

# Creating Living Scenery Technology Routes with WED 2.6 Shapes

## Introduction

Living Scenery Technology (LST) is a free plugin for X-Plane 11 and 12 that allows for animation of objects along arbitrary routes, with a high degree of flexibility. LST runs on a per-package basis, and operates on .lst files in each package, the **Init.lst** (sets up package-wide parameters, such as activation bounds), **Objects.lst** (defines static objects, and routes), **Datarefs.lst** (defines Datarefs to create, and set), and **PCL.lst** (for pilot controlled lighting – not discussed in this document). Only the Init.lst file is required.

WED 2.6 introduces a new feature – **Shapes**. Shapes are not an exportable scenery element, rather, they are simply polygon data that is exported into GIS formats (OSM and KML), which are entirely ignored by X-Plane. This feature is perfect for 3<sup>rd</sup> party utilities, such as LST, which rely on polygon data that should be ignored by the sim itself.

This guide will discuss how to use WED Shapes to create animation routes for LST. Before proceeding, it is recommended that you have a working knowledge of WED, and a basic knowledge of LST development concepts. Please reference the LST Developer Manual for further details on LST commands.

## Creating Shapes

To use the Shape tool in WED, select the shapes tool (as circled in the below screenshot). Create the desired path for a highway or loop using the shape tool, just like you would using a line tool.



Shapes are directional. Notice in the below screenshot one end has an arrow. LST uses this directional feature to determine the direction of the route – the end with the arrow is the end of the route, the end without an arrow is the start. You can reverse the direction of the route using WED's Edit -> Reverse function.

LST has simple curve handling along routes, so when adding normal curves do not add nodes to smooth the curve, make a 90 degree angle and let LST handle it. If you have a large curve, adding a few nodes to extend the curve may be desirable. Please see the blue lines in the below screenshot for a rough idea of LST's automatic curve handling.

Note that LST will cap the curve radius to  $\frac{1}{2}$  the length of the shortest of the legs the curve connects, so if you set up a route, and discover a turn needs to be tighter, simply add an additional node close to the corner to tighten the curve.



# Properties

Every shape and every node have a set of properties (see below screenshot). These properties are used to define behaviors at a route level, and waypoint level.

## Shape Properties

- Closed: This field is ignored by LST since this behavior is defined by whether the route is a LOOP or HIGHWAY.
- Properties: This field is used to define what the route does. This text will be inserted at the beginning of the route. Enter a HIGHWAY or LOOP command, with the appropriate arguments. Since this argument is a single line, you can use a semicolon as a newline delimiter. For example, for a highway that has its spawn rate adjusted by a Dataref, your shape property would look something like this:

```
"HIGHWAY,my_lib/Objects/People/Passenger_Animated_Random.obj,10,25;SPAWNFACTO  
R,my_scenery/icao/passenger_density"
```

Note the semicolon denoting the start of a new line.

## Node Properties:

Every node has its own set of properties that affect that waypoint in the route:

- Z Value: This is the speed at this waypoint in kilometers per hour.
- Description: This works just like the shape properties – put your waypoint modifiers here.

## Using WED Shapes for automatic LST route branching

LST has a unique feature for combining multiple routes – branching. When object reaches a waypoint, it has a chance to “branch”, or switch, to being on a different route. Put simply, branching is a fork in the road. That branching chance can be based on a condition, or a percent (ie 30% branch to another route, 70% remain on the current route). This allows for objects to dynamically pass through different routes, as opposed to always following a fixed route, allowing for a natural flow of traffic.

For clarity's sake, in this discussion we will use *source route* to denote the route the object starts on, and *target route* to denote the route the object could switch to if branching occurs.

In LST, branching is implemented with a BRANCH, or BRANCHIF command, which contains a zero-based *index* (ie the first route in the file is route 0, the second is route 1, etc) of a *target route*, and a condition, or random chance (ie 0.5, 50% of objects will be branched). Being index based, this is hard to set up by hand, and even *harder* to maintain.

Using WED shapes, LST branching can be implemented far easier. To set up branching simply collocate one node of the *source route* with the start node of the *target route*.

By default, the branch chance is 0.5, that is, 50% of objects will switch to the *target route*, and 50% will remain on the *source route*. One exception is, if this is the last node of the *source route*, the branch chance will be 100% as it is assumed you don't want some objects to continue to the *target route* and some to simply cease to exist. To specify the branch chance, or a condition for branching, you will need to add a special BRANCH command to the collocated node on the *source route*.

To specify a chance, use the command "BRANCH\_<chance>" where <chance> is a value between 0.0-1.0, with 0.0 causing no objects to branch, and 1.0 to cause every object to branch.

To specify a condition, use the command BRANCHIF\_<operand 1>,<comparison operator>,<operand 2>. Operand 1 and 2 can be a dataref or numeric value. Comparison operator can be <, =, or, >. If the equation evaluates to true, the objects will branch, otherwise they will remain on the route. For example, with the command: "BRANCHIF\_1,>,0" all objects would branch since 1 is greater than 0. Likewise, the command "BRANCHIF\_sim/cockpit2/switches/beacon\_on,=,1" would result in objects branching if the beacon switch is on, and continuing on the current route if it is off.

## Conversion to LST Formats

After you have set up your routes in WED, the final step is to convert it into the LST formats. A simple tool is provided for this – LST Generator. Simply run LST Generator.exe, then drag the scenery folder into the command line interface to paste its path (or paste the path manually) and hit enter. Init.lst and Objects.lst files will be generated in this folder. If you already had these LST files generated, the tool will automatically recycle these files.

Please note that this tool *is* case sensitive and makes *no* sanity or syntax checks. It does have thorough logging, so if the program fails to generate the files, (or crashes), and doesn't show an error message, you can check the LST Generator Log.txt, generated in the same folder as LST Generator.exe. You may be able to find at what point it is failing and thereby be able to check if there is an issue with your routes (i.e. a typo). If you do encounter such a situation, please do reach out *with* the log and the doc.osm from your scenery that it failed to convert, and I'll be happy to take a look to see what I can do to either avoid this going forward or provide better error messages.

## Example Package

Included is an example scenery, demonstrating various LST features, implemented via the shapes feature in WED. You can view it in sim at the airport "LST Test", and in WED version 2.6 or greater.

I hope this brief guide has been helpful. If you have any questions or have a specific use case that LST isn't handling well, please do reach out and I'd be happy to try to answer your questions and/or discuss what you need.