5.1.4 Interfaz de tramos

Una vez arrancada la aplicación y cargada la red vial, en el formulario de herramientas puede seleccionarse el botón representado por un dedo señalando un tramo que une dos nodos. Este botón lanza la creación del interfaz de tramos.

El interfaz de tramos tiene el aspecto que se muestra en la figura 5.1.4.1.

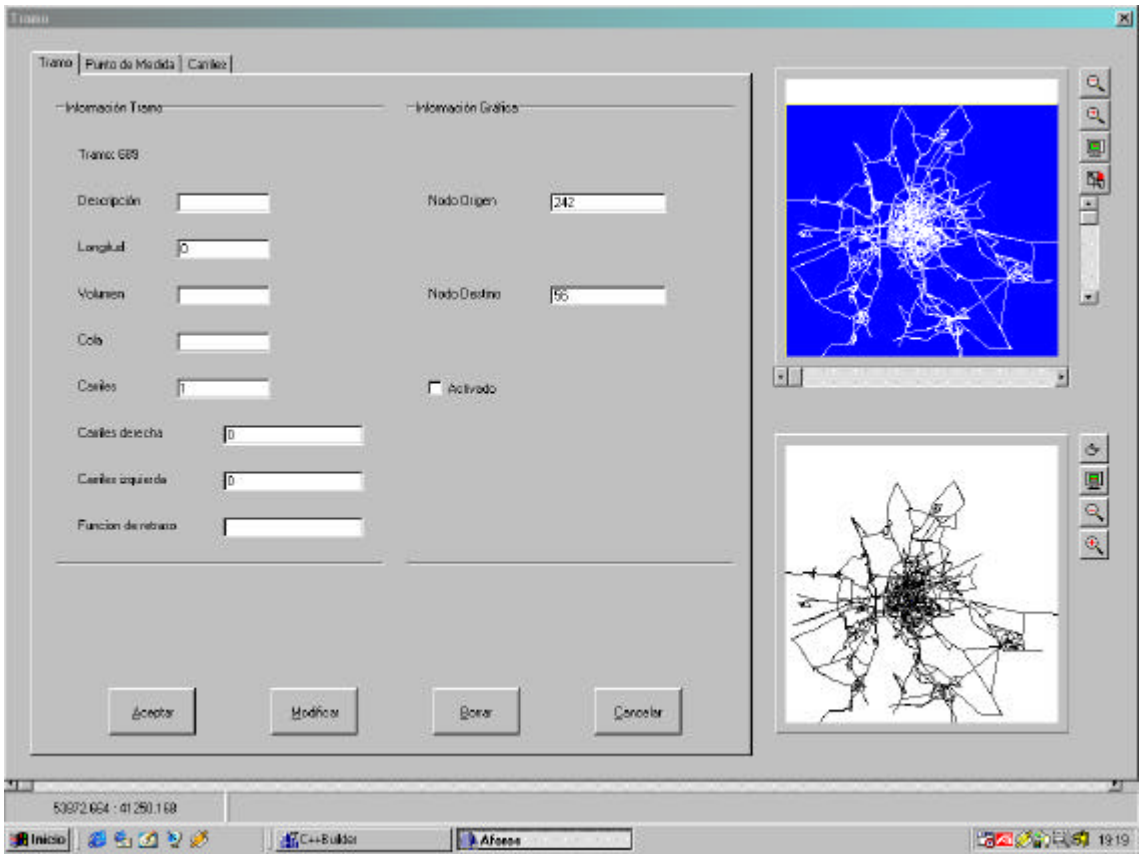

Figura 5.1.4.1: Interfaz de tramo

Como puede observarse en dicha figura, este interfaz también se divide en varias páginas. Pinchando con el ratón sobre la solapa de una cualquiera de las páginas se consigue que la página elegida sea la que se muestra en pantalla.

Al igual que en otros interfaces, en la parte derecha de la pantalla aparecen las dos imágenes que pueden usarse para seleccionar un nuevo tramo o simplemente, hacer operaciones de zoom y windowing sobre ellas.

En el marco izquierdo se muestra información básica del tramo: en primer lugar el código del mismo. A continuación, su descripción, la longitud del mismo, el volumen, la cola y por ejemplo, el numero de carriles.

Se definió tramo como uno de los sentidos de circulación de una vía. Eso implica que puede tener varios carriles. Pero además aparecen otros dos datos, carriles derecha y carriles izquierda. Se refieren a carriles adicionales que pueden crearse bien por la derecha, bien por la izquierda del tramo.

Finalmente aparece la función de retraso.

En el marco derecho, con el título de Información Gráfica aparece nodo origen , nodo destino y activado.

En la parte inferior aparecen los botones: Aceptar, Modificar, Borrar y Cancelar.

Estos botones lanzan las funciones adecuadas al ser pulsados. La función concreta de cada uno de ellos se ve en detalle posteriormente.

Si el usuario pincha con ratón sobre la solapa Puntos de Medida debe aparecer en pantalla el interfaz de Puntos de Medida, tal y como se muestra en la figura 5.1.4.2.

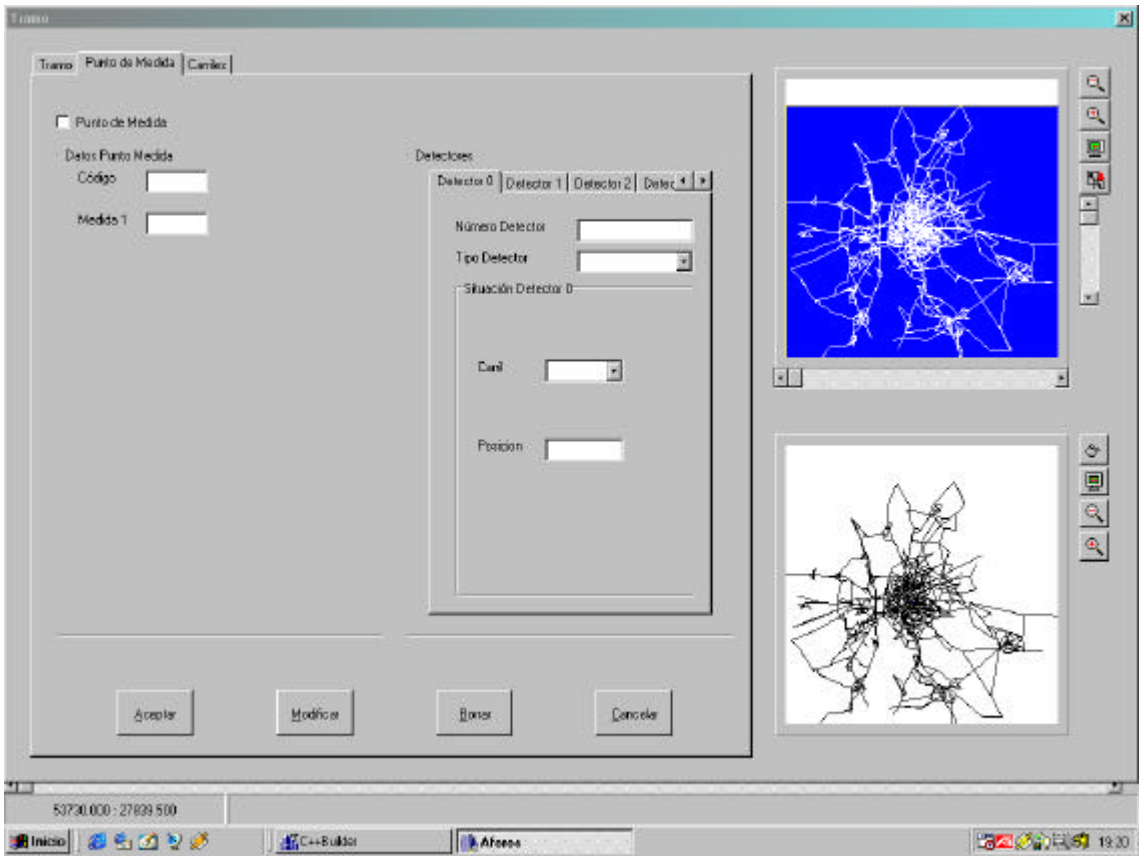

Figura 5.1.4.2: Interfaz de Puntos de Medida

En la parte izquierda aparece un marco, con el nombre Datos Punto Medida. En él aparece el código del punto de medida y la medida 1.

En el marco derecho aparece el marco llamado Detectores. Hay una página para cada uno de los detectores.

Finalmente, abajo aparecen cuatro botones que nos permiten realizar las funciones más usuales sobre los puntos de medida y los detectores.

Si el usuario pincha en la solapa Carriles, debe aparecer en pantalla el interfaz de Carriles, tal y como se muestra en la figura 5.1.4.3.

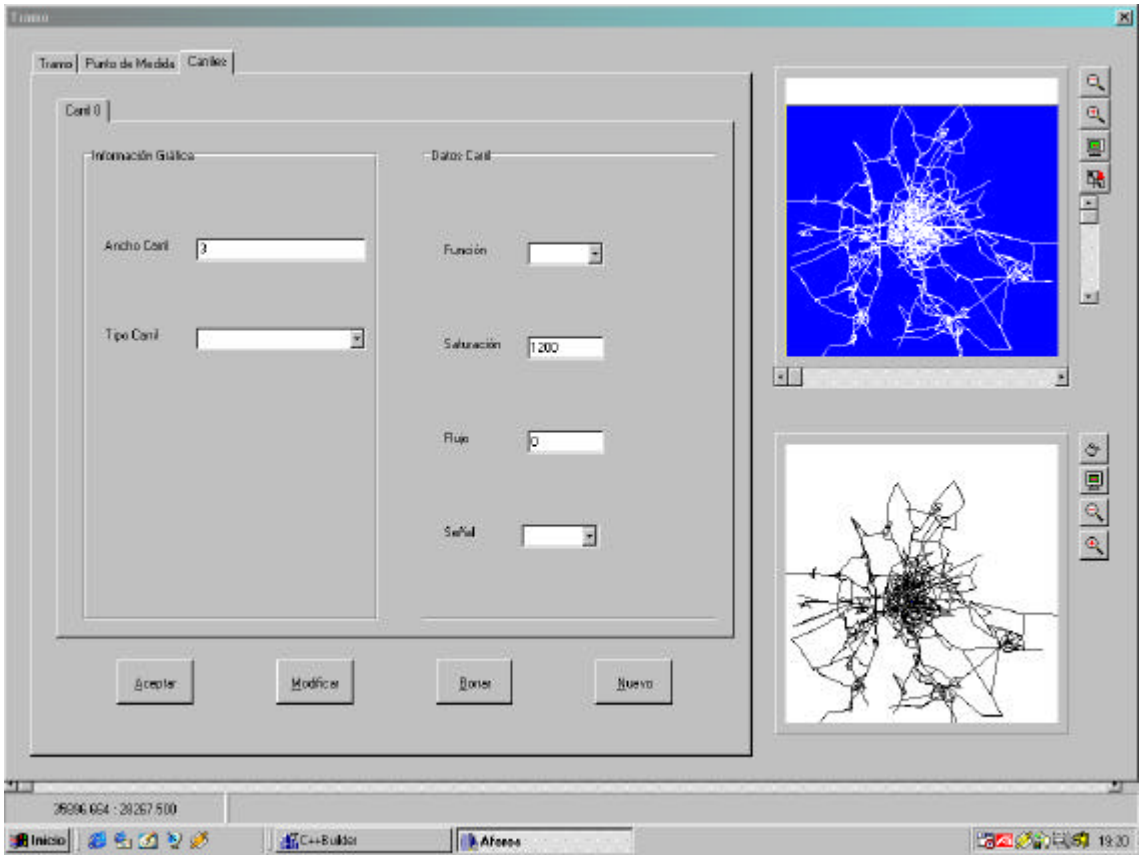

Figura 5.1.4.3: Interfaz de carriles

De nuevo aparecen dos marcos, derecho e izquierdo. El marco izquierdo muestra el ancho del carril de qué tipo es. En el marco de la derecha aparece la función del carril, su saturación, su flujo y la señal.

Por supuesto en la parte inferior los cuatro botones, con sus funciones.

Implementación del interfaz de tramos

Las clases que se usan son:

```
class Datos_carril
{
 unsigned int cod_tramo;
 unsigned char cod_carril;
 float saturacion;
 float longitud;
 float ancho;
 float flujo;
 float posicion_comienzo;
 unsigned int funcion;
 unsigned int senal;
 unsigned int tipo;
```

```
public:
//métodos
};
```

Esta clase almacena toda la información necesaria sobre un carril de un tramo. La propiedad *cod_tramo* es el código del tramo al que pertenece, la propiedad *cod_carril* es el código de carril dentro del tramo. El resto de las propiedades son características del carril.

Todos los métodos de la clase sirven para obtener o actualizar el valor de alguna de las propiedades.

```
class Tramo_interactivo
{
  unsigned int cod_tramo;
  TForm *formulario;
  TPageControl *pagina;
  TTabSheet *tramo;
  TTabSheet *pm;
  TTabSheet *carril;
  TCheckBox *Activado;
  Pm_interactivo p_m;
  Carril_interactivo car;
  Informacion_tramo inf_tramo;
public:
  void Generar_formulario(TForm *padre,Tramo d_tramo);
  void Destruir_formulario();
  void Actualiza_evento();
   void __fastcall Sel_tramo (TObject *Sender, TMouseButton Button, TShiftState Shift,
int X, int Y);
  void Activacion_seleccion_tramos (Contexto *context);
};
```

La mayoría de la propiedades, excepto *cod_tramo, p_m* y *inf_tramo*, son objetos que tienen como fin construir el interfaz de tramos.

Además, como se verá posteriormente, las clases a las que pertenecen los objetos *p_m (Pm_interactivo), car (Carril_interactivo)* y *inf_tramo (Informacion_tramo)* tienen a su vez propiedades que en su mayoría son de nuevo objetos gráficos, salvo alguna excepción.

La propiedad *cod_tramo* es el código del tramo. El resto de las propiedades se verán a continuación cuando se traten cada una de las clases.

El método *Generar_formulario* crea gran cantidad de componentes gráficos para formar el interfaz de tramos. Para ello usa además varios métodos de los objetos *car* y *inf_tramo*.

El método *Destruir_formulario* libera la memoria ocupada por todos los eltos creados después de llamar a la función *Grabar_datos_tramos*. Esta función lo que hace es salvar a disco la lista *Lista_datos_tramos*. Esta lista está formada por objetos de la clase *Datos_tramos*, que se detalla a continuación.

El método Actualiza_evento se usa para asignar ciertas funciones a los eventos onClick sobre los botones. En concreto asigna la función *Aceptar_Click* al botón *Aceptar*, *Modificar_Click* al botón *Modificar* y *Borrar_Click* al botón *Borrar.*

El método *Sel_tramo* de encarga de determinar el tramo seleccionado sobre la imagen y a continuación llama a los métodos necesarios para actualizar la información.

El método *Activacion_selección_tramos* simplemente asigna los eventos necesarios a la imagen para cuando el ratón de desplace sobre ella: al liberar el botón del ratón tras pulsarlo se llama al método *Sel_tramo*. Mientras el ratón se desplaza por la imagen se llama a la función que se encargar de imprimir las coordenadas globales del punto sobre el que está el ratón en la barra de estado.

```
class Datos_tramos
{
char descripcion[50];
 float volumen;
 float cola;
public:
// métodos para manipular propiedades
};
```

En esta clase todos los métodos tienen como función actualizar o devolver el valor de alguna de las propiedades, excepto la sobrecarga de los operadores, necesarias para todo objeto que forme parte de una lista de tipo *Lista_inf*.

Las propiedad *descripcion* es una breve descripcion del tramo. Las propiedades *volumen* y *cola* almacenan información sobre el tramo.

```
class Pm_interactivo
{
  TGroupBox *GroupBox1;
  TGroupBox *GroupBox2;
  TCheckBox *activado;
  TLabel *l_codigo;
  TLabel *l_medida1;
  TLabel *l_medida2;
  TEdit *e_codigo;
  TEdit *e_medida1;
  TImage *imagen;
  TPageControl *pagina;
  TBitBtn *Aceptar;
  TBitBtn *Modificar;
```

```
  TBitBtn *Borrar;
  TBitBtn *Cancelar;
  Lista_Detector_interactivo lista;
public:
    void Generar_formulario (TWinControl *padre,TForm *formulario);
    void Genera_detectores(unsigned int numero,int Ancho,int Alto);
    void Genera_pm (TWinControl *padre,int Ancho_p,int Alto_p);
    void Destruir_formulario();
};
```

De nuevo todas las propiedades con la excepción de *lista* son objetos gráficos para representar el interface de la página de puntos de medida.

La clase *Lista_Detector_interactivo* a la que pertenece el objeto lista sólo tiene una propiedad, una lista de tipo *TList*. Los únicos métodos que incluye son para manipular dicha lista: crearla, añadir elementos, etcétera.

El método *Generar_formulario* crea todos los objetos gráficos necesarios para la representación gráfica del interfaz de puntos de medida. Para ello llama a los métodos *Genera_pm* y *Genera_detectores*.

El método *Genera_detectores* crea tantos objetos de la clase *Detector_interactivo* como se le pasan en el parámetro *numero*. Además añade cada detector que crea a la lista *Lista_Detector_interactivo.* La clase *Detector_interactivo* es bastante simple, como puede apreciarse:

```
class Detector_interactivo
{
  TTabSheet *pag;
  TEdit *posicion;
  TGroupBox *Grupo1;
  TEdit *codigo;
  TLabel *l_codigo;
  TLabel *l_carril;
  TLabel *l_tipo;
  TLabel *l_posicion;
  TComboBox *Tipo_detector;
  TComboBox *carril;
public:
void Generar_formulario (TWinControl *padre,int numero,int Ancho_p,int Alto_p);
void Destruir_formulario();
};
```

Como puede apreciarse, su única función es crear el formulario para cada uno de los puntos de medida.

El método *Genera_pm* genera el formulario para puntos de medida.

El método *Destruir_formulario* libera todos los objetos gráficos creados.

Esto puede apreciarse de manera más clara en la figura 5.1.4.4.

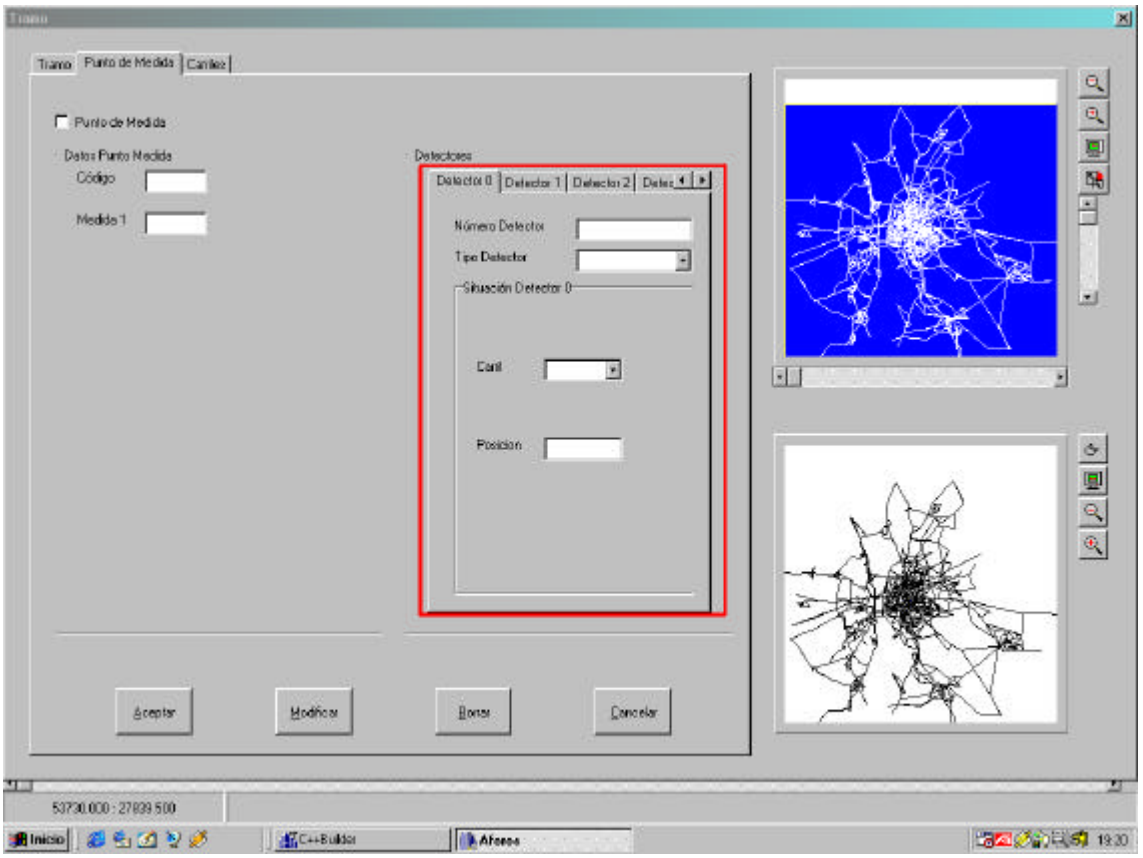

Figura 5.1.4.4: Interfaz de puntos de medida indicando clases

En esta figura se puede apreciar mejor qué parte del formulario se encarga de representar cada clase, *Pm_interactivo* y la lista de objetos de la clase *Detector_interactivo*.

La clase *Pm_interactivo* se encarga de representar prácticamente todo el interfaz, con excepción de la zona incluida en rectángulo de color rojo. Incluso se encarga de crear el objeto *TPageControl*, el controlador necesario para luego poder crear cada una de las páginas *TTabSheet*.

Además *Pm_interactivo* contiene una propiedad que es un lista de objetos de la clase *Detector_interactivo*. Pues bien, cada objeto de la clase *Detector_interactivo* dibuja una de las diferentes páginas que aparecen en el interior del rectángulo rojo.

```
class Carril_interactivo
{
 unsigned int tramo;
TPageControl *pagina;
 TBitBtn *Aceptar;
 TBitBtn *Modificar;
 TBitBtn *Borrar;
```

```
 TBitBtn *Nuevo_carril;
 Lista_inf<Datos_Carril_int > lista_carriles;
public:
   void Destruir_formulario();
   void Generar_formulario (TWinControl *padre,int numero_carriles,unsigned int tramo,int Ancho_p,int Alto_p);
   void __fastcall Borrar_click (TObject *Check);
   void __fastcall Modificar_click (TObject *Check);
   void __fastcall Aceptar_click (TObject *Check);
   void __fastcall Nuevo_click (TObject *Check);
   void Borrar_carril();
   void Actualiza_eventos();
   void Obtener_datos (Datos_carril *d);
   void Borrar_carril (unsigned int orden);
   void Add_carril (unsigned int i);
   unsigned int Obtener_tipo (unsigned int carril);
     bool Comprobacion_coherencia_ensanchamiento (unsigned int carril,unsigned int tipo);
};
```

La función de esta clase es representar la página correspondiente a los carriles. La propiedad *lista_carriles* es una lista de objetos de la clase *Datos_Carril_int*.

Los métodos *Generar_formulario* y *Destruir_formulario* son los encargados de crear todos los objetos gráficos necesarios para representar la página de carriles y después destruirlos, respectivamente. Para ello, el primero de estos métodos debe llamar a otros, ya que crea la lista *lista_tipo_carril* para poder presentar los tipos de carril existentes y *Lista_datos_carril* para cargar los datos de los carriles. Además llama a *Actualiza_eventos* y a *Actualiza_datos*.

Lista_datos_carril es una lista de objetos de la clase *Datos_carril*.

El método *Borrar_carril* destruye el formulario de cada carril y borra todos los carriles. Si se le pasa un número como parámetro sólo borra ese carril.

El método *Borrar_Click* se encargar de borrar un carril tanto de la lista de *Lista_datos_carril* como de la lista *lista_carriles*.

El método *Modificar_Click* modifica adecuadamente la lista *Lista_datos_carril* y también llama al método *Actualiza_tramo* de la clase *Informacion_tramo*, o sea, actualiza los datos representados en el interfaz.

El método *Aceptar_Click* hace lo mismo que *Modificar_Click*, realmente.

El método *Nuevo_Click* modifica adecuadamente el elemento en la lista *Lista_tramos*. A continuación llama al método *Actualiza_tramo* de la clase *Informacion_tramo*, o sea, actualiza los datos representados en el interfaz, y a *Actualiza_datos.*

El método *Actualiza_eventos* asigna al evento *onClick* sobre los botones *Aceptar, Modificar, Borrar* y *Nuevo_carril* a los métodos *Aceptar_Click, Modificar_Click, Borrar_Click* y *Nuevo_Click*, respectivamente.

El método *Obtener_datos* obtiene los datos de la lista *lista_carriles* y los guarda en el objeto que se le pasa como parámetro. Los datos los obtiene del formulario.

El método *Add_carril* añade un nuevo elemento a la lista *lista_carriles*.

El método *Obtener_tipo* devuelve el tipo de carril.

El método *Comprobacion_coherencia_ensanchamiento* comprueba si el ensanchamiento es coherente según el tipo de carril.

```
class Datos_Carril_int
{
  unsigned char carril;
  TGroupBox *GroupBox1;
  TGroupBox *GroupBox2;
  TGroupBox *GroupBox3;
  TLabel *l_tipo;
  TLabel *l_saturacion;
  TLabel *l_senal;
  TLabel *l_flujo;
  TLabel *l_longitud;
  TLabel *l_comienzo;
  TLabel *l_funcion;
  TLabel *l_ancho;
  TEdit *e_comienzo;
  TEdit *e_longitud;
  TEdit *e_saturacion;
  TEdit *e_flujo;
  TEdit *e_ancho;
  TTabSheet *pagina;
  TComboBox *tipo_carril;
  TComboBox *funcion;
  TComboBox *senal;
public:
   void Genera_carril (TWinControl *padre,int numero_carril,int Ancho_p,int Alto_p);
  void Destruir_formulario();
  void Actualiza_tipo_carril ();
  void Obtener_datos (Datos_carril *d);
  float Obtener_flujo ();
  float Obtener_ancho ();
  unsigned int Obtener_funcion ();
  float Obtener_comienzo ();
  float Obtener_longitud ();
  float Obtener_saturacion ();
  unsigned int Obtener_tipo_carril ();
  unsigned int Obtener_senal ();
```

```
  void Actualiza_datos (Datos_carril d);
};
```

La clase *Datos_carril_int* tiene como función representar un formulario donde mostrar o recopilar información sobre los carriles de un tramo. La propiedad *carril* sirve para identificar al carril.
Cada objeto de esta clase representa gráficamente una de las páginas que aparecen en la página de carriles.

El método *Genera_carril* crea los objetos necesarios, llamando a *Actualiza_tipo_carril* para obtener los tipos de carriles existentes. Este método lee los tipos de carriles existentes de la lista *Lista_tipo_carril.* Esta lista contiene objetos de la clase *Tipo_carril*, cuya clase se muestra a continuación:

```
class Tipo_carril
{
  unsigned int codigo;
  char nombre[20];
  char descripcion[50];
  float ancho;
public:
//métodos
};
```

La clase tiene varias propiedades en las que almacena información sobre los tipos de carril existentes. La propiedad *codigo* identifica a los tipos de carril dentro de la lista. Las propiedades *nombre* y *descripcion* son autodescriptivas. La propiedad *ancho* indica el ancho del carril.

El método *Destruir_formulario* libera la memoria usada por los objetos que representan gráficamente el formulario.

El método *Obtener_datos* almacena en el objeto de clase *Dato_carril* que se le pasa como parámetro toda la información del carril usando sus métodos, que extraen la información de los cuadros de diálogo del formulario.

Todos los demás métodos lo que hacen es obtener los datos correspondientes del formulario, excepto *Actualiza_datos*, que tiene la función contraria. Lo que hace es presentar en cada uno de los campos del formulario la información contenida en el objeto de la clase *Datos_carril* que se le pasa.

```
class Informacion_tramo
{
 TLabel *l_codigo;
 TLabel *l_descripcion;
 TLabel *l_funcion;
 TLabel *l_longitud;
 TLabel *l_volumen;
 TLabel *l_cola;
```

```
 TLabel *l_carriles;
 TLabel *l_carriles_der;
 TLabel *l_carriles_izq;
 TLabel *l_origen;
 TLabel *l_destino;
 TMemo *e_descripcion;
 TEdit *e_longitud;
 TEdit *e_funcion;
 TEdit *e_volumen;
 TEdit *e_cola;
 TEdit *e_carriles;
 TEdit *e_carriles_der;
 TEdit *e_carriles_izq;
 TEdit *e_origen;
 TEdit *e_destino;
 TCheckBox *activado;
 TGroupBox *GroupBox1;
 TGroupBox *GroupBox2;
 TBitBtn *Aceptar;
 TBitBtn *Modificar;
 TBitBtn *Borrar;
 TBitBtn *Cancelar;
public:
   void Generar_datos_tramos (TWinControl *padre,int Ancho_p,int Alto_p);
   void Destruir_formulario();
   void Actualiza_tramo (Tramo tramo);
   void __fastcall Borrar_click (TObject *Check);
   void __fastcall Modificar_click (TObject *Check);
   void __fastcall Aceptar_click (TObject *Check);
   void Actualiza_eventos();
   void Obtener_datos(Tramo *l,Datos_tramos *d);
};
```

Todas las propiedades de esta clase son objetos gráficos que se usan para representar el formulario de información de tramo.

El método *Generar_datos_tramos* se encarga de generar el formulario y de cargar la información desde disco a la lista *Lista_datos_tramos*. También llama a *Actualiza_eventos*.

El método *Destruir_formulario* graba en disco la información de la lista *Lista_datos_tramos* y después libera la memoria usada por esta lista y por todos los objetos gráficos creados.

El método *Actualiza_eventos* asigna los eventos *onClick* sobre los botones *Aceptar*, *Modificar* y *Borrar* a los métodos *Aceptar_Click*, *Modificar_Click* y *Borrar_Click*, respectivamente.

El método *Actualiza_tramo* presenta la información del objeto tramo en el formulario.

El método *Borrar_Click* borra el tramo de la lista *Lista_datos_tramos* y de la lista *Lista_tramos*. A continuación borra la imagen y representa la nueva.

El método *Modificar_Click* modifica el elemento de la lista *Lista_datos_tramos*.

El método *Aceptar_Click* modifica el elemento de la lista *Lista_datos_tramos* o lo crea si no existía.

El método *Obtener_datos* almacena los datos del formulario según corresponda en los objetos que se le pasan como parámetro.

Procedimiento de uso

Al pulsar el usuario sobre el botón que selecciona el interfaz de tramos se ejecuta la función *Activacion_selección_tramo*. Esta función asigna ciertas funciones a los eventos que puedan tener lugar a continuación sobre la imagen que representa la red vial. En concreto el comportamiento que se persigue es que al desplazar el ratón sobre la imagen aparezcan las coordenadas globales del punto sobre el que está el ratón en ese instante en la barra de estado de la aplicación. Si se pulsa el botón del ratón no pasa nada y cuando se libera el botón del ratón se llama al método *Seleccionar_tramo_inicial* de la clase *Contexto.*

Si el usuario estuviese ya en el interfaz de tramos, estos mismos pasos se siguen si se pulsa el botón representado por un dedo que aparece junto a la imagen que representa la porción de red vial seleccionada, la que aparece en la parte inferior derecha de la pantalla.

El método *Seleccionar_tramo_inicial* toma el tramo seleccionado, usando el método *Buscar_tramo* de la clase *Mapa* y a continuación llama al método *Generar_formulario* de la clase *Tramo_interactivo.*

Este método crea el interfaz gráfico de tramos, usando para ellos los métodos adecuados de las clases *Carril_interactivo, Pm_interactivo* e *Informacion_tramo*. Por supuesto también llama al método *Pinta_mapa* de la clase *Informacion_pantalla* para representar las dos imágenes que aparecen en la parte derecha del interfaz.

El método *Generar_formulario* de la clase *Carril_interactivo* se encarga de la representación gráfica de la página de carriles dentro del interfaz de tramos. Además se encarga de crear la lista *lista_tipo_carril* para poder presentar los tipos de carril existentes y *Lista_datos_carril* para cargar los datos de los carriles. Además llama a *Actualiza_eventos* y a *Actualiza_datos*, para actualizar los eventos sobre los botones de la imagen inferior derecha y representar los datos del carril correctamente en el formulario.

Una vez hecho esto, cuando el usuario seleccione la página de carriles este interfaz presenta toda la información relativa a carriles sobre el tramo seleccionado, aunque éste se haya seleccionado en otra página. Además, los eventos posibles, que son la interacción del usuario sobre la imagen o la pulsación de alguno de los botones del interfaz, tienen correctamente asignados los métodos correspondientes para que el interfaz se comporte de la manera adecuada.

El método llamado de la clase *Pm_interactivo* también es *Generar_formulario*. Este método de nuevo lo que hace es crear todos los objetos gráficos para representar el interfaz de medida.

Y finalmente el método que se llama de la clase *Informacion_pantalla* es *Generar_datos_tramos*. El método *Generar_datos_tramos* se encarga de generar la página de tramos y de cargar la información desde el fichero en disco a la lista *Lista_datos_tramos*. También llama a *Actualiza_eventos*.

De esta manera todas las diferentes páginas están listas para representar en cada caso la información adecuada del tramo seleccionado, aunque se haya seleccionado en una página distinta, y listas para ejecutar el código adecuado ante la ocurrencia de cualquier evento. Los eventos ante los que responde el interfaz son, como ya se dijo anteriormente, la selección de un tramo en la imagen inferior derecha tras pulsar el botón de selección de tramo y la pulsación de cada uno de los botones que aparecen en cada una de las páginas.