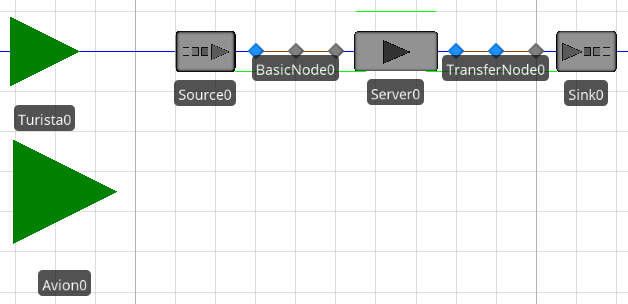
|  |  |
| --- | --- |
| ***Universidad de San Carlos de Guatemala***  ***Facultad de Ingeniería***  ***Laboratorio de Modelación y Simulación 1***  ***23/09/2020*** | Universidad de San Carlos de Guatemala - Wikipedia, la ... |

***Práctica 3***

|  |  |
| --- | --- |
| **Pareja No. 26** | |
| Ariel Alejandro Bautista Méndez | 201503910 |
| Melyza Alejandra Rodríguez Contreras | 201314821 |

1. **Justificación del modelo base**



El modelo base está compuesto por los diferentes objetos utilizados para la realización de la práctica, se agregó un solo elemento como ejemplo para el modelo final generado con la API de SIMIO.

Lo objetos agregados son:

1. Server
2. Source
3. Sink
4. Basic Node
5. Transfer Node
6. Path
7. Conveyor

Entidades utilizadas

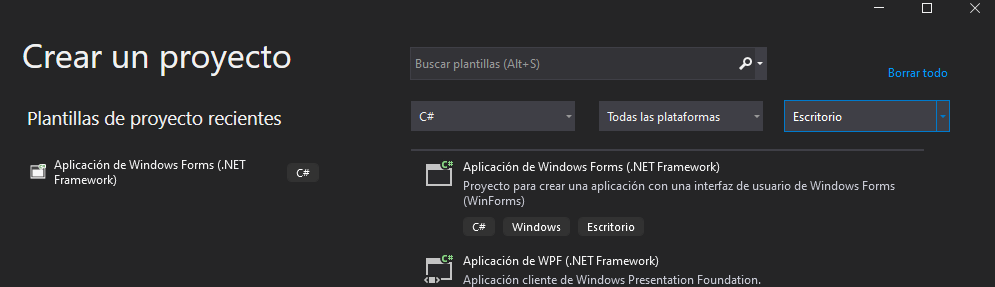
|  |  |
| --- | --- |
| **Entidad** | **Descripción** |
| **Turista** | Representa a los turistas que se mueven dentro del territorio nacional, entre regiones y que pueden llegar a Guatemala a través de un aeropuerto o bien ser turistas nacionales. |
| **Avión** | Representa a los aviones pertenecientes a las fuerzas armadas, estos sobrevuelan el territorio nacional manteniendo el orden en las fronteras marítimas y terrestres de Guatemala. |

La manera en que son empleados cada uno de los objetos y entidades listados anteriormente se explicará posteriormente.

1. **Pasos para la implementación de la API de SIMIO**

**Creación de un proyecto de C#**

Inicialmente se debe crear un proyecto para trabajar la API, en este caso fue creado utilizando el lenguaje C#. Se trabajo con el IDE Visual Studio en su versión Community 2019.



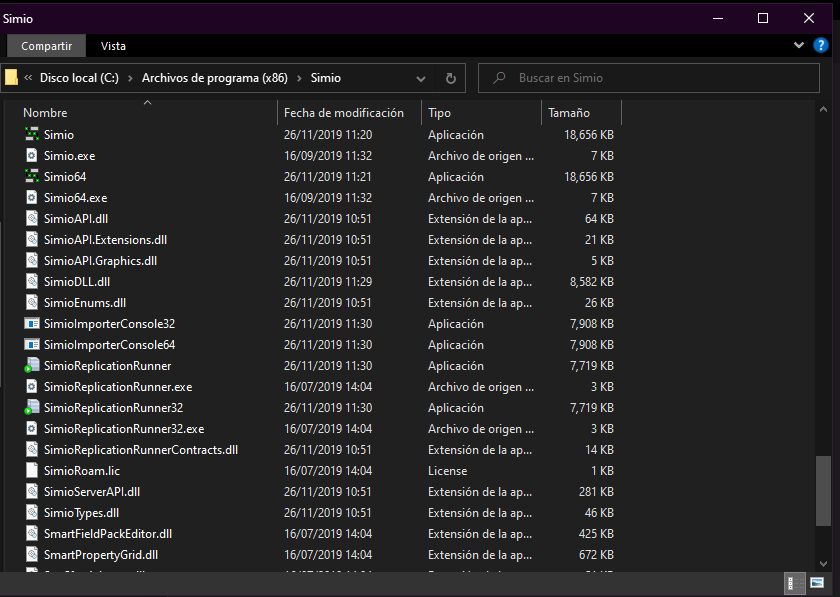
**Configuración de librerías**

Para la utilización de la API de SIMIO, se debe incluir las siguientes DLL.

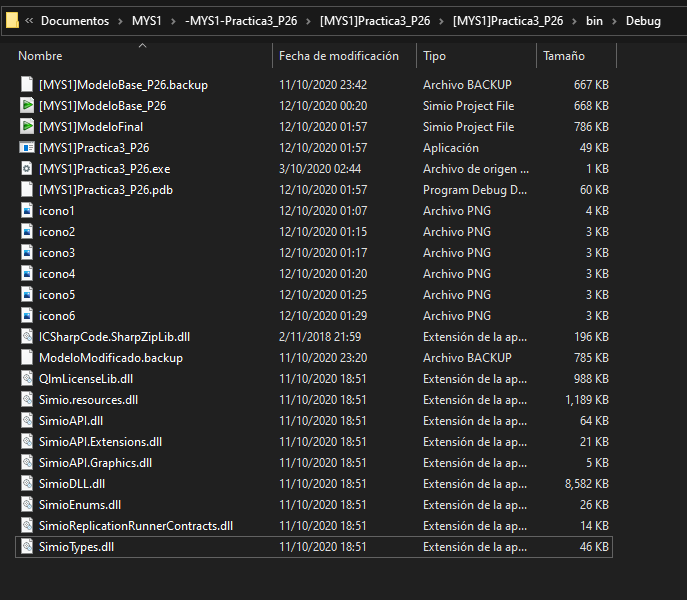
1. ICSharpCode.SharpZipLib
2. QlmLicenseLib
3. Simio.resources
4. SimioAPI
5. SimioAPI.Extensions
6. SimioAPI.Graphics
7. SimioDLL
8. SimioEnums
9. SimioReplicationRunnerContracts
10. SimioTypes

Estas DLL pueden ser encontradas en la carpeta de instalación de SIMIO, es importante que se utilicen las librerías que se encuentran en dicha carpeta ya que están asociadas directamente con la instalación que tenemos en nuestra computadora, esto nos evitará problemas de compatibilidad y licencia.

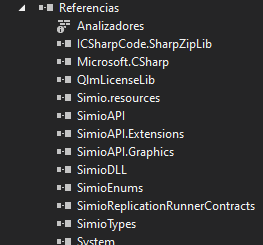
A continuación, se puede observar el directorio de instalación de SIMIO en la cual podemos encontrar las DLL listadas



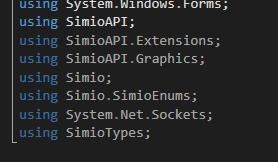
Una vez que se tengan localizadas las librerías, se deben copiar en la carpeta Debug, ubicada dentro de la carpeta Bin de nuestro proyecto de C#.



Además, se debe agregar una referencia a estas nuevas librerías dentro de nuestro proyecto.

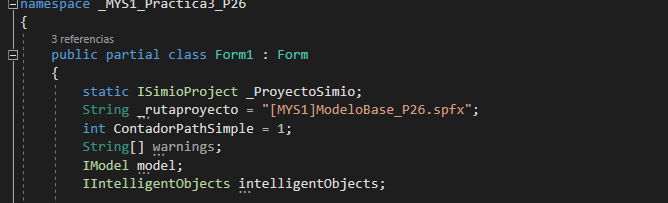


Finalmente, debemos agregar los imports correspondientes para poder acceder a los métodos de las librerías añadidas en nuestro proyecto.



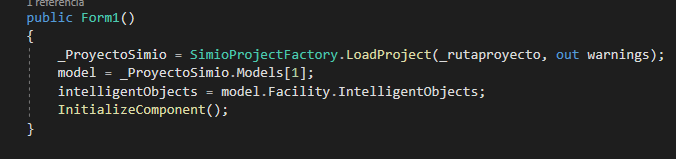
**Declaración de componentes a utilizar**

Se deben establecer los objetos o componentes que utilizaremos para formar nuestro modelo final.

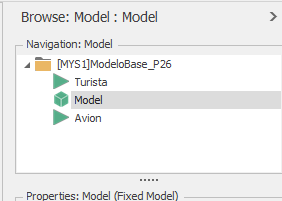


* Se debe declarar un objeto de tipo ISimioProyect, este representa el proyecto en el cual añadiremos los diferentes objetos que conformarán nuestro modelo final.
* Así mismo, se declara la ruta en la que se encuentra el modelo base que servirá como ejemplo de los componentes utilizados.
* Adicionalmente, podemos agregar contadores para manejar el número de objetos generados ya que la API no lleva control del mismo.
* Además, se debe crear un arreglo que permita almacenar las advertencias generadas al crear el modelo.
* También se debe declarar un objeto de tipo IModel, este será nuestro tapiz para agregar objetos.
* Por último, se debe declarar un objeto de tipo IIntelligentObjects, este almacenará cada uno de los objetos que generemos.

Después de realizar la declaración de los objetos a utilizar, se debe realizar lo siguiente.



* Se debe cargar el proyecto creado anteriormente, indicándole la ruta y el contenedor de advertencias.
* Así mismo, se debe asignar el modelo a utilizar, en este caso asignamos el valor 1, haciendo referencia a la posición del modelo en nuestro modelo base.



* Por último, asignamos nuestro contenedor de objetos al modelo seleccionado.

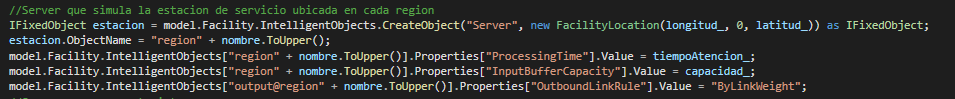
**Creación de objetos**

El objetivo de la utilización de la API de SIMIO es crear componentes mediante código para luego formar un modelo, para esto se debe generar los componentes, indicando el tipo y la ubicación en la que este será ubicado dentro del tapiz que representa el modelo.

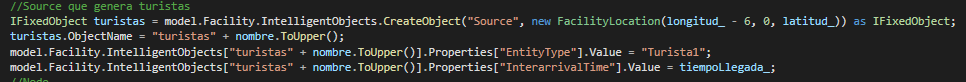
Así mismo, se pueden editar sus propiedades para asignar comportamientos a cada uno de los objetos generados.

A continuación, se muestra como crear dichos objetos.

Creación de un objeto de tipo Server.



Creación de un objeto de tipo Source.



Creación de un objeto de tipo Basic Node.



Creación de un objeto de tipo Transfer Node.

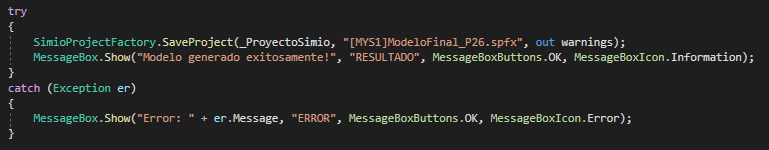


Creación de un enlace de tipo Path.



**Generación del modelo final**

Cuando se ha terminado de agregar objetos se procede a crear el modelo final, para ello se debe usar la propiedad SaveProject, indicando la ruta y el nombre en que será generado dicho modelo.



En caso de que todo esté correcto, se indica que el modelo se ha generado con éxito, en caso contrario, se muestran los errores existentes para su corrección.

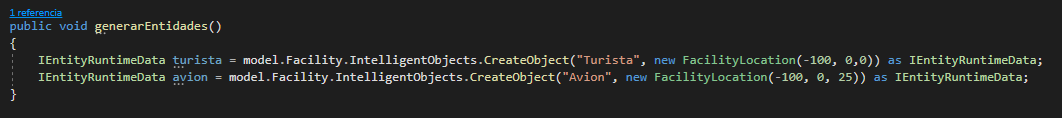
1. **Descripción de métodos**

**generarEntidades()**

No recibe parámetros.

Este método genera dos entidades, una de tipo Avión, representando a las naves de las fuerzas armadas, la cual será utilizada como tipo de entidad del objeto SOURCE que representa la base militar.

La segunda entidad que se genera es de tipo Turista, esta será utilizada como tipo de entidad que generan los objetos SOURCE ubicados en cada una de las regiones, así mismo, también será empleada como tipo de entidad para el objeto SOURCE que representa a cada uno de los aeropuertos existentes. Esta entidad se moverá dentro del territorio nacional entre regiones.



**crearRegion()**

Los parámetros que recibe son:

* Nombre, de tipo STRING, representa el nombre de la región.
* Longitud y Latitud, de tipo INTEGER, representan la ubicación de la región.
* Capacidad, de tipo INTEGER, representa la capacidad de atención que tiene la estación de servicio de la región.
* TiempoLlegada, de tipo INTEGER, representa el tiempo entre llegadas de turistas.
* UnidadTiempo, de tipo STRING, es la unidad de tiempo para la llegada de turistas.
* TiempoAtención, de tipo INTEGER, representa el tiempo que tarda un turista en la estación de servicio de la región.

Este método se encarga de generar cada una de las regiones existentes en el territorio nacional, inicialmente genera un objeto de tipo SERVER, este tendrá el nombre de la región a la que representa.

Se modifican sus propiedades de:

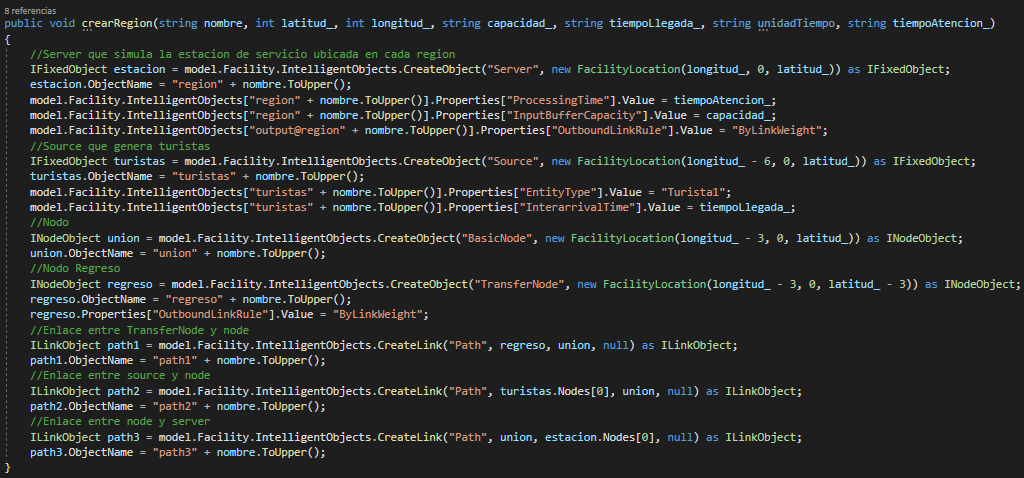
* Tiempo de procesamiento
* Capacidad
* Selección por peso de su nodo de salida

Así mismo se crea un objeto de tipo SOURCE que generará los turistas que viajarán dentro del territorio nacional, también son modificadas sus propiedades.

* Tipo de entidad que genera.
* Tiempo entre llegadas

Se crean dos nodos, uno de tipo TRANSFER NODE que representa el punto en el que convergen los turistas que regresan a la región y otro de tipo BASIC NODE, en este se reúnen los turistas procedentes del extranjero y los que proceden del territorio nacional.

Posteriormente se enlazan los diferentes elementos.

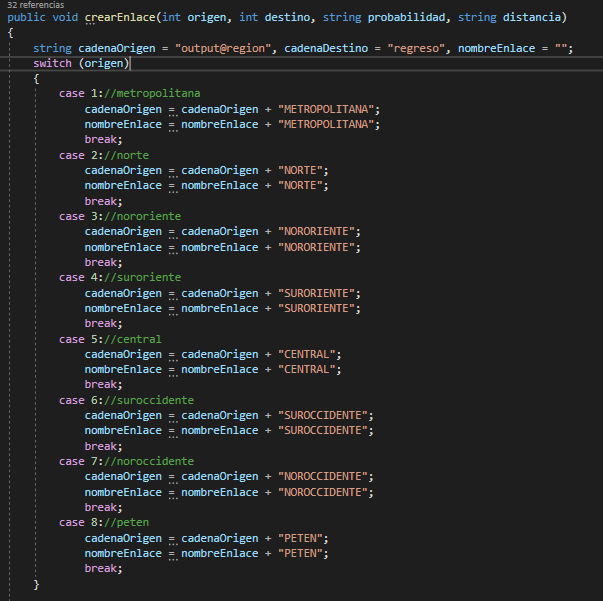


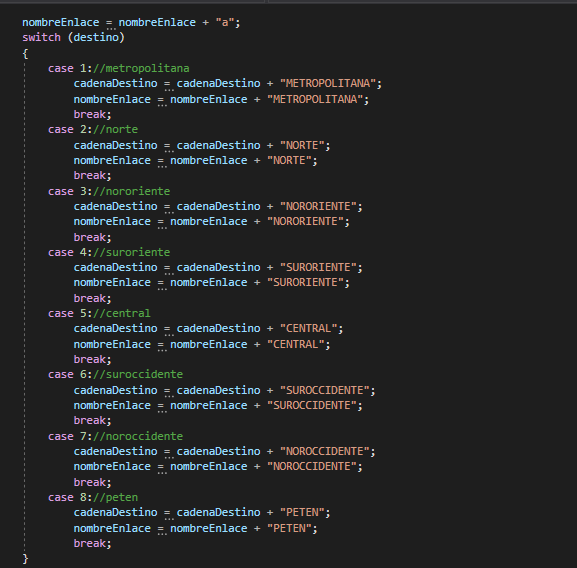
**crearEnlace()**

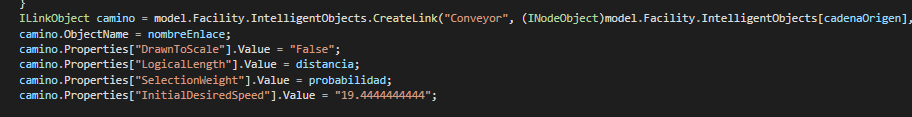
Los parámetros que recibe son:

* Origen, de tipo INTEGER, representa la región desde donde parte el enlace.
* Destino, de tipo INTEGER, representa la región hacia donde se dirige el enlace.
* Probabilidad, de tipo INTEGER, presenta el peso que se le asignará al enlace.
* Distancia, de tipo INTEGER, representa la longitud lógica del enlace en metros.

La funcionalidad de este método es comunicar los diferentes elementos que conformar las regiones del territorio nacional, así mismo, sirve para que las entidades de tipo Turista viajen entre dichas regiones.







**crearAeropuerto()**

Los parámetros que recibe son:

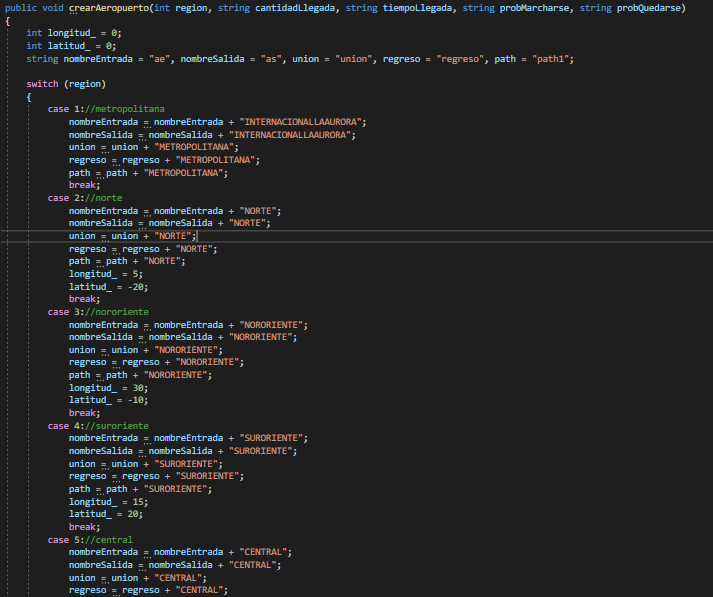
* Región, de tipo INTEGER, representa la región en la que se encuentra el aeropuerto.
* CantidadLlegada, de tipo INTEGER, representa la cantidad de personas que llegan en un vuelo.
* TiempoLlegada, de tipo STRING, representa el tiempo en minutos entre la llegada de los diferentes vuelos.
* ProbMarcharse, de tipo INT, representa la probabilidad que existe de que un turista decida marcharse del territorio nacional cuando llega a una región que posee un aeropuerto.
* ProbQuedarse, de tipo INT, representa la probabilidad que existe de que un turista decida quedarse en el territorio nacional cuando llega a una región que posee un aeropuerto.

Este método modela los aeropuertos existentes en el territorio nacional, para ellos se implementó un objeto de tipo SOURCE, que genera los turistas que llegan al país. Se modificaron sus propiedades como se indica a continuación, utilizando los parámetros correspondientes:

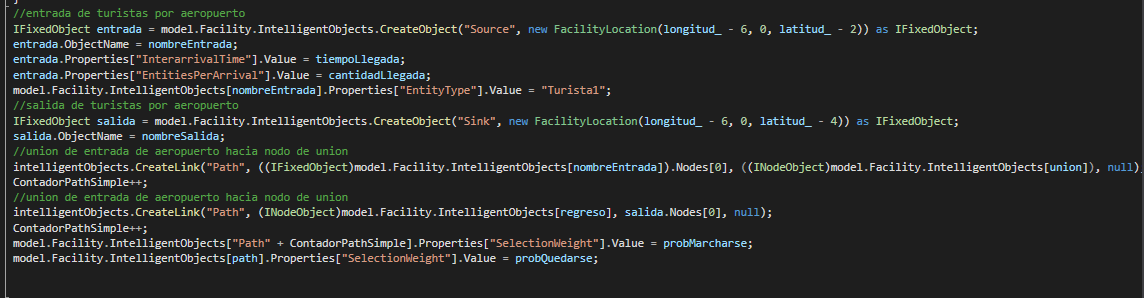
* Tiempo entre llegadas.
* Tipo de entidad que genera.
* Cantidad de entidades por arribo.

Así mismo, se crea un objeto de tipo SINK, que representa un medio de salida para los turistas que deciden marcharse del país.

Finalmente se conectan los diferentes componentes mediante enlaces de tipo PATH, los cuales poseen los pesos correspondientes a las probabilidades indicadas.







**Dibujar#()**

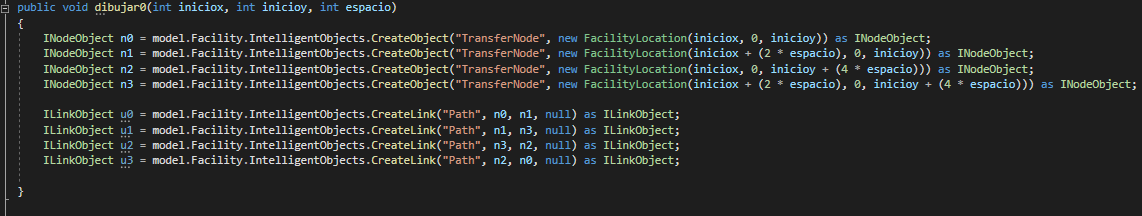
Los parámetros que recibe son:

* InicioX, representa la coordenada x en la que comienza el número que se desea dibujar.
* InicioY, representa la coordenada y en la que comienza el número que se desea dibujar.
* Espacio, representa el espacio de separación entre los vértices del número.

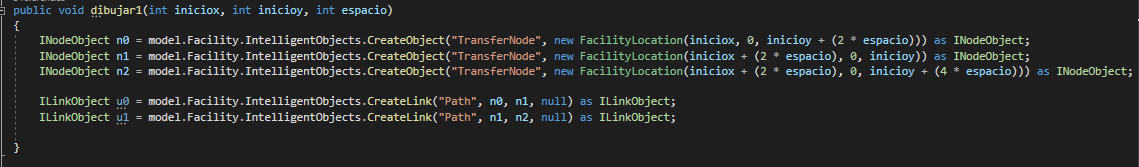
Este método va del 0 – 9. Es la representación de los números dígitos que servirán posteriormente para formar el carné de cada uno de los integrantes. Para ello se crean nodos de tipo TRANSFER NODE que representan los vértices que tienen los números correspondientes, estos nodos se conectan a través de enlaces de tipo PATH creando la forma del número.

A continuación, se muestran ejemplos de los métodos para dibujar los diferentes números.

Dibujar0()



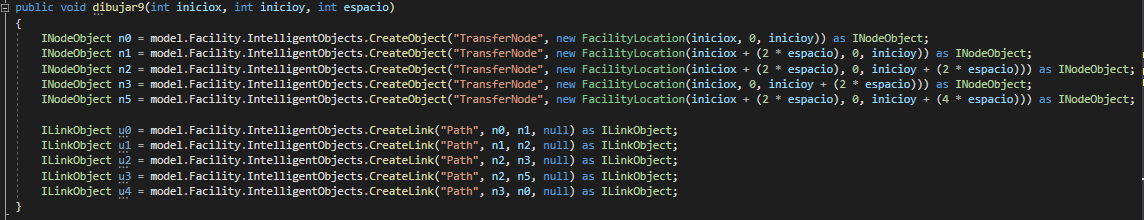
Dibujar1()



Dibujar8()



Dibujar9()

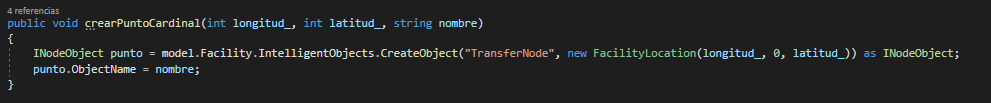


**crearPuntoCardinal()**

Los parámetros que recibe son:

* Longitud y Latitud, de tipo INTEGER, representa la ubicación del punto cardinal.
* Nombre, de tipo STRING, representa el nombre del punto cardinal.

Este método genera un punto cardinal que servirá de guía para saber hacia dónde se dirigen los turistas dentro del territorio nacional, para ello se creo un objeto de tipo TRANSFER NODE en la ubicación y nombre descritos por los parámetros.



**pintarMapa()**

El método en cuestión no recibe parámetros.

Este método es utilizado para graficar el contorno del mapa de guatemala, para ello, va colocando BasicNode’s en coordenadas determinadas a modo de representar la silueta de dicho mapa. El método inicia pintando dos nodos en la ubicación indicada, y luego estos son unidos mediante un Conveyor, posteriormente, se modifican las propiedades “DrawnToScale”, “InitialDesiredSpeed” y “LogicalLength” del Conveyor, el primero se coloca en “False” para poder colocarle una longitud lógica y que no la dibuje a escala según la distancia real que separa a ambos nodos. La segunda propiedad modificada obliga a las entidades que se desplacen por él a recorrerlo con una velocidad específia, en este caso 16.67 m/s que equivalen a 60 km/h. Y el último parámetro modificado coloca una longitud lógica al conveyor, en este caso es diferente para cada arista ya que se nos dio la longitud real entre frontera y frontera, y dado que el modelo se trató de dibujar lo más a escala posible se trató la manera de hacer una distribución uniforme de estas distancias.

Las distancias relativas pueden apreciarse en las siguientes tablas:

**Frontera Con El Pacífico**

|  |  |  |
| --- | --- | --- |
| **Arista** | **L. Real (m)** | **L. Lógica (m)** |
| v2 -> v3 | 19.65 | 56671.97 |
| v3 -> v4 | 32.06 | 92463.27 |
| v4 -> v5 | 36.36 | 104864.77 |
| **TOTAL** | 88.07 | 254000 |

**Frontera Con México**

|  |  |  |
| --- | --- | --- |
| **Arista** | **L. Real (m)** | **L. Lógica (m)** |
| v5 -> v6 | 15.23 | 54701.54 |
| v6 -> v7 | 12.81 | 46009.63 |
| v7 -> v8 | 36.06 | 129516.58 |
| v8 -> v9 | 52 | 186768.22 |
| v9 -> v10 | 12.53 | 45003.96 |
| v10 -> v11 | 10.82 | 38862.16 |
| v11 -> v12 | 11.18 | 40155.17 |
| v12 -> v13 | 19.21 | 68996.49 |
| v13 -> v14 | 15 | 53875.45 |
| v14 -> v15 | 13 | 46692.05 |
| v15 -> v16 | 70 | 251418.76 |
| **TOTAL** | **267.84** | **962000** |

**Frontera con Belice**

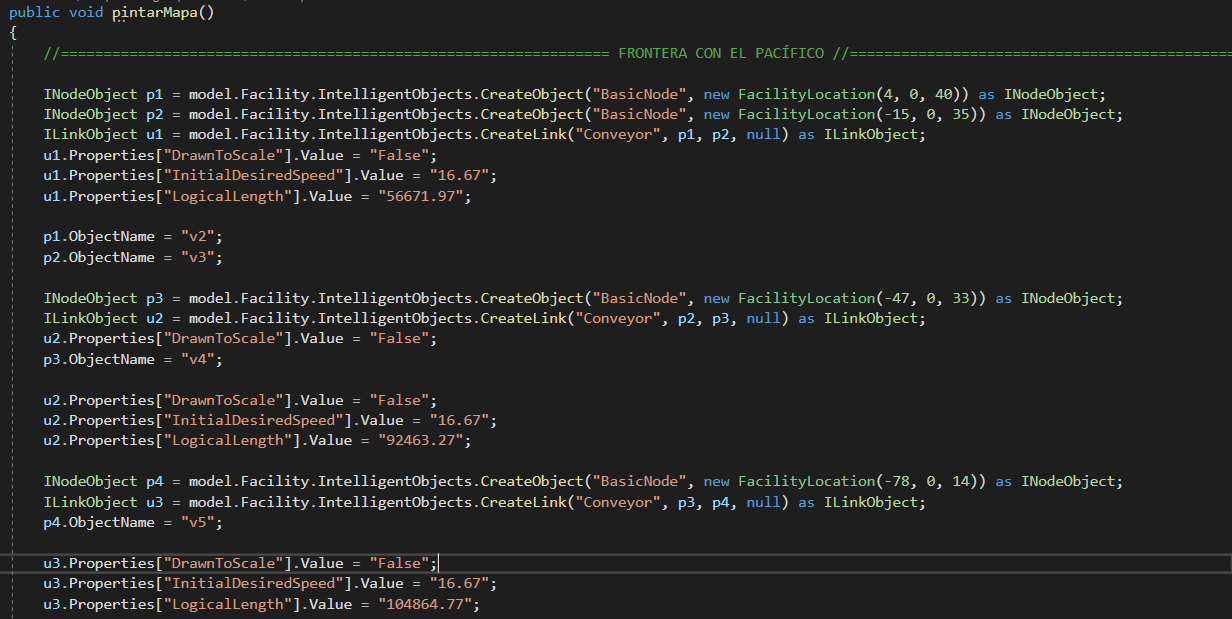
|  |  |  |
| --- | --- | --- |
| **Arista** | **L. Real (m)** | **L. Lógica (m)** |
| v16 -> v17 | 59 | 200947.50 |
| v17 -> v18 | 19.1 | 65052.50 |
| **TOTAL** | **78.1** | **266000** |

**Frontera Con El Atlántico**

|  |  |  |
| --- | --- | --- |
| **Arista** | **L. Real (m)** | **L. Lógica (m)** |
| v18 -> v19 | 17.49 | 25415.02 |
| v19 -> v20 | 3.16 | 4591.85 |
| v20 -> v21 | 2.83 | 4112.32 |
| v21 -> v22 | 3.16 | 4591.85 |
| v22 -> v23 | 2.83 | 4112.32 |
| v23 -> v24 | 3.61 | 5245.75 |
| v24 -> v25 | 3.16 | 4591.85 |
| v25 -> v26 | 3.61 | 5245.75 |
| v26 -> v27 | 10.44 | 15170.54 |
| v27 -> v28 | 4.12 | 5986.84 |
| v28 -> v29 | 10.82 | 15722.73 |
| v29 -> v30 | 5 | 7265.59 |
| v30 -> v31 | 2.83 | 4112.32 |
| v31 -> v32 | 4.24 | 6161.22 |
| v32 -> v33 | 5.1 | 7410.90 |
| v33 -> v34 | 7.21 | 10476.98 |
| v34 -> v35 | 2.24 | 3254.98 |
| v35 -> v36 | 10 | 14531.17 |
| **TOTAL** | **101.85** | **148000** |

**Frontera con Honduras**

|  |  |  |
| --- | --- | --- |
| **Arista** | **L. Real (m)** | **L. Lógica (m)** |
| v36 -> v37 | 36.24 | 129212.26 |
| v37 -> v38 | 7.62 | 27168.80 |
| v38 -> v39 | 11.18 | 39861.84 |
| v39 -> v40 | 16.76 | 59757.10 |
| **TOTAL** | **71.8** | **256000** |

****

**Frontera Con El Salvador**

|  |  |  |
| --- | --- | --- |
| **Arista** | **L. Real (m)** | **L. Lógica (m)** |
| v40 -> v41 | 6.08 | 35889.50 |
| v41 -> v42 | 17 | 100348.94 |
| v42 -> v2 | 11.31 | 66761.56 |
| **TOTAL** | **34.39** | **203000** |

****

****

****

****

****

****

****

****

****

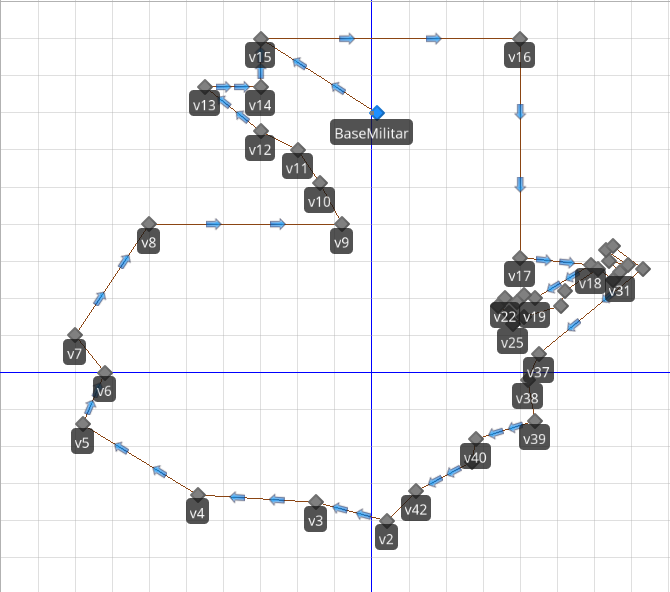
Como se puede apreciar en las capturas se modificaron las mismas tres propiedades en cada uno de los conveyor creados. Y finalmente se crea un Source al que se le coloca por nombre “BaseMilitar” y se cambia el valor de las propiedades “InterarrivalTime”, “MaximumArrivals” y “EntityType”, al primero se le coloca el valor de 15, para que llegue un avión cada 15 minutos al sistema, a la segunda propiedad se le coloca también el valor de 15 para indicarle que genere un máximo de 15 aviones, el último parámetro coloca el tipo de entidad que manejará el Source, en este caso el avión como tal.

1. **Evolución del modelo final**

**Creación del contorno del mapa y la base militar**

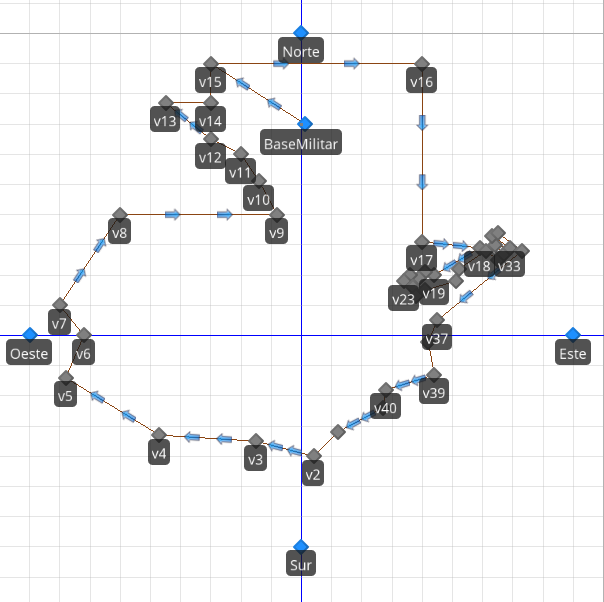
Se crea el contorno del mapa de Guatemala mediante nodos de tipo BASIC NODE, conectados por medio de enlaces CONVEYOR, estos enlaces poseen una longitud lógica determinada que representa la longitud de las diferentes fronteras marítimas y terrestres del país.

Así mismo, se crea la base militar ubicada en la selva de Petén, esta base militar está representada con un objeto SOURCE que genera entidades de tipo AVION que representan las naves de las fuerzas armadas, actualmente se cuenta con 15 aviones, los cuales tienen un tiempo de separación de 15 minutos.



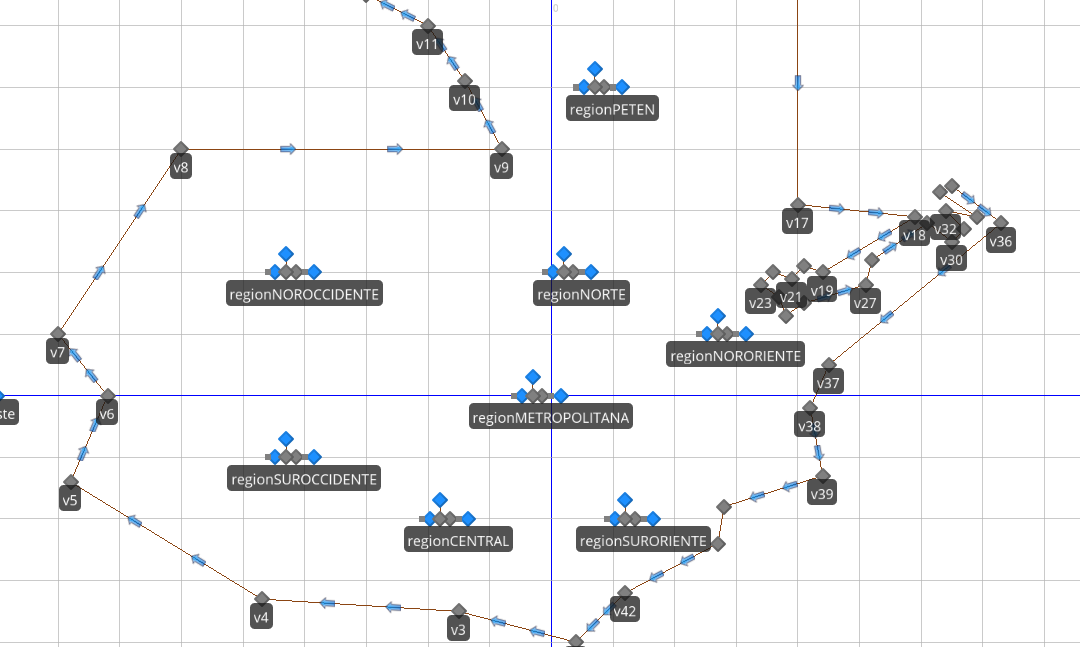
**Creación de puntos cardinales**

Se crean los puntos cardinales por medio de objetos TRANSFER NODE, estos sirven de guía para conocer el rumbo hacia el cual se mueven los turistas dentro del territorio nacional.



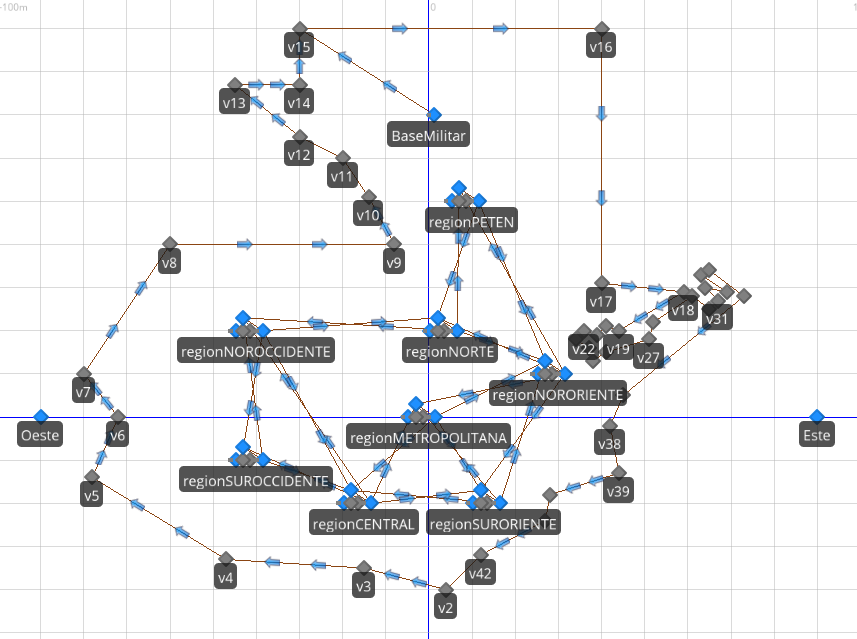
**Creación de regiones**

Se crean las diferentes regiones del territorio nacional, mismas que están compuestas por una estación de servicio representada con un objeto SERVER, un objeto SOURCE, que genera turistas que viajan entre regiones y dos nodos en los cuales convergen los turistas que llegan a una región determinada. Se crean 7 regiones diferentes.



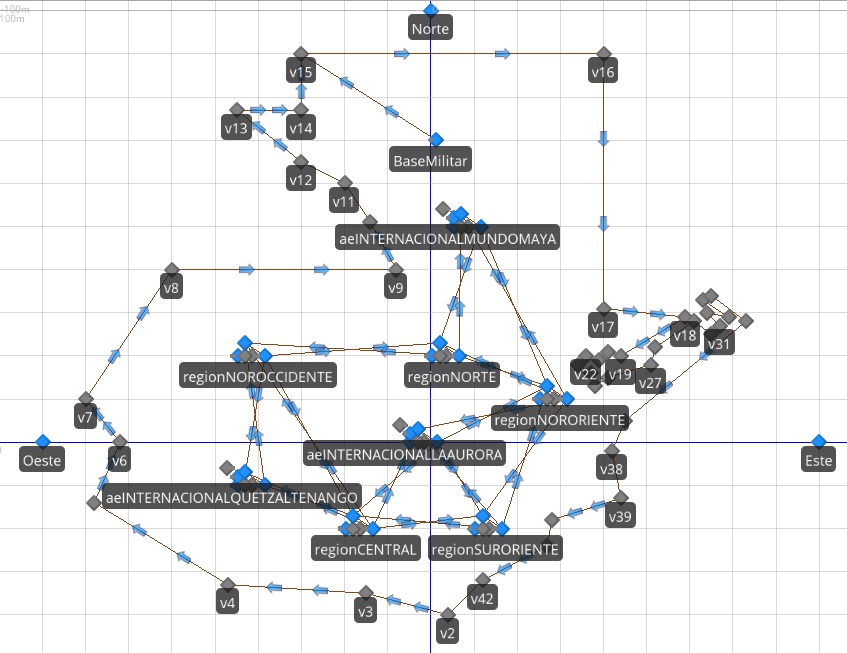
**Enlaces entre regiones**

Se conectan las regiones mediante objetos de tipo CONVEYOR, la salida de la estación de servicio de una región determinada se conecta con el nodo de regreso en la región de destino, se implementó la selección por peso en el nodo de salida de la estación para poder insertar las probabilidades que tienen los turistas de dirigirse a una región en particular.



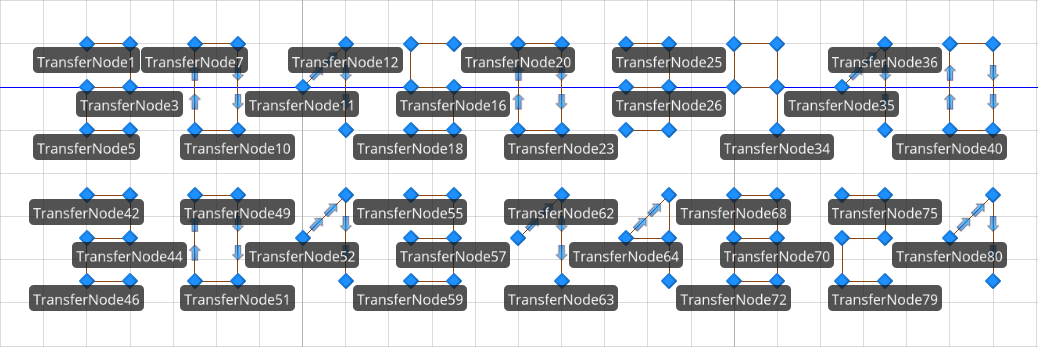
**Creación de aeropuertos**

Se crean los aeropuertos existentes dentro del territorio nacional, estos sirven como medio de llegada para los turistas internacionales y como punto de salida para los turistas que deciden marcharse del territorio nacional. Se implementó la selección por peso para determinar los turistas que se quedan en la región y los que se marchan del país.



**Creación de carnés**

Se dibujan los carnés de los integrantes por medio de objetos de tipo TRANSFER NODE conectados por medio de enlaces PATH, estos forman los diferentes números.

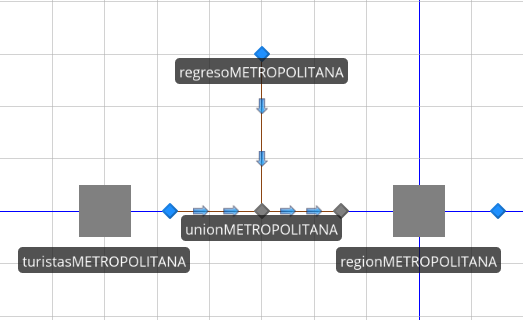


1. **Diseño del modelo final**

**Regiones**

Para modelar las regiones, se implementó lo siguiente:

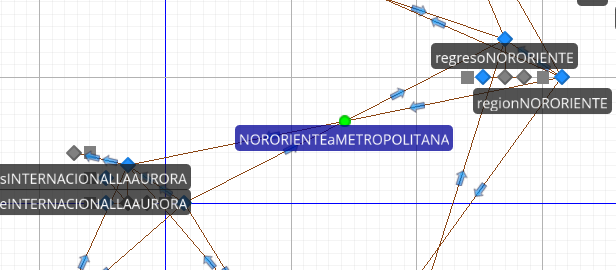
* Un objeto de tipo SOURCE para generar los turistas que viajan entre regiones, el nombre de este es “turistas” acompañado de el nombre de la región en la cual se encuentra. El tiempo en minutos entre llegadas está dado por una distribución de Poisson.
* Un objeto de tipo SERVER representando la estación de servicio con un tiempo de procesamiento definido por una distribución exponencial dada por el enunciado de trabajo, así como la capacidad de turistas que posee cada estación. El nombre que identifica la región es “región” acompañado de el nombre de la región en la cual se encuentra ubicado.
* Un nodo de tipo BASIC NODE que representa un punto de unión entre la salida del objeto SOURCE que genera los turistas nacionales, la salida del objeto SOURCE que genera los turistas internaciones (explicado posteriormente) y los turistas que ingresan a la región provenientes de otra región y deciden quedarse en el territorio nacional.
* Un nodo de tipo TRANSFER NODE que sirve de punto de ingreso a una región cuando los turistas provienen de otra región. En este nodo se implementó selección por peso debido a que es en este punto en que los turistas deciden si se quedan dentro del territorio nacional o se marchan del país.
* Se implementó también selección por peso en el nodo de salida del objeto SERVER que representa la estación de servicio de la región, esto debido a que en este punto los turistas deciden hacia qué región quieren viajar.



**Enlaces entre regiones**

Para conectar las diferentes regiones se utilizaron enlaces de tipo CONVEYOR debido a que se indica la distancia que los turistas deben recorre para llegar desde una región hacia otra, así mismo se indica la velocidad con que estos lo hacen.

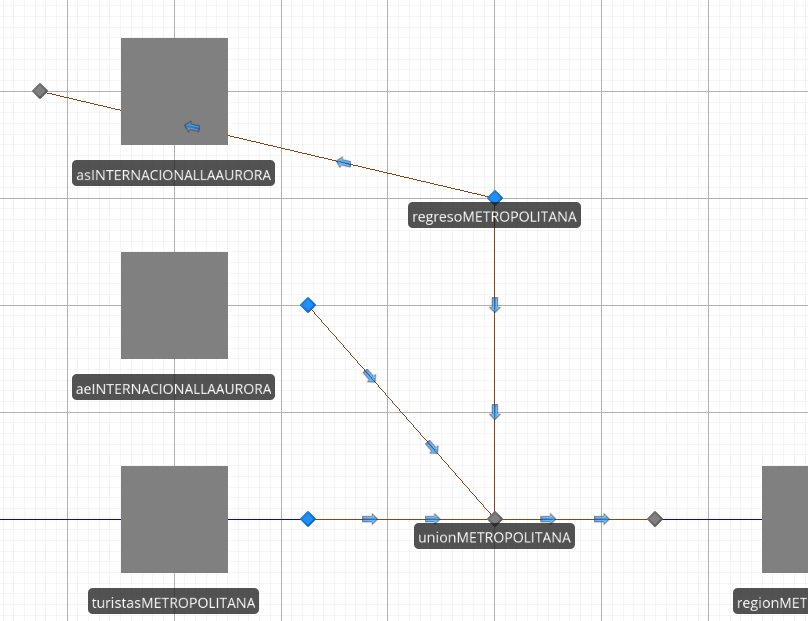
El nombre que identifica a cada uno de los enlaces se compone de la región origen, la palabra “a” y la región destino del turista. La longitud lógica del enlace es la distancia mencionada anteriormente, así como la velocidad.



**Aeropuertos**

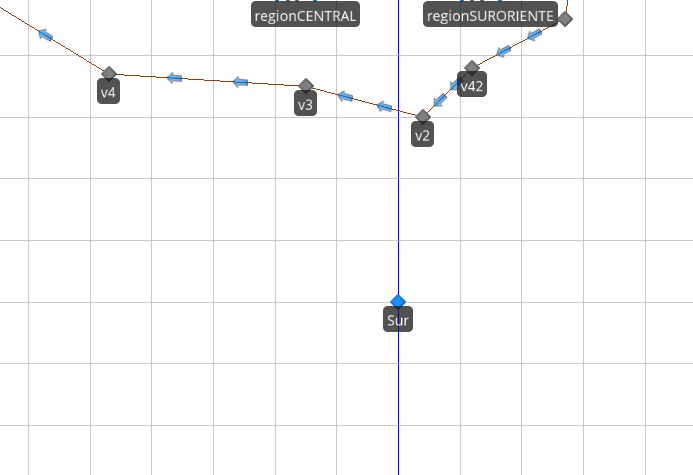
Para los aeropuertos se implementó lo siguiente:

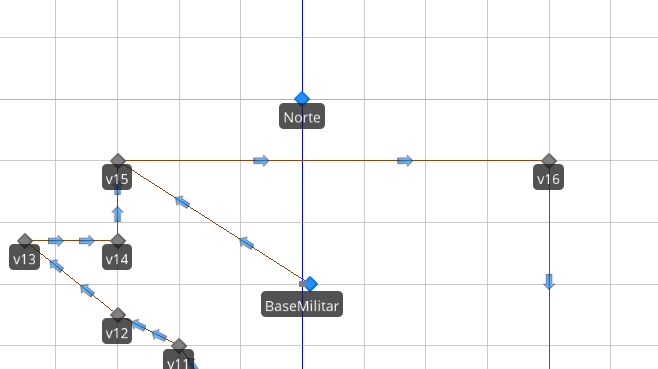
* Un objeto de tipo SOURCE que se encarga de generar los turistas internacionales que llegan al territorio nacional, se modificó el tiempo entre llegadas, mismo que está dado por una distribución exponencial. Así mismo, se modificó la propiedad de entidades por llegada, se colocó la cantidad indicada por el enunciado. La salida de este objeto está conectada mediante un enlace de tipo PATH al nodo de unión de la región en la que se encuentra ubicado el aeropuerto.
* Se creó también un objeto de tipo SINK que representa la salida del territorio nacional, este objeto está conectado desde el nodo de regreso de la región por medio de un enlace de tipo PATH, el cual posee el peso correspondiente a la probabilidad de que los turistas decidan marcharse del país.



**Puntos cardinales**

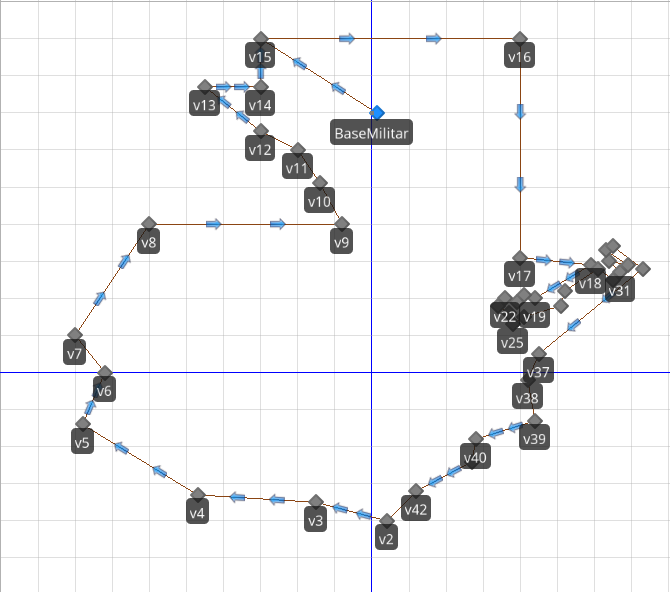
Los puntos cardinales fueron representados por medio de nodos de tipo TRANSFER NODE ya que fue dejado a discreción del estudiante y a que dichos puntos cardinales solamente sirven como guía para indicar el rumbo de los turistas dentro del país. Estos nodos no poseen enlaces.





**Contorno del Mapa**

Para dibujar el contorno del mapa se utilizaron BasicNodes como vértices para el grafo, y Conveyors como aristas, dado que la única función de los vértices es darle forma a la silueta se utilizaron estos tipos de componentes (BasicNode’s), y los Conveyors se utilizaron para delimitar una velocidad de desplazamiento sobre las aristas.



**Base Militar**

Para la base militar se utilizó un Source ya que se simulará el lugar de despegue de 15 aviones que al final serán entidades que ingresarán cada 15 minutos al sistema.

1. **Conclusiones**

* Los modelos de simulación ayudan en gran medida para predecir comportamientos en base a datos recopilados previamente, para este caso, se analizó el desplazamiento de personas por centros turísticos, importantes del país, algo que puede ayudar a las autoridades locales a preparar medidas para manejar la aglomeración de personas.
* Simio ha demostrado ser una potente herramienta de simulación que con el uso de la API hace más fácil la integración de modelos de simulación con lenguajes de programación populares, en este caso C#, potenciando enormemente la gama de posibilidades que tenemos al trabajar con estas herramientas.
* Dependiendo de la forma en que se integre la API de Simio con el lenguaje de programación deseado, puede darnos una ventaja al momento de crear un modelo de simulación, ya que por ejemplo mediante una sentencia de ciclo for o while se podrían crear una cantidad considerable de componentes en cuestión de segundos.
* La creación de modelos mediante archivos de entrada implementando un mini compilador podría reducir enormemente el trabajo y tiempo que se requiere para implementar un modelo de simulación