# City.Net Synthesis Template User’s Guide

* Version 3, 30-May, 2011: Corrected SVN repository URL and completed coordinate reference frames section. Updated node region and edge region types. Added partial quick start tutorial as Appendix B (missing portions of transportation system node and edge definition).
* Version 2, 24-May, 2011: Added graphical examples of node and edge regions, added section on coordinate frames.
* Version 1, 10-May, 2011: Initial Version

Paul Grogan, ptgrogan@mit.edu

## Introduction

The City.Net Synthesis Template was created to help synthesize the various systems to be analyzed within the City.Net decision support tool. The Synthesis Template is an object-oriented MATLAB program which allows system representations to be entered, checked, and visualized.

The focus of Synthesis is simply to represent data across various systems in a common format, which should not require any additional programming. In future development, the Synthesis Template will be extended to Analysis and Evaluation, which both will require custom behaviors to be programmed.

## Installation

The Synthesis Template source code can be accessed via the City.Net subversion (SVN) repository hosted at:

<https://spacenet.mit.edu/citynet>

The server uses a self-signed SSL certificate (which may be expired), so if you encounter a security warning dialog box, please accept. To access the source code, you will also need to enter your provided username and password. In addition to securing the files, these credentials also help to track changes.

Although you can browse the files using your web browser, it is much more efficient to install a SVN client. On the Windows platform, I recommend installing the Tortoise SVN client, which acts as a shell extension (i.e. additional right-click options). You can download Tortoise SVN from its project homepage:

<http://tortoisesvn.tigris.org/>

Once you have installed Tortoise SVN (or a similar SVN client), you can check out the City.Net repository by completing the following steps.

1. Create a new folder where you want to store the repository files
2. Right-click and select “SVN Checkout…”

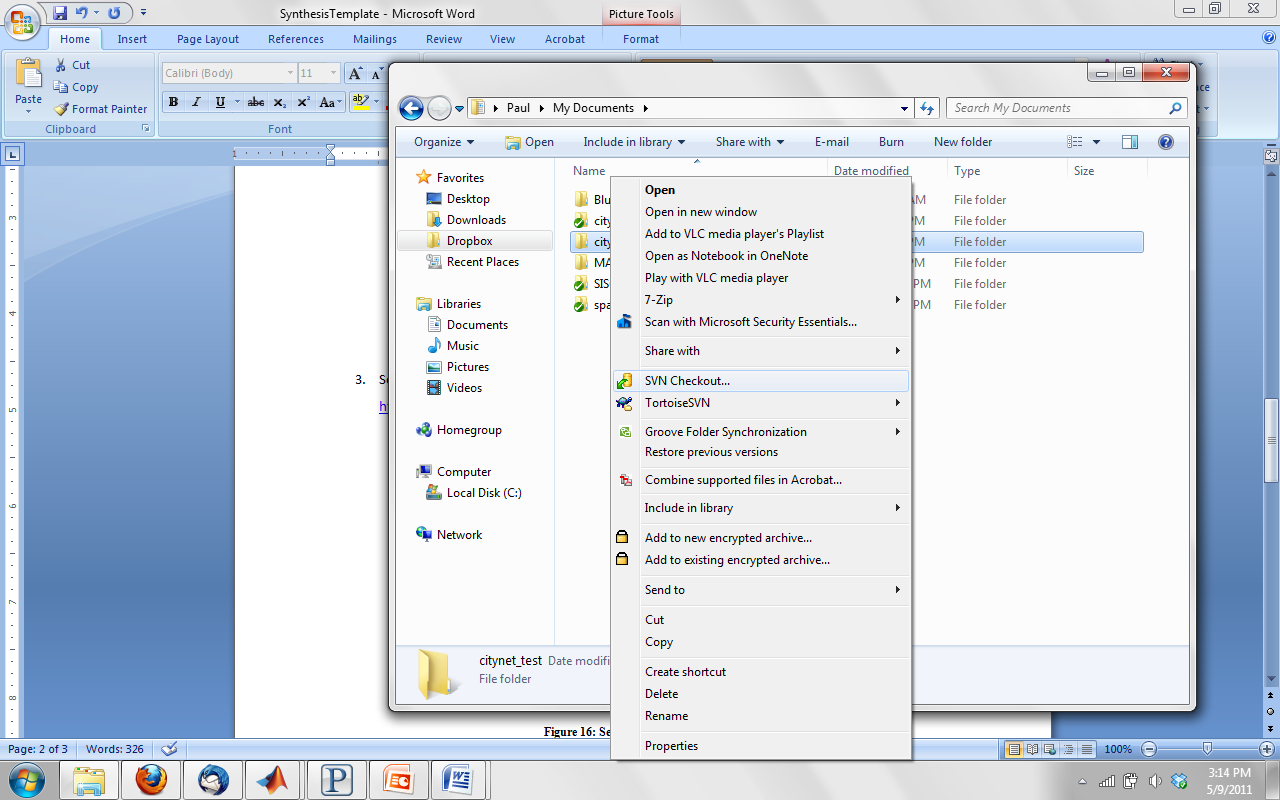


Figure : Tortoise SVN Checkout

1. Set the repository URL to:   
   <https://spacenet.mit.edu/citynet/trunk>   
   *Note: Repositories are set up with a “trunk” for current development and several “branches” for releases. This allows “code freeze” points without limiting future development.*

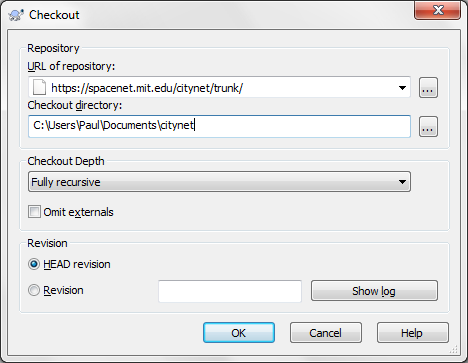


Figure : Setting the Repository URL

1. HEAD revision means the most current version of all files. Click OK to start the download process.
   1. You may be prompted with a warning about the self-signed certificate, please accept.
   2. You will also be prompted to enter your username and password for authentication.

## Synthesis Primer

Before you start synthesizing the city systems, let’s first discuss some of the high-level topics on this model construction. These will include definitions of coordinate reference frames, cells and layers (spatial and functional positioning), and nodes and edges (static and dynamic resources).

### Coordinate Reference Frames

The primary reference frame used for geo-spatial systems is the geographic coordinate frame, which uses a spherical coordinate system of latitude, longitude, and altitude (above sea level) to define points on the surface of the Earth. To better accommodate city features which do not follow the orthogonal cardinal directions (north-south and east-west), the synthesis template allows a rotation from the geographic coordinates to a local city coordinate frame.

On the city scale, coordinates are assumed to be planar Cartesian (i.e. orthogonal). The local city coordinate frame has both an X- and a Y-axis; typically the Y-axis runs more north-south, and the X-axis more east-west. Positive Y- is usually more south, positive X is usually more east. The rotation between the local coordinate frame and the geographic frame is specified in degrees counterclockwise, from the local coordinate frame to the geographic coordinate frame. For an example, see Figure 3 below.

In Figure 3, the local coordinate frame origin is defined at the longitude, latitude pair (θ, φ). The local coordinate frame axes are rotated by approximately 50 degrees (α) counter-clockwise from cardinal directions. The positive X-axis points to the south east and the positive Y-axis points to the south west.

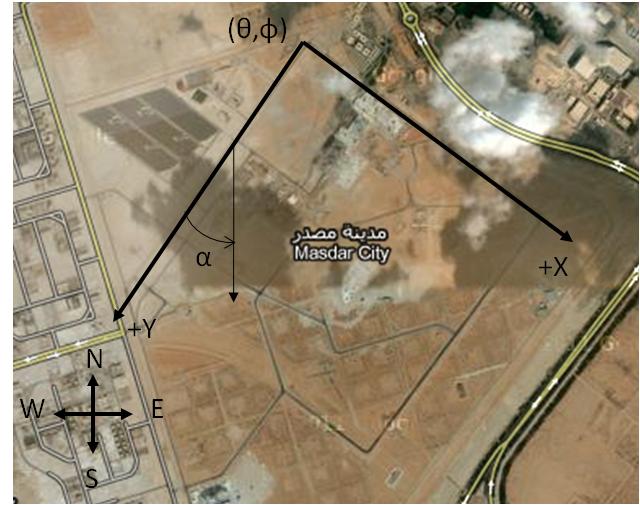


Figure : Reference Frames for a Sample City

### Cell and Layer Definition

The spatial area of a city is divided into rectangular cells which determine the level of analysis fidelity. Cells need not be square nor share common boundaries; however this is the most common format. Cells may either be defined in the synthesis template input file, or automatically generated during the synthesis execution. In the former case, each individual cell is specified by its vertices, while in the latter case cells would be specified by a bounding area and meshing rules (e.g. number of rows and columns).

|  |  |
| --- | --- |
| Figure : Sample City with 64 Cells | untitled.jpg  Figure : Sample City with 5 Layers |

The functional aspects of a city are divided into logical layers which often are illustrated in the z-direction in visualizations. There can be any number of layers within a city to represent several components within the same cell across functional domains. Layers must be specified in the synthesis template input, as they are used within the definitions of nodes and edges (covered in the next section).

### Node Definition

A node is defined by the intersection of a cell (spatial details) with a specific layer (functional details). Each node is provided a type which in turn inherits specific attributes – this formulation allows the user to change attributes in a single location to update all associated nodes.

Although it is the ultimate goal to define the set of nodes within a system, this task is complicated by the definition of cells. To allow a transition between multiple levels of analysis fidelity, nodes are preferable defined through the use of *node regions*. A node region is an area where nodes are generated at any cells that meet the generation conditions. A node region is only specified at a single layer, and as one associated node type for assignment to the generated nodes.

There are three available types of node regions: polygon, polyline, and polypoint. A polygon node region generates nodes at cells that have sufficient overlap with a polygon region. A polyline node region generates nodes at cells that intersect a multi-segment line. Finally, a polypoint node region generates nodes at cells that contain vertices. Examples of the three types of node regions are shown below in Figure 6.

|  |  |  |  |
| --- | --- | --- | --- |
| a) | b) | c) | d) |

Figure : a) Node Region Vertices with Type b) Polygon c) Polyline d) Polypoint

For example, consider the figure below. A potential node region is drawn as a polygon overlaying a default cell grid. During the node generation process, nodes will be generated at any cell having a critical fraction of overlap with the node region. Examples of 0.0, 0.5, and 1.0 for a minimum overlap fraction are shown below in Figure 6.

|  |  |  |  |
| --- | --- | --- | --- |
| a) | b) | c) | d) |

Figure : a) Node Region with Minimum Overlap Fraction b) 0.0, c) 0.5, and d) 1.0

### Edge Definition

An edge is defined by a connection between two nodes (each positioned at a cell and within a layer). Edges will always connect nodes within the same system, but may span two different layers. Just like in the case of the nodes, each edge is provided a type which in turn inherits specific attributes – this formulation allows the user to change attributes in a single location to update all associated edges.

Again, although it is the ultimate goal of synthesis to define the set of edges within a system, this task is complicated by the definition of cells. Similar to node regions, edge regions can be defined to specify regions where edges should be generated. There are two types of edge regions in use: linear and spatial. Linear edge regions (polyline and polypoint) specify segments between locations where an edge should exist. Spatial edge regions (polygon orthogonal, adjacent, and connected) specify regions where all nodes should be considered for connection under a generation rule.

For example, consider the figure below. A potential edge region is drawn as a polygon overlaying a default cell grid. During the edge generation process, edges will be generated between existing nodes according to the edge region connectivity option. The polyline perimeter option generates edges between adjacent vertices of the polygon. The orthogonal and all neighbors options generate edges in orthogonal and diagonal directions between any nodes inside the region (according to the minimum overlap fraction, covered previously in the node region section). Although not shown, the fully connected option would create edges between each two nodes contained within the region.

|  |  |  |  |
| --- | --- | --- | --- |
| a) | b) | c) | d) |

Figure : a) Edge Region with b) Polypoint, c) Polygon Orthogonal, and d) Polygon Adjacent

## Spreadsheet Template

To help the system synthesis process, a spreadsheet template is available for use. This guide will walk through the example transportation system which is available in the \_example directory in the SVN trunk.

*Note: On non-Windows operating systems, MATLAB sometimes has difficulty reading Excel spreadsheets. Although the ‘basic’ mode is used for the xlsread functions, sometimes it may still fail due to formatting changes between versions of Excel. If you encounter this issue, save the spreadsheet as an Excel 95 file format.*

### City Worksheet

The city worksheet defines high-level parameters for the city being modeled. These values should not have to be modified.

Table : City Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| **City Name** | Name of the city being modeled. | String | - | - |
| **Latitude** | Latitude of the local reference frame origin (North is positive, South is negative). | Double | Degrees | [-90,90] |
| **Longitude** | Longitude of the local reference frame origin (West is positive, East is negative). | Double | Degrees | (-180,180] |
| **Rotation** | Rotation of the local reference frame with respect to the North-South and West-East lines. | Double | Degrees counterclockwise | (-180,180] |
| **Image Path** | File path to a ground underlay image. | String | - | - |
| **Image Vertices X** | X-coordinate locations of the image vertices (counterclockwise from upper left corner). | Double[4] | *Distance* | (-inf,inf) |
| **Image Vertices Y** | Y-coordinate locations of the image vertices (counterclockwise from upper left corner). | Double[4] | *Distance* | (-inf,inf) |
| **Min. Intersection Fraction** | Parameter that sets the minimum fraction of a cell intersected by a region such that a node is generated. | Double | - | [0,1] |

### Node Types Worksheet

The node types worksheet defines the classifications of nodes that will comprise each system. The worksheet depends on the node type attributes worksheet to assign characteristics to each node type.

Table : Node Type Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Node Type ID | Unique identifier of each node type. | Integer | - | [1,inf) |
| Name | Short name of the node type. | String | - | - |
| Description | Longer description of the node type. | String | - | - |
| Hex Color | Color code for display in visualizations. Required to start with “0x” followed by three pairs of hexadecimal (0-9, a-f) values for the amount of red, green, and blue. For example, 0xff0000 is pure red and 0x666666 is a medium gray. | String | - | 0x###### |

### Node Type Attributes Worksheet

The node type attributes worksheet assigns quantitative characteristics to each node type. Presently, attributes are limited to double-type (decimal), however future development may explicitly allow integers, Booleans, enumerations, strings, and other formats.

Table : Node Type Attribute Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Attribute ID | Unique identifier of each node type attribute. | Integer | - | [1,inf) |
| Node Type ID | Reference to node type to which this attribute is assigned. | Integer | - | Valid Node ID |
| Name | Short name of the attribute. | String | - | - |
| Description | Longer description of the attribute. | String | - | - |
| Units | Abbreviation for units used to define attribute. | String | - | - |
| Bounds | Annotation for acceptable range of values. | String | - | - |
| Value | Quantitative value for attribute. Currently limited to double values. | Double | - | - |

### Edge Types Worksheet

The edge types worksheet defines the classifications of edges that will comprise each system. The worksheet depends on the edge type attributes worksheet to assign characteristics to each edge type.

Table : Edge Type Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Edge Type ID | Unique identifier of each edge type. | Integer | - | [1,inf) |
| Name | Short name of the edge type. | String | - | - |
| Description | Longer description of the edge type. | String | - | - |
| Hex Color | Color code for display in visualizations. Required to start with “0x” followed by three pairs of hexadecimal (0-9, a-f) values for the amount of red, green, and blue. For example, 0xff0000 is pure red and 0x666666 is a medium gray. | String | - | 0x###### |

### Edge Type Attributes Worksheet

The edge type attributes worksheet assigns quantitative characteristics to each edge type. Presently, attributes are limited to double-type (decimal), however future development may explicitly allow integers, Booleans, enumerations, strings, and other formats.

Table : Edge Type Attribute Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Attribute ID | Unique identifier of each edge type attribute. | Integer | - | [1,inf) |
| Edge Type ID | Reference to edge type to which this attribute is assigned. | Integer | - | Valid Edge ID |
| Name | Short name of the attribute. | String | - | - |
| Description | Longer description of the attribute. | String | - | - |
| Units | Abbreviation for units used to define attribute. | String | - | - |
| Bounds | Annotation for acceptable range of values. | String | - | - |
| Value | Quantitative value for attribute. Currently limited to double values. | Double | - | - |

### Layers Worksheet

The layers worksheet defines the functional layers that exist within the city. Each system will have at least one, and likely several, layers to represent various functions that take place within a single cell.

Table : Layer Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Layer ID | Unique identifier of each layer. | Integer | - | [1,inf) |
| Name | Short name of the attribute. | String | - | - |
| Description | Longer description of the attribute. | String | - | - |
| Display Height | The z-coordinate at which to display this layer in three-dimensional plots. | Double | *Distance* | [0,inf) |

### Systems Worksheet

The systems worksheet defines the major systems that are being modeled within a city. You should not have to edit this sheet.

Table : System Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| System ID | Unique identifier of each system. | Integer | - | [1,inf) |
| Name | Short name of the attribute. | String | - | - |
| Description | Longer description of the attribute. | String | - | - |

### Node Regions Worksheet

The node regions worksheet defines areas within the city that share similar characteristics and should be represented by nodes of the same type. It is independent of the cell definition such that a single node region definition can be used across several different levels of analysis fidelity.

Table : Node Region Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Node Region ID | Unique identifier of each node region. | Integer | - | [1,inf) |
| System ID | Reference to the system identifier that contains this node region. | Integer | - | Valid System ID |
| Node Type ID | Reference to the node type identifier that should be used to generate nodes. | Integer | - | Valid Node Type ID |
| Layer ID | Reference to the layer identifier that should contain the generated nodes. | Integer | - | Valid Layer ID |
| Vertices X | X-coordinate locations of vertices that define the polygonal region. Single points and lines are also allowed: in these cases, only the points specified will be used to generate nodes. | Double[] | *Distance* | (-inf,inf) |
| Vertices Y | Y-coordinate locations of vertices that define the polygonal region. Single points and lines are also allowed: in these cases, only the points specified will be used to generate nodes. | Double[] | *Distance* | (-inf,inf) |
| Generation Type | Selection of node generation type: polygon, polyline, or polypoint | String | - | {polygon, polyline, polypoint} |
| Description | Optional description of node region. | String | - | - |

### Edge Regions Worksheet

Similar to node regions, the edge regions worksheet is used to define regions that should be connected using edges in a particular way. There are four types of regions currently available:

* **Polyline Perimeter**: Connects nodes that are specified along the perimeter of a polygon.
* **Orthogonal Neighbors:** Connects all nodes within a region in orthogonal directions (e.g. North-South and West-East). Assumes a square or rectangular grid of cells.
* **All Neighbors**: Connects all nodes within a region in orthogonal and diagonal directions. Assumes a square or rectangular grid of cells.
* **Fully Connected**: Connects each node within a region to all other nodes within a region.

Table : Edge Region Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Edge Region ID | Unique identifier of each edge region. | Integer | - | [1,inf) |
| System ID | Reference to the system identifier that contains this node region. | Integer | - | Valid System ID |
| Edge Type ID | Reference to the edge type identifier that should be used to generate edges. | Integer | - | Valid Edge Type ID |
| Layer IDs | Reference to the layer identifiers that should contain the generated edges. | Integer[] | - | Valid Layer ID |
| Vertices X | X-coordinate locations of vertices that define the polygonal region. Polyline perimeter must be defined in counterclockwise order. | Double[] | *Distance* | (-inf,inf) |
| Vertices Y | Y-coordinate locations of vertices that define the polygonal region. Polyline perimeter must be defined in counterclockwise order. | Double[] | *Distance* | (-inf,inf) |
| Connection Type | Selection of connection generation type: polyline, polypoint, polygon orthogonal (orthogonal), polygon adjacent (adjacent) or polygon fully connected (connected). | String | - | {polypoint, polyline, orthogonal, adjacent, connected} |
| Directed | Specifies whether the edges are directed (1) or undirected (0). | Boolean | - | {0,1} |
| Description | Optional description of edge region. | String | - | - |

### Cell Regions Worksheet

The cell regions worksheet allows automatic generation of rectangular cells within a region, similar to the node region. Presently, cell regions only support rectangular bounding boxes as regions, and will use the minimum and maximum X- and Y-coordinates to define such. All cells generated will be rectangular, square if the width per column is equal to the height per row.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Cell Region ID | Unique identifier of each cell region. | Integer | - | [1,inf) |
| Vertices X | X-coordinate locations of vertices that define the polygonal region. Vertices must be defined in counterclockwise order, and currently only rectangular bounding regions are used. | Double[] | *Distance* | (-inf,inf) |
| Vertices Y | Y-coordinate locations of vertices that define the polygonal region. Vertices must be defined in counterclockwise order, and currently only rectangular bounding regions are used. | Double[] | *Distance* | (-inf,inf) |
| Number Rows | Number of rows (along Y-axis) of cells to generate within region. | Integer | - | [1,inf) |
| Number Columns | Number of columns (along X-axis) of cells to generate within region. | Integer | - | [1,inf) |
| Description | Optional description of cell region. | String | - | - |

### Other Worksheets (Cells, Nodes, Edges)

The other three worksheets – Cells, Nodes, and Edges – are typically product of a generation process, rather than system-specific input. Although it is possible to explicitly define the cells, nodes, and edges, this ties the specific implementation to one cell grid.

Table : Cell Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Cell ID | Unique identifier for cell. | Integer | - | [1,inf) |
| Location X | X-coordinate location of upper-left hand corner. | Double | *Distance* | (-inf,inf) |
| Location Y | Y-coordinate location of upper-left hand corner. | Double | *Distance* | (-inf,inf) |
| Dimension X | Width of cell (in x-axis direction). | Double | *Distance* | [0,inf) |
| Dimension Y | Height of cell (in y-axis direction). | Double | *Distance* | [0,inf) |

Table : Node Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Node ID | Unique identifier for node. | Integer | - | [1,inf) |
| System ID | Reference to system identifier to contain node. | Integer | - | Valid System ID |
| Node Type ID | Reference to node type identifier to assign to node. | Integer | - | Valid Node Type ID |
| Cell ID | Reference to cell identifier to spatially position node. | Integer | - | Valid Cell ID |
| Layer ID | Reference to layer identifier to functionally position node. | Integer | - | Valid Layer ID |

Table : Edge Worksheet Inputs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable** | **Description** | **Data Type** | **Units** | **Bounds** |
| Edge ID | Unique identifier for edge. | Integer | - | [1,inf) |
| System ID | Reference to system identifier to contain node. | Integer | - | Valid System ID |
| Edge Type ID | Reference to edge type identifier to assign to edge. | Integer | - | Valid Edge Type ID |
| Origin Node ID | Reference to node identifier for origin of edge. | Integer | - | Valid Node ID |
| Destination Node ID | Reference to node identifier for destination of edge. | Integer | - | Valid Node ID |
| Directed | Specifies whether the edges are directed (1) or undirected (0). | Boolean | - | {0,1} |

## Appendix A: Object Class Diagram

The City.Net Synthesis Template is programmed using object-oriented MATLAB scripts. To help understand the technical details of implementation, including class dependencies, an object class diagram is presented below. In addition, please see the detailed comments and documentation provided in the source code and examples.

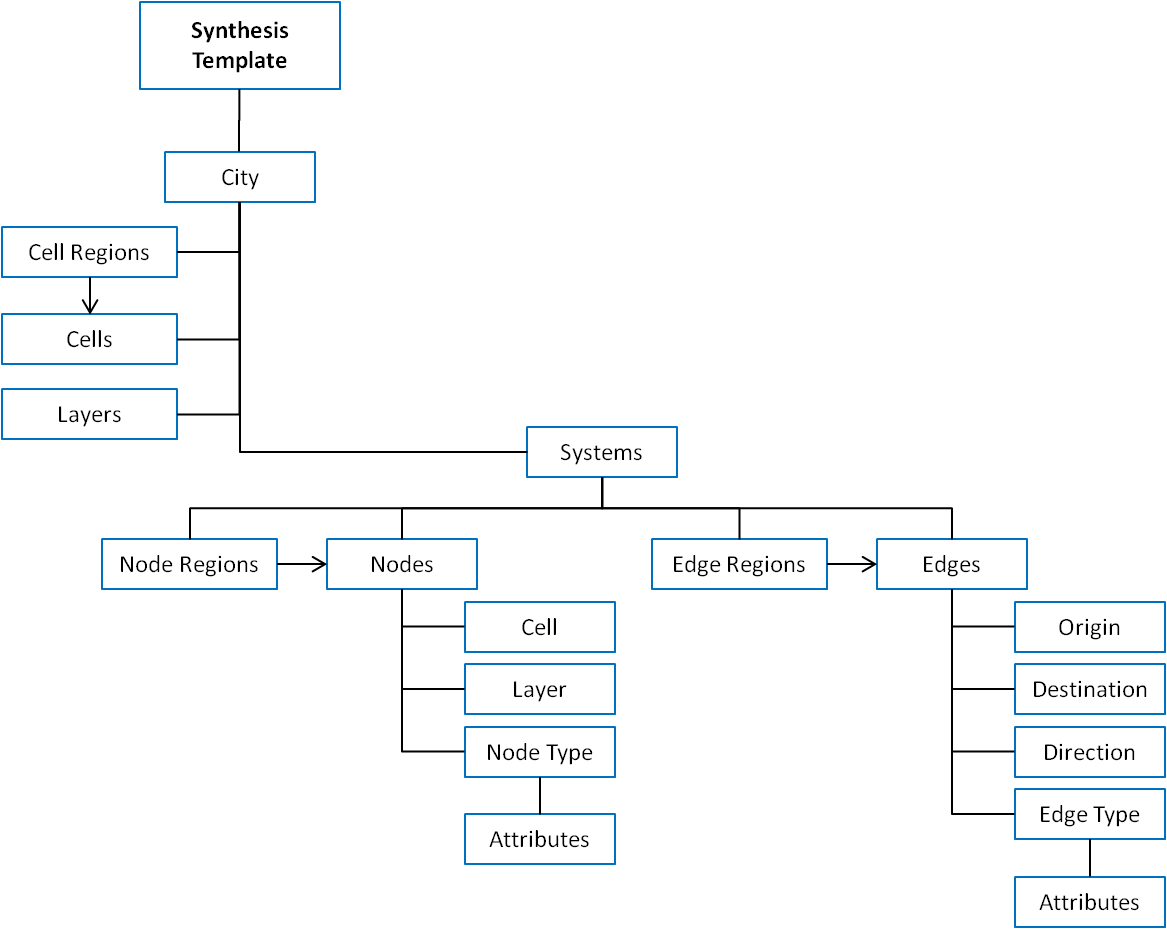


Figure : Synthesis Template Object Class Diagram

## Appendix B: Quick Start Tutorial

### Introduction

This quick start tutorial is intended to introduce the main components of the City.Net Synthesis Template in a step-by-step tutorial format. We will be modeling a sample city with notional attributes to simplify the learning process. Please use the files provided in the /\_tutorial directory in the SVN repository. There are both copies of the blank and completed files for use and reference.

This tutorial is designed to take approximately one hour to complete. Feel free to skip or copy-and-paste blocks of text where desired to speed up the process, such as for database definitions.

### City and Systems Definition

This tutorial is based on the city-center of Boston. The area we will be working with is approximately 3 kilometers by 3 kilometers as displayed in Figure 9 below. The image spans from local coordinates (x,y)=(0,0) in the northwest corner, to (x,y)=(3,3) in the southeast corner. For reference, the major districts in this area include the North End, Downtown, South Boston, the Boston Common and Public Garden, Beacon Hill, and the Esplanade.



Figure : Sample Boston City Map (3 km x 3 km)

The local reference frame origin will be placed in the upper-left hand corner of this image. This corresponds to 42.369734°N, 71.073127°W. The first step in this tutorial is setting high-level city parameters.

1. Open the provided Excel spreadsheet, **synthesisTutorial.xls**, and open the “**city**” worksheet
2. In cell **B1** enter the city name: Boston
3. In cell **B2** enter the latitude: 42.369734
4. In cell **B3** enter the longitude: -71.073127  
   *Note: longitudes in the Western Hemisphere are negative.*
5. In cell **B4** enter the local frame rotation: 0  
   *Note: the coordinate frame is not rotated for this example because the lack of structure in the city allows cardinal directions work as well as any for defining cells.*
6. In cell **B5** enter the image path: boston\_3km.jpg  
   *Note: the image path is relative to the MATLAB path, which will likely include the City.Net application folder and the local folder containing your city synthesis script. In this tutorial, the image is stored in the /\_tutorial directory along with the synthesis script.*
7. In cell **B6** enter the image x-vertices: [0 0 3 3]  
   *Note: brackets are required for evaluation by MATLAB. Some MATLAB functions require vertex definition in counter-clockwise order. In this case, the vertices are defined in the following order: northwest, southwest, southeast, northeast.*
8. In cell **B7** enter the image y-vertices: [0 3 3 0]
9. In cell **B8** enter the minimum intersection fraction: 0.5  
   *Note: this parameter helps determine which cells should include a node, based on the intersection fraction between a cell and a region. In future versions of City.Net, this feature will likely be moved to a region-specific value with a city-wide default.*
10. *Future versions of City.Net will also include a user-definable unit of distance. Present versions default to kilometers.*

The completed city worksheet is shown below, in Table 13.

Table : Completed City Worksheet

|  |  |
| --- | --- |
| **City Name:** | Boston |
| **Latitude:** | 42.369734 |
| **Longitude:** | -71.073127 |
| **Rotation:** | 0 |
| **Image Path:** | boston\_3km.jpg |
| **Image Vertices X:** | [0 0 3 3] |
| **Image Vertices Y:** | [0 3 3 0] |
| **Min. Intersection Fraction:** | 0.5 |

At this point, you should be able to run portions of the Synthesis Template program. Although we have not yet defined cells, nodes, or edges, we can still visualize the city using the specified image.

1. Open the MATLAB script, **mainTutorial.m**, and add the following lines of code:
   1. First, clear all variables out of the MATLAB memory, close any existing figures, and clear the console output:  
      **clear classes  
      close all  
      clc**
   2. Next, add the path to the City.Net program files; the default installation will simply be the next directory up from the tutorial directory:  
      **addpath(‘..’)**
   3. Create a reference to the Synthesis Template object:  
      **synthTemp = SynthesisTemplate.instance();**  
      *Note: the Synthesis Template application follows a Singleton design pattern which restricts it to only have one instance running at a time. To access this instance, the static instance() function is used.*
   4. Next, instruct the Spreadsheet Reader object to read in your template file:  
      **SpreadsheetReader.ReadTempate(‘synthesisTutorial.xls’);**
   5. Finally, generate a new figure and render your city model:  
      **figure(1)  
      synthTemp.RenderCity();**
2. Run the **mainTutorial.m** script; the figure generated should appear like that in Figure 10.

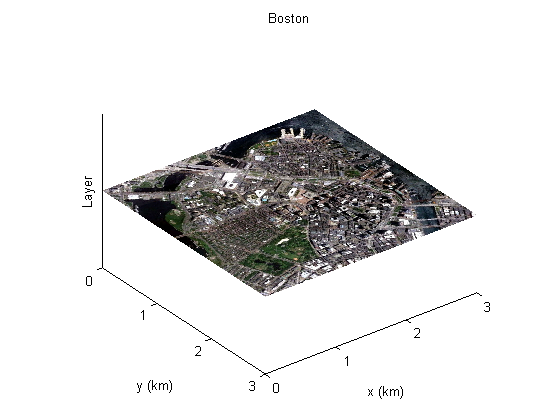


Figure : Initial City Rendering

We will use the default systems definitions, covering the building, energy, transportation, waste, and water systems, as shown in Table 14. Although all five are defined, we will only be defining model components for the building and transportation systems.

Table : Completed Systems Worksheet

|  |  |  |
| --- | --- | --- |
| **System ID** | **Name** | **Description** |
| 1 | Building | System to model the buildings and urban system. |
| 2 | Energy | System to model the generation, distribution, and consumption of energy. |
| 3 | Transportation | System to model the movement of people and cargo across several modes of transportation. |
| 4 | Waste | System to model the generation, distribution, and consumption of water. |
| 5 | Water | System to model the generation and processing of waste. |

### Cell Regions and Layers

First, we will specify the functional layers in which we will perform analysis. Layers act as a “third dimension” in the functional view of a spatial area, thus we will define four layers corresponding to the high-level functional decomposition of the city: buildings, recreation, public transportation, and private transportation. The buildings and recreation layers will ultimately be tied to the building system and the public and private transportation layers will be tied to the transportation system. This is not a unique decomposition – the layers used are primarily driven by the desired visualization and analysis coupling.

1. Open the “**layers**” worksheet
2. Add a row for layer 1:
   * Layer ID: 1
   * Name: **Buildings**
   * Description: Residential, commercial, and industrial buildings
   * Display Height: 0.5
3. Add a row for layer 2:
   * Layer ID: 2
   * Name: **Recreation**
   * Description: Parks and other recreational areas
   * Display Height: 1.0
4. Add a row for layer 3:
   * Layer ID: 3
   * Name: **Public Trans.**
   * Description: Metro lines and other public transportation
   * Display Height: 1.5
5. Add a row for layer 4:
   * Layer ID: 4
   * Name: **Private Trans.**
   * Description: Automobile roadways and other private transportation
   * Display Height: 2.0

The completed layers worksheet is shown below in Table 15.

Table : Completed Layers Worksheet

|  |  |  |  |
| --- | --- | --- | --- |
| **Layer ID** | **Name** | **Description** | **Display Height** |
| 1 | Buildings | Residential, commercial, and industrial buildings | 0.5 |
| 2 | Recreation | Parks and other recreational areas | 1 |
| 3 | Public Trans. | Metro lines and other public transportation | 1.5 |
| 4 | Private Trans. | Automobile roadways and other private transportation | 2 |

Next, we will define the cells within which to perform analysis. A cell region is a polygon in which cells should be generated through a meshing procedure. To simplify this example, we will only create a single region that spans the entirety of the specified area from (0,0) to (3,3). This tutorial will start with a 20 by 20 grid, totaling 400 cells.

1. Open the “**cell\_regions**” worksheet
2. Add a row for the single cell region:
   * Cell Region ID: 1
   * VerticesX: [0 0 3 3]
   * VerticesY: [0 3 3 0]
   * Rows: 20
   * Columns: 20  
     *Note that, similar to image vertices, cell region vertices must be specified in counterclockwise order.*

The completed cell region worksheet is shown below in Table 16.

Table : Completed Cell Region Worksheet

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cell Region ID** | **VerticesX** | **VerticesY** | **Rows** | **Columns** |
| 1 | [0 0 3 3] | [0 3 3 0] | 20 | 20 |

Finally, now that we have defined a cell region and layers, we must explicitly generate the cells to display. In a graphical user interface, this process would be automated.

1. Open the MATLAB script, **mainTutorial.m**, and add the following lines of code after the ReadTemplate function call to generate the cells contained within any cells regions:  
   **synthTemp.GenerateCells();**
2. Run the **mainTutorial.m** script; the figure generated should appear like that in Figure 11.

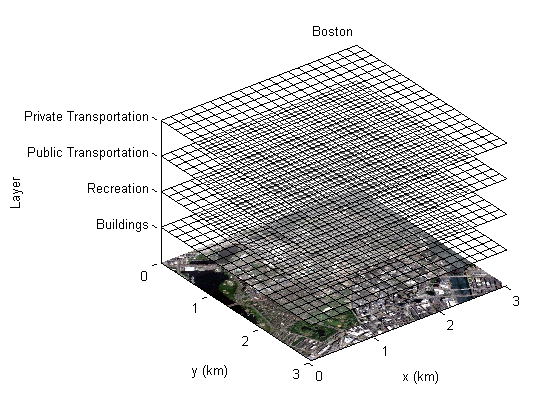


Figure : Rendered City with Layers and Cells

### Conceptual Regions

In the next few sections, we will define the nodes and edges which make up the bulk of the Synthesis Template. First, however, let’s look at a conceptual map of the Boston area to help identify the various regions which we will model, shown in Figure 12.

The region has been divided into polygons that roughly are characteristics of regions we would like to model. For example, the red regions are primarily residential, blue commercial, yellow industrial, and green recreational within the building system. For the transportation system, approximate locations of highways and metro lines are sketched, which will correspond to both nodes and edges.

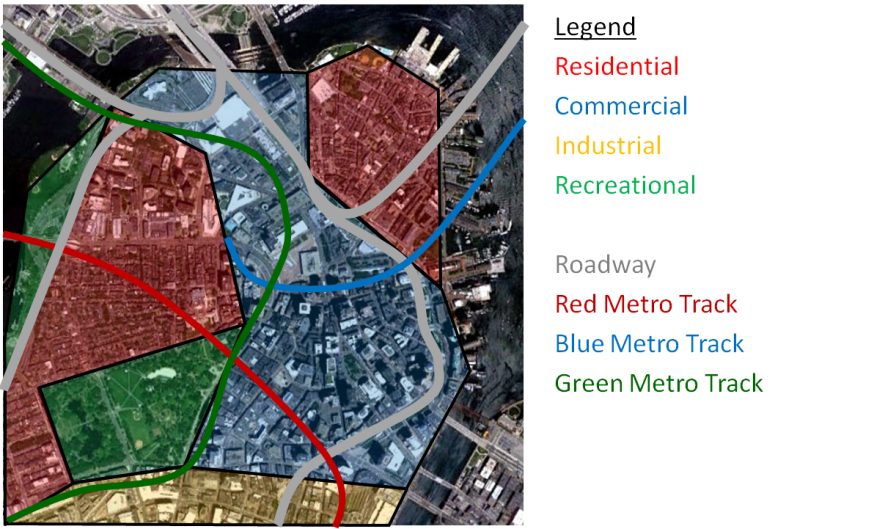


Figure : Conceptual Mapping of Nodes and Edges

The next few sections will describe in detail how to model the nodes and edges within each system and layer.

### Building System Nodes

The building system nodes lie in two separate layers: the building layer and the recreational layer. The building layer is comprised of three node types: residential, commercial, and industrial. The recreational layer is comprised of one node type: recreational.

1. Open the “**node\_types**” worksheet
2. Add a row for node type 1:
   * Node Type ID: 1
   * Name: **Residential**
   * Description: Housing units in North End and Beacon Hill
   * Hex Color: 0x99333
3. Add a row for node type 2:
   * Node Type ID: 2
   * Name: **Commercial**
   * Description: Office buildings in downtown Boston
   * Hex Color: 0x333399
4. Add a row for node type 3:
   * Node Type ID: 3
   * Name: **Industrial**
   * Description: Wharf and harbor area just south of Boston
   * Hex Color: 0x999933
5. Add a row for node type 4:
   * Node Type ID: 4
   * Name: **Recreational**
   * Description: Parks in Boston Common, Public Garden, and Esplanade
   * Hex Color: 0x339933

These steps are summarized in Table 17, below.

Table : Building System Node Types

|  |  |  |  |
| --- | --- | --- | --- |
| **Node Type ID** | **Name** | **Description** | **Hex Color** |
| 1 | Residential | Housing units in Back Bay and Beacon Hill | 0x993333 |
| 2 | Commercial | Office buildings in downtown Boston | 0x333399 |
| 3 | Industrial | Wharf and harbor area just south of Boston | 0x999933 |
| 4 | Recreational | Parks in Boston Common, Public Garden, and Esplanade | 0x339933 |

We will also define node type attributes for each node type. Node type attributes are the values used in evaluation to help perform analysis. For the building system node types, we will assign values for three different attributes: resident density, employee density, and energy consumption, all on a per-area basis. *Note: all attribute values are fictitious.*

1. Open the “**node\_type\_attributes**” worksheet
2. Add three rows for the residential node type (ID 1)
   * residentDensity: 20000 1/km^2
   * employeeDensity: 1000 1/km^2
   * energyConsumption: 500 kW-h/km^2
3. Add three rows for the commercial node type (ID 2)
   * residentDensity: 7500 1/km^2
   * employeeDensity: 50000 1/km^2
   * energyConsumption: 1250 kW-h/km^2
4. Add three rows for the industrial node type (ID 3)
   * residentDensity: 3000 1/km^2
   * employeeDensity: 30000 1/km^2
   * energyConsumption: 2000 kW-h/km^2
5. Add three rows for the recreational node type (ID 4)
   * residentDensity: 0 1/km^2
   * employeeDensity: 100 1/km^2
   * energyConsumption: 200 kW-h/km^2

These steps are summarized in Table 18, below.

Table : Building System Node Type Attributes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute ID** | **Node Type ID** | **Name** | **Description** | **Units** | **Bounds** | **Value** |
| 1 | 1 | residentDensity | Number of residents per area | 1/km^2 | [0,inf) | 20000 |
| 2 | 1 | employeeDensity | Number of employees per area | 1/km^2 | [0,inf) | 1000 |
| 3 | 1 | energyConsumption | Annual energy consumption per area | kW-h/km^2 | [0,inf) | 500 |
| 4 | 2 | residentDensity | Number of residents per area | 1/km^2 | [0,inf) | 7500 |
| 5 | 2 | employeeDensity | Number of employees per area | 1/km^2 | [0,inf) | 50000 |
| 6 | 2 | energyConsumption | Energy consumption per area | kW-h/km^2 | [0,inf) | 1250 |
| 7 | 3 | residentDensity | Number of residents per area | 1/km^2 | [0,inf) | 3000 |
| 8 | 3 | employeeDensity | Number of employees per area | 1/km^2 | [0,inf) | 30000 |
| 9 | 3 | energyConsumption | Energy consumption per area | kW-h/km^2 | [0,inf) | 2000 |
| 10 | 4 | residentDensity | Number of residents per area | 1/km^2 | [0,inf) | 0 |
| 11 | 4 | employeeDensity | Number of employees per area | 1/km^2 | [0,inf) | 100 |
| 12 | 4 | energyConsumption | Energy consumption per area | kW-h/km^2 | [0,inf) | 200 |

Next, we will be using node regions to define the nodes, but first let’s mark approximate coordinates of the vertices of the polygonal regions, as shown in Figure 13.

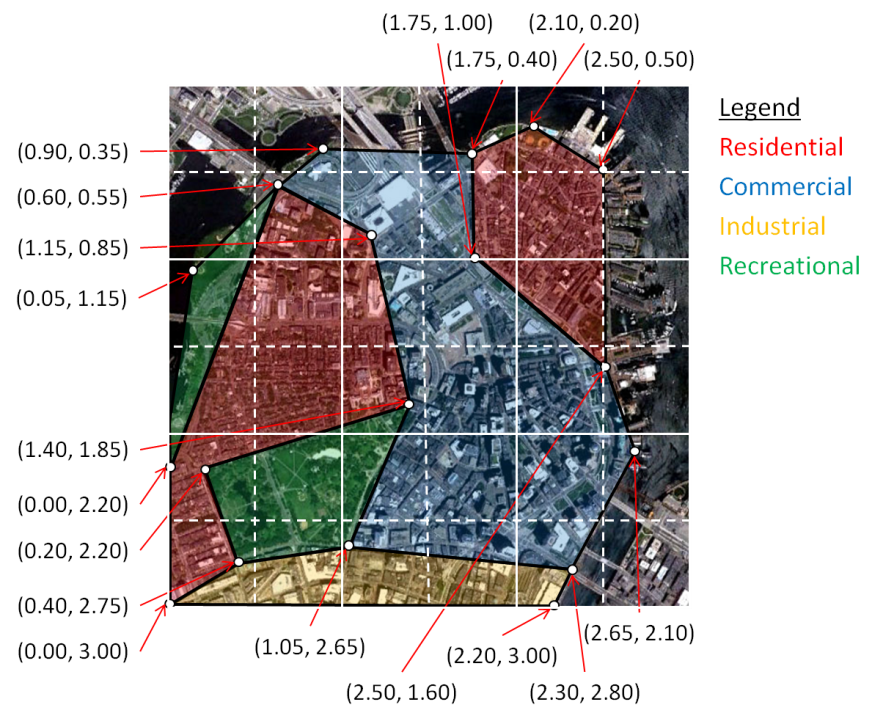


Figure : Coordinates of Building Node Region Vertices

There are a total of six regions in this sketch: two residential areas (Beacon Hill and North End), one commercial area (Downtown), one industrial area (South Boston), and two recreational areas (Boston Common and Public Garden and Esplanade).

1. Open the “**node\_regions**” worksheet
2. Add a row for Beacon Hill
   * Node Region ID: 1
   * System ID: 1
   * Node Type ID: 1
   * Layer ID: 1
   * VerticesX: [0.60 0.00 0.00 0.40 0.20 1.40 1.15]
   * VerticesY: [0.55 2.20 3.00 2.75 2.20 1.85 0.85]
   * Type: polygon
   * Description: Beacon Hill
3. Add a row for the North End
   * Node Region ID: 2
   * System ID: 1
   * Node Type ID: 1
   * Layer ID: 1
   * VerticesX: [1.75 1.75 2.50 2.50 2.10]
   * VerticesY: [0.40 1.00 1.60 0.50 0.20]
   * Type: polygon
   * Description: North End
4. Add a row for Downtown
   * Node Region ID: 3
   * System ID: 1
   * Node Type ID: 2
   * Layer ID: 1
   * VerticesX: [0.60 1.15 1.40 1.05 2.30 2.65 2.50 1.75 1.75 0.90]
   * VerticesY: [0.55 0.85 1.85 2.65 2.80 2.10 1.60 1.00 0.40 0.35]
   * Type: polygon
   * Description: Downtown
5. Add a row for South Boston
   * Node Region ID: 4
   * System ID: 1
   * Node Type ID: 3
   * Layer ID: 1
   * VerticesX: [0.00 2.20 2.30 1.05 0.40]
   * VerticesY: [3.00 3.00 2.80 2.65 2.75]
   * Type: polygon
   * Description: South Boston
6. Add a row for Boston Common and Public Garden
   * Node Region ID: 5
   * System ID: 1
   * Node Type ID: 4
   * Layer ID: 2
   * VerticesX: [0.20 0.40 1.05 1.40]
   * VerticesY: [2.20 2.75 2.65 1.85]
   * Type: polygon
   * Description: Boston Common and Public Garden
7. Add a row for the Esplanade
   * Node Region ID: 6
   * System ID: 1
   * Node Type ID: 4
   * Layer ID: 2
   * VerticesX: [0.60 0.05 0.00]
   * VerticesY: [0.55 1.15 2.20]
   * Type: polygon
   * Description: Esplanade

Table : Building System Node Regions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Node Region ID** | **System ID** | **Node Type ID** | **Layer ID** | **VerticesX** | **VerticesY** | **Type** | **Description** |
| 1 | 1 | 1 | 1 | [0.60 0.00 0.00 0.40 0.20 1.40 1.15] | [0.55 2.20 3.00 2.75 2.20 1.85 0.85] | polygon | Beacon Hill |
| 2 | 1 | 1 | 1 | [1.75 1.75 2.50 2.50 2.10] | [0.40 1.00 1.60 0.50 0.20] | polygon | North End |
| 3 | 1 | 2 | 1 | [0.60 1.15 1.40 1.05 2.30 2.65 2.50 1.75 1.75 0.90] | [0.55 0.85 1.85 2.65 2.80 2.10 1.60 1.00 0.40 0.35] | polygon | Downtown |
| 4 | 1 | 3 | 1 | [0.00 2.20 2.30 1.05 0.40] | [3.00 3.00 2.80 2.65 2.75] | polygon | South Boston |
| 5 | 1 | 4 | 2 | [0.20 0.40 1.05 1.40] | [2.20 2.75 2.65 1.85] | polygon | Boston Common and Public Garden |
| 6 | 1 | 4 | 2 | [0.60 0.05 0.00] | [0.55 1.15 2.20] | polygon | Esplanade |

Now that the node regions have been defined, let’s take a look at the resulting city model. First, however, we should specify that the nodes should be generated from the node regions.

1. Open the MATLAB script, **mainTutorial.m**, and add the following lines of code after the GenerateCells function call to generate the nodes contained within any node regions:  
   **synthTemp.GenerateNodes();**
2. Run the **mainTutorial.m** script; the figure generated should appear like that in.

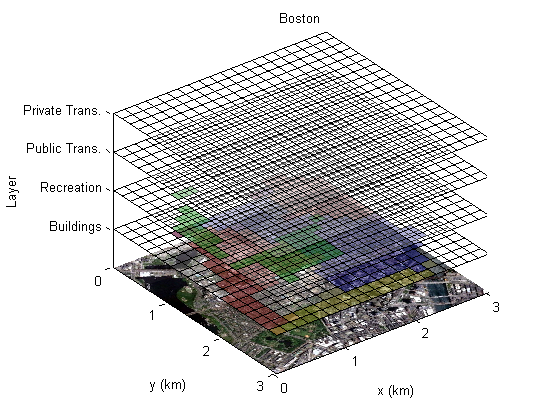


Figure : City Rendering with Building System Nodes

1. If you would like to generate images for the building and recreation layers alone, add the following commands:  
   **figure(2)  
   subplot(1,2,1)  
   synthTemp.RenderLayer(1);  
   subplot(1,2,2)  
   synthTemp.RenderLayer(2);**

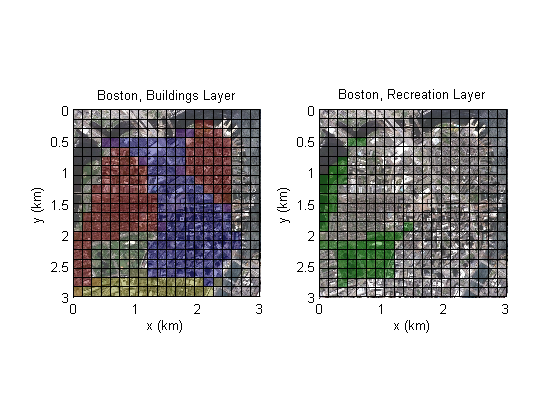


Figure : Buildings and Recreation Layers

1. Furthermore, if you would like to generate the boundaries of the actual node regions (useful for debugging), add the following commands:  
   figure(2)  
   subplot(1,2,1)  
   synthTemp.RenderLayer(1); **synthTemp.RenderNodeRegion2d(1);   
   synthTemp.RenderNodeRegion2d(2);   
   synthTemp.RenderNodeRegion2d(3);   
   synthTemp.RenderNodeRegion2d(4);**subplot(1,2,2)  
   synthTemp.RenderLayer(2); **synthTemp.RenderNodeRegion2d(5);   
   synthTemp.RenderNodeRegion2d(6);**

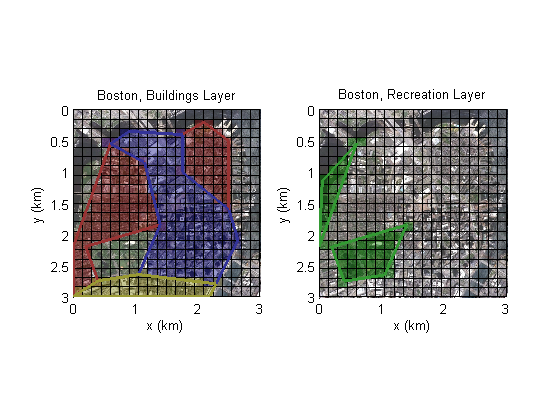


Figure : Buildings and Recreation Layers with Node Regions

### Transportation System Nodes

Similar to the building system, the transportation system also has several types of nodes to display.

1. Open the “**node\_types**” worksheet
2. Add a row for node type 5:
   * Node Type ID: 5
   * Name: **Roadway**
   * Description: Major automobile thoroughfares
   * Hex Color: 0x99999
3. Add a row for node type 6:
   * Node Type ID: 6
   * Name: **Red** **Metro Line**
   * Description: Selection of metro lines through Boston
   * Hex Color: 0xff9999
4. Add a row for node type 8:
   * Node Type ID: 7
   * Name: **Green** **Metro Line**
   * Description: Selection of metro lines through Boston
   * Hex Color: 0x99ff99
5. Add a row for node type 7:
   * Node Type ID: 8
   * Name: **Blue** **Metro Line**
   * Description: Selection of metro lines through Boston
   * Hex Color: 0x9999ff

The transportation system entries in the node type worksheet are shown below, in Table 18.

Table : Transportation System Node Types

|  |  |  |  |
| --- | --- | --- | --- |
| **Node Type ID** | **Name** | **Description** | **Hex Color** |
| 5 | Roadway | Major automobile thoroughfares | 0x333333 |
| 6 | Red Metro Line | MBTA red line | 0xff9999 |
| 7 | Green Metro Line | MBTA green line | 0x99ff99 |
| 8 | Blue Metro Line | MBTA blue line | 0x9999ff |

For each node type a set of node type attributes will be used as inputs for future evaluation and analysis. For this tutorial, we will greatly simplify the node type attributes to only a few for each node type such as: resident density, employee density, and energy consumption. Note that nearly all node type attributes should be defined on a per area basis.

1. Open the “**node\_type\_attributes**” worksheet
2. Add a row for the roadway node type (ID 5)
   * energyConsumption: 50 kW-h/km^2
3. Add a row for the red metro line node type (ID 6)
   * energyConsumption: 100 kW-h/km^2
4. Add a row for the green metro line node type (ID 7)
   * energyConsumption: 150 kW-h/km^2
5. Add a row for the blue metro line node type (ID 8)
   * energyConsumption: 75 kW-h/km^2

These steps are summarized in Table 21, below.

Table : Transportation System Node Type Attributes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Attribute ID** | **Node Type ID** | **Name** | **Description** | **Units** | **Bounds** | **Value** |
| 13 | 5 | energyConsumption | Annual energy consumption per area | 1/km^2 | [0,inf) | 50 |
| 14 | 6 | energyConsumption | Annual energy consumption per area | 1/km^2 | [0,inf) | 100 |
| 15 | 7 | energyConsumption | Annual energy consumption per area | kW-h/km^2 | [0,inf) | 150 |
| 16 | 8 | energyConsumption | Annual energy consumption per area | 1/km^2 | [0,inf) | 75 |

Next, we will be using node regions to define the nodes, but first let’s mark approximate coordinates of the vertices of the polyline regions, as shown in Figure 18. Note that the transportation system, rather than using polygonal regions like the building system, uses polyline regions to define nodes.

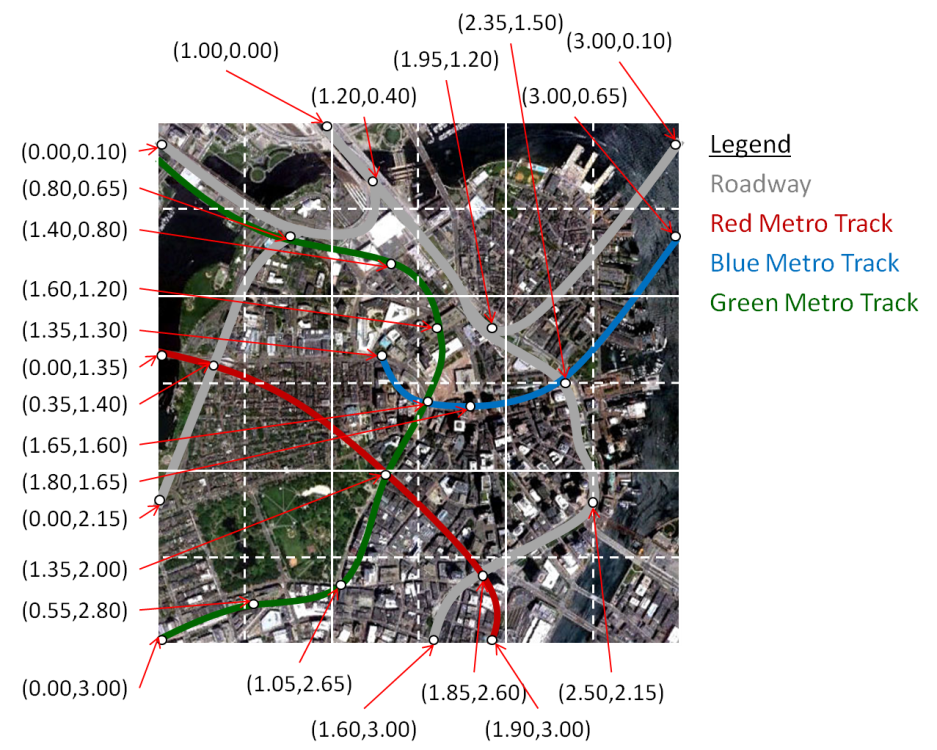


Figure : Coordinates of Transportation System Node Region Vertices

TODO: Add node region, synthesis script.

### Transportation System Edges

Also, we will consider four edge types within the transportation system.

1. Open the “**edge\_types**” worksheet
2. Add a row for edge type 1:
   * Edge Type ID: 1
   * Name: **Automobile**
   * Description: Private automobile traffic on major roadways
   * Hex Color: 0x333333
3. Add a row for edge type 2:
   * Edge Type ID: 2
   * Name: **Red Line Metro**
   * Description: Public metro service on red metro line
   * Hex Color: 0xff0000
4. Add a row for edge type 3:
   * Edge Type ID: 3
   * Name: **Green Line Metro**
   * Description: Public metro service on green metro line
   * Hex Color: 0x00ff00
5. Add a row for edge type 4:
   * Edge Type ID: 4
   * Name: **Blue Line Metro**
   * Description: Public metro service on blue metro line
   * Hex Color: 0x0000ff

The completed edge types worksheet is shown below, in Table 21.

Table : Transportation System Edge Types

|  |  |  |  |
| --- | --- | --- | --- |
| **Edge Type ID** | **Name** | **Description** | **Hex Color** |
| 1 | Automobile | Private automobile traffic on roadways | 0xffffff |
| 2 | Red Metro | Public metro service on red metro lines | 0xff0000 |
| 3 | Green Metro | Public metro service on green metro lines | 0x00ff00 |
| 4 | Blue Metro | Public metro service on blue metro lines | 0x0000ff |

For the four edge types, we will consider attributes including speed, capacity, and emissions.

TODO: Add edge type attributes, vertices map, edge regions, synthesis script.