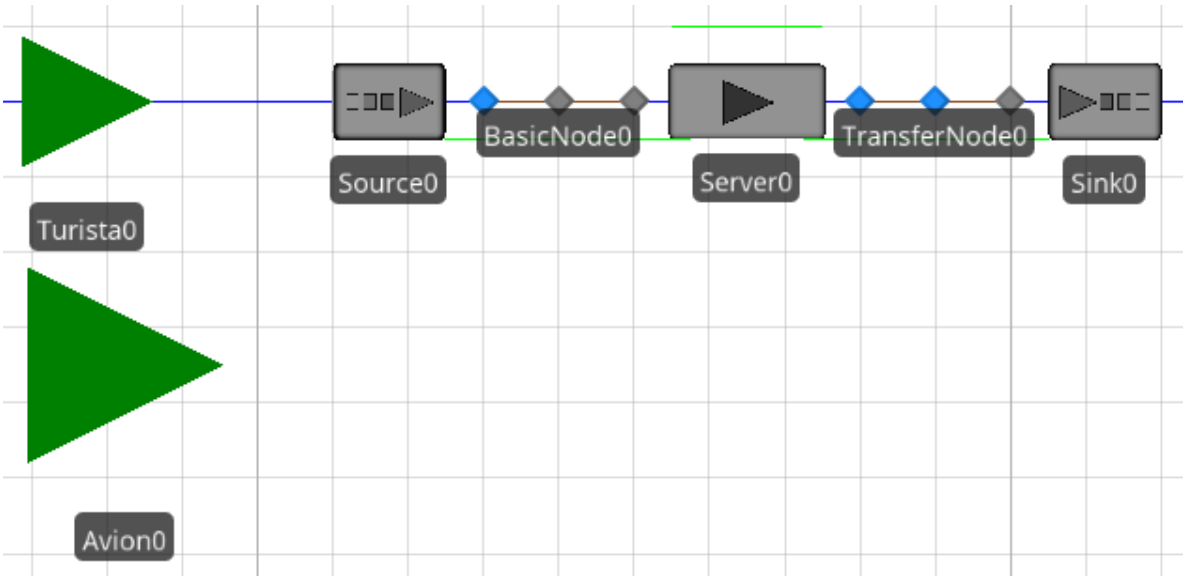
Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Laboratorio de Modelación y Simulación 1
23/09/2020

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

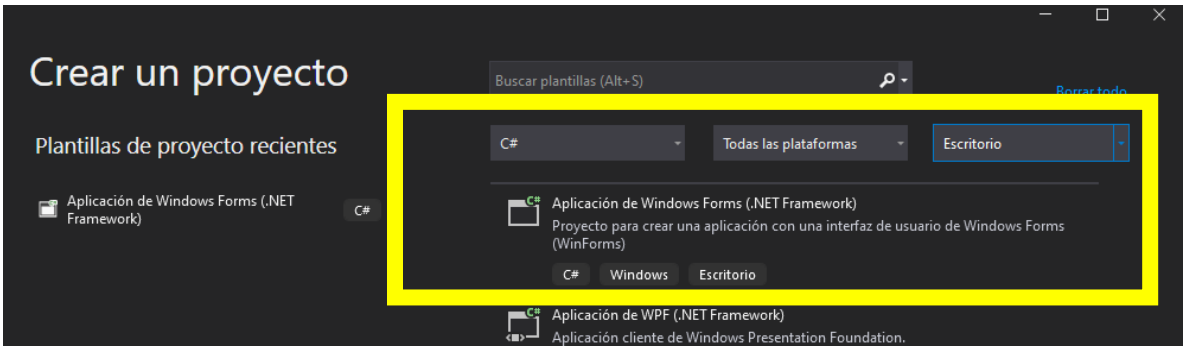
Entidad	Descripción
Turista	Representa a los turistas que se mueven dentro del territorio nacional, entre regiones y que pueden llegar a Guatemala a través de un aeropuerto o bien ser turistas nacionales.
Avión	Representa a los aviones pertenecientes a las fuerzas armadas, estos sobrevuelan el territorio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

2- 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.



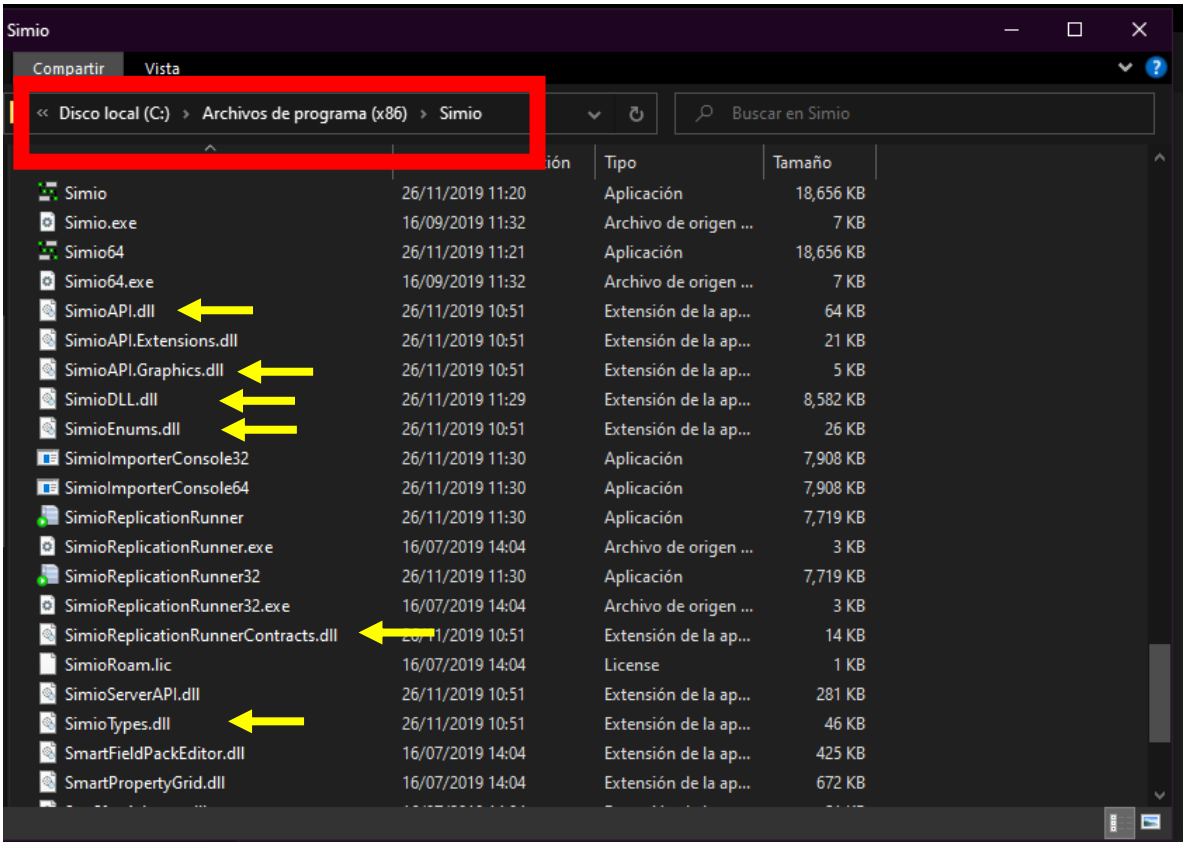
Configuración de librerías

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

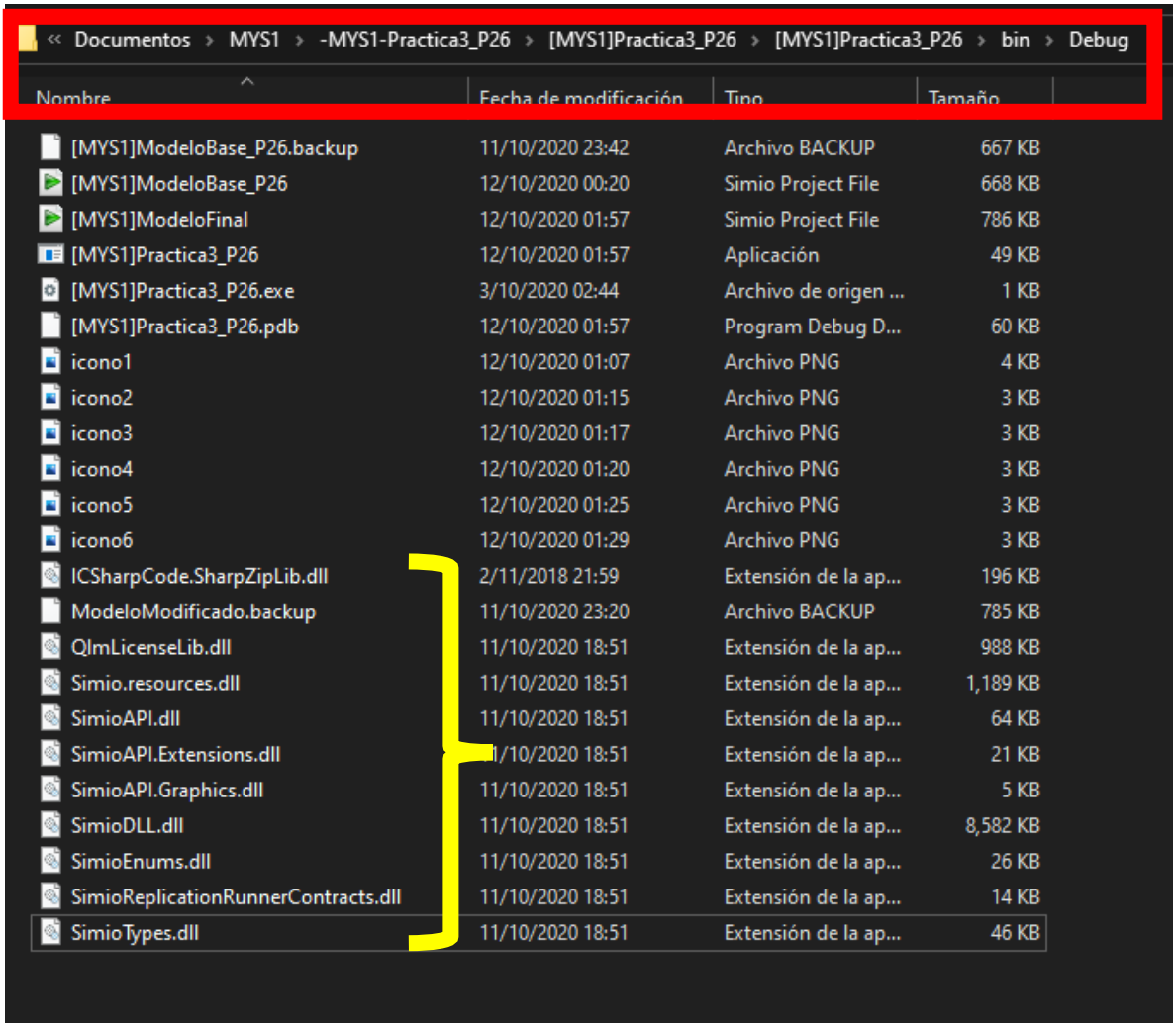
1- ICSharpCode.SharpZipLib
2- QlmLicenseLib
3- Simio.resources
4- SimioAPI
5- SimioAPI.Extensions
6- SimioAPI.Graphics
7- SimioDLL
8- SimioEnums
9- SimioReplicationRunnerContracts
10- SimioTypes

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.

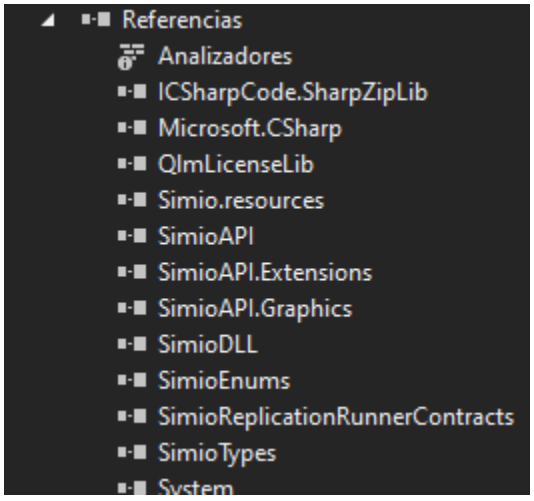
A continuación, se puede observar el directorio de instalación de SIMIO en la cual podemos encontrar las DLL listadas



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.

```
using System.Windows.Forms;
using SimioAPI;
using SimioAPI.Extensions;
using SimioAPI.Graphics;
using Simio;
using Simio.SimioEnums;
using System.Net.Sockets;
using SimioTypes;
```

Declaración de componentes a utilizar

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

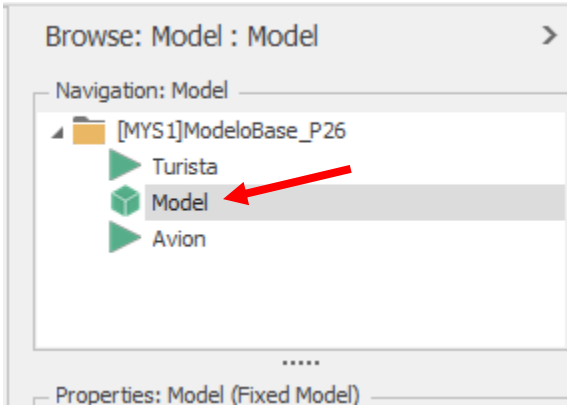
```
namespace _MYS1_Practica3_P26
{
    3 referencias
    public partial class Form1 : Form
    {
        static ISimioProject _ProyectoSimio;
        String _rutaproyecto = "[MYS1]ModeloBase_P26.spfx";
        int ContadorPathSimple = 1;
        String[] warnings;
        IModel model;
        IIntelligentObjects intelligentObjects;
```

- 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.

```
1 referencia
public Form1()
{
    _ProyectoSimio = SimioProjectFactory.LoadProject(_rutaproyecto, out warnings);
    model = _ProyectoSimio.Models[1];
    intelligentObjects = model.Facility.IntelligentObjects;
    InitializeComponent();
}
```

- 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.

```
//Server que simula la estacion de servicio ubicada en cada region
IFixedObject estacion = model.Facility.IntelligentObjects.CreateObject("Server", new FacilityLocation(longitud_, 0, latitud_)) as IFixedObject;
estacion.ObjectName = "region" + nombre.ToUpper();
model.Facility.IntelligentObjects["region" + nombre.ToUpper()].Properties["ProcessingTime"].Value = tiempoAtencion_;
model.Facility.IntelligentObjects["region" + nombre.ToUpper()].Properties["InputBufferCapacity"].Value = capacidad_;
model.Facility.IntelligentObjects["output@region" + nombre.ToUpper()].Properties["OutboundLinkRule"].Value = "ByLinkWeight";
```

Creación de un objeto de tipo Source.

```
//Source que genera turistas
IFixedObject turistas = model.Facility.IntelligentObjects.CreateObject("Source", new FacilityLocation(longitud_ - 6, 0, latitud_)) as IFixedObject;
turistas.ObjectName = "turistas" + nombre.ToUpper();
model.Facility.IntelligentObjects["turistas" + nombre.ToUpper()].Properties["EntityType"].Value = "Turista1";
model.Facility.IntelligentObjects["turistas" + nombre.ToUpper()].Properties["InterarrivalTime"].Value = tiempoLlegada_;
```

Creación de un objeto de tipo Basic Node.

```
//Nodo
INodeObject union = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(longitud_ - 3, 0, latitud_)) as INodeObject;
union.ObjectName = "union" + nombre.ToUpper();
```

Creación de un objeto de tipo Transfer Node.

```
INodeObject regreso = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(longitud_ - 3, 0, latitud_ - 3)) as INodeObject;
regreso.ObjectName = "regreso" + nombre.ToUpper();
regreso.Properties["OutboundLinkRule"].Value = "ByLinkWeight";
```

Creación de un enlace de tipo Path.

```
//Enlace entre TransferNode y node
ILinkObject path1 = model.Facility.IntelligentObjects.CreateLink("Path", regreso, union, null) as ILinkObject;
path1.ObjectName = "path1" + nombre.ToUpper();
```

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.

```
try
{
    SimioProjectFactory.SaveProject(_ProyectoSimio, "[MYS1]ModeloFinal_P26.spfx", out warnings);
    MessageBox.Show("Modelo generado exitosamente!", "RESULTADO", MessageBoxButtons.OK, MessageBoxIcon.Information);
}
catch (Exception er)
{
    MessageBox.Show("Error: " + er.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

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.

3- 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.

```
1 referencia
public void generarEntidades()
{
    IEntityRuntimeData turista = model.Facility.IntelligentObjects.CreateObject("Turista", new FacilityLocation(-100, 0,0)) as IEntityRuntimeData;
    IEntityRuntimeData avion = model.Facility.IntelligentObjects.CreateObject("Avion", new FacilityLocation(-100, 0, 25)) as IEntityRuntimeData;
}
```

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.

```
8 referencias
public void crearRegion(string nombre, int latitud_, int longitud_, string capacidad_, string tiempoLlegada_, string unidadTiempo, string tiempoAtencion_)
{
    //Server que simula la estacion de servicio ubicada en cada region
    IFixedObject estacion = model.Facility.IntelligentObjects.CreateObject("Server", new FacilityLocation(longitud_, 0, latitud_)) as IFixedObject;
    estacion.ObjectName = "region" + nombre.ToUpper();
    model.Facility.IntelligentObjects["region" + nombre.ToUpper()].Properties["ProcessingTime"].Value = tiempoAtencion_;
    model.Facility.IntelligentObjects["region" + nombre.ToUpper()].Properties["InputBufferCapacity"].Value = capacidad_;
    model.Facility.IntelligentObjects["output@region" + nombre.ToUpper()].Properties["OutboundLinkRule"].Value = "ByLinkWeight";
    //Source que genera turistas
    IFixedObject turistas = model.Facility.IntelligentObjects.CreateObject("Source", new FacilityLocation(longitud_ - 6, 0, latitud_)) as IFixedObject;
    turistas.ObjectName = "turistas" + nombre.ToUpper();
    model.Facility.IntelligentObjects["turistas" + nombre.ToUpper()].Properties["EntityType"].Value = "Turista1";
    model.Facility.IntelligentObjects["turistas" + nombre.ToUpper()].Properties["InterarrivalTime"].Value = tiempoLlegada_;
    //Nodo
    INodeObject union = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(longitud_ - 3, 0, latitud_)) as INodeObject;
    union.ObjectName = "union" + nombre.ToUpper();
    //Nodo Regreso
    INodeObject regreso = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(longitud_ - 3, 0, latitud_ - 3)) as INodeObject;
    regreso.ObjectName = "regreso" + nombre.ToUpper();
    regreso.Properties["OutboundLinkRule"].Value = "ByLinkWeight";
    //Enlace entre TransferNode y node
    ILinkObject path1 = model.Facility.IntelligentObjects.CreateLink("Path", regreso, union, null) as ILinkObject;
    path1.ObjectName = "path1" + nombre.ToUpper();
    //Enlace entre source y node
    ILinkObject path2 = model.Facility.IntelligentObjects.CreateLink("Path", turistas.Nodes[0], union, null) as ILinkObject;
    path2.ObjectName = "path2" + nombre.ToUpper();
    //Enlace entre node y server
    ILinkObject path3 = model.Facility.IntelligentObjects.CreateLink("Path", union, estacion.Nodes[0], null) as ILinkObject;
    path3.ObjectName = "path3" + nombre.ToUpper();
}
```

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.

```
32 referencias
public void crearEnlace(int origen, int destino, string probabilidad, string distancia)
{
    string cadenaOrigen = "output@region", cadenaDestino = "regreso", nombreEnlace = "";
    switch (origen)
    {
        case 1://metropolitana
            cadenaOrigen = cadenaOrigen + "METROPOLITANA";
            nombreEnlace = nombreEnlace + "METROPOLITANA";
            break;
        case 2://norte
            cadenaOrigen = cadenaOrigen + "NORTE";
            nombreEnlace = nombreEnlace + "NORTE";
            break;
        case 3://nororiente
            cadenaOrigen = cadenaOrigen + "NORORIENTE";
            nombreEnlace = nombreEnlace + "NORORIENTE";
            break;
        case 4://suroriente
            cadenaOrigen = cadenaOrigen + "SURORIENTE";
            nombreEnlace = nombreEnlace + "SURORIENTE";
            break;
        case 5://central
            cadenaOrigen = cadenaOrigen + "CENTRAL";
            nombreEnlace = nombreEnlace + "CENTRAL";
            break;
        case 6://suroccidente
            cadenaOrigen = cadenaOrigen + "SUROCCIDENTE";
            nombreEnlace = nombreEnlace + "SUROCCIDENTE";
            break;
        case 7://noroccidente
            cadenaOrigen = cadenaOrigen + "NOROCCIDENTE";
            nombreEnlace = nombreEnlace + "NOROCCIDENTE";
            break;
        case 8://peten
            cadenaOrigen = cadenaOrigen + "PETEN";
            nombreEnlace = nombreEnlace + "PETEN";
            break;
    }
```

```
nombreEnlace = nombreEnlace + "a";
switch (destino)
{
    case 1://metropolitana
        cadenaDestino = cadenaDestino + "METROPOLITANA";
        nombreEnlace = nombreEnlace + "METROPOLITANA";
        break;
    case 2://norte
        cadenaDestino = cadenaDestino + "NORTE";
        nombreEnlace = nombreEnlace + "NORTE";
        break;
    case 3://nororiente
        cadenaDestino = cadenaDestino + "NORORIENTE";
        nombreEnlace = nombreEnlace + "NORORIENTE";
        break;
    case 4://suroriente
        cadenaDestino = cadenaDestino + "SURORIENTE";
        nombreEnlace = nombreEnlace + "SURORIENTE";
        break;
    case 5://central
        cadenaDestino = cadenaDestino + "CENTRAL";
        nombreEnlace = nombreEnlace + "CENTRAL";
        break;
    case 6://suroccidente
        cadenaDestino = cadenaDestino + "SUROCCIDENTE";
        nombreEnlace = nombreEnlace + "SUROCCIDENTE";
        break;
    case 7://noroccidente
        cadenaDestino = cadenaDestino + "NOROCCIDENTE";
        nombreEnlace = nombreEnlace + "NOROCCIDENTE";
        break;
    case 8://peten
        cadenaDestino = cadenaDestino + "PETEN";
        nombreEnlace = nombreEnlace + "PETEN";
        break;
}
```

```
}
ILinkObject camino = model.Facility.IntelligentObjects.CreateLink("Conveyor", (INodeObject)model.Facility.IntelligentObjects[cadenaOrigen],
camino.ObjectName = nombreEnlace;
camino.Properties["DrawnToScale"].Value = "False";
camino.Properties["LogicalLength"].Value = distancia;
camino.Properties["SelectionWeight"].Value = probabilidad;
camino.Properties["InitialDesiredSpeed"].Value = "19.4444444444";
```

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.

```
public void crearAeropuerto(int region, string cantidadLlegada, string tiempoLlegada, string probMarcharse, string probQuedarse)
{
    int longitud_ = 0;
    int latitud_ = 0;
    string nombreEntrada = "ae", nombreSalida = "as", union = "union", regreso = "regreso", path = "path1";

    switch (region)
    {
        case 1://metropolitana
            nombreEntrada = nombreEntrada + "INTERNACIONALLAAURORA";
            nombreSalida = nombreSalida + "INTERNACIONALLAAURORA";
            union = union + "METROPOLITANA";
            regreso = regreso + "METROPOLITANA";
            path = path + "METROPOLITANA";
            break;
        case 2://norte
            nombreEntrada = nombreEntrada + "NORTE";
            nombreSalida = nombreSalida + "NORTE";
            union = union + "NORTE";
            regreso = regreso + "NORTE";
            path = path + "NORTE";
            longitud_ = 5;
            latitud_ = -20;
            break;
        case 3://nororiente
            nombreEntrada = nombreEntrada + "NORORIENTE";
            nombreSalida = nombreSalida + "NORORIENTE";
            union = union + "NORORIENTE";
            regreso = regreso + "NORORIENTE";
            path = path + "NORORIENTE";
            longitud_ = 30;
            latitud_ = -10;
            break;
        case 4://suroriente
            nombreEntrada = nombreEntrada + "SURORIENTE";
            nombreSalida = nombreSalida + "SURORIENTE";
            union = union + "SURORIENTE";
            regreso = regreso + "SURORIENTE";
            path = path + "SURORIENTE";
            longitud_ = 15;
            latitud_ = 20;
            break;
        case 5://central
            nombreEntrada = nombreEntrada + "CENTRAL";
            nombreSalida = nombreSalida + "CENTRAL";
            union = union + "CENTRAL";
            regreso = regreso + "CENTRAL";
```

```
        longitud_ = -15;
        latitud_ = 20;
        break;
    case 6://suroccidente
        nombreEntrada = nombreEntrada + "INTERNACIONALQUETZALTENANGO";
        nombreSalida = nombreSalida + "INTERNACIONALQUETZALTENANGO";
        union = union + "SUROCCIDENTE";
        regreso = regreso + "SUROCCIDENTE";
        path = path + "SUROCCIDENTE";
        longitud_ = -40;
        latitud_ = 10;
        break;
    case 7://noroccidente
        nombreEntrada = nombreEntrada + "NOROCCIDENTE";
        nombreSalida = nombreSalida + "NOROCCIDENTE";
        union = union + "NOROCCIDENTE";
        regreso = regreso + "NOROCCIDENTE";
        path = path + "NOROCCIDENTE";
        longitud_ = -40;
        latitud_ = -20;
        break;
    case 8://peten
        nombreEntrada = nombreEntrada + "INTERNACIONALMUNDOMAYA";
        nombreSalida = nombreSalida + "INTERNACIONALMUNDOMAYA";
        union = union + "PETEN";
        regreso = regreso + "PETEN";
        path = path + "PETEN";
        longitud_ = 10;
        latitud_ = -50;
        break;
}
```

```
//entrada de turistas por aeropuerto
IFixedObject entrada = model.Facility.IntelligentObjects.CreateObject("Source", new FacilityLocation(longitud_ - 6, 0, latitud_ - 2)) as IFixedObject;
entrada.ObjectName = nombreEntrada;
entrada.Properties["InterarrivalTime"].Value = tiempoLlegada;
entrada.Properties["EntitiesPerArrival"].Value = cantidadLlegada;
model.Facility.IntelligentObjects[nombreEntrada].Properties["EntityType"].Value = "Turista1";
//salida de turistas por aeropuerto
IFixedObject salida = model.Facility.IntelligentObjects.CreateObject("Sink", new FacilityLocation(longitud_ - 6, 0, latitud_ - 4)) as IFixedObject;
salida.ObjectName = nombreSalida;
//union de entrada de aeropuerto hacia nodo de union
intelligentObjects.CreateLink("Path", ((IFixedObject)model.Facility.IntelligentObjects[nombreEntrada]).Nodes[0], ((INodeObject)model.Facility.IntelligentObjects[union]), null);
ContadorPathSimple++;
//union de entrada de aeropuerto hacia nodo de union
intelligentObjects.CreateLink("Path", (INodeObject)model.Facility.IntelligentObjects[regreso], salida.Nodes[0], null);
ContadorPathSimple++;
model.Facility.IntelligentObjects["Path" + ContadorPathSimple].Properties["SelectionWeight"].Value = probMarcharse;
model.Facility.IntelligentObjects[path].Properties["SelectionWeight"].Value = probQuedarse;
```

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()

```
public void dibujar0(int iniciox, int inicioy, int espacio)
{
    INodeObject n0 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox, 0, inicioy)) as INodeObject;
    INodeObject n1 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox + (2 * espacio), 0, inicioy)) as INodeObject;
    INodeObject n2 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox, 0, inicioy + (4 * espacio))) as INodeObject;
    INodeObject n3 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox + (2 * espacio), 0, inicioy + (4 * espacio))) as INodeObject;

    ILinkObject u0 = model.Facility.IntelligentObjects.CreateLink("Path", n0, n1, null) as ILinkObject;
    ILinkObject u1 = model.Facility.IntelligentObjects.CreateLink("Path", n1, n3, null) as ILinkObject;
    ILinkObject u2 = model.Facility.IntelligentObjects.CreateLink("Path", n3, n2, null) as ILinkObject;
    ILinkObject u3 = model.Facility.IntelligentObjects.CreateLink("Path", n2, n0, null) as ILinkObject;

}
```

Dibujar1()

```
public void dibujar1(int iniciox, int inicioy, int espacio)
{
    INodeObject n0 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox, 0, inicioy + (2 * espacio))) as INodeObject;
    INodeObject n1 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox + (2 * espacio), 0, inicioy)) as INodeObject;
    INodeObject n2 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox + (2 * espacio), 0, inicioy + (4 * espacio))) as INodeObject;

    ILinkObject u0 = model.Facility.IntelligentObjects.CreateLink("Path", n0, n1, null) as ILinkObject;
    ILinkObject u1 = model.Facility.IntelligentObjects.CreateLink("Path", n1, n2, null) as ILinkObject;

}
```

Dibujar8()

```
public void dibujar8(int iniciox, int inicioy, int espacio)
{
    INodeObject n0 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox, 0, inicioy)) as INodeObject;
    INodeObject n1 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox + (2 * espacio), 0, inicioy)) as INodeObject;
    INodeObject n2 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox + (2 * espacio), 0, inicioy + (2 * espacio))) as INodeObject;
    INodeObject n3 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox, 0, inicioy + (2 * espacio))) as INodeObject;
    INodeObject n4 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox, 0, inicioy + (4 * espacio))) as INodeObject;
    INodeObject n5 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox + (2 * espacio), 0, inicioy + (4 * espacio))) as INodeObject;

    ILinkObject u0 = model.Facility.IntelligentObjects.CreateLink("Path", n0, n1, null) as ILinkObject;
    ILinkObject u1 = model.Facility.IntelligentObjects.CreateLink("Path", n1, n2, null) as ILinkObject;
    ILinkObject u2 = model.Facility.IntelligentObjects.CreateLink("Path", n2, n3, null) as ILinkObject;
    ILinkObject u3 = model.Facility.IntelligentObjects.CreateLink("Path", n3, n4, null) as ILinkObject;
    ILinkObject u4 = model.Facility.IntelligentObjects.CreateLink("Path", n4, n5, null) as ILinkObject;
    ILinkObject u5 = model.Facility.IntelligentObjects.CreateLink("Path", n5, n2, null) as ILinkObject;
    ILinkObject u6 = model.Facility.IntelligentObjects.CreateLink("Path", n3, n0, null) as ILinkObject;
}
```

Dibujar9()

```
public void dibujar9(int iniciox, int inicioy, int espacio)
{
    INodeObject n0 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox, 0, inicioy)) as INodeObject;
    INodeObject n1 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox + (2 * espacio), 0, inicioy)) as INodeObject;
    INodeObject n2 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox + (2 * espacio), 0, inicioy + (2 * espacio))) as INodeObject;
    INodeObject n3 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox, 0, inicioy + (2 * espacio))) as INodeObject;
    INodeObject n5 = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(iniciox + (2 * espacio), 0, inicioy + (4 * espacio))) as INodeObject;

    ILinkObject u0 = model.Facility.IntelligentObjects.CreateLink("Path", n0, n1, null) as ILinkObject;
    ILinkObject u1 = model.Facility.IntelligentObjects.CreateLink("Path", n1, n2, null) as ILinkObject;
    ILinkObject u2 = model.Facility.IntelligentObjects.CreateLink("Path", n2, n3, null) as ILinkObject;
    ILinkObject u3 = model.Facility.IntelligentObjects.CreateLink("Path", n2, n5, null) as ILinkObject;
    ILinkObject u4 = model.Facility.IntelligentObjects.CreateLink("Path", n3, n0, null) as ILinkObject;
}
```

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.

```
4 referencias
public void crearPuntoCardinal(int longitud_, int latitud_, string nombre)
{
    INodeObject punto = model.Facility.IntelligentObjects.CreateObject("TransferNode", new FacilityLocation(longitud_, 0, latitud_)) as INodeObject;
    punto.ObjectName = nombre;
}
```

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**

```
public void pintarMapa()
{
    //============================================================ FRONTERA CON EL PACÍFICO //=======================================

    INodeObject p1 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(4, 0, 40)) as INodeObject;
    INodeObject p2 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-15, 0, 35)) as INodeObject;
    ILinkObject u1 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p1, p2, null) as ILinkObject;
    u1.Properties["DrawnToScale"].Value = "False";
    u1.Properties["InitialDesiredSpeed"].Value = "16.67";
    u1.Properties["LogicalLength"].Value = "56671.97";

    p1.ObjectName = "v2";
    p2.ObjectName = "v3";

    INodeObject p3 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-47, 0, 33)) as INodeObject;
    ILinkObject u2 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p2, p3, null) as ILinkObject;
    u2.Properties["DrawnToScale"].Value = "False";
    p3.ObjectName = "v4";

    u2.Properties["DrawnToScale"].Value = "False";
    u2.Properties["InitialDesiredSpeed"].Value = "16.67";
    u2.Properties["LogicalLength"].Value = "92463.27";

    INodeObject p4 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-78, 0, 14)) as INodeObject;
    ILinkObject u3 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p3, p4, null) as ILinkObject;
    p4.ObjectName = "v5";

    u3.Properties["DrawnToScale"].Value = "False";
    u3.Properties["InitialDesiredSpeed"].Value = "16.67";
    u3.Properties["LogicalLength"].Value = "104864.77";
```

```
//============================================================ FRONTERA CON MÉXICO //=============================================

INodeObject p5 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-72, 0, 0)) as INodeObject;
ILinkObject u4 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p4, p5, null) as ILinkObject;
p5.ObjectName = "v6";

u4.Properties["DrawnToScale"].Value = "False";
u4.Properties["InitialDesiredSpeed"].Value = "16.67";
u4.Properties["LogicalLength"].Value = "54701.54";

INodeObject p6 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-80, 0, -10)) as INodeObject;
ILinkObject u5 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p5, p6, null) as ILinkObject;
p6.ObjectName = "v7";

u5.Properties["DrawnToScale"].Value = "False";
u5.Properties["InitialDesiredSpeed"].Value = "16.67";
u5.Properties["LogicalLength"].Value = "46009.63";

INodeObject p7 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-60, 0, -40)) as INodeObject;
ILinkObject u6 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p6, p7, null) as ILinkObject;
p7.ObjectName = "v8";

u6.Properties["DrawnToScale"].Value = "False";
u6.Properties["InitialDesiredSpeed"].Value = "16.67";
u6.Properties["LogicalLength"].Value = "129516.58";

INodeObject p8 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-8, 0, -40)) as INodeObject;
ILinkObject u7 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p7, p8, null) as ILinkObject;
p8.ObjectName = "v9";

u7.Properties["DrawnToScale"].Value = "False";
u7.Properties["InitialDesiredSpeed"].Value = "16.67";
u7.Properties["LogicalLength"].Value = "186768.22";
```

```
INodeObject p9 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-14, 0, -51)) as INodeObject;
ILinkObject u8 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p8, p9, null) as ILinkObject;
p9.ObjectName = "v10";

u8.Properties["DrawnToScale"].Value = "False";
u8.Properties["InitialDesiredSpeed"].Value = "16.67";
u8.Properties["LogicalLength"].Value = "45003.96";

INodeObject p10 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-20, 0, -60)) as INodeObject;
ILinkObject u9 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p9, p10, null) as ILinkObject;
p10.ObjectName = "v11";

u9.Properties["DrawnToScale"].Value = "False";
u9.Properties["InitialDesiredSpeed"].Value = "16.67";
u9.Properties["LogicalLength"].Value = "38862.16";

INodeObject p11 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-30, 0, -65)) as INodeObject;
ILinkObject u10 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p10, p11, null) as ILinkObject;
p11.ObjectName = "v12";

u10.Properties["DrawnToScale"].Value = "False";
u10.Properties["InitialDesiredSpeed"].Value = "16.67";
u10.Properties["LogicalLength"].Value = "40155.17";

INodeObject p12 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-45, 0, -77)) as INodeObject;
ILinkObject u11 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p11, p12, null) as ILinkObject;
p12.ObjectName = "v13";

u11.Properties["DrawnToScale"].Value = "False";
u11.Properties["InitialDesiredSpeed"].Value = "16.67";
u11.Properties["LogicalLength"].Value = "68996.49";

INodeObject p13 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-30, 0, -77)) as INodeObject;
ILinkObject u12 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p12, p13, null) as ILinkObject;
p13.ObjectName = "v14";
```

```
u12.Properties["DrawnToScale"].Value = "False";
u12.Properties["InitialDesiredSpeed"].Value = "16.67";
u12.Properties["LogicalLength"].Value = "53875.45";

INodeObject p14 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(-30, 0, -90)) as INodeObject;
ILinkObject u13 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p13, p14, null) as ILinkObject;
p14.ObjectName = "v15";

u13.Properties["DrawnToScale"].Value = "False";
u13.Properties["InitialDesiredSpeed"].Value = "16.67";
u13.Properties["LogicalLength"].Value = "46692.05";

INodeObject p16 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(40, 0, -90)) as INodeObject;
ILinkObject u15 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p14, p16, null) as ILinkObject;
u15.Properties["DrawnToScale"].Value = "False";
p16.ObjectName = "v16";

u15.Properties["DrawnToScale"].Value = "False";
u15.Properties["InitialDesiredSpeed"].Value = "16.67";
u15.Properties["LogicalLength"].Value = "251418.76";


//============================================================ FRONTERA CON BELICE //=============================================

INodeObject p17 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(40, 0, -31)) as INodeObject;
ILinkObject u16 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p16, p17, null) as ILinkObject;
p17.ObjectName = "v17";

u16.Properties["DrawnToScale"].Value = "False";
u16.Properties["InitialDesiredSpeed"].Value = "16.67";
u16.Properties["LogicalLength"].Value = "200947.50";

INodeObject p18 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(59, 0, -29)) as INodeObject;
ILinkObject u17 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p17, p18, null) as ILinkObject;
p18.ObjectName = "v18";
```

```
u17.Properties["DrawnToScale"].Value = "False";
u17.Properties["InitialDesiredSpeed"].Value = "16.67";
u17.Properties["LogicalLength"].Value = "65052.50";


//============================================================ FRONTERA CON EL ATLÁNTICO //=========================================

INodeObject p19 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(44, 0, -20)) as INodeObject;
ILinkObject u18 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p18, p19, null) as ILinkObject;
p19.ObjectName = "v19";

u18.Properties["DrawnToScale"].Value = "False";
u18.Properties["InitialDesiredSpeed"].Value = "16.67";
u18.Properties["LogicalLength"].Value = "25415.02";


INodeObject p20 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(41, 0, -21)) as INodeObject;
ILinkObject u19 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p19, p20, null) as ILinkObject;
p20.ObjectName = "v20";

u19.Properties["DrawnToScale"].Value = "False";
u19.Properties["InitialDesiredSpeed"].Value = "16.67";
u19.Properties["LogicalLength"].Value = "4591.85";

INodeObject p21 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(39, 0, -19)) as INodeObject;
ILinkObject u20 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p20, p21, null) as ILinkObject;
p21.ObjectName = "v21";

u20.Properties["DrawnToScale"].Value = "False";
u20.Properties["InitialDesiredSpeed"].Value = "16.67";
u20.Properties["LogicalLength"].Value = "4112.32";

INodeObject p22 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(36, 0, -20)) as INodeObject;
ILinkObject u21 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p21, p22, null) as ILinkObject;
p22.ObjectName = "v22";
```

```
u21.Properties["DrawnToScale"].Value = "False";
u21.Properties["InitialDesiredSpeed"].Value = "16.67";
u21.Properties["LogicalLength"].Value = "4591.85";

INodeObject p23 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(34, 0, -18)) as INodeObject;
ILinkObject u22 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p22, p23, null) as ILinkObject;
p23.ObjectName = "v23";

u22.Properties["DrawnToScale"].Value = "False";
u22.Properties["InitialDesiredSpeed"].Value = "16.67";
u22.Properties["LogicalLength"].Value = "4112.32";

INodeObject p24 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(37, 0, -16)) as INodeObject;
ILinkObject u23 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p23, p24, null) as ILinkObject;
p24.ObjectName = "v24";

u23.Properties["DrawnToScale"].Value = "False";
u23.Properties["InitialDesiredSpeed"].Value = "16.67";
u23.Properties["LogicalLength"].Value = "5245.75";

INodeObject p25 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(38, 0, -13)) as INodeObject;
ILinkObject u24 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p24, p25, null) as ILinkObject;
p25.ObjectName = "v25";

u24.Properties["DrawnToScale"].Value = "False";
u24.Properties["InitialDesiredSpeed"].Value = "16.67";
u24.Properties["LogicalLength"].Value = "4591.85";

INodeObject p26 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(41, 0, -15)) as INodeObject;
ILinkObject u25 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p25, p26, null) as ILinkObject;
p26.ObjectName = "v26";

u25.Properties["DrawnToScale"].Value = "False";
u25.Properties["InitialDesiredSpeed"].Value = "16.67";
u25.Properties["LogicalLength"].Value = "5245.75";
```

```
INodeObject p27 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(51, 0, -18)) as INodeObject;
ILinkObject u26 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p26, p27, null) as ILinkObject;
p27.ObjectName = "v27";

u26.Properties["DrawnToScale"].Value = "False";
u26.Properties["InitialDesiredSpeed"].Value = "16.67";
u26.Properties["LogicalLength"].Value = "15170.54";

INodeObject p28 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(52, 0, -22)) as INodeObject;
ILinkObject u27 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p27, p28, null) as ILinkObject;
p28.ObjectName = "v28";

u27.Properties["DrawnToScale"].Value = "False";
u27.Properties["InitialDesiredSpeed"].Value = "16.67";
u27.Properties["LogicalLength"].Value = "5986.84";

INodeObject p29 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(61, 0, -28)) as INodeObject;
ILinkObject u28 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p28, p29, null) as ILinkObject;
p29.ObjectName = "v29";

u28.Properties["DrawnToScale"].Value = "False";
u28.Properties["InitialDesiredSpeed"].Value = "16.67";
u28.Properties["LogicalLength"].Value = "15722.73";

INodeObject p30 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(65, 0, -25)) as INodeObject;
ILinkObject u29 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p29, p30, null) as ILinkObject;
p30.ObjectName = "v30";

u29.Properties["DrawnToScale"].Value = "False";
u29.Properties["InitialDesiredSpeed"].Value = "16.67";
u29.Properties["LogicalLength"].Value = "7265.59";

INodeObject p31 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(67, 0, -27)) as INodeObject;
ILinkObject u30 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p30, p31, null) as ILinkObject;
p31.ObjectName = "v31";
```

```
u30.Properties["DrawnToScale"].Value = "False";
u30.Properties["InitialDesiredSpeed"].Value = "16.67";
u30.Properties["LogicalLength"].Value = "4112.32";

INodeObject p32 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(64, 0, -30)) as INodeObject;
ILinkObject u31 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p31, p32, null) as ILinkObject;
p32.ObjectName = "v32";

u31.Properties["DrawnToScale"].Value = "False";
u31.Properties["InitialDesiredSpeed"].Value = "16.67";
u31.Properties["LogicalLength"].Value = "6161.22";

INodeObject p33 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(69, 0, -29)) as INodeObject;
ILinkObject u32 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p32, p33, null) as ILinkObject;
p33.ObjectName = "v33";

u32.Properties["DrawnToScale"].Value = "False";
u32.Properties["InitialDesiredSpeed"].Value = "16.67";
u32.Properties["LogicalLength"].Value = "7410.90";

INodeObject p34 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(63, 0, -33)) as INodeObject;
ILinkObject u33 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p33, p34, null) as ILinkObject;
p34.ObjectName = "v34";

u33.Properties["DrawnToScale"].Value = "False";
u33.Properties["InitialDesiredSpeed"].Value = "16.67";
u33.Properties["LogicalLength"].Value = "10476.98";

INodeObject p35 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(65, 0, -34)) as INodeObject;
ILinkObject u34 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p34, p35, null) as ILinkObject;
p35.ObjectName = "v35";

u34.Properties["DrawnToScale"].Value = "False";
u34.Properties["InitialDesiredSpeed"].Value = "16.67";
u34.Properties["LogicalLength"].Value = "3254.98";
```

```
u35.Properties["DrawnToScale"].Value = "False";
u35.Properties["InitialDesiredSpeed"].Value = "16.67";
u35.Properties["LogicalLength"].Value = "14531.17";


//============================================================ FRONTERA CON HONDURAS //=============================================

INodeObject p37 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(45, 0, -5)) as INodeObject;
ILinkObject u36 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p36, p37, null) as ILinkObject;
p37.ObjectName = "v37";

u36.Properties["DrawnToScale"].Value = "False";
u36.Properties["InitialDesiredSpeed"].Value = "16.67";
u36.Properties["LogicalLength"].Value = "129212.26";

INodeObject p38 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(42, 0, 2)) as INodeObject;
ILinkObject u37 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p37, p38, null) as ILinkObject;
p38.ObjectName = "v38";

u37.Properties["DrawnToScale"].Value = "False";
u37.Properties["InitialDesiredSpeed"].Value = "16.67";
u37.Properties["LogicalLength"].Value = "27168.80";

INodeObject p39 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(44, 0, 13)) as INodeObject;
ILinkObject u38 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p38, p39, null) as ILinkObject;
p39.ObjectName = "v39";

u38.Properties["DrawnToScale"].Value = "False";
u38.Properties["InitialDesiredSpeed"].Value = "16.67";
u38.Properties["LogicalLength"].Value = "39861.84";

INodeObject p40 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(28, 0, 18)) as INodeObject;
ILinkObject u39 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p39, p40, null) as ILinkObject;
p40.ObjectName = "v40";
```

```
    u39.Properties["DrawnToScale"].Value = "False";
    u39.Properties["InitialDesiredSpeed"].Value = "16.67";
    u39.Properties["LogicalLength"].Value = "59757.10";

    INodeObject p41 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(27, 0, 24)) as INodeObject;
    ILinkObject u40 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p40, p41, null) as ILinkObject;
    p41.ObjectName = "v41";

    u40.Properties["DrawnToScale"].Value = "False";
    u40.Properties["InitialDesiredSpeed"].Value = "16.67";
    u40.Properties["LogicalLength"].Value = "35889.50";

    INodeObject p42 = model.Facility.IntelligentObjects.CreateObject("BasicNode", new FacilityLocation(12, 0, 32)) as INodeObject;
    ILinkObject u41 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p41, p42, null) as ILinkObject;
    p42.ObjectName = "v42";

    u41.Properties["DrawnToScale"].Value = "False";
    u41.Properties["InitialDesiredSpeed"].Value = "16.67";
    u41.Properties["LogicalLength"].Value = "100348.94";

    ILinkObject u42 = model.Facility.IntelligentObjects.CreateLink("Conveyor", p42, p1, null) as ILinkObject;

    u42.Properties["DrawnToScale"].Value = "False";
    u42.Properties["InitialDesiredSpeed"].Value = "16.67";
    u42.Properties["LogicalLength"].Value = "66761.56";

    IFixedObject base_militar = model.Facility.IntelligentObjects.CreateObject("Source", new FacilityLocation(0, 0, -70)) as IFixedObject;
    ILinkObject u43 = model.Facility.IntelligentObjects.CreateLink("Path", base_militar.Nodes[0], p14, null) as ILinkObject;
    base_militar.ObjectName = "BaseMilitar";
    base_militar.Properties["InterarrivalTime"].Value = "15";
    base_militar.Properties["MaximumArrivals"].Value = "15";
    model.Facility.IntelligentObjects["BaseMilitar"].Properties["EntityType"].Value = "Avion1";
}
```

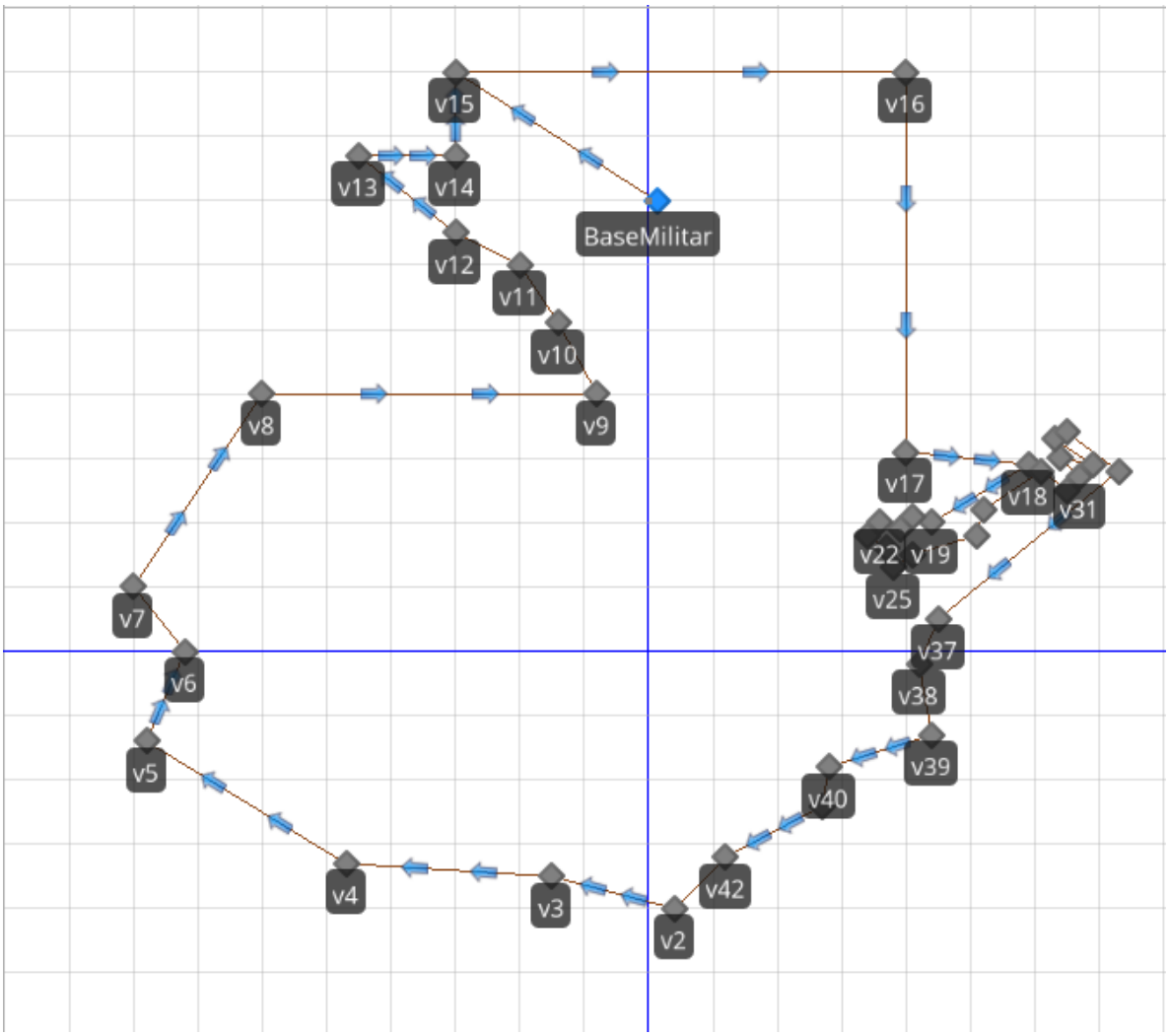
Como se puede apreciar en las capturas se modificaron las mismas tres propiedades en cada uno de los conveyor creados. Y finalmente se crea un Source al que se le coloca por nombre “BaseMilitar” y se cambia el valor de las propiedades “InterarrivalTime”, “MaximumArrivals” y “EntityType”, al primero se 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.

4- Evolución del modelo final

Creación del contorno del mapa y la base militar

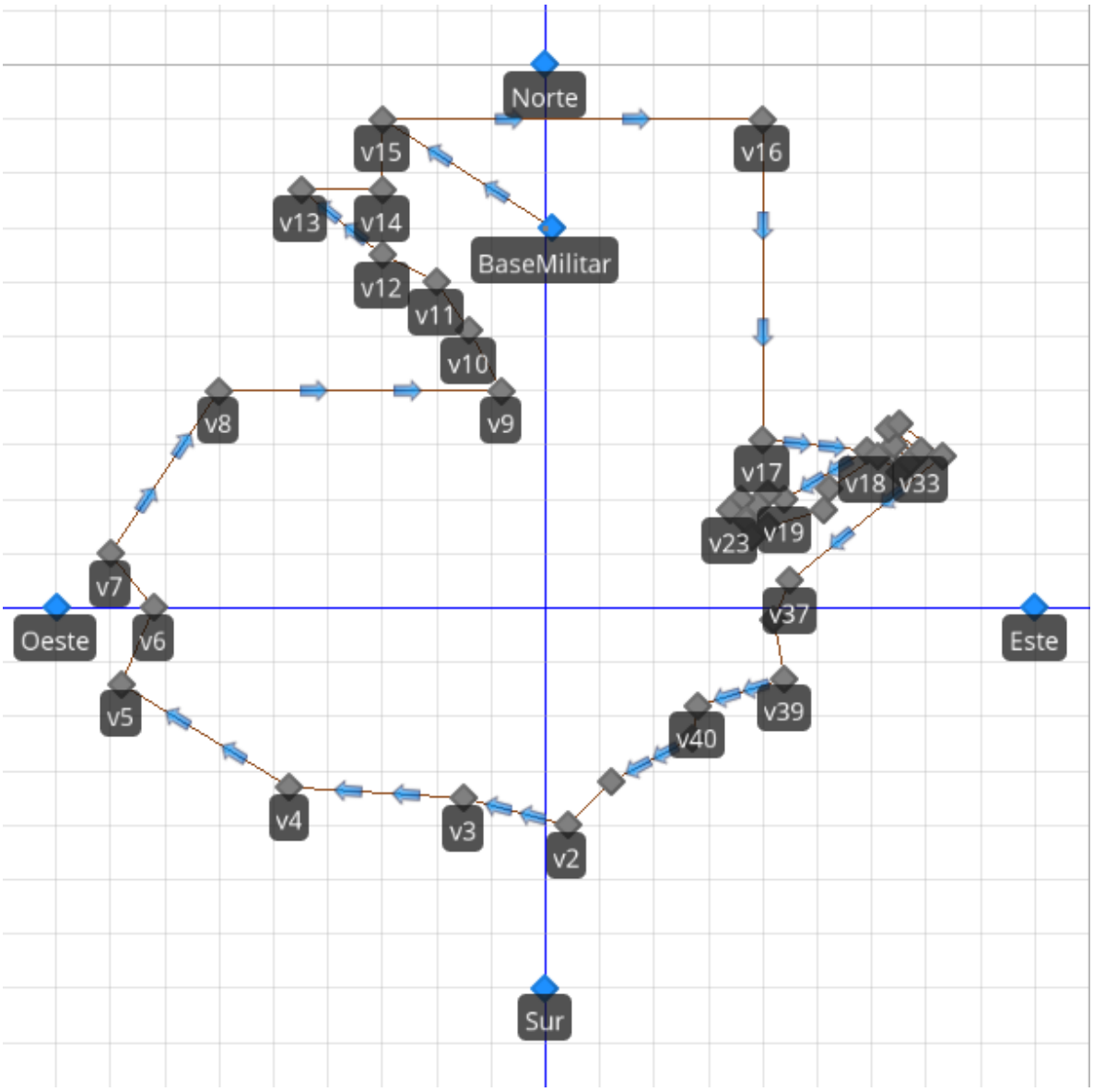
Se crea el contorno del mapa de Guatemala mediante nodos de tipo BASIC NODE, conectados por medio de enlaces CONVEYOR, estos enlaces poseen una longitud lógica determinada que representa la longitud de las diferentes fronteras marítimas y terrestres del país.

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.



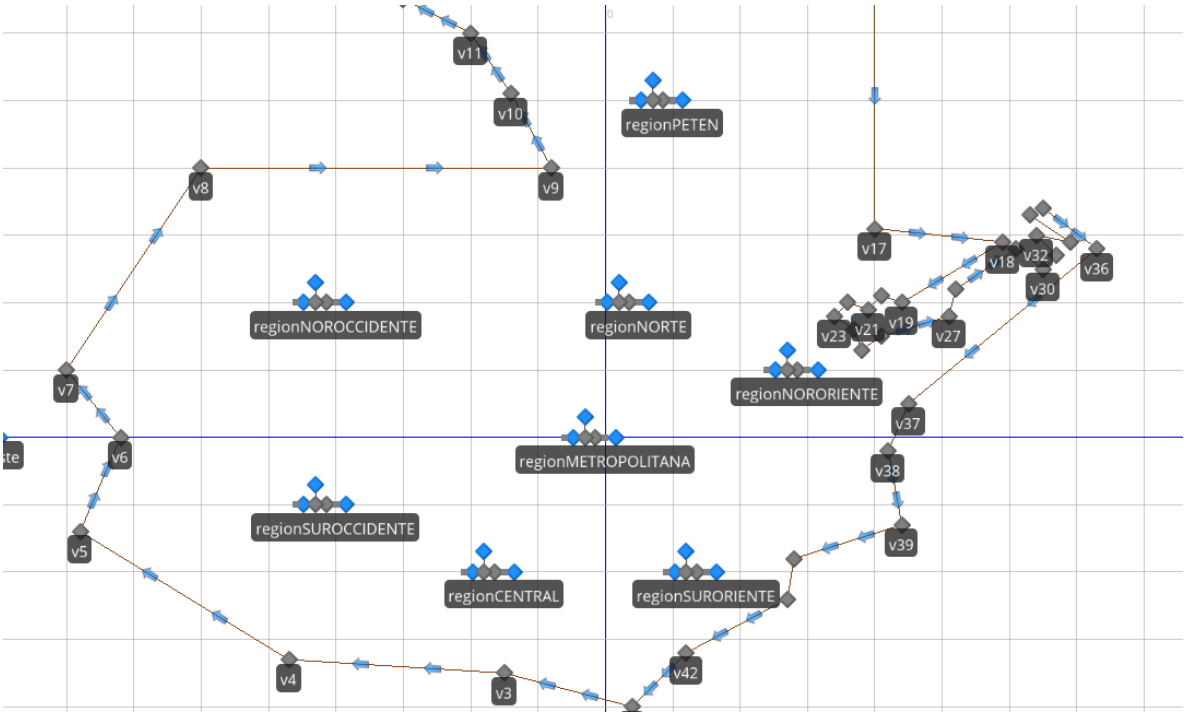
Creación de puntos cardinales

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.



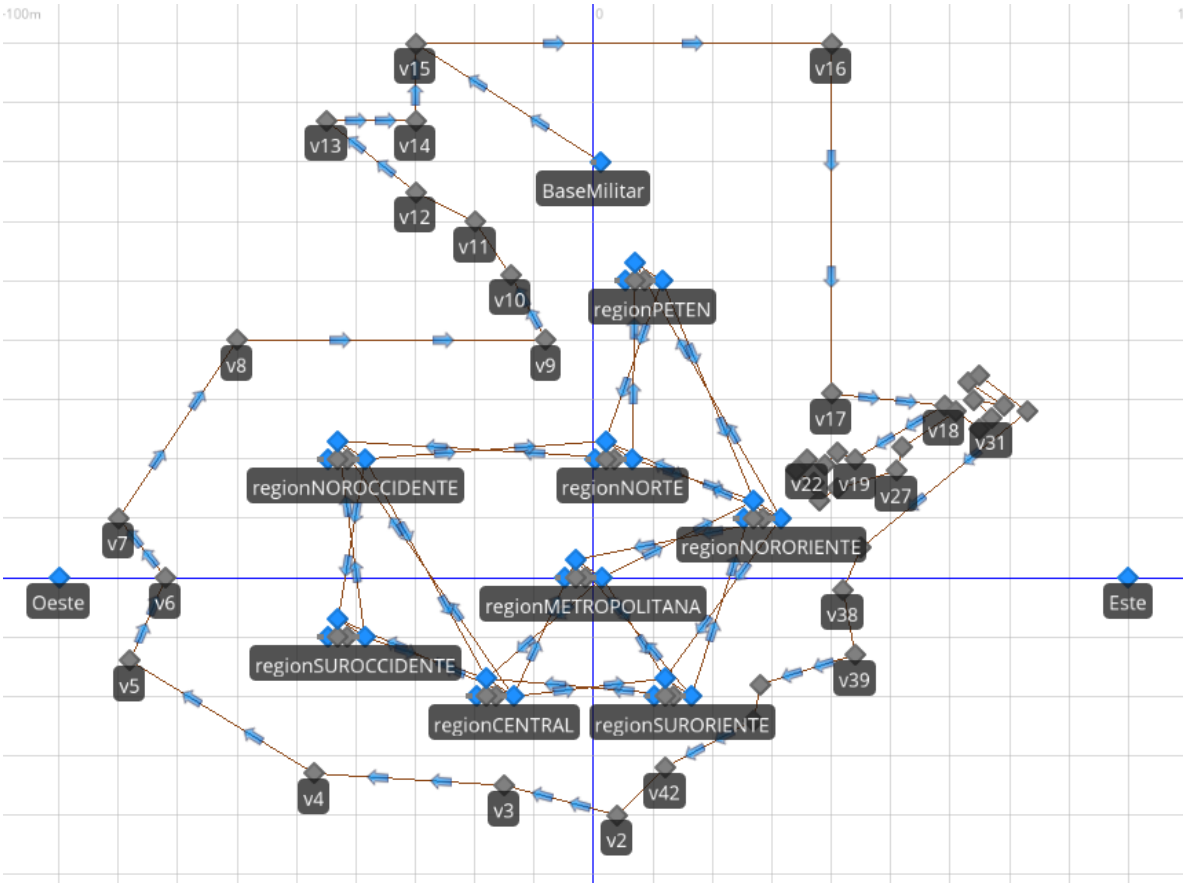
Creación de regiones

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.



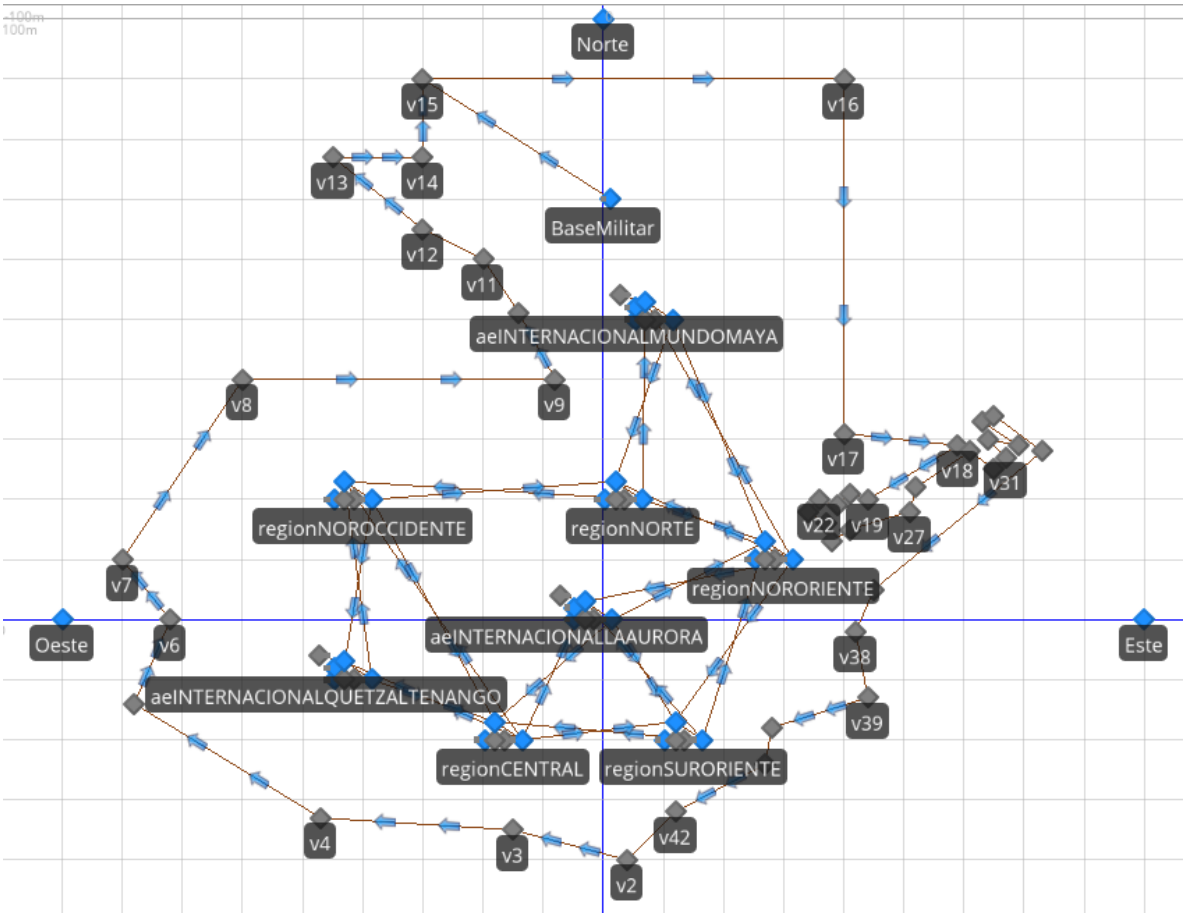
Enlaces entre regiones

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.



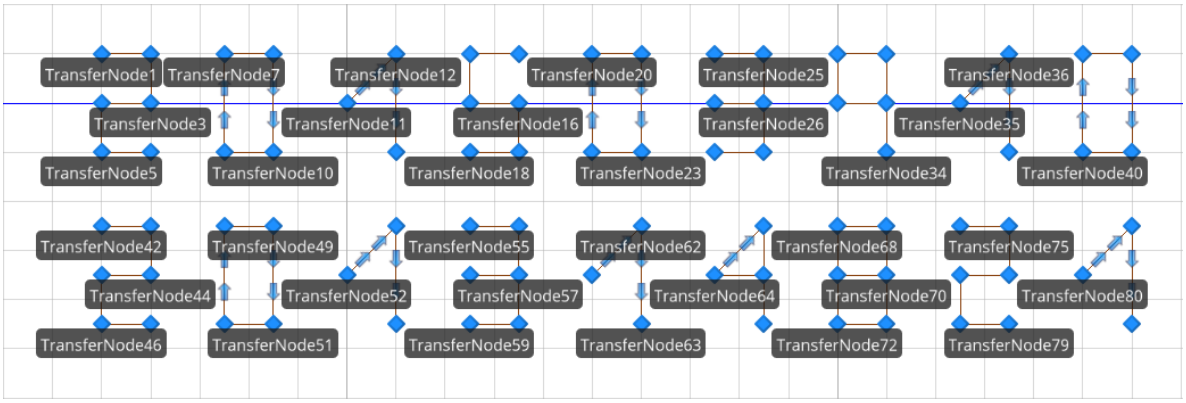
Creación de aeropuertos

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.

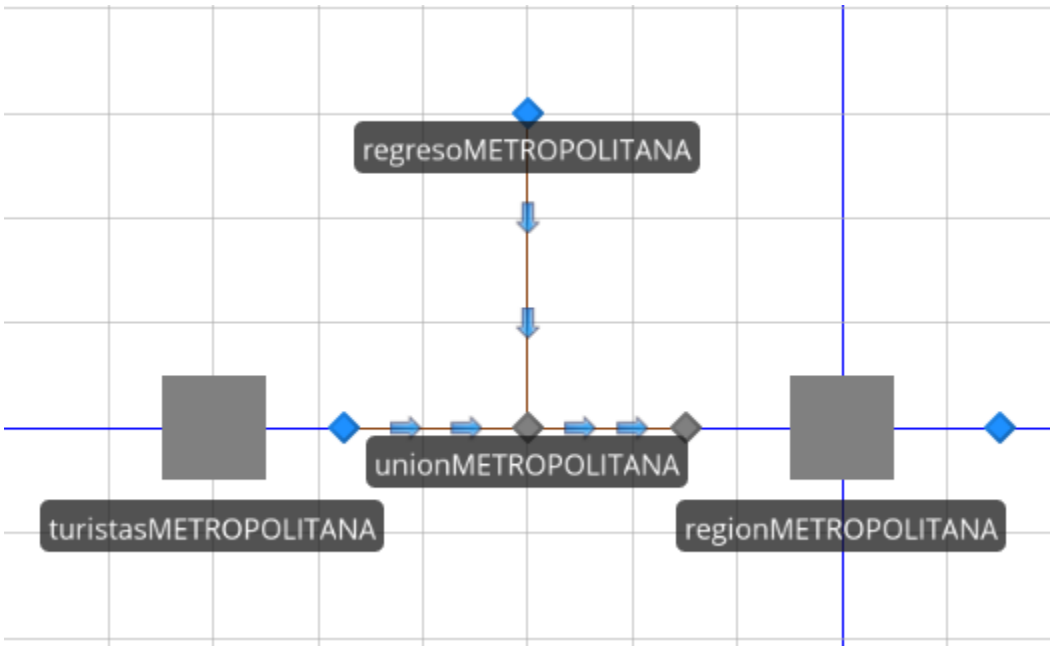


5- Diseño del modelo final

Regiones

Para modelar las regiones, se implementó lo siguiente:

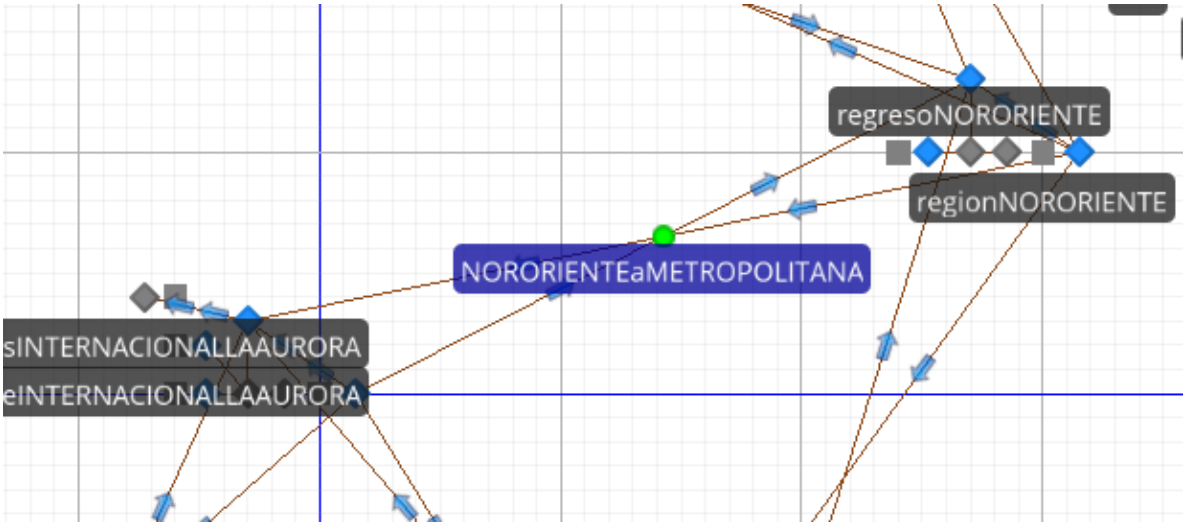
- Un objeto de tipo SOURCE para generar los turistas que viajan entre regiones, el nombre de este es “turistas” acompañado del nombre de la región en la cual se encuentra. El tiempo en minutos entre llegadas está dado por una distribución de Poisson.
- Un objeto de tipo SERVER representando la estación 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 del 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.

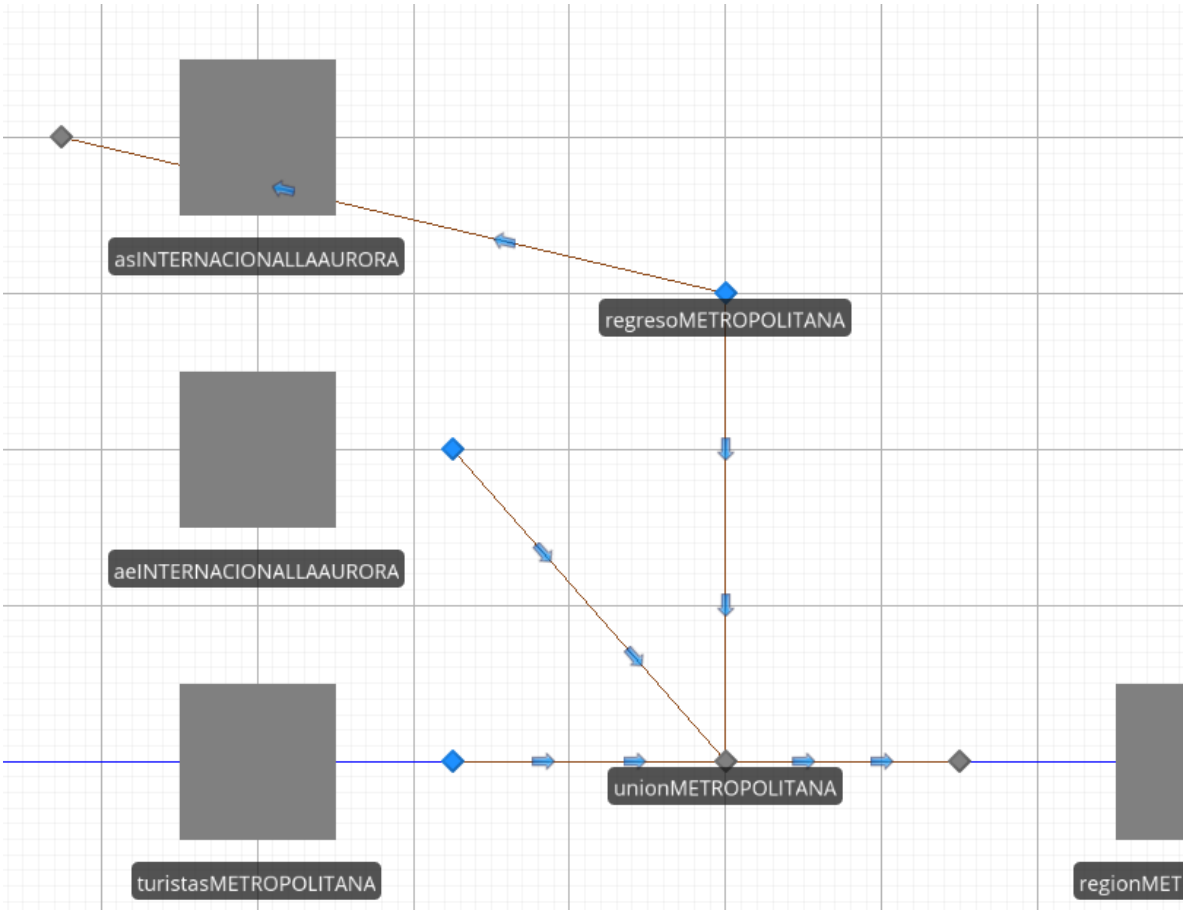
El nombre que identifica a cada uno de los enlaces se compone de la región origen, la palabra “a” y la región destino del turista. La longitud lógica del enlace es la distancia mencionada anteriormente, así como la velocidad.



Aeropuertos

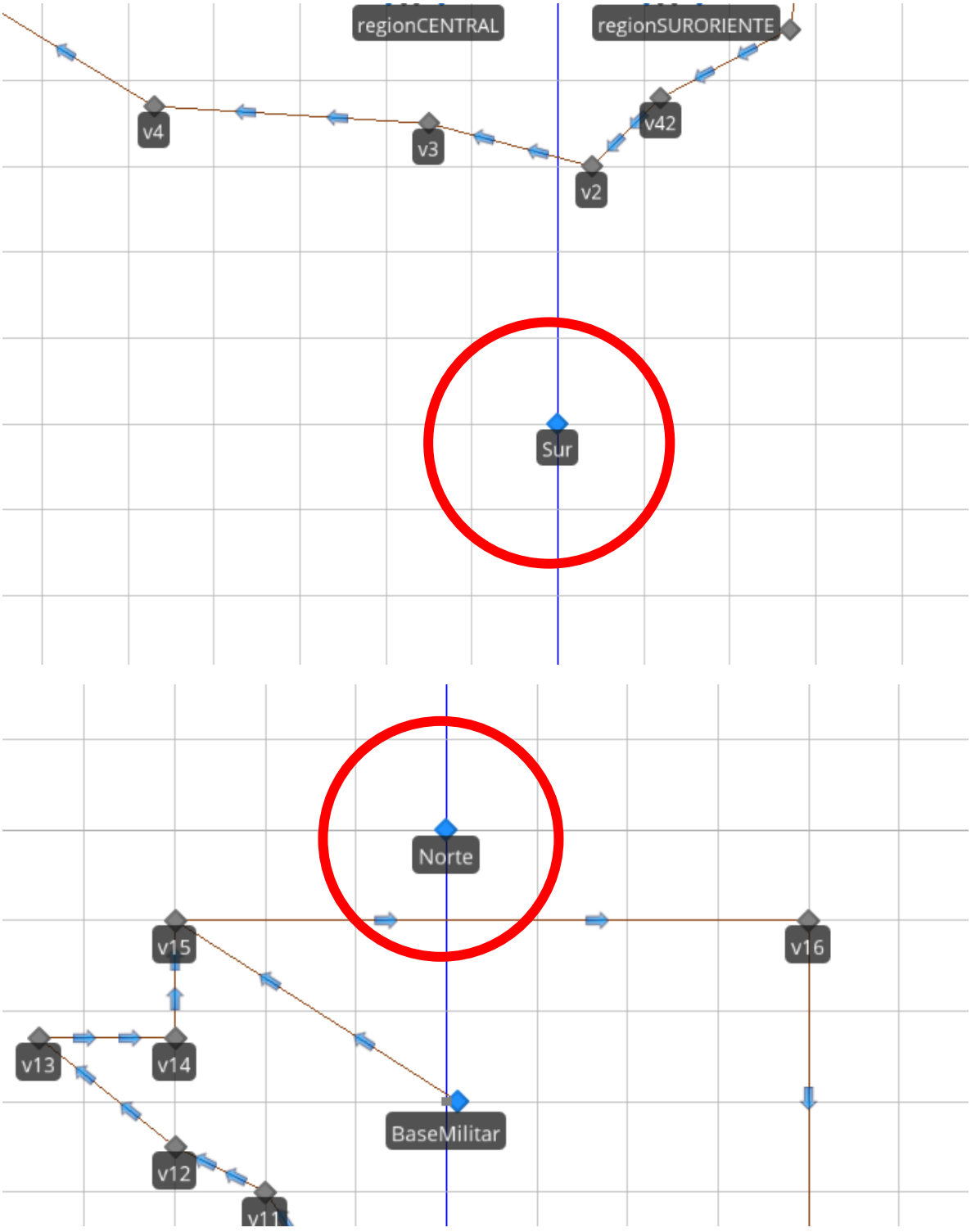
Para los aeropuertos se implementó lo siguiente:

- 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.



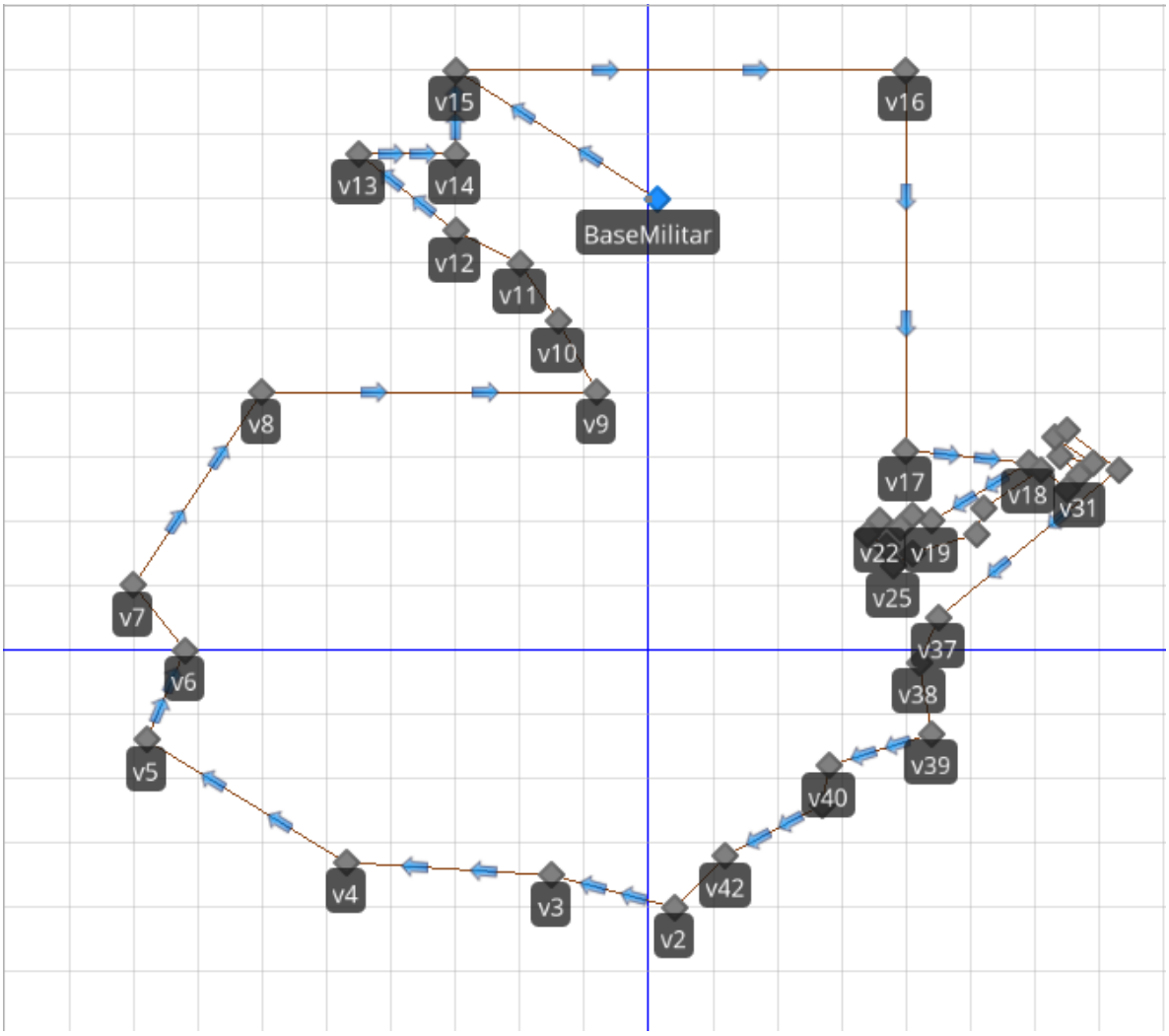
Puntos cardinales

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.


regionCENTRAL
regionSURORIENTE
v4
v3
v2
v42
Sur
Norte
v15
v16
v13
v14
BaseMilitar
v12
v11

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.

6- 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.

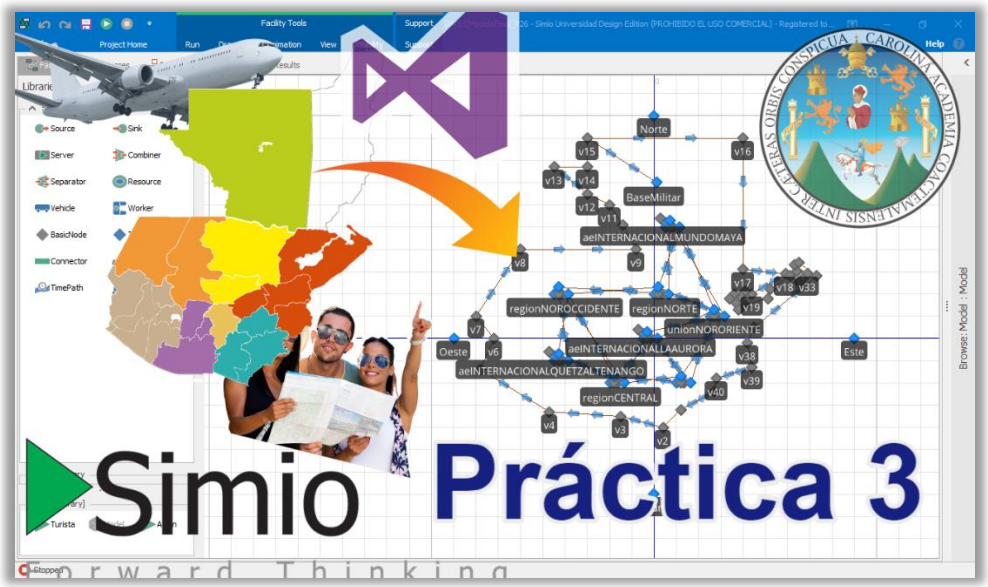
- Dependiendo de la forma en que se integre la API de Simio con el lenguaje de programación deseado, puede darnos una ventaja al momento de crear un modelo de simulación, ya que por ejemplo mediante una sentencia de ciclo FOR o WHILE se podrían crear una cantidad considerable de componentes en cuestión 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

- El estudio del comportamiento de los turistas, tanto nacionales como internacionales, es muy importante para poder implementar estrategias que permitan dar a conocer los diferentes destinos que posee nuestro país, haciendo énfasis en los que se encuentran en las regiones menos visitadas y que así puedan estas explotar su potencial en cuanto a turismo.

7- Vídeo

El vídeo con la descripción de la práctica se encuentra pulsando la imagen o en el siguiente enlace.



Enlace:

https://www.youtube.com/watch?v=oo-juL-LIQ0