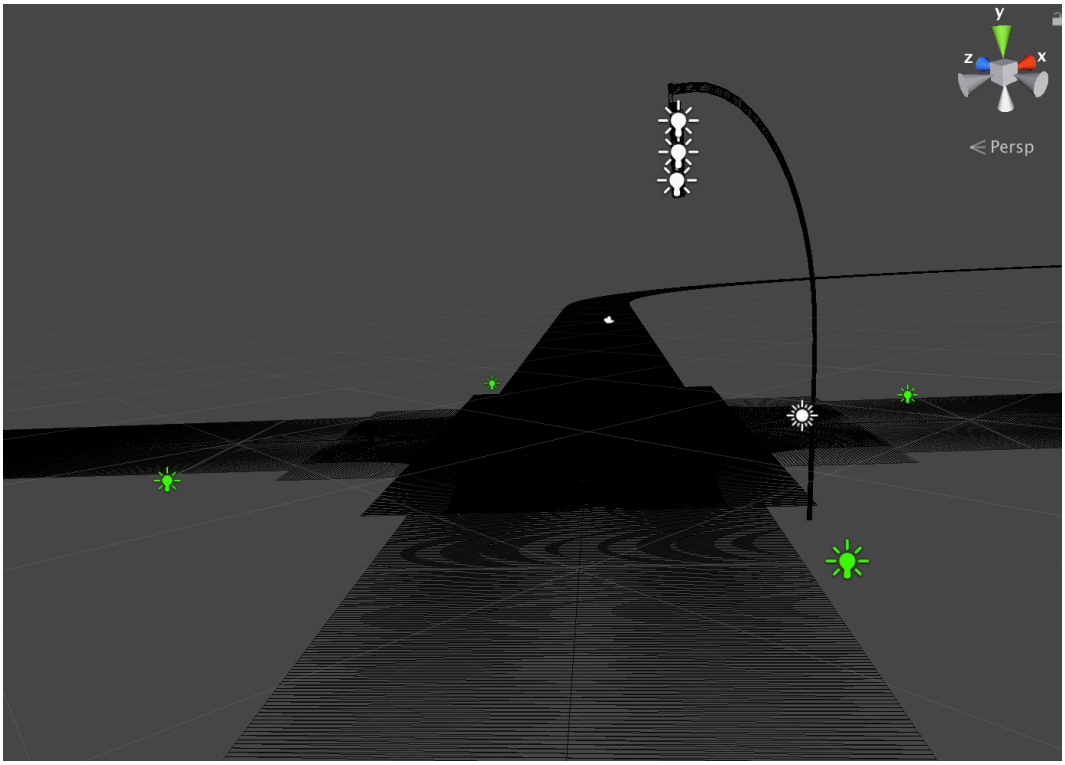


Figure 4. 13: right direction

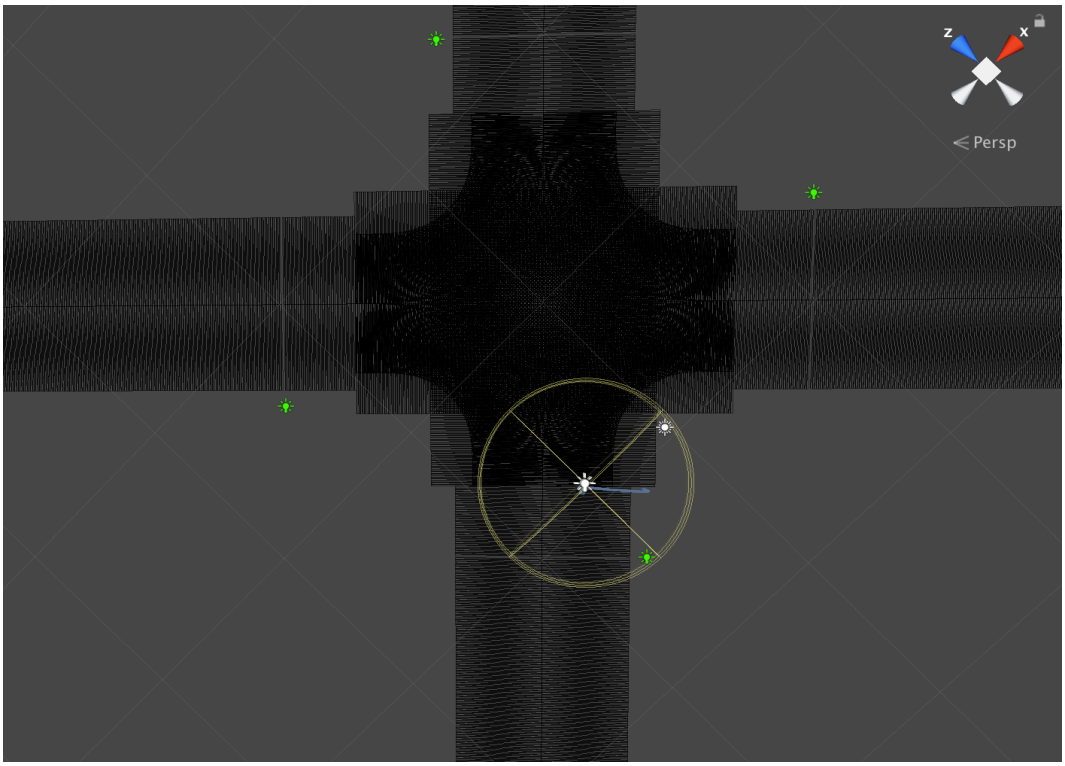


Figure 4. 14: right direction

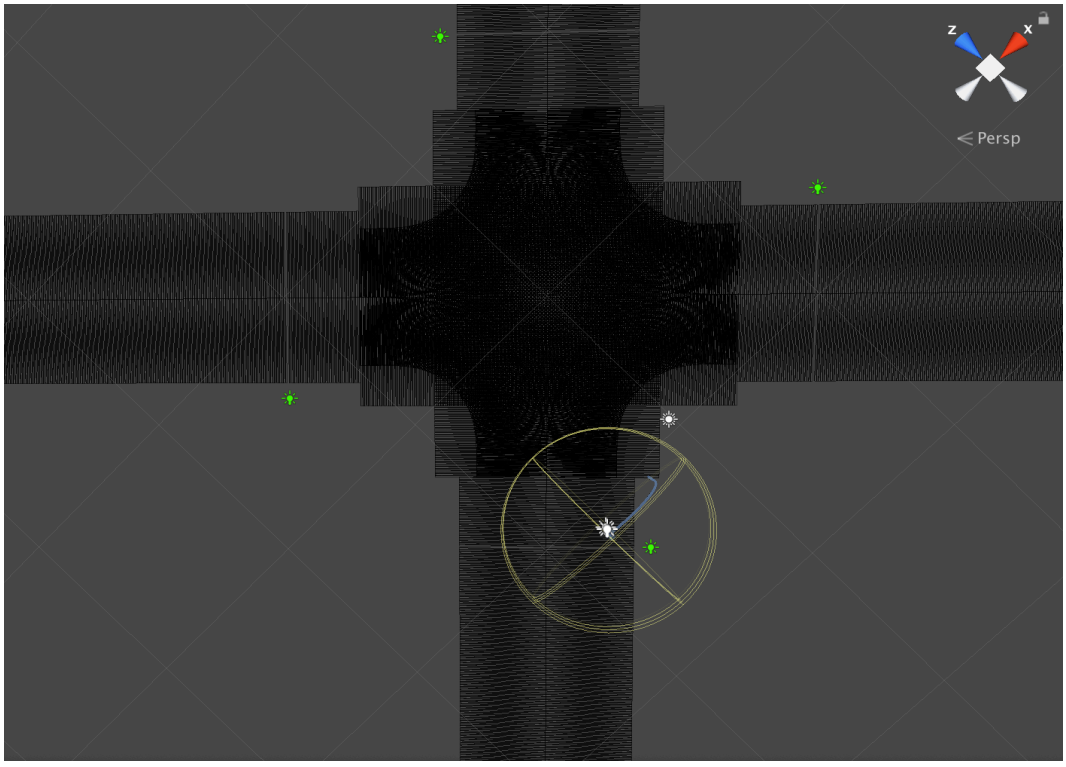


Figure 4. 15: right direction

- **Direction problems**

As figure 4.13 and figure 4.14 showed, after correctly put the signal in the right position, they should rotate the signal in the right direction which the signal vertical the road direction. As figure 4.15 showed, though it is in the right position, it is not in the right direction.

The way to rotate the signal in the right direction is illustrated in the following step:

1. Set the direction of the signal on default as a model in Unity3D.
2. Calculate the direction of the road section which the signal belong to.
3. Rotate the signal model according to the angle which between the signal default direction and the direction of the road.

- **Combine program structure**

Combine each step mentioned before in this chapter, come out with figure 4.16 and figure 4.17 a program flow chart and a Function diagram. According to the program flow chart can establish a road network in unity3d showed in figure 4.18 figure 4.19 and figure 4.20.

But it does not has the lane texture, in the next I will explain how to correctly generate lane texture in Unity3D.

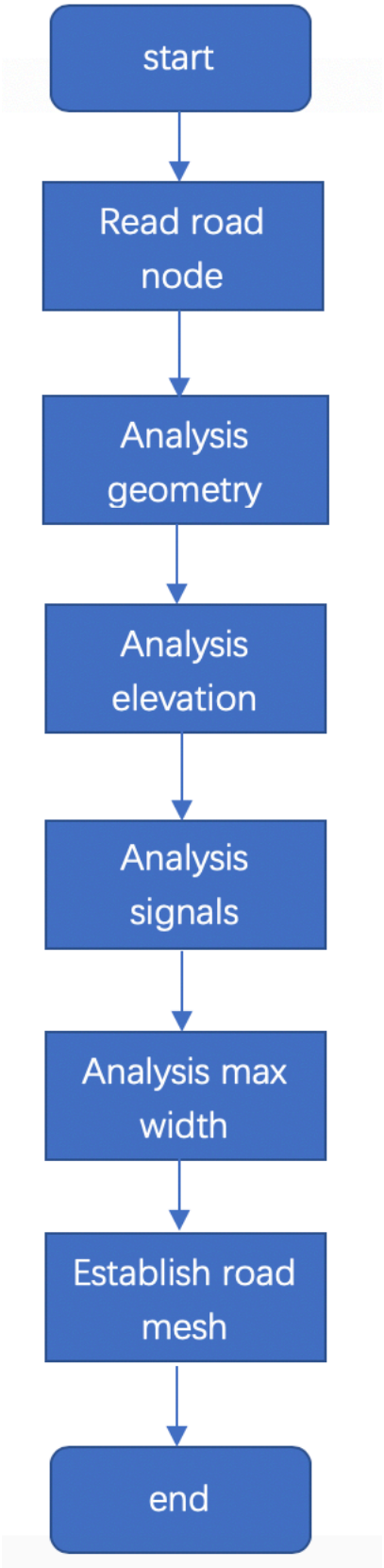


Figure 4. 16: Program flow chart

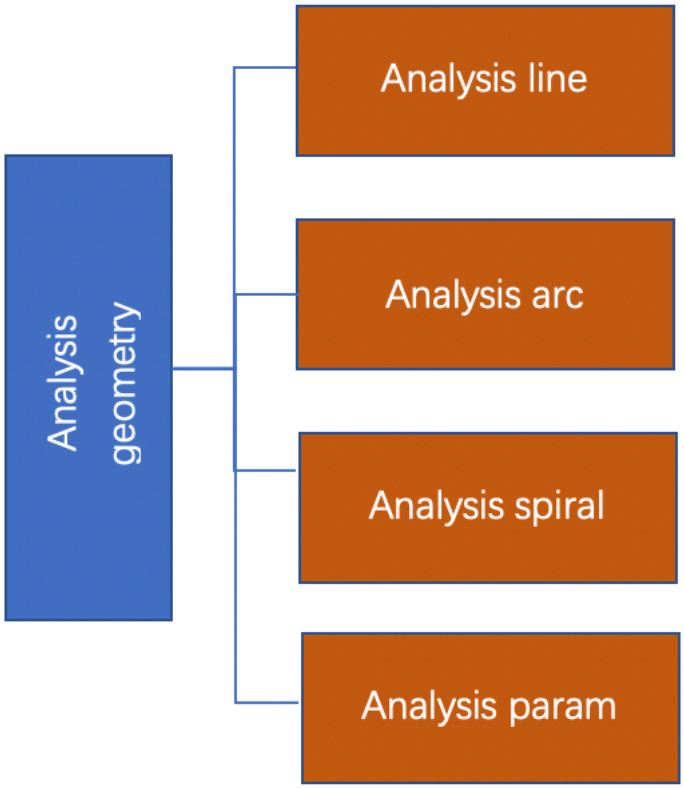


Figure 4. 17: Function diagram

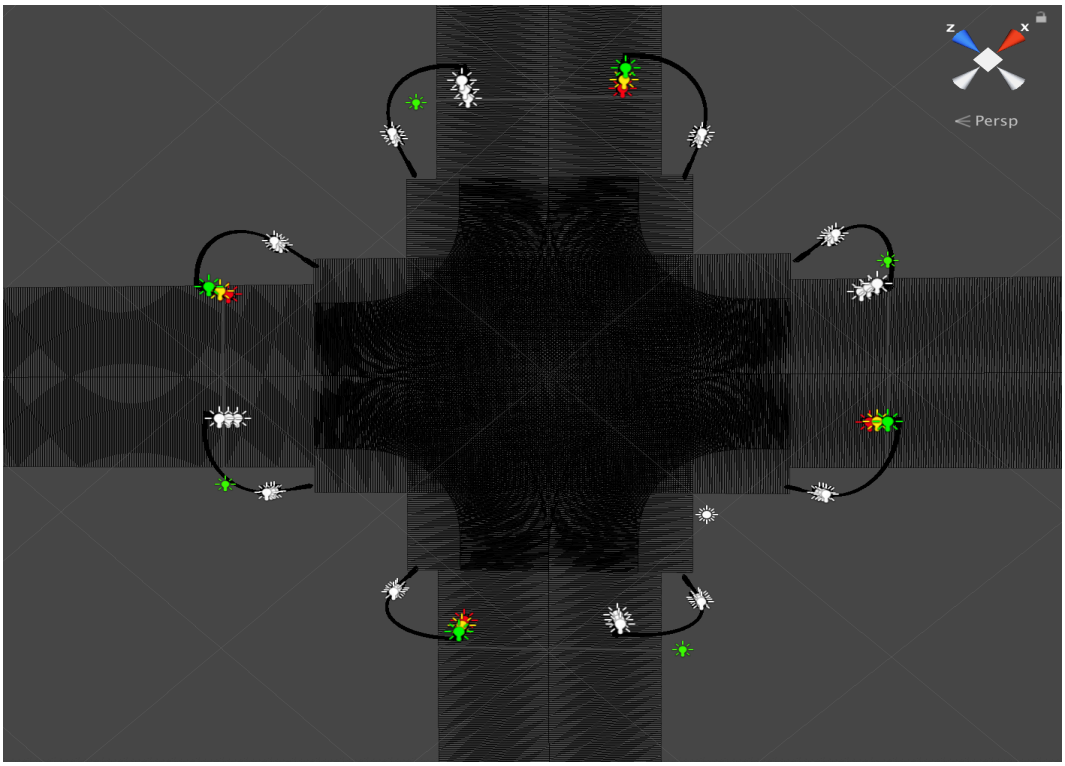


Figure 4. 18: road network 1

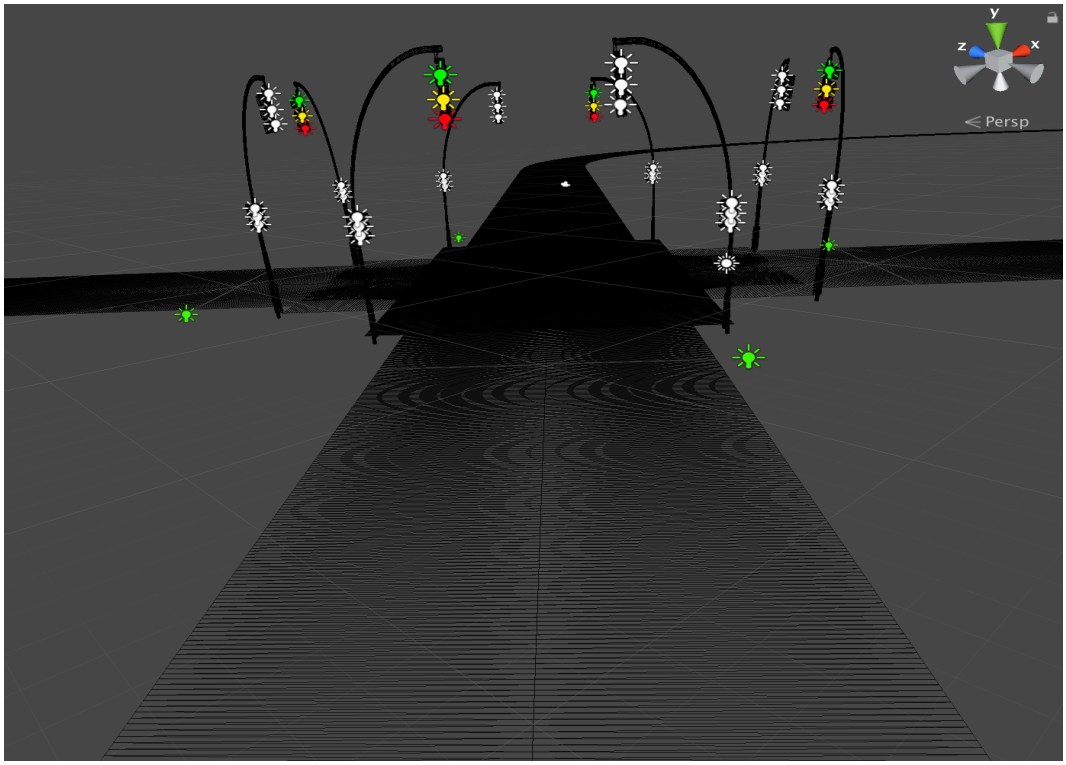


Figure 4. 19: road network 2

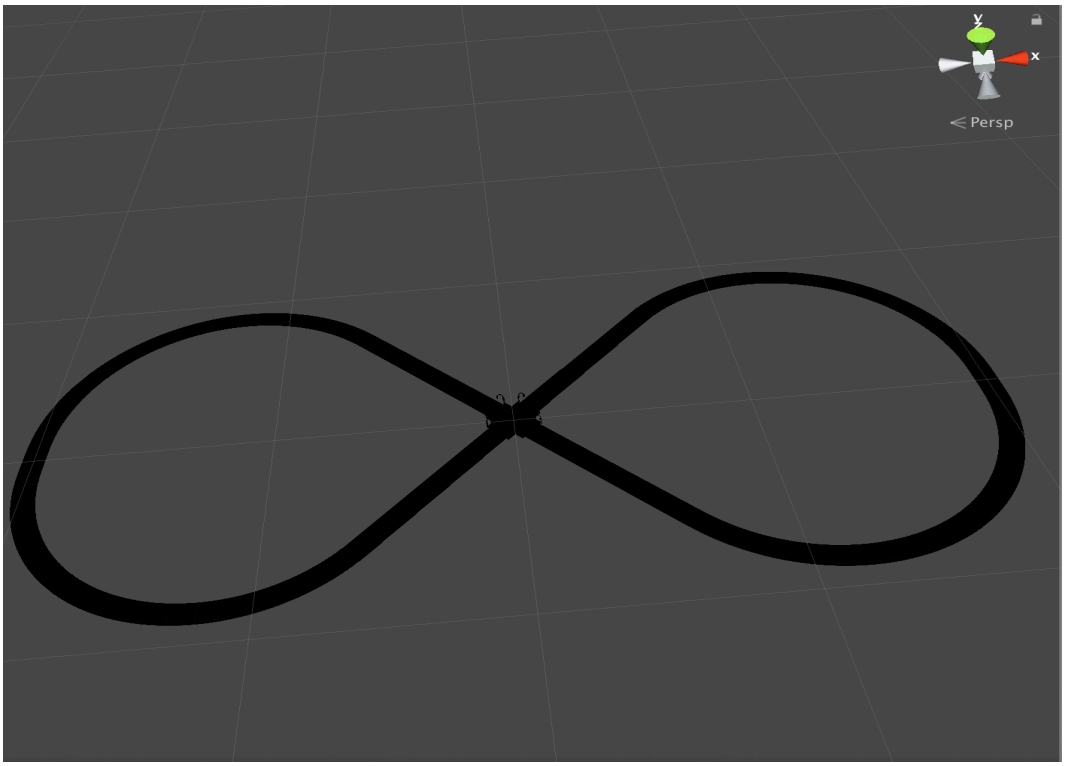


Figure 4. 20: road network 3

Code implementation

Algorithm: establish road network mesh in Unity3D
1: for I in range(0,road.number) step+1 2: create geometry 3: analysis elevation 4: create road mesh 5: Return road network mesh

4.2 Texture generation

After research, I found it is better to dynamic draw lanes on texture and apply the texture on the road section.

4.2.1 Texture introduction

Normally, the mesh geometry of an object only gives a rough approximation of the shape as figure 4.11 showed while most of the fine detail is supplied by Textures. A texture is just a standard bitmap image that is applied over the mesh surface as showed in figure 4.21. You can think of a texture image as though it were printed on a rubber sheet that is stretched and pinned onto the mesh at appropriate positions[10].

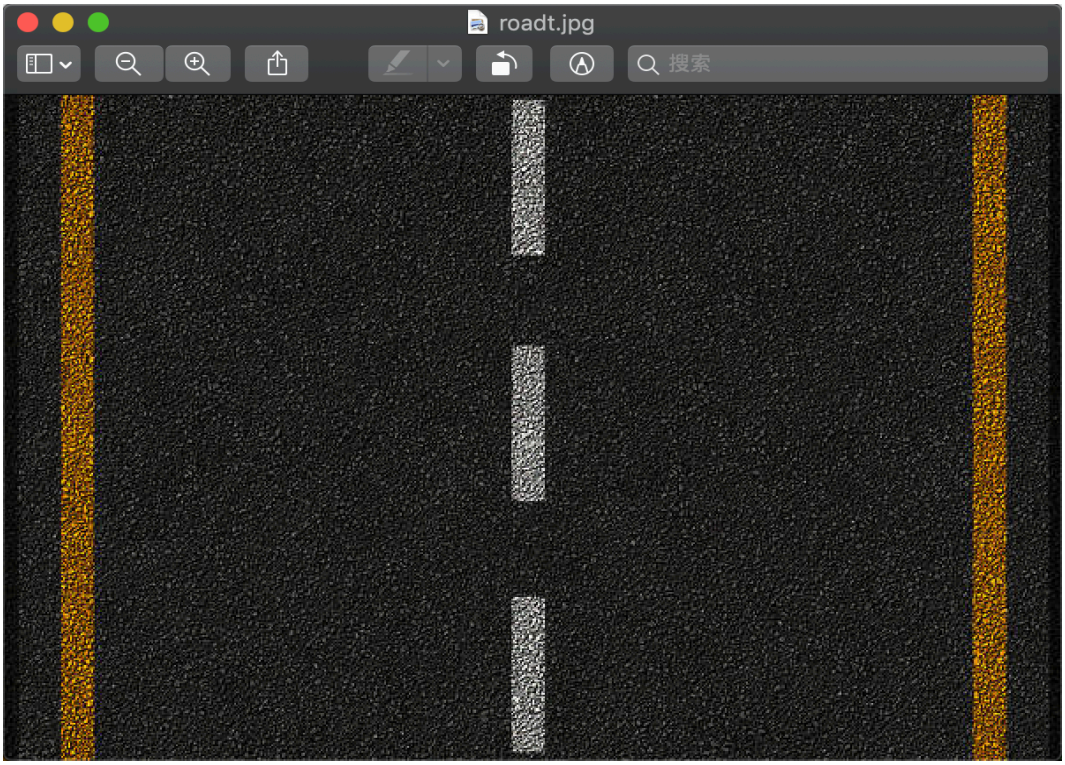


Figure 4. 21: texture

4.2.2 Draw texture

The length of the road section is equal to the width of the texture, and the road max left + road max right=the height of the texture.

After deciding the size of the texture and draw a blank texture, we will draw lanes on the texture.

According to specification 1.5, we know all the lanes are described along the lane reference line. And the lane reference line is described along the road reference line. So before drawing the lanes, we have to decide the road reference line and lane reference line first.

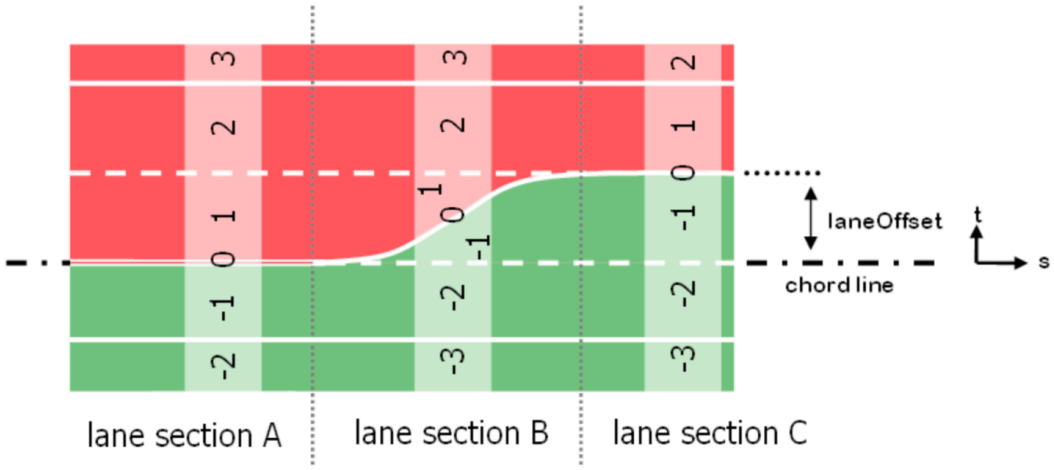


Figure 4. 22: laneOffset[2]

As figure 4.22 showed, lane 0 is lane reference line and it is described by road reference(chord) line and laneOffset. Each lane has a width record means the width of the lane. When the lane is in the red(left) area the id of the lane is positive and in the green(right) area, the id is negative[3]. The outer border of the lane is the track of the width record. So each lane has a width track, and the next width lane track is based on last width lane track. For example, in figure 4.22, lane 1 outer border is the broken white line drawn along the lane 0 according to the lane 1 width record. The lane 2 outer borders soild white line drawn along the lane 1 outer borderline according to lane 2 width record.

Actually, it is complex to draw the lane line which is the white line(broken and soild) in figure 4.22. you have to consider in which area what type of the line is as well as what the track of the line is. I will explain a simple situation to easily understand.

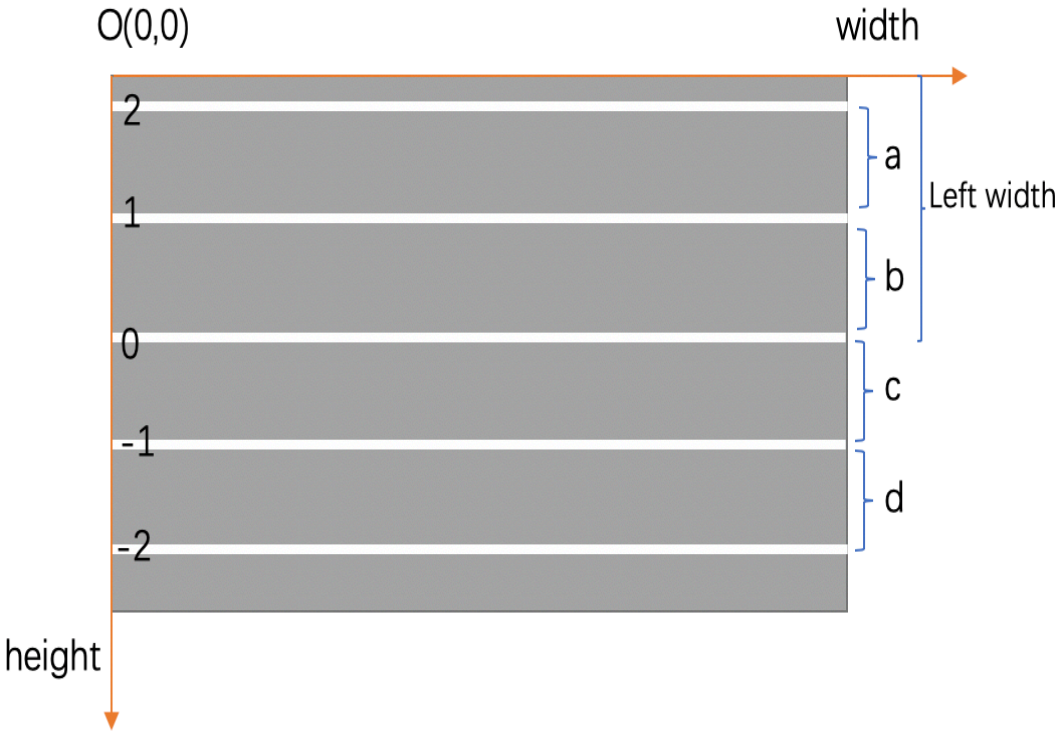


Figure 4. 23: Lane Texture

As figure 4.23 show, it is a road texture with five-lane outer border tracks in the texture coordinate system. The number is the lane id. The variate(a,b,c and d) is the width of each lane calculated by the width record.

Lane 1 border is calculate by following equation:

$$Lane1=Left\ width-b \tag{4.31}$$

Lane 2 border is calculate by following equation:

$$Lane2=Left\ width-b-a \tag{4.32}$$

Lane -1 border is calculate by following equation:

$$Lane-1=Left\ width+c \tag{4.33}$$

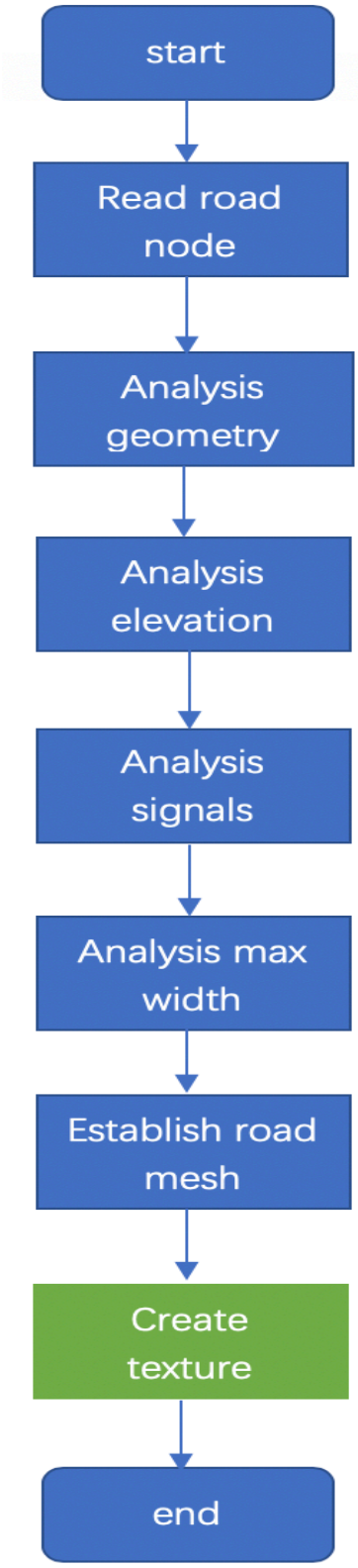


Figure 4. 24: program flow chart

As figure 4.23 showed after add create texture function to the program, can draw the road network as figure 4.24, figure 4.25, and figure 4.26 showed. They have the lanes on the road face.

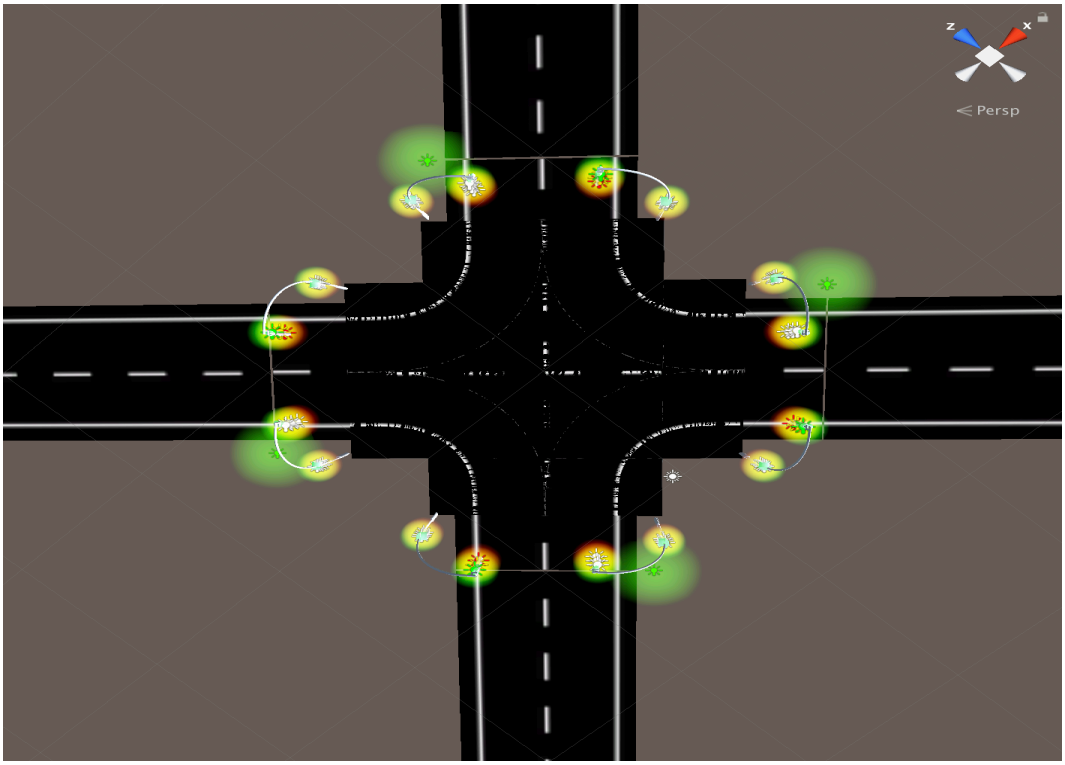


Figure 4. 25 road network with texture 1

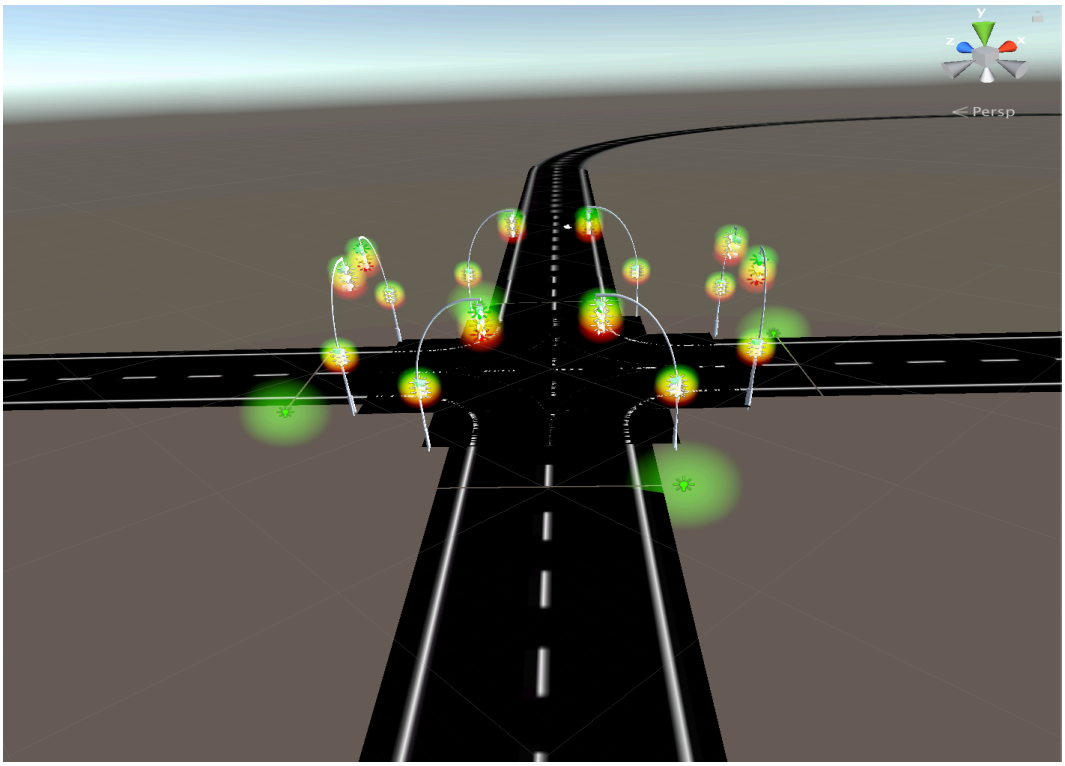


Figure 4. 26 road network with texture 2

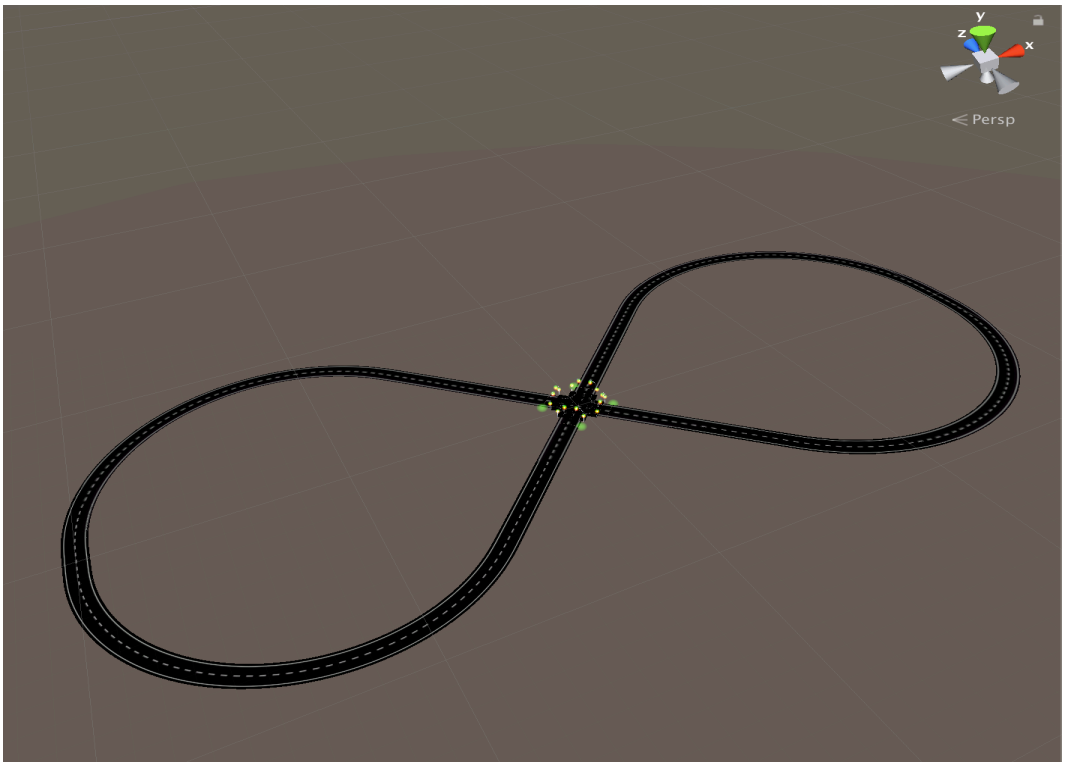


Figure 4. 27 road network with texture 3

Chapter 5

results and discussion

results

After testing all the data from the official website and compare the road network model which established by the program(see in appendix) with the official model[1], it is obvious that the algorithm of the program is correct, including the geometry of the road, lane, signals and the elevation of the road are all correctly created.

Though the program till now can only support limit record nodes in OpenDrive file which is enough to support driving simulation, the thesis already establish a frame and algorithm for the program which can easy add new node in the program to make the road network more real.

.

discussion

5.1 attribute order problem

During read the OpenDRIVE data format into the predefined data structure, I found it is complex to read the data when different OpenDRIVE file has a different order of the attributes.

For example

```
<roadMark sOffset="0.000000000000000e+00" type="solid" weight="standard" color="standard" width="1.300000000000000e-01"/>
```

Figure 5. 1 roadmark 1

```
<roadMark sOffset="0" color="standard" width="0.22988857012664093" weight="bold" type="solid">
```

Figure 5. 2 roadmark 2

When read the attribute type value, in the different file, have to first decide the index of the attribute type then use the index to read the value.

But if they have the same order, if the type attribute always in the second in the roadMark node. Can quickly use roadmark to read the value of the attribute type.

5.2 Texture problem

In the OpenDRIVE the lane coordinate is a double format, but the program use the texture to draw the lane, the coordinate of texture is integer (pixel) format. So there is a small mistake during transfer double to integer.

5.3 Max width improvement

During decide the road width, use max left width and max right width.

The main goal is to decrease the code complex. But it makes the road network not looks good. As the figure 5.3 showed.

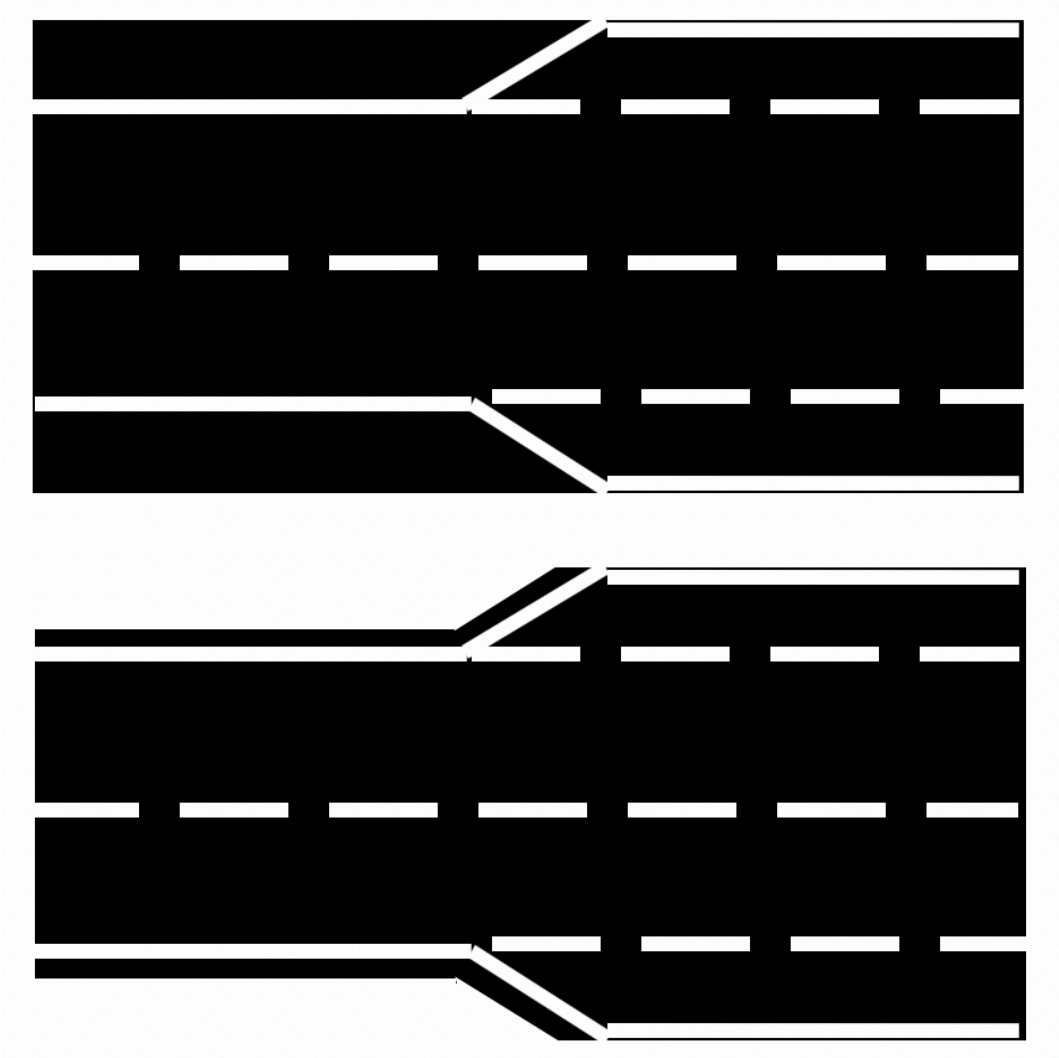


Figure 5. 3 dynamic width

Obviously, the second figure looks better. The next work is using dynamic width, not static width.

Bibliography

[1] "OpenDRIVE - Home", *Opendrive.org*, 2019. [Online]. Available: http://www.opendrive.org/index.html. [Accessed: 11- Apr- 2019].

[2] A. Barsi, V. Poto and V. Tihanyi, "Creating OpenCRG Road Surface Model from Terrestrial Laser Scanning Data for Autonomous Vehicles", 2019. .

[3] *Opendrive.org*, 2019. [Online]. Available: http://opendrive.org/docs/OpenDRIVEFormatSpecRev1.5H.pdf. [Accessed: 11- Apr- 2019].

[4] "opendrive roads | Add OpenDRIVE Roads to Driving Scenario - MATLAB &am", *Keywordniches.com*, 2019. [Online]. Available: http://www.keywordniches.com/search/opendrive-roads. [Accessed: 11- Apr- 2019].

[5] I. Zamojc, "Introduction to Unity3D", *Code Envato Tuts+*, 2019. [Online]. Available: https://code.tutsplus.com/tutorials/introduction-to-unity3d--mobile-10752. [Accessed: 11- Apr- 2019].

[6] "Unity (game engine)", *En.wikipedia.org*, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Unity_(game_engine). [Accessed: 11- Apr- 2019].

[7] 2019. [Online]. Available: https://www.merriam-webster.com/dictionary/simulator. [Accessed: 11- Apr- 2019].

[8] "Learn C# in 7 days", *O'Reilly | Safari*, 2019. [Online]. Available: https://www.oreilly.com/library/view/learn-c-in/9781787287044/0b58f137-5542-4066-a516-00d18b8dcea7.xhtml. [Accessed: 11- Apr- 2019].

[9] 2019. [Online]. Available: http://all-searches.com/recherche/Unity::2d::Guide/web/3. [Accessed: 11- Apr- 2019].

[10] U. Technologies, "Unity - Manual: Textures", *Docs.unity3d.com*, 2019. [Online]. Available: https://docs.unity3d.com/Manual/Textures.html. [Accessed: 11- Apr- 2019].

Appendix

The following figures are created by the program according the data file published on OpenDRIVE offical website and Mr ###.

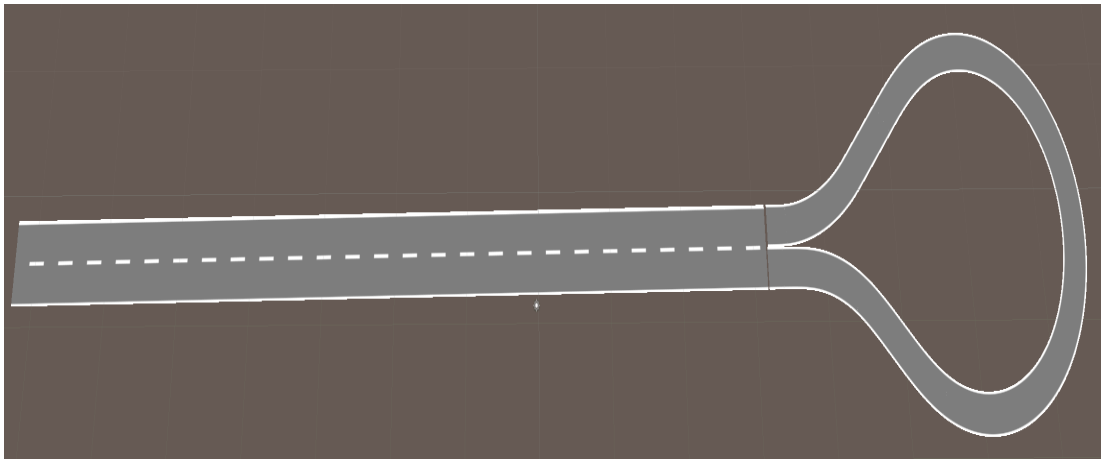


Figure A.1 sample of dead-end road

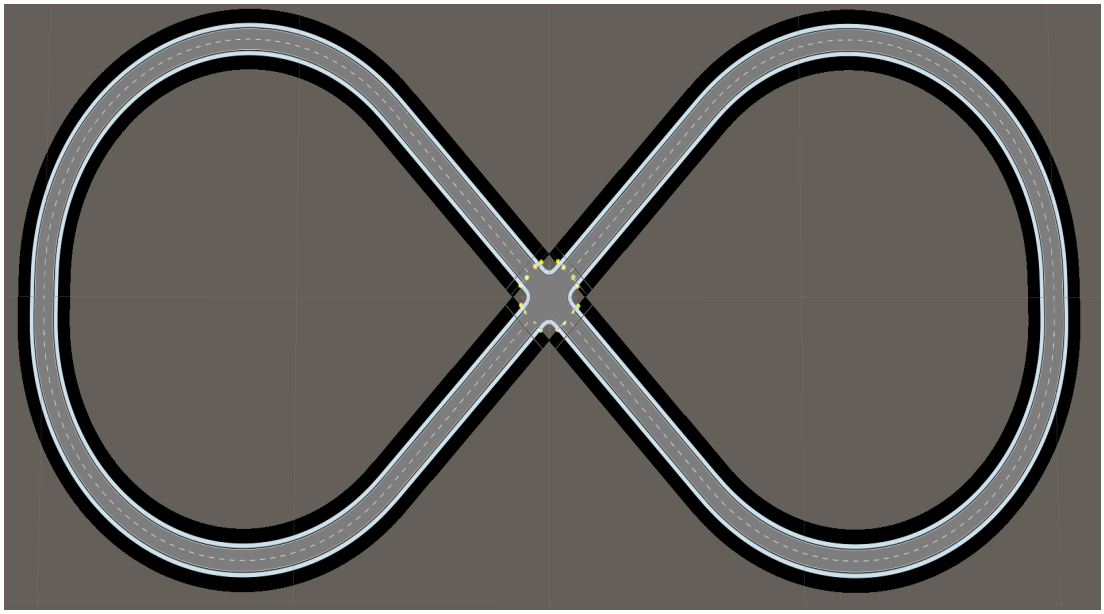


Figure A.2 sample of a standard crossing with traffic lights

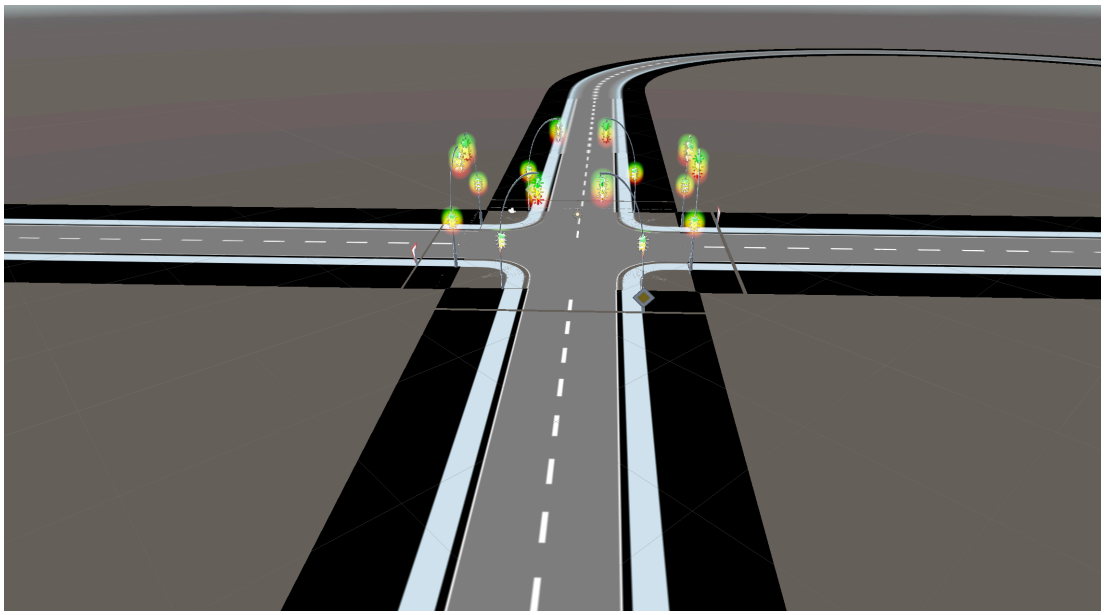


Figure A.3 sample of a standard crossing with traffic lights

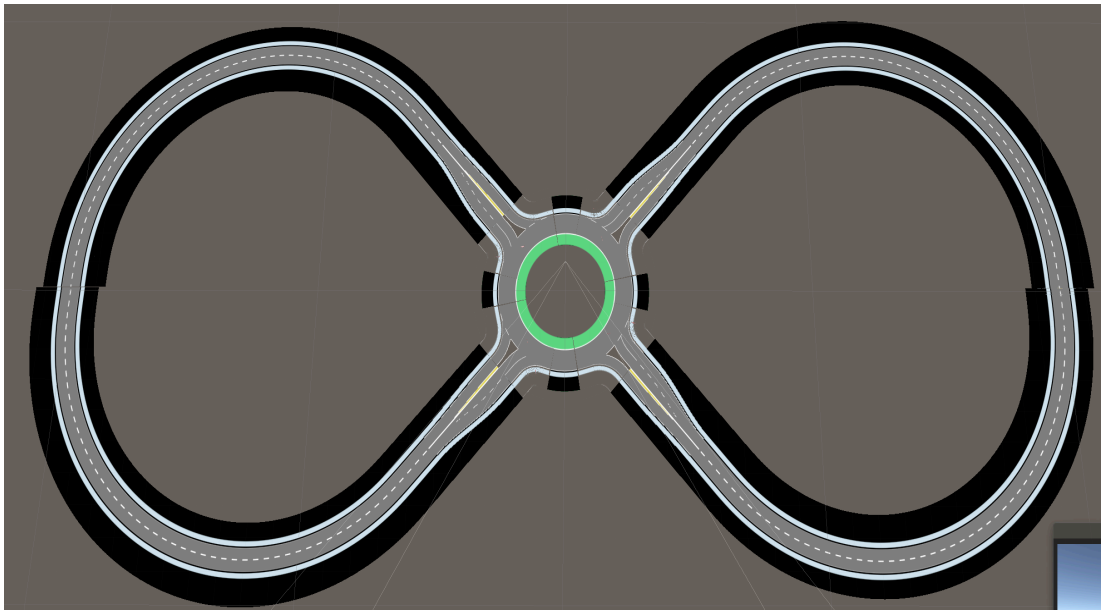


Figure A.4 sample of a roundabout

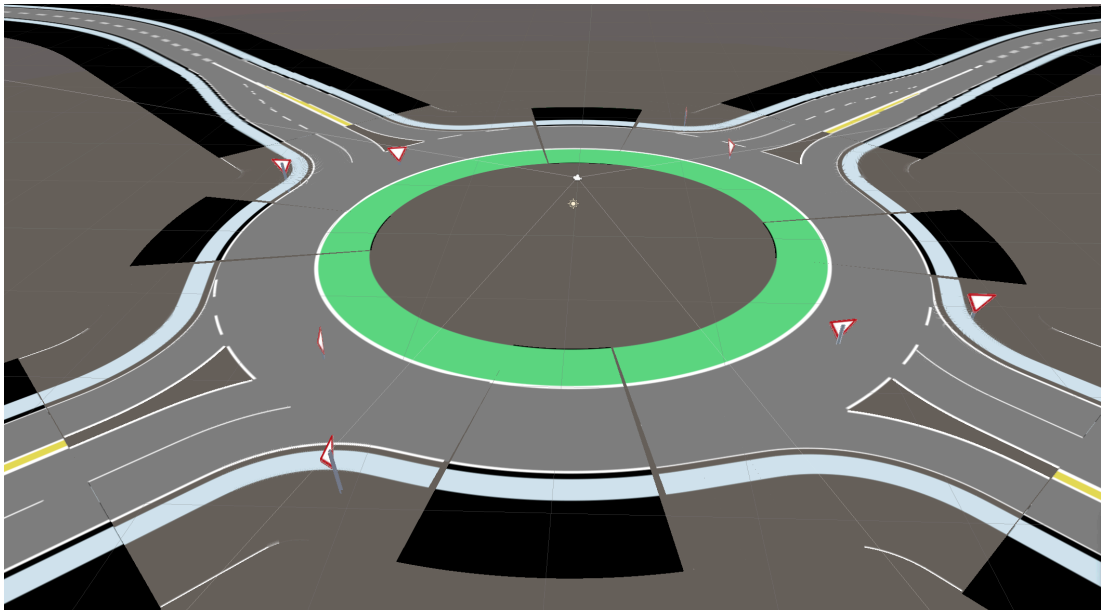


Figure A.5 sample of a roundabout

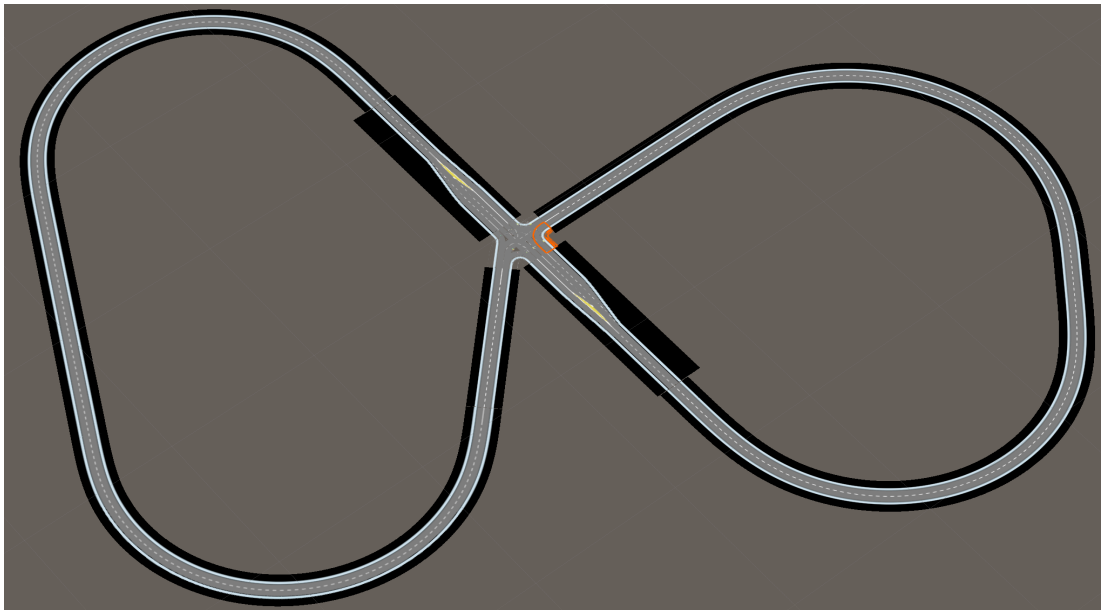


Figure A.6 sample of a complex crossing with traffic lights

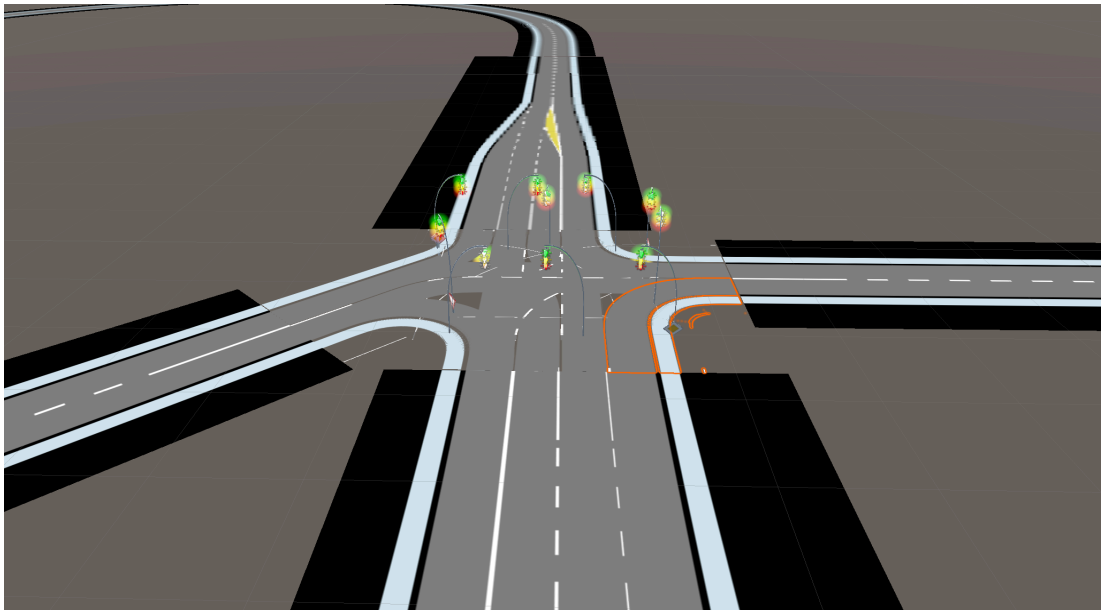


Figure A.7 sample of a complex crossing with traffic lights

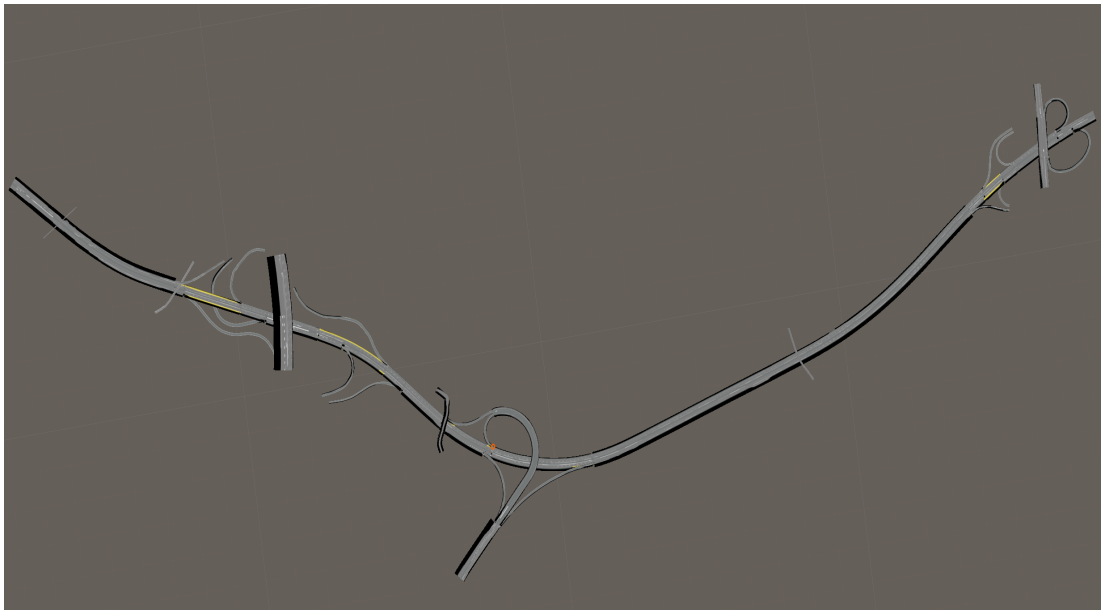


Figure A.8 sample by Atlatec

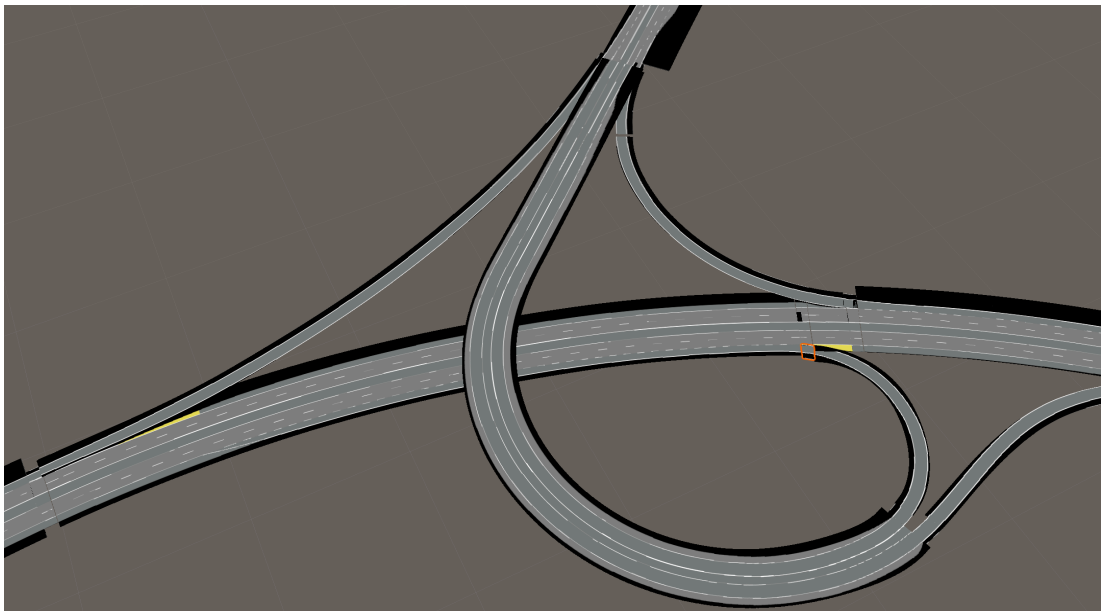


Figure A.9 sample by Atlatec

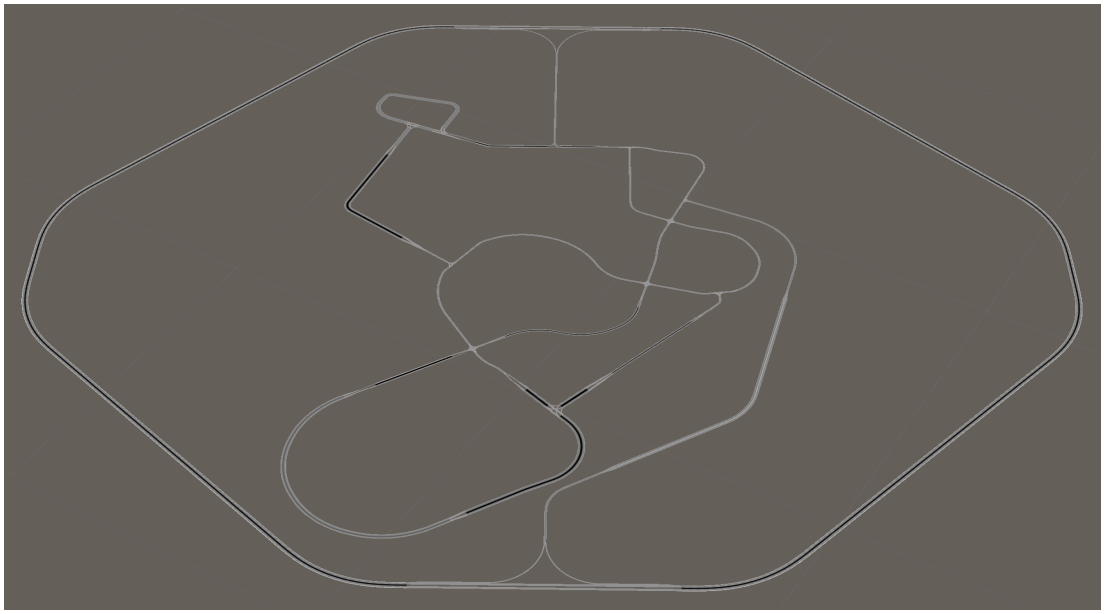


Figure A.10 sample by

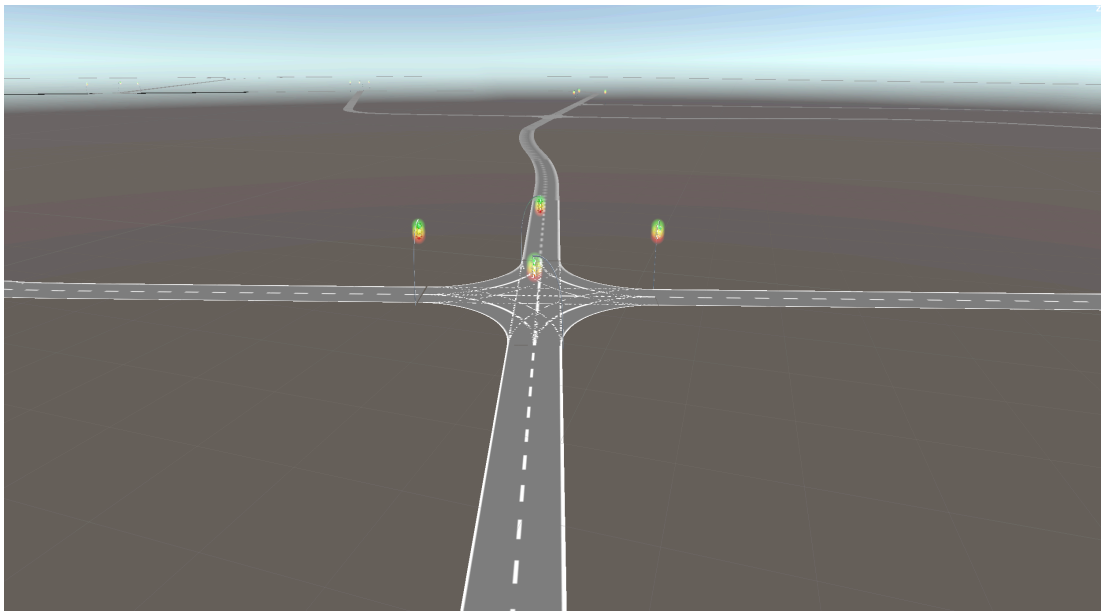


Figure A.11 sample by