JunctionArt

Release 1.0.0

Golam Md Muktadir, Abdul Jawad, Augmented Design Labs

Jul 31, 2023

CONTENTS:

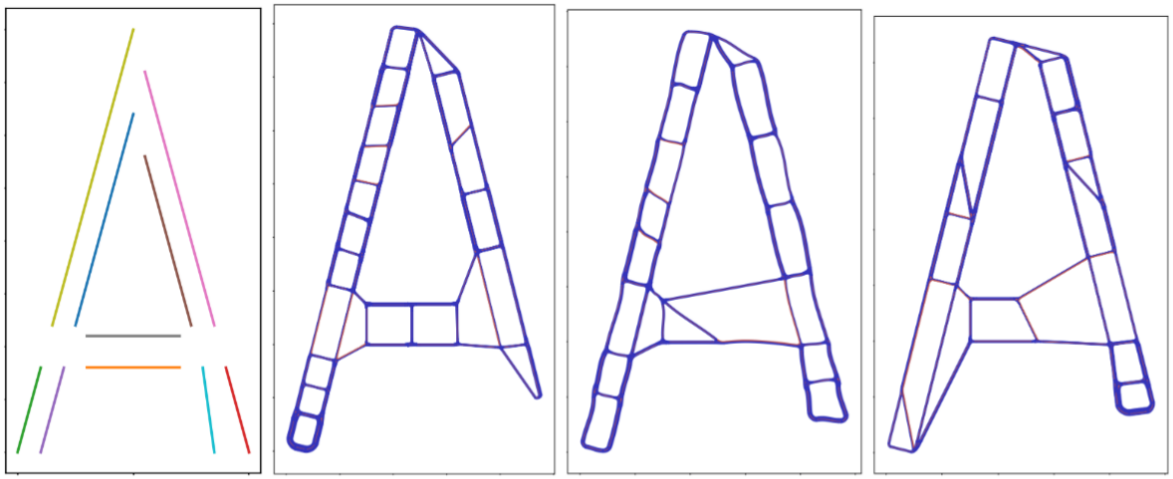
This documentation website is still under construction. Please post an issue on our github repository if you cannot install it or use it correctly. https://github.com/AugmentedDesignLab/junction-art

With JunctionArt, you can create complete city maps, interesting intersections and roundabouts. Here are some examples:

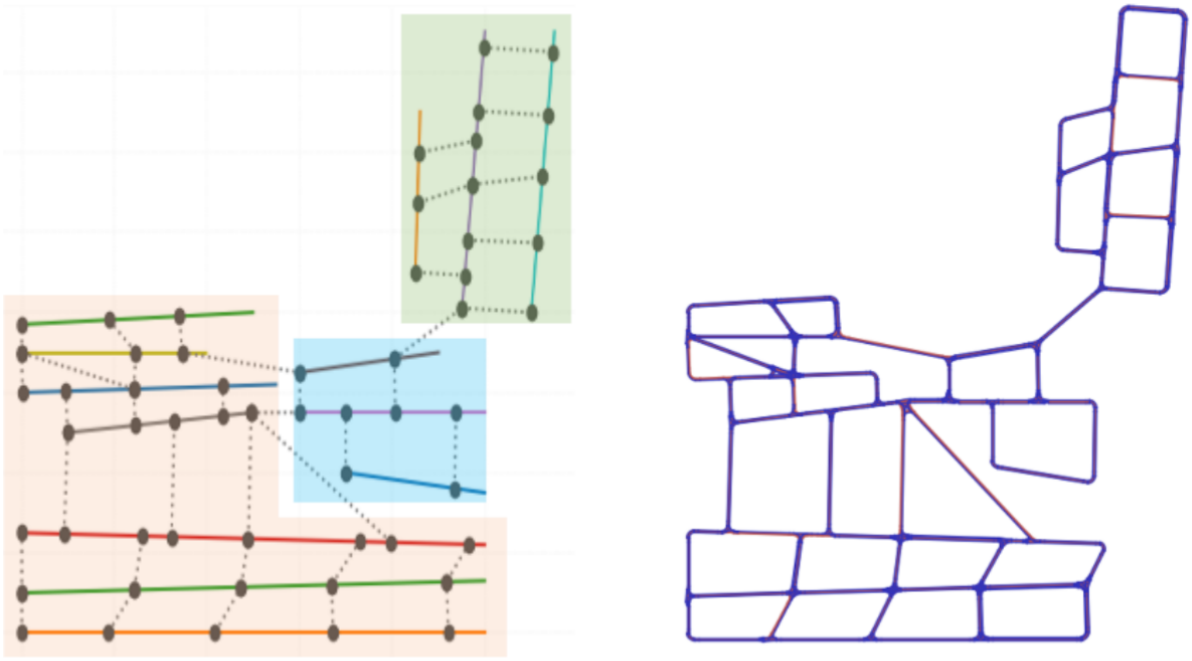
CHAPTER

ONE

1. ROAD GENERATOR



(a) (b) (c) (d)



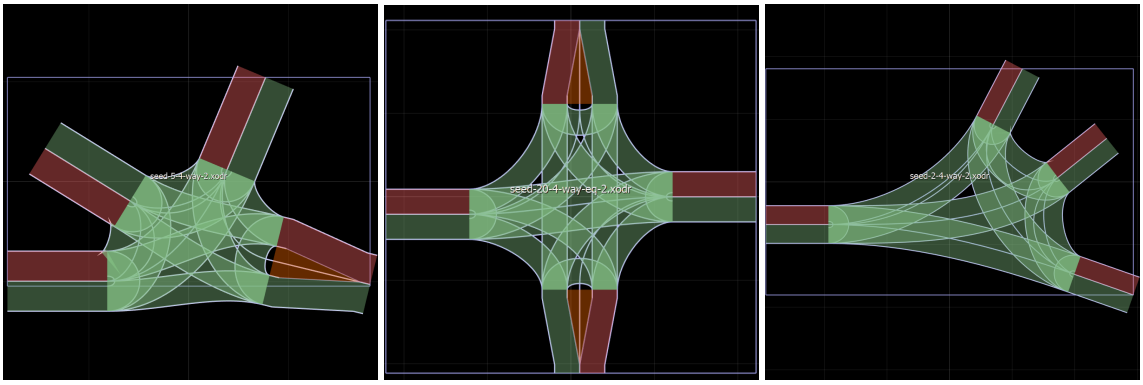
(a) (b)

CHAPTER

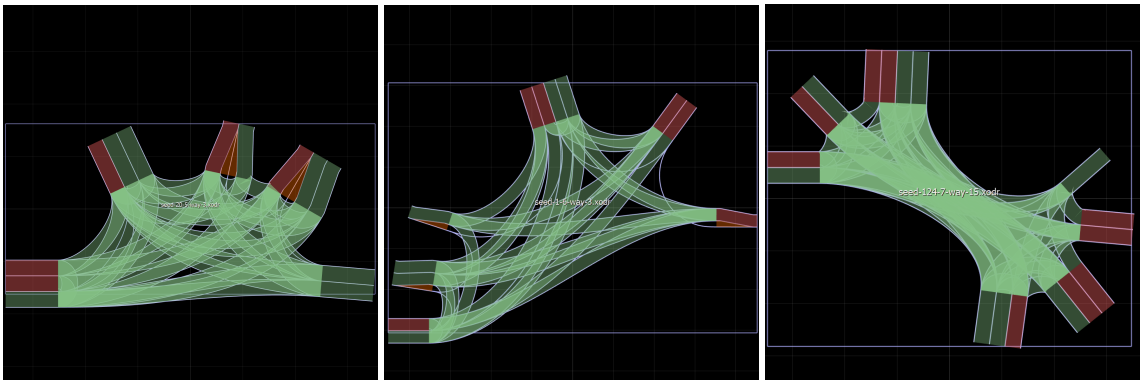
TWO

2. INTERSECTION GENERATOR

Four-way intersections:



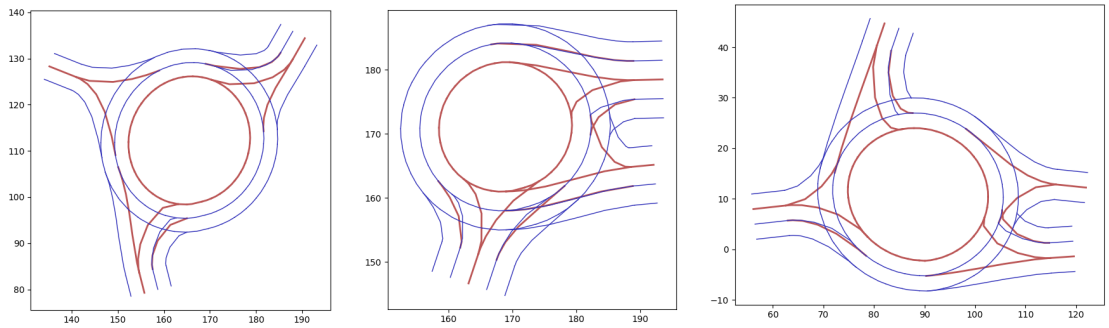
Varying way intersections:



CHAPTER

THREE

3. ROUNDABOUT GENERATOR



3.1 Installation

You can use JunctionArt as an intersection and road generator, or you can use it as a library to build features upon. Both needs the package to be installed. You can install it from *the source* or *the pypi package*. To view the roads, you need a tool like esmini or Mathworks Roadrunner. Many of our test codes are integrated with esmini odrplot.

Python version: Works in python 3.7.9. There are some internal python library issues with 3.7.11.

There are some libraries that may face permission errors. In such cases, run your terminal in administrator mode

3.1.1 To use our generatiors in your project

1. Install via PIP

```
pip install junctionart
```

1. (Optional) Install https://trac.osgeo.org/osgeo4w/ for analysis tools
2. Read the *User Manual*

3.1.2 Extending JunctionArt

Extending JunctionArt requires installing the source and installing poetry to publish in the PyPi repository.

Installing from source

1. Clone the repository from *https://github.com/AugmentedDesignLab/junction-art*
2. Install poetry and conda https://python-poetry.org/docs/ https://www.anaconda.com/
3. Go to the root folder of the junction-art. Create a new virtual environment with conda.

Run these commands in order

```
poetry config virtualenvs.create false --local
conda env update -f requirements.yml --prune
pip install --upgrade pip
conda install -c conda-forge shapely
conda install -c conda-forge matplotlib
conda install -c conda-forge tabulate
conda install -c conda-forge psutil
conda install -c conda-forge scikit-spatial
poetry install
```

OR

```
poetry config virtualenvs.create false --local
pip install -r requirements-pip.txt
pip install --upgrade pip
conda install -c conda-forge shapely
conda install -c conda-forge matplotlib
conda install -c conda-forge tabulate
conda install -c conda-forge psutil
conda install -c conda-forge scikit-spatial
poetry install
```

If you are going to use the analysis tools, you need to make sure libgeos_c is installed (the conda install -c conda-forge shapely command should install it by default.)

Option 1: Install via conda (recommended)

1. Activate or create a conda environment for the project with python 3.7+
2. Put the name of your environment in the first line of requirements.yml file (located in the root).

```
name: your_env_name
```

and run:

```
$ conda env update -f requirements.yml --prune
```

–prune

Remove **–prune** if you need the existing python packages installed in your environment.

Option 2: Install via pip

```
$ pip install -r requirements-pip.txt
```

3.2 User Manual

JunctionArt produces HD roadmaps and intersections.

3.2.1 Create intersection from incident points

In order to create an intersections using JunctionArt, there are some functionalities implemented in the **JunctionBuilderFromPointsAndHeading** class. Initialize the object **JunctionBuilderFromPointsAndHeading** using **CountryCode** and **laneWidth** for all the roads. Default country is US and the default width is three.

First import the following modules.

```
import pyodrx
from junctionart.extensions.CountryCodes import CountryCodes
from library.Configuration import Configuration
from junctionart.junctions.JunctionBuilderFromPointsAndHeading import␣
↪JunctionBuilderFromPointsAndHeading
import junctionart.extensions, os
```

Create the **JunctionBuilderFromPointsAndHeading** object. A;so create the **configuration** object to fetch the directory of esmini **odr_viewer**.

```
builder = JunctionBuilderFromPointsAndHeading(country=CountryCodes.US, laneWidth=3)
configuration = Configuration()
```

Specificly, function **createIntersectionFromPointsWithRoadDefinition** takes in odr ID, road definition (described below), road ID for the first road, length of the defined straight roads,and the format of the output. If getAsOdr is false, the function returns an intersection object.

```
odr = builder.createIntersectionFromPointsWithRoadDefinition(odrID=0, firstRoadId=100,␣
↪roadDefinition=roadDefinition, straightRoadLen=40, getAsOdr=True)
```

1. **odrID** = unique ID for the resultant OpenDRIVE file
2. **firstRoadId** = ID of the first road, ID of the rest of the roads are greater than **firstRoadId**
3. **roadDefinition** = List of dictionaries for defining the incident roads
4. **straightRoadLen** = length of the defined straight roads. All the incident roads have the same length.
5. **getAsOdr** = a boolean to define othe output format. If **True** the output will an xodr file, otherwise the function returns an **Intersection** object.

Road Definition

Road definition is a list of dictionary to describe the incident roads of an intersection formatted as below:

```
roadDefinition = [

            {'x': -30, 'y': 30, 'heading': 2, 'leftLane': 2, 'rightLane': 2, 'medianType
↪': 'partial', 'skipEndpoint': pyodrx.ContactPoint.start},
            {'x':   0, 'y': 30, 'heading': 1,  'leftLane': 2, 'rightLane': 3, 'medianType
↪': None, 'skipEndpoint': None},
            {'x':   0, 'y':  0, 'heading': -1.5, 'leftLane': 1, 'rightLane': 1,
↪'medianType': 'partial', 'skipEndpoint': pyodrx.ContactPoint.end},
        ]
```

1. **x, y** is the cartesian coordinate of the point where the straight road is connected with the intersection. **heading** is the direction of the road outwards to the intersection. **heading** needs to be defined in **radians**.
2. **leftLane** and **rightLane** define the number of lanes of the incident roads.
3. **medianType** can takes the value **None/partial**. This parameter adds an island at the end of the road. Users can skip one end point by using the parameter **skipEndpoint** (possible values are **pyodrx.ContactPoint.start** and **pyodrx.ContactPoint.end**).
4. **Roads needs to be defined in clock-wise order**

To view the intersection generated by using the above-mentioned road definition use the following statement:

```
extensions.view_road(odr, os.path.join('..',configuration.get("esminipath")))
```

3.3 Extending JunctionArt

This guide is for developers who wants to add new features in JunctionArt or build their new types of road artifacts and generators.

3.3.1 Preparing the development environment

- Install from source *Installing from source*
- Setup esmini build 1067. Newer versions do not work well with JunctionArt.

3.3.2 Essential Concepts

Keypoints of Junction Architecture

1. The successor or predecessor of an incident road is a junction object, not a road object.
2. For each incident road, we are creating a set of connection roads from its incoming lanes. So, in the juncttion description, these connection roads need to be included. The data we need here are: laneConfigurations, contact points between the incoming road and connection road, lane links of the incoming road and the connection road.
3. Now, we need a builder, that can take all this input and build the junction.

Steps for the builder:

1. take junction id, outside roads and connection road as inputs
2. For each connection road, if it's successor or predecessor is an incident road: a. create a junction object a. set incident roads junction id and type b. add the link to the junction object

Strategies for lane configurations

Two roads with different lane configurations can be connected by a connection road. There are may ways this connection road can be created. Connection roads can connect all the lanes or some of them.

Basic steps:

1. Create all the roads
2. Drop all the lanes from the connection roads and rebuild the lanes.

Intersections

Non-intersections (2-roads only)

With new connection road strategies:

In case of two roads, they are connected by a middle road if they have different lane configurations.

Merge at edge

Lanes closer to the median are connected first. Lanes at the edges are merged if there are no unique relationships possible. This is useful when roads are not already placed on inertial system.

```
LaneBuilder::createLanesForConnectionRoad
```

Merge by distance:

This is useful when roads alreay have fixed intertial co-ordinates. A pair of lanes having lower distance is connected before a pair with higher distance. When no unique relationships are possible, create merge lanes. We can start by finding the predecessor lanes to the reference line which are cloest. It has many corner cases and produces lane crossings which leads to unsafe paths.

Merge at median:

Opposite of Merge at edge. Pretty useful if edges are closer than medians.

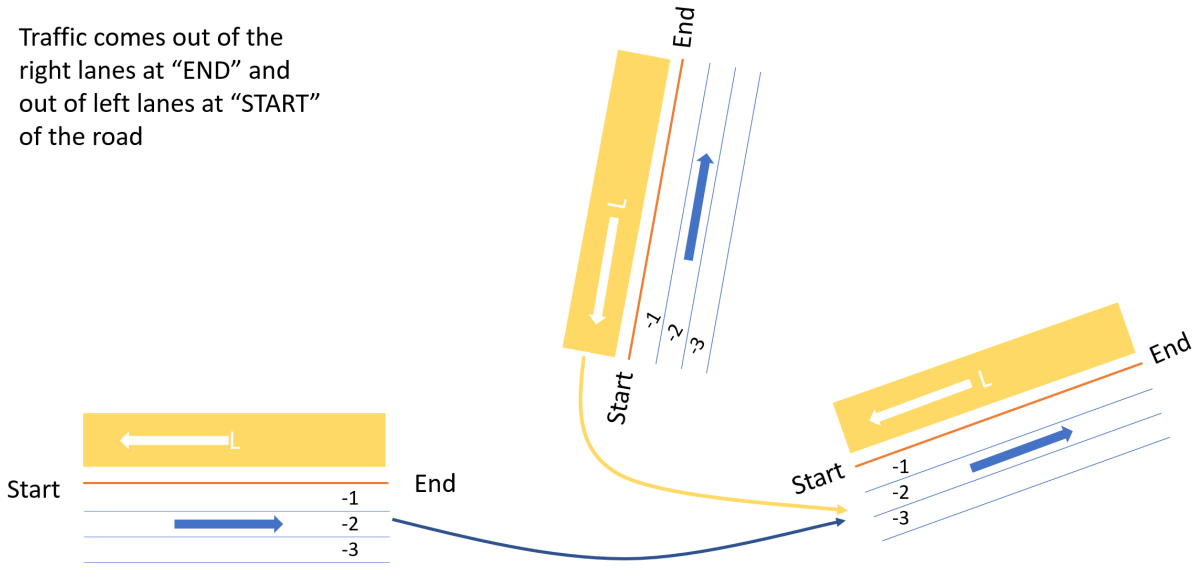
Merge in the middle:

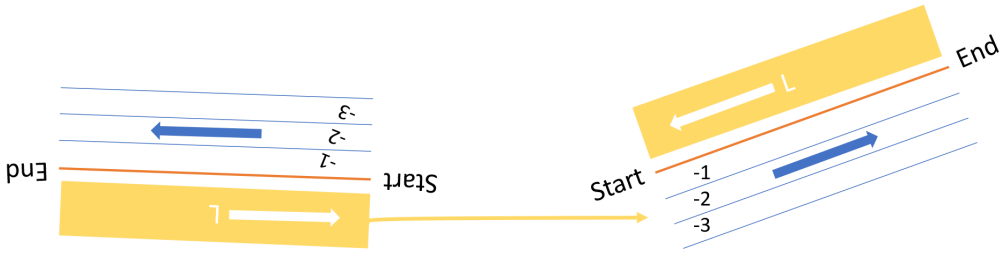
1. Connect (-1, -1)
2. Connect (-edge1, -edge2)
3. for each un connected lane, if no unique relationship, merge based on the distances from edge and median.

Lane Linker

Background:

Right lanes and left lanes in a road are relative to the start and endpoints of the road. This is not the same in a driver's point of view. When two roads are connected, it's important to consider their contact points. Check the following figures:





Traffic comes out of the
right lanes at “END” and
out of left lanes at “START”
of the road

It’s important to note that, when two roads have the **same contact point** (start-start or end-end), left lanes are to be linked with right lanes, and vice versa. In case of the **different contact points** (start-end or end-start), left lanes are to be linked with left lanes and right lanes with right lanes.

The lane linker class considers this concept and currently implemented for pairs where one is a connection road.

Sequential HD Map Builder

1. A collection of intersections, curves into unplaced segments
2. A collection of road segments(no intersection) into unplaced connection roads
3. Loop: 1.1. A Blank map with grid cells, 1.x1 meter cells. 1.2. Place the first intersection in the center of the map. Fill the occupied cells. 1.3. Update the list of open slots and 1.4. Choose a slot. Find a segment that can be connected to the slot without overlapping with any other segments in the map with a distance between (min, max). This would be a grid search. If no such candidates. End this loop. A scoring system for connecting to more slots? 1.5. So, now we have a location on the map for the new intersection. Rotate the intersection so that a matching incident road has the same heading as the chosen slot. 1.6. Connect the slot with the incident point with a straight road. Transform the new intersection and fill the cells for both the straight road and new intersection. 1.7. Connect 50% of the incident roads of the new intersection with existing slots without creating an overlap and a lot of curve. (This needs some greedy approach, otherwise it would be very slow). Maybe another grid search within a box. 1.8 update open slots.

Dynamic cell groups:

Each segment has a cell group. If we want to find the segment on the left. We need to find the group on the left. The group on the left can be bigger or smaller. How can we search efficiently?

When we fill the cell, we can store, segment ref, group ref.

So, it’s more like ray tracing. from a point. we trace cells in a direction. Before a placement, the bounding box must completely fill into a place without filled cells. Gap constraints?

Turn and Merge Lanes

Usage

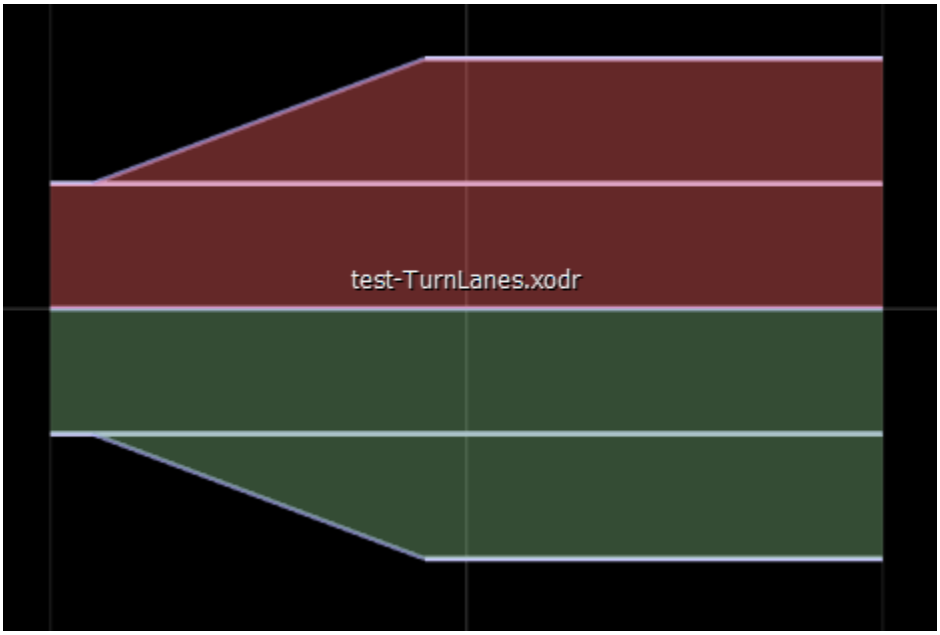
There are 3 variations of turn lanes:

1. Turn lanes at the end of a side:

method: **StraightRoadBuilder::create**

example:

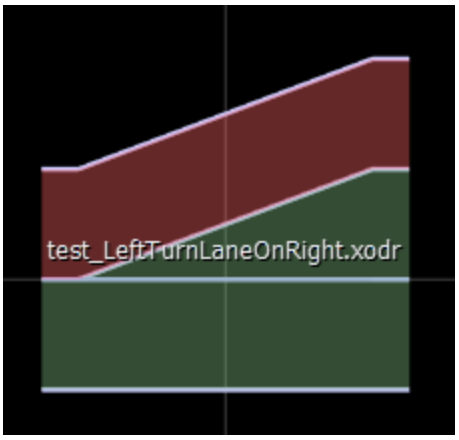
```
StraightRoadBuilder::create(0, length = 10,
                                        laneSides=LaneSides.RIGHT,
                                        isLeftTurnLane=True)
```



2. Turn lanes at the beginning of a side and merge lanes at the beginning on the other side

method: **StraightRoadBuilder::create** example:

```
StraightRoadBuilder::create(0, laneSides=LaneSides.RIGHT, isLeftTurnLane=True)
StraightRoadBuilder::create(0, laneSides=LaneSides.LEFT, isRighttTurnLane=True)
```



3. Turn lanes at the beginning of a side and no merge lanes at the beginning on the other side

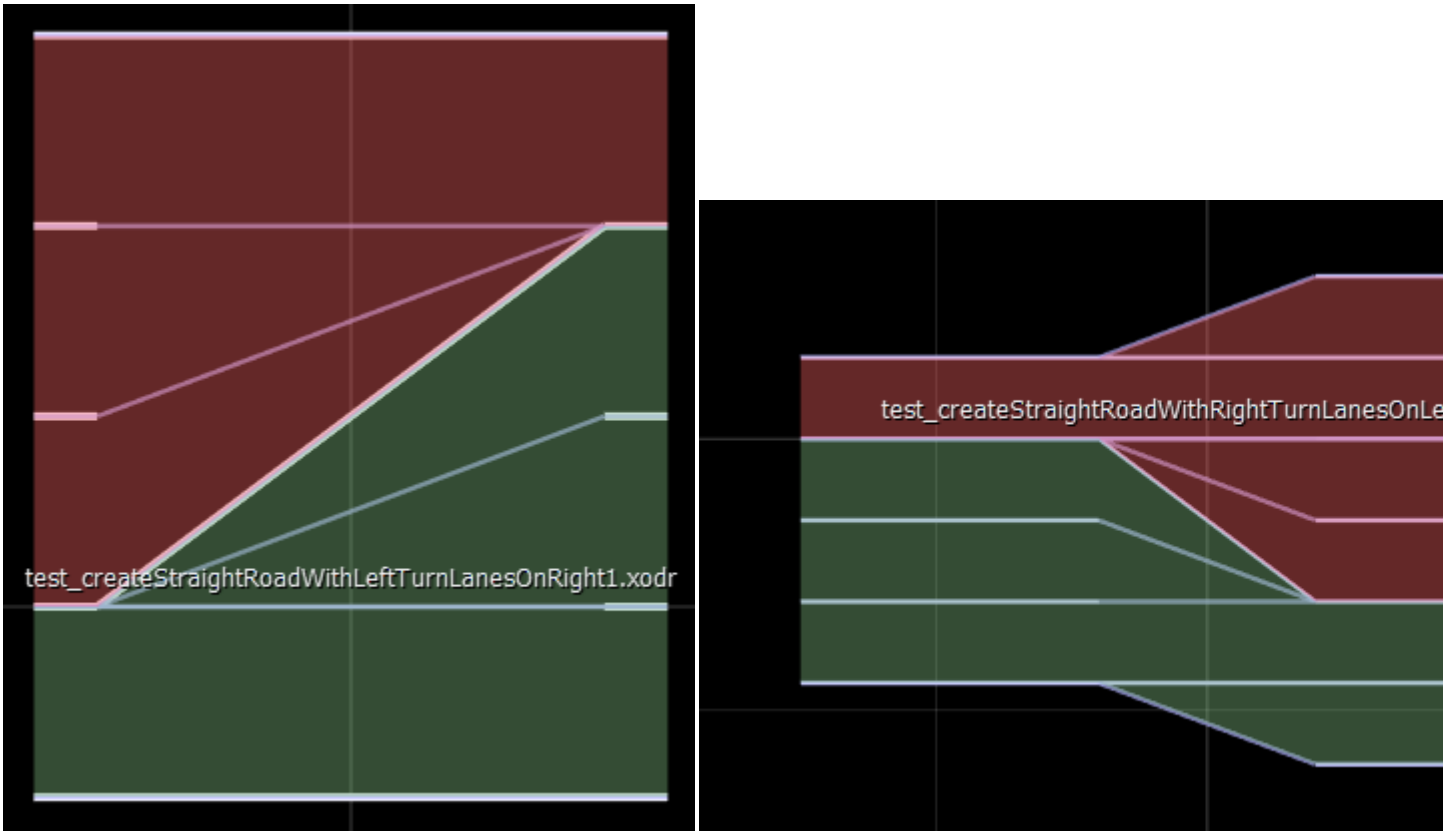
method: **StraightRoadBuilder::createWithRighttTurnLanesOnLeft**

method: **StraightRoadBuilder::createWithLeftTurnLanesOnRight**

example:

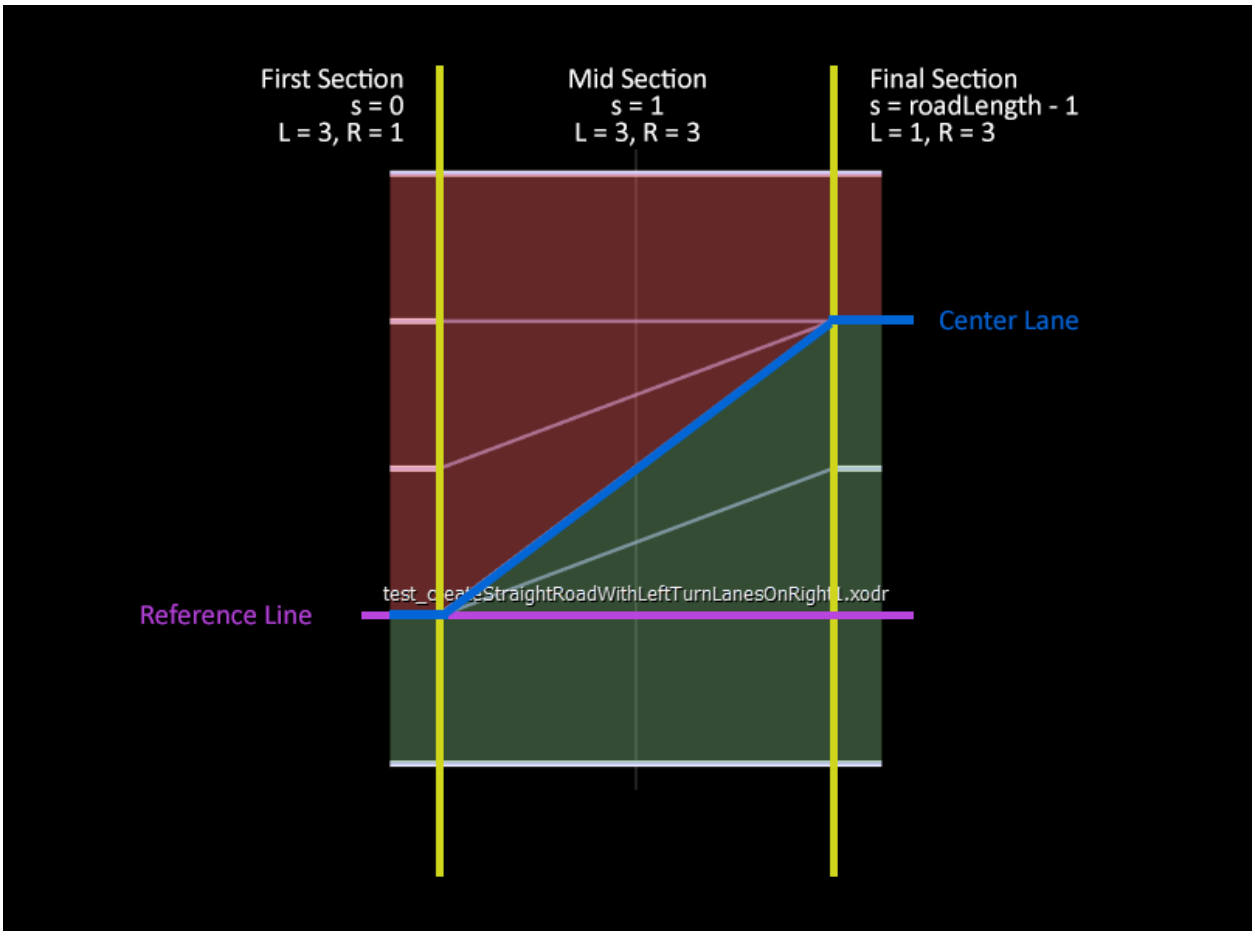
```
StraightRoadBuilder::createWithLeftTurnLanesOnRight(1, length = 10, n_lanes=1,
                                                                  isLeftTurnLane=True,
                                                                  isRightTurnLane=True,
                                                                 ␣
↪numberOfLeftTurnLanesOnRight=2))

StraightRoadBuilder::createWithRightTurnLanesOnLeft(1, length = 10, n_lanes=1,
                                                                  isLeftTurnLane=True,
                                                                  isRightTurnLane=True,
                                                                 ␣
↪numberOfRightTurnLanesOnLeft=2))
```



Similarly, merge lanes have similar structures. Merge lanes at the beginning of a side is only used in case of turn lanes at the beginning of a side to make space for the turns.

Architecture



3 Lane Sections

To create turn lanes we create 3 consecutive lane sections starting at s=0. s=1, s=roadLength-1.

In case of turn lanes, the first section does not have the turn lane, the mid section has the curve, and the final section has a straight lane.

In case of merge lanes, the first section has straight lanes, mid has the curves, the final section does not have the merge lanes.

So, for on turn lane in a side with an existing lane, the first section will have 1 lane, mid section will have 2, final will have 2.

only mid section has a curve

This simplifies a lot of calculations for consecutive roads, road signs, etc.

Lane Offset Calculations:

For turn lanes at *only the beginning* of a side, the center lane is shifted up or down from the reference line (Which have the same effect has reference lines shifted down or up). Special functions available in RoadLinker to adjust laneOffsets of sucessor roads.

Road Utilities

Transforming roads

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search