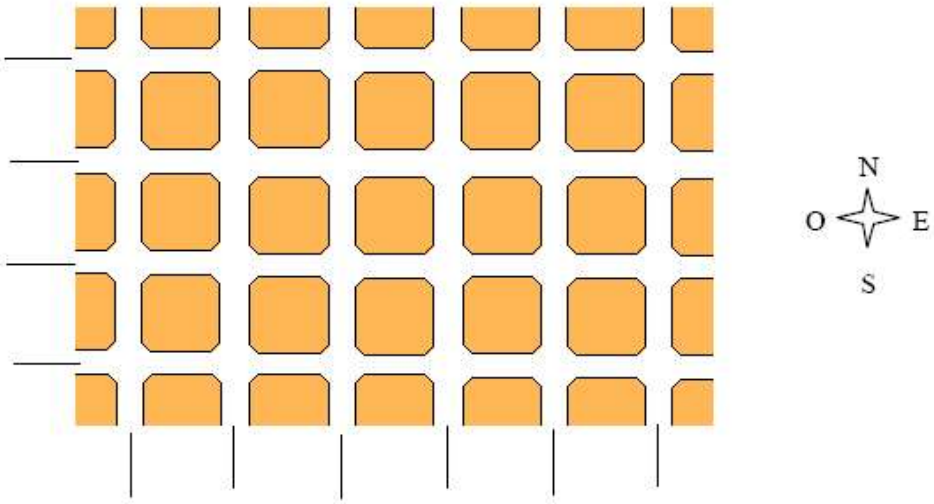
1. Enunciat

1.1. Objectiu

Implementació en Ada 95 d'un sistema de control del trànsit pels carrers d'una ciutat destinat a millorar la seva fluïdesa. Per a comprovar el seu funcionament cal fer també un simulador d'aquest trànsit.

1.2. Ciutat

- ✔ Suposarem que la ciutat té l'estructura següent:



- ✔ Formalment, la nostra ciutat està formada per N carrers orientats d'est a oest i M carrers orientats de nord a sud, que es creuen en $N \times M$ cruïlles.

- ✔ Suposarem que tots el carrers són d'un sol sentit de circulació, disposen de dos carrils, i satisfan condicions de contorn periòdiques: quan un vehicle arriba al final d'un carrer torna a aparèixer per l'altre extrem del mateix carrer.

- ✔ Els sentits de circulació estan alternats, per la qual cosa és convenient prendre N i M parells.

1.3. Vehicles

- ✔ Existeixen quatre tipus de vehicles: cotxe, ciclomotor, camió i bicicleta. Cadascun d'ells està caracteritzat per la seva velocitat màxima dins de la ciutat.

- ✔ Com els carrers disposen de dos carrils, els vehicles més ràpids poden avançar als més lents.

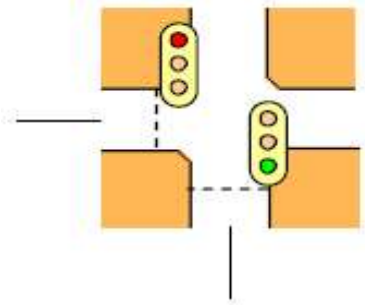
- ✔ Cada vegada que un vehicle arriba a una cruïlla decideix si segueix recte o gira (segons la cruïlla en qüestió serà a esquerra o a dreta). Evidentment, s'haurà d'aturar si el semàfor està en vermell, i no podrà continuar la marxa fins que es posi verd.

- ✔ Tots els vehicles tenen una certa probabilitat d'espatllar-se, quedant-se aturats. En aquest cas la grua (suposem que és de tipus camió i que no s'espatlla mai), alertada pel sistema de control del trànsit, sortirà del garatge, trobarà el vehicle espatllat, el portarà al taller de la ciutat i retornarà al seu garatge.

- ✔ El camió d'escombraries (també suposem que no s'espatlla mai) fa periòdicament, cada T_{CE} hores, un recorregut per tots els carrers de la ciutat, aturant-se durant X_{CE} minuts davant de cada contenidor d'escombraries, que estan situats en les voreres de la dreta (segons el sentit de circulació) a mig camí entre cada parell de cruïlles consecutives.

1.4. Cruïlles i semàfors

- ✔ A cada cruïlla hi ha un parell de semàfors en fases oposades: quan un està verd l'altre està vermell.



- ✔ Suposem que el semàfor corresponent al carrer orientat d'est a oest inicia la seva fase verda a l'instant t_0, i que T és el temps que triga a tornar a iniciar la propera fase verda (es diu que T és el període d'aquest semàfor). Aleshores, suposant que les fases en verd duren igual que les fases en vermell, els instants d'inici d'aquestes fases seran:

	t_0	$t_0 + T/2$	$t_0 + T$	$t_0 + 3T/2$	$t_0 + 2T$
Est/Oest	verd	vermell	verd	vermell	verd
Nord/Sud	vermell	verd	vermell	verd	vermell

- ✔ Inicialment els valors de t_0 i de T han de ser els mateixos per a totes les cruïlles.

1.5. Control del trànsit

- El sistema de control del trànsit té per objectiu millorar la fluïdesa del trànsit, és dir, minimitzar els temps d'espera dels vehicles en els semàfors, i retirar de les vies públiques els vehicles espatllats.

- Per a fer la seva tasca el sistema de control pot consultar en qualsevol moment l'estat (verd o vermell) de tots els semàfors i el número de vehicles aturats en cadascun d'ells.

- L'actuació del sistema de control sobre el trànsit es fa únicament avançant o endarrerint el moment de transició de fase dels semàfors de les cruïlles, és dir, modificant els valors de t_0. Es poden modificar per separat i de forma diferent els valors de t_0 de cadascuna de les cruïlles.

- Haureu de definir indicadors que permetin valorar la fluïdesa del trànsit en cada moment, i com aquesta evoluciona: estem realment millorant amb el nostre sistema de control la fluïdesa del trànsit?

1.6. Interfície gràfica

- Permet modificar els paràmetres del simulador i del control.

- Mostra de forma gràfica tota la informació rellevant: sentit de circulació dels carrers, estat dels semàfors, posició dels vehicles, tipus i estat dels vehicles (circulant, aturat en un semàfor, espatllat), indicació de les actuacions del control, valoració de la fluïdesa del trànsit i la seva evolució, etc.

- Tota aquesta informació s'ha d'anar actualitzant contínuament, intentant, però, que interfereixi el mínim possible en la resta d'aspectes de la simulació.

- S'ha d'implementar utilitzant JEWL, GWindows o GtkAda. No es pot fer servir AdaGraph, ja que no funciona correctament en moltes de les versions del sistema operatiu Windows (principalment perquè no està preparat per a treballar en sistemes concurrents).

1.7. Paràmetres

- Unitat de temps de simulació (UTS): si volem que un segon davant de la pantalla representi mig minut de vida a la nostra ciutat, fem UTS=30. Mentre més gran sigui UTS més ràpida serà la simulació.

- Tamany de la ciutat: N, M.

- Distància entre les cruïlles (en metres).

- Velocitats màximes dels diferents vehicles dins de la ciutat (en Km/h).

- ✔ Nombre inicial de vehicles de cada tipus.
- ✔ Probabilitat que un vehicle segueixi recte a l'arrivar a una cruïlla (e.g. P=0.7).
- ✔ Distància mitjana recorreguda per cada tipus de vehicle abans d'espatllar-se, i la seva desviació estàndard (e.g. d(camió)=5000m ± 100m).
- ✔ Camió d'escombraries: TCE, XCE.
- ✔ Tots els paràmetres han de ser configurables des de la interfície gràfica, i han de portar valors per defecte per no haver d'introduir- los tots cada vegada.

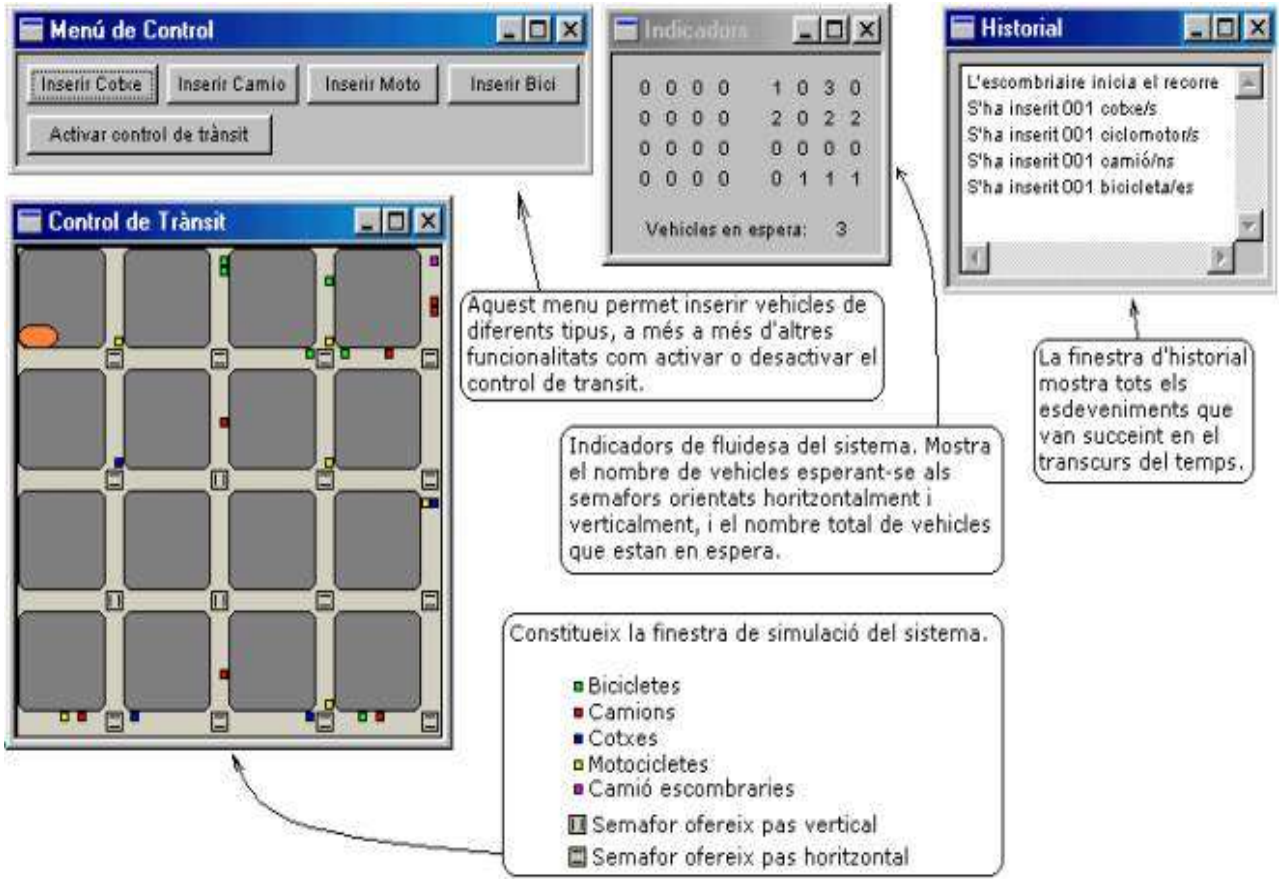
2. Manual d'usuari

2.1. Software utilitzat

A continuació s'explica el conjunt d'eines que hem utilitzat per al desenvolupament de la pràctica, i per tant, són necessàries per a la compilació d'aquesta. El programa ha estat implementat en GNAT Ada 95 en un sistema operatiu Windows 98. Per a poder implementar les operacions que fan possible el disseny de l'entorn gràfic del programa, s'han utilitzat les llibreries gràfiques JEWL 1.6. Tot i que el programa ha estat desenvolupat sota un sistema Windows 98, s'observa una millora en el rendiment d'aquest quan s'executa en un sistema Windows XP o superior. Per tant, es recomana usar aquest programa en un sistema operatiu Windows XP.

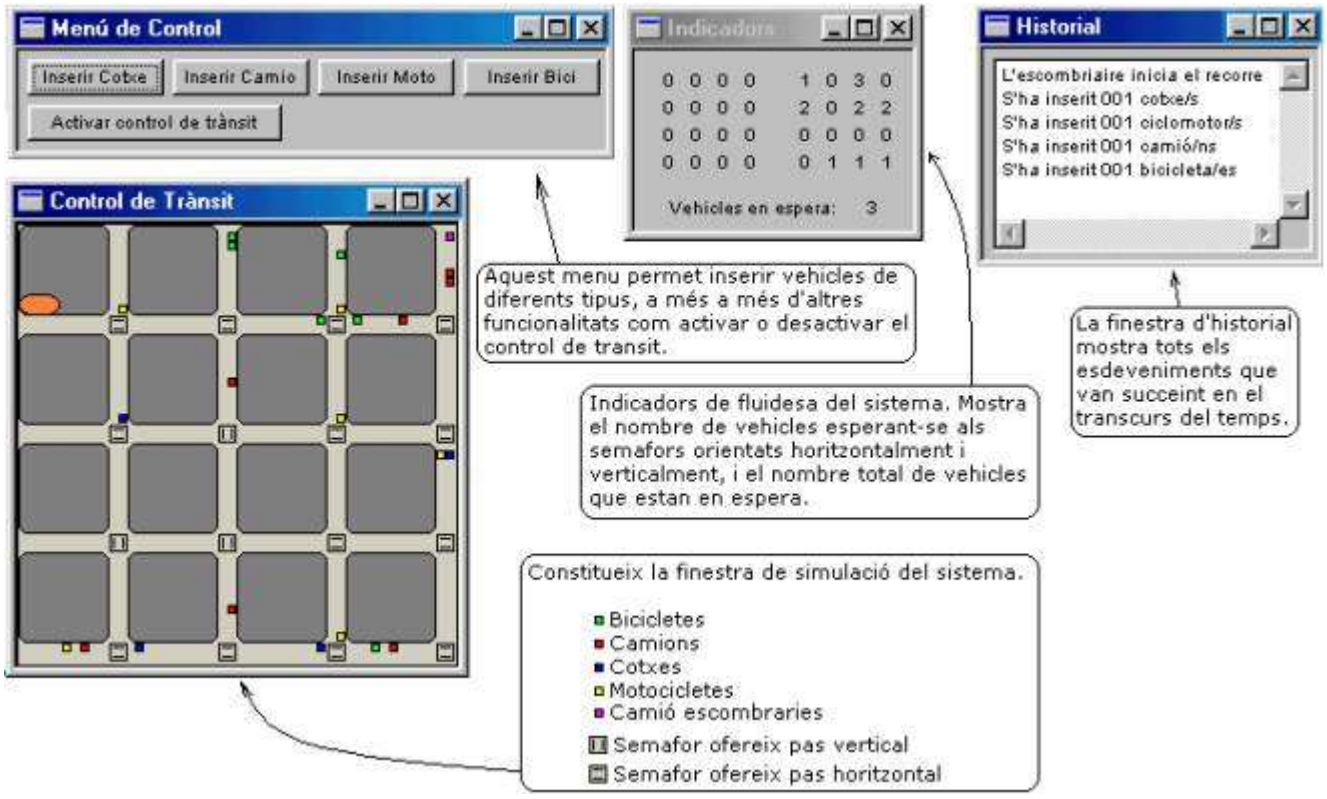
2.2. Descripció i utilització del programa

Inicialment el programa demana a l'usuari que insereixi els paràmetres de configuració de l'aplicació. Tots aquests paràmetres són configurables per mitjà d'una finestra que incorpora un formulari amb totes les opcions possibles, inicialment el formulari ja porta tots els camps amb valors per defecte per no haber-los d'introduïr tots de nou a cada execució. La següent figura mostra detalladament la finalitat de cadascun dels camps del formulari:



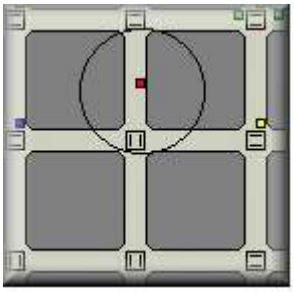
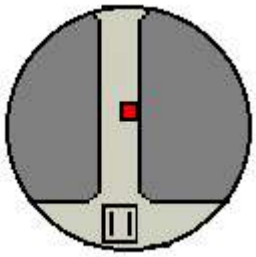
Finalment, un cop editats els camps convenients, el programa ja està llest per posar-se en marxa, i per tant, ja es pot clicar al botó *iniciar*.

Al clicar damunt del botó *iniciar*, el sistema es començarà a posar en marxa, i es començaran a crear esglaonadament tots els components que el formen: carrers, semàfors, vehicles, tasques de control i monitorització, etc... Finalment han d'aparèixer quatre finestres: la finestra de simulació, la finestra d'indicadors de fluidesa, la finestra del menú de control i la finestra del historial.

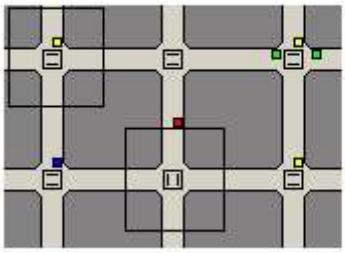
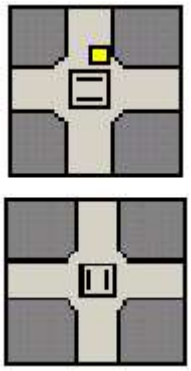


La finestra de simulació representa de forma gràfica el comportament de tots els components que composen l'entorn. Els vehicles es mostren en forma de petits quadres de colors que es desplacen pel mapa, cada tipus de vehicle té associat un color diferent:

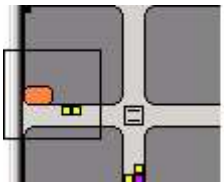
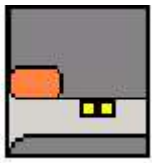
Els vehicles simulen el comportament d'un vehicle del món real, en aquest sentit, cal dir que hi ha tipus de vehicles que corren més que d'altres, hi ha vehicles que efectuen abançaments a altres vehicles, els vehicles s'aturen quan tenen el corresponent semàfor en vermell, reanuden la marxa altrament, s'espatllen al superar un determinat kilometratge, etc... Els vehicles es mouen a trabés dels carrers, exactament a trabés de trams. Considerem *trams* les regions de carretera compresses entre dos cruïlles. En aquest sentit cada tram està compost per dos carrils del mateix sentit de circulació, un dels dos carrils està dissenyat per efectuar abançaments a altres vehicles. Cada tram té associat un semàfor al final d'aquest.

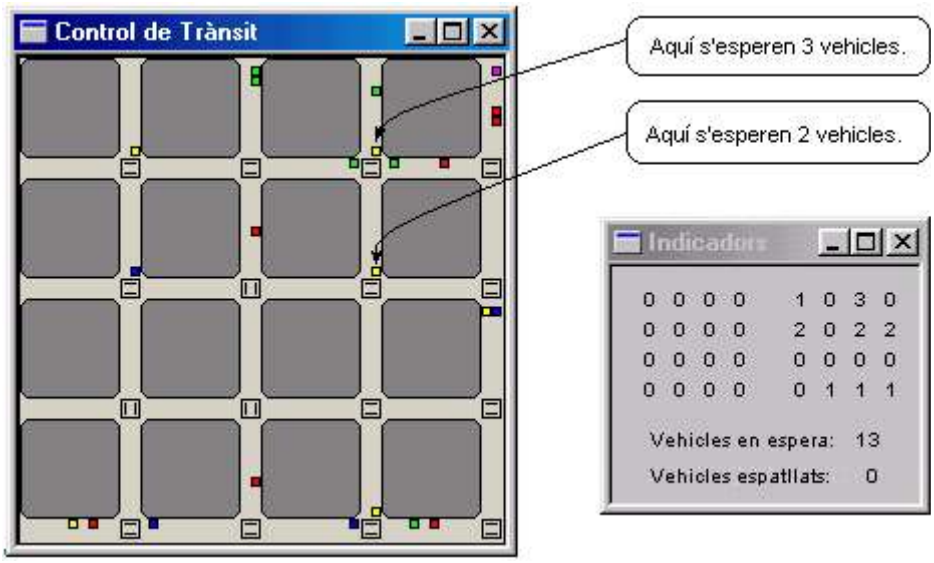
A la figura anterior es mostra un tram orientat verticalment amb un vehicle de tipus camió circulant-hi. Al final del tram s'hi troba un semàfor que regula el pas dels vehicles de procedència vertical i horitzontal. La representació gràfica dels semàfors consta de dos estats: oferint pas vertical i oferint pas horitzontal:

A la figura anterior s'observen els dos casos, en primer lloc tenim un semàfor que dóna pas als vehicles que circulen horitzontalment i en segon lloc tenim un semàfor que dóna pas als vehicles que circulen verticalment, independentment del sentit de la circulació del carrer en ambdós casos. Per altra banda al mapa hi ha un taller on la grua diposita els vehicles espatllats que recull, el taller sempre es troba al principi del primer tram horitzontal del mapa i constitueix una petita regió de color marró clar, tal i com mostra la següent figura:

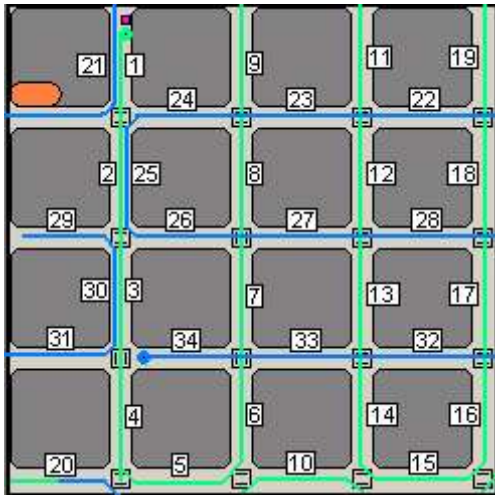



La finestra d'indicadors de fluidesa mesura la fluidesa del trànsit. Consta de dos matrius de números que indiquen el nombre de vehicles que s'estan esperant en aquell semàfor. La matriu esquerra fa referència a les cues de vehicles que s'estan esperant als trams orientats horitzontalment, la matriu dreta fa referència a les cues de vehicles que s'estan esperant als trams orientats verticalment. Mostra també el total de vehicles que s'estan esperant i el nombre de vehicles espatllats. Aquests valors són actualitzats en intèrvals de temps de 0.05s, és a dir, una freqüència de mostreig de 20Hz.



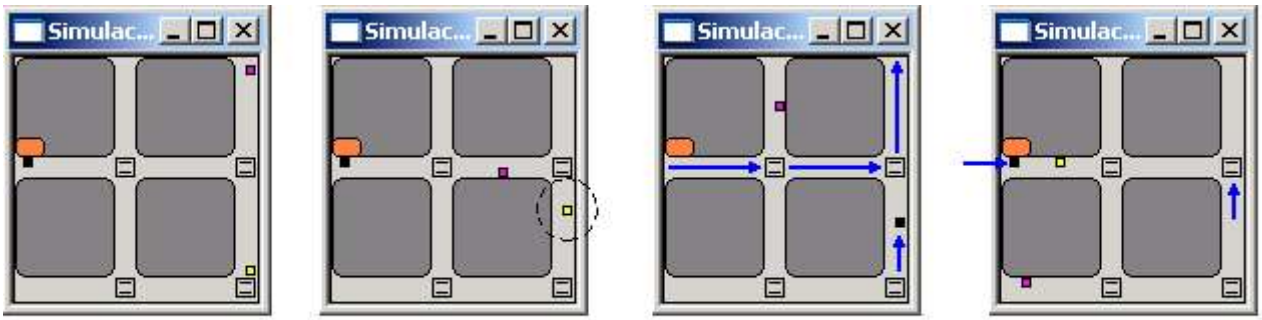
La finestra del menú de control permet una mínima interacció amb el sistema en temps d'execució. Les operacions que es poden efectuar són la insersió de nous vehicles i l'activavió i desactivació del control de trànsit. Els vehicles quan s'insereixen apareixen pel primer tram horitzontal del mapa, just al costat del taller. La finestra de l'historial manté un llistat de tots els esdeveniments que van succeeïnt al sistema, conservant l'ordre en que aquests apareixen. El camió d'escombraries efectua un recorregut per tots els carrers del mapa.

A continuació s'explica a grans trets el procediment que realitza per recorrer-los tots, a les pròximes pàgines es descriurà més detalladament com es porta a terme. El camió d'escombraries sempre parteix del tram vertical superior esquerra del mapa per iniciar el seu recorregut. En primer lloc recorre els carrers verticals i finalment recorre els carrers horitzontals. La següent figura en mostra el recorregut per a un mapa de 4 per 4 carrers:



Quan el camió d'escombraries no pot recòrrer tots els carrers dins del temps assignat, aleshores aborta el recorregut en curs i torna a iniciar-lo desde la posició inicial de sortida. Altrament, quan aquest finalitza el seu recorregut dins del termini de temps, aleshores s'atura i s'espera a que torni a produïr-se la nova ordre de sortida per tornar a sortir desde la posició inicial.

Per altra banda hi ha la grua que només es posa en funcionament quan el control de trànsit està activat. La grua surt del taller i localitza el primer vehicle espatllat que es troba a la seva cua, és a dir, a la cua de vehicles espatllats que soliciten sér remolcats fins al taller. Un cop la grua ha cercat el vehicle, aquest desapareix del mapa i la grua es dirigeix cap al taller. Quan la grua arriba al taller, el vehicle espatllat és arreglat automàticament i torna a formar part de la circulació, acte seguit la grua procedeix a la cerca del següent vehicle de la cua o es queda aturada en cas que no hi hagi vehicles espatllats als quals remolcar.



La grua s'espera a que s'espatlli un vehicle.

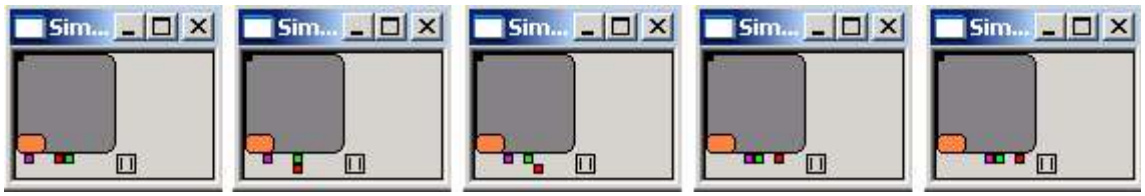
Un vehicle s'ha espatllat i solicita els serveis de la grua.

La grua localitza el vehicle i aquest desapareix visualment del mapa.

La grua torna al taller a deixar el vehicle remolcat. El vehicle és reparat i torna a circular.

Si el control de trànsit és desactivat en el moment en que la grua està efectuant una cerca o remolc d'algun vehicle, eleshores la grua aturarà els seus serveis quan hagi finalitzat les seves operacions en curs. La velocitat en que la grua cerca i remolca els vehicles dependrà de la velocitat assignada als vehicles de tipus camió, ja que aquesta es considera un vehicle amb característiques de la categoria dels camions.

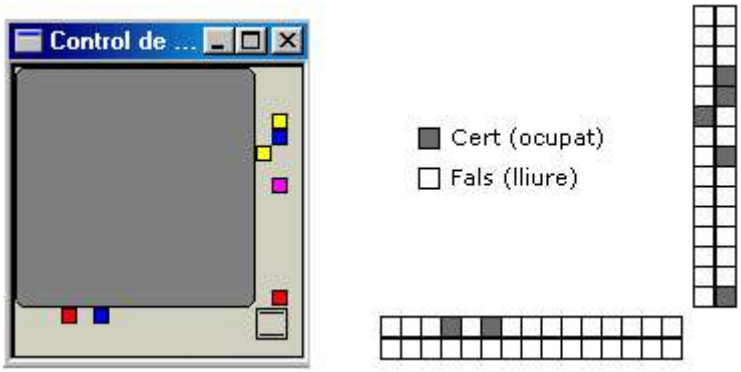
Els avançaments eviten que els vehicles més lents bloquegin el pas als més ràpids. Per a poder fer possible això s'utilitza el carril d'abançament. Un bon ús dels abançaments és possible gràcies al comportament tolerant amb el que han estat programats els vehicles ja que sinó es produirien abançaments que no acabarien mai. Els abançaments que no acaben mai són aquells en que el vehicle abançat i vehicle abançador es mouen a la mateixa velocitat i per tant circulen paral·lelament en carrils diferents. Per evitar aquest problema els vehicles abançats redueixen la seva velocitat per permetre que el vehicle abançador pugui efectuar l'abançament amb èxit.



3. Decisions de disseny

3.1. Trams

Els trams són els fragments de carrer que hi ha entre cruïlles. En aquests hi circulen els vehicles que constantment estan demanant la següent posició dins del tram en el que es troben. En aquest sentit, els trams controlen i serveixen posicions als vehicles que hi estan circulan, per tant, hem definit els trams com a objectes protegits que controlen l'accés a les posicions ocupades i lliures dels carrils que el composen. Per a simplificar el problema, els trams només saben quines posicions (metres quadrats) estan essent ocupades o lliures pels vehicles, per a això s'usen dos vectors de booleans. Per altra banda, són els propis vehicles els que coneixen en quina posició i carril es troben dins del tram per on circulen.



Els atributs i mètodes que encapsula el tipus protegit *Tram* es descriuen a continuació:

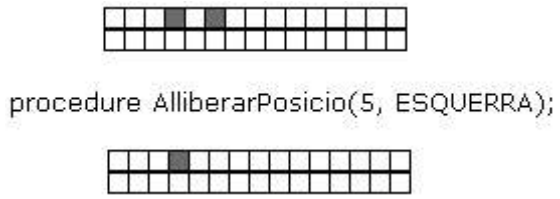
```
Sentit: Ttipussentit;      -- Sentit del tram (NordSud, SudNord, etc...)
Tipus: Ttipustram;         -- Tipus de tram (Vertical o Horitzontal)
Longitud: Natural;         -- Longitud en metres del tram
Carrildret: Ptcarril;      -- Vector de booleans de longitud Longitud
Carrilesquerra: Ptcarril; -- Vector de booleans de longitud Longitud
Semafor: Pparellsemafor;   -- Semàfor associat al tram
Posicio: Tcoordenada;      -- Coordenada gràfica del tram dins del mapa
Adreça:  Tcoordenada;      -- Coordenada lògica del tram (0..M,0..N)
```

```
procedure Inicialitzar (...);
```

El procediment *Inicialitzar* inicialitza els atributs de l'objecte. S'usa a l'inici de l'aplicació quan es modela i creen els diferents trams que composen el sistema.

```
procedure AlliberarPosicio(Pos: in Integer; Carril: in Ttipuscarril);
```

Allibera una posició concreta del tram. S'usa principalment quan els vehicles s'espatllen i han d'alliberar la seva posició per a no bloquejar el pas als altres vehicles.

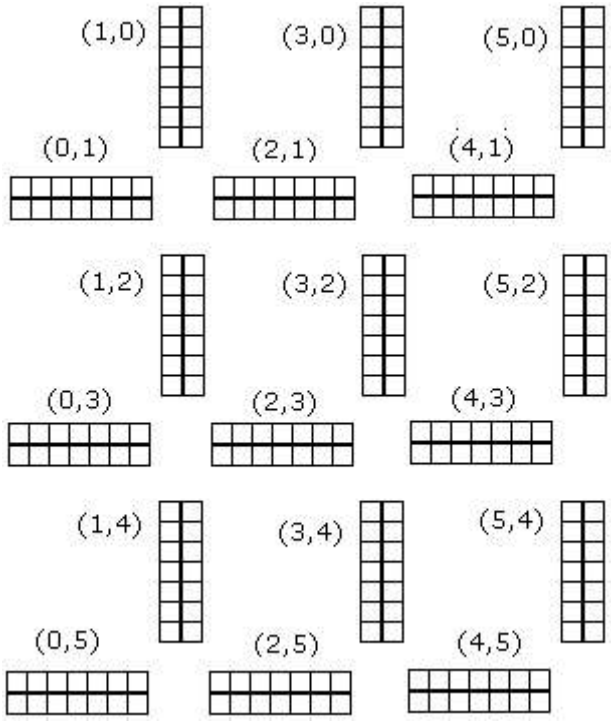



```
function ObtenirTipusTram return TtipusTram;
```

Aquesta funció retorna el valor de l'atribut *Tipus*.

```
function ObtenirAdreça return Tcoordenada;
```

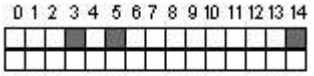
Aquesta funció retorna el valor de l'atribut *Adreça*. Aquest valor correspon a la posició lògica del tram al mapa i es utilitzat per la tasca grua per poder fer les localitzacions dels vehicles espatllats i del taller. La següent figura mostra les assignacions d'adreces lògiques als trams d'un mapa de 3 per 3 carrers:



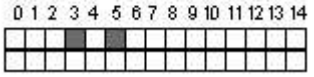
```
procedure ObtenirSeguentPosicio (Pos: in out Integer; Carril: in out Ttipuscarril;
C: out Tcoordenada; Fitram : in out Boolean; Migtram: out Boolean; Sem: out
Pparellsemafor; Tipustram: out Ttipustram);
```

És el procediment més important d'aquest tipus protegit. Els vehicles quan volen abançar una posició (metre) criden aquest procediment en el tram per on circulen. *ObtenirSeguentPosicio* comprova que el vehicle que es troba a la posició *Pos* del carril *Carril* pugui abançar una posició en aquest. Es poden produïr diverses cituacions que s'expliquen a continuació.

- Quan el vehicle està al final del tram:

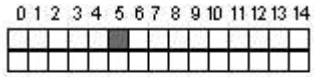


ObtenirSeguentPosicio (14, ESQUERRA, C, Fitram, Migtram, Sem, Tipustram);

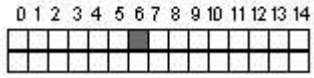


El vehicle de la posició 14 del carril esquerra vol avançar, *Fitram* pren per valor cert, *TipusTram* pren per valor el tipus del tram i *Sem* apunta al semàfor associat al tram. A partir d'aquí el vehicle s'encua a la cua d'espera del semàfor *Sem*.

- Quan el vehicle està al carril esquerra i la posició de davant està lliure:

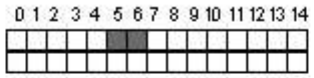


ObtenirSeguentPosicio (5 , ESQUERRA, C, Fitram, Migtram, Sem, Tipustram);

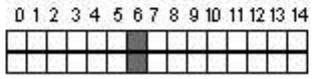


El vehicle avança a la posició de davant. *Pos* incrementa en una unitat.

- Quan el vehicle està al carril esquerra i la posició de davant està ocupada per un altre:



ObtenirSeguentPosicio (5 , ESQUERRA, C, Fitram, Migtram, Sem, Tipustram);

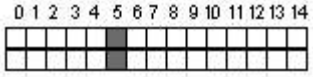


En aquest cas s'efectua un avançament. *Carril* pren per valor *dreta* i s'incrementa *Pos*.

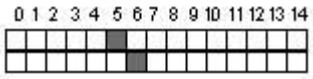
- Quan el vehicle està a l'esquerra i a la mateixa posició del carril dret hi ha un vehicle.

En aquest cas no es fa res ja que, tal i com s'ha explicat a l'apartat del manual d'usuari, es deixa que el vehicle avançador acabi d'efectuar l'avançament amb èxit. Així s'evita que hi hagi vehicles que es desplacen a la mateixa velocitat i circulin paral·lelament i per tant bloquegin el pas a altres vehicles.

- Quan el vehicle està a la dreta i a la mateixa posició del carril esquerra hi ha un vehicle.

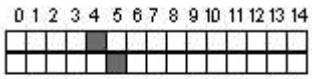


ObtenirSeguentPosicio (5 , DRETA , C, Fitram, Migtram, Sem, Tipustram);

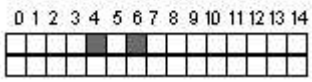


Aquest cas vol dir que el vehicle està efectuant un avançament i, per tant, ha de continuar avançant pel carril dret fins aconseguir trobar el carril esquerra lliure.

- Quan el vehicle està a la dreta i a la mateixa posició més u del carril esquerra està lliure.



ObtenirSeguentPosicio (5 , DRETA , C, Fitram, Migtram, Sem, Tipustram);



En aquest cas el vehicle avançador s'incorpora de nou al carril esquerra. Per tant, *Pos* s'incrementa en una unitat i *Carril* pren per valor *Esquerra*.

```
procedure DemanarSituarseAlCarrilEsquerra (Pos: in out Integer, C: out Tcoordenada,
Carril: in out Ttipuscarril );
```

Aquest procediment permet al vehicle situar-se al carril esquerra sempre que sigui possible, si no es possible el vehicle haurà de tornar a cridar aquest mètode més tard. Només és usat pel camió d'escombraries per situar-se al carril esquerra per recollir les escombraries dels containers.

3.2. Package MapeigEntorn

Aquest *package* conté procediments encarregats de crear i mapejar els elements estàtics que composen el mapa: trams i semàfors. Aquests s'han de situar a la seva posició lògica dins del mapa, per a fer-ho hem usat dos matrius de trams i una matriu de semàfors que permeten treballar amb més facilitat amb aquests elements. Per una banda hi ha la matriu de trams verticals i per l'altra la matriu de trams horitzontals:

```
TramsHoritzontals: PTMatriuTrams;
TramsVerticals: PTMatriuTrams;
```

Amb aquestes dos matrius es pot accedir a l'apuntador d'un determinat tram en funció de la seva posició lògica. De la mateixa manera també es disposa d'una matriu per als semàfors, que permet accedir a un determinat semàfor en funció, també, de la seva posició lògica.

```
Semafors: PTMatriuParellSemafors;
```

Els procediments que el formen són els següents

```
procedure CrearTrams(M: in Integer; N: in Integer);
```

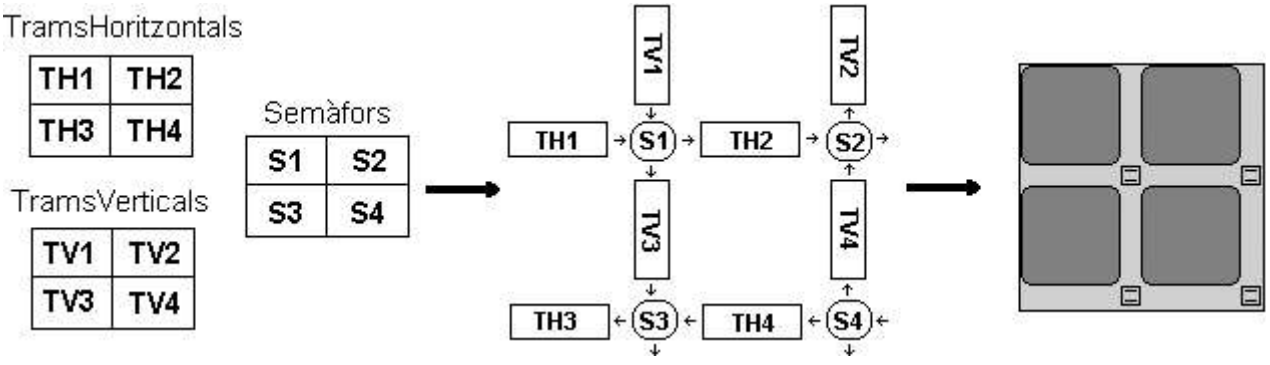
Crea les dos matrius d'apuntadors de trams i els corresponents trams.

```
Procedure CrearSemafors(M: in Integer; N: in Integer; Cua: in Pqueue; Tsem: in
Integer; uts: in Integer);
```

Crea la matriu d'apuntadors de semàfors i els corresponents semàfors.

```
Procedure Inicialitzar_Trams_i_Semafors(M: in Integer; N: in Integer; llarg: in
Integer; d: in PDibuixant; Cua: in Pqueue; MotorGrafic: in TipusMotorGrafic);
```

Inicialitza els trams i semàfors de les matrius. Concretament, assigna semàfors, coordenades, sentits de circulació, etc... als trams. I també assigna trams als semàfors.



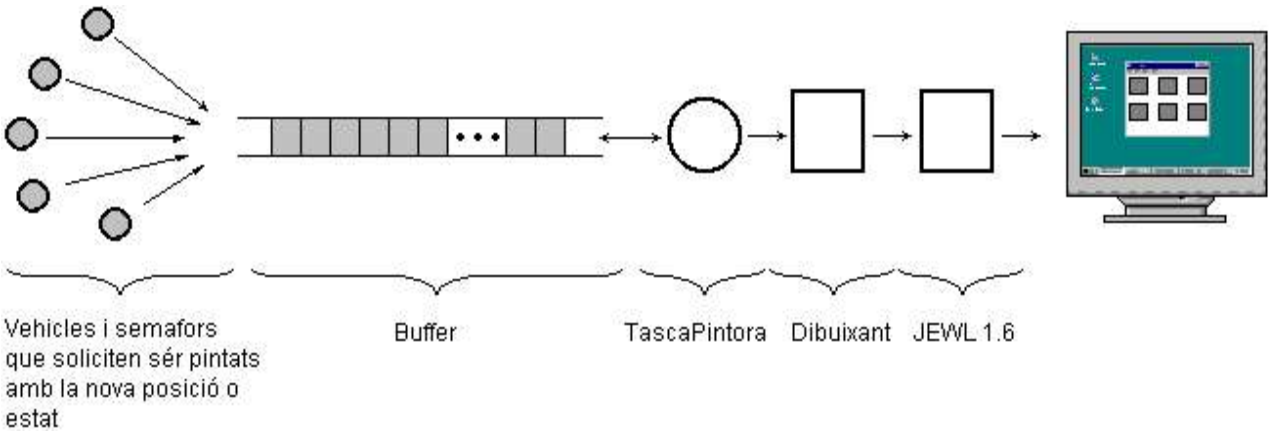
3.3. Tasca pintora (Motor gràfic basat en el sistema de buffering)

Aquesta tasca només s'usa en el cas en que l'usuari hagi escollit el motor gràfic basat en el sistema de *buffering*. Aquest mètode permet pintar els elements dinàmics del mapa sense interferir amb el funcionament del sistema. Per a fer possible això s'usa un buffer implementat en el package *QueueArray* que permet escriure i llegir les commandes de pintar per pantalla que efectuen els vehicles i semàfors. Amb aquest sistema, el temps de còmput que necessiten les llibreries gràfiques (JEWL 1.6) per pintar els objectes per pantalla no interfereix (bloqueja) amb el funcionament desitjat de les tasques. Aquesta tasca pintora llegeix constantment del buffer, i mitjançant crides als corresponents mètodes de l'objecte protegit *Dibuixant* es realitzen els canvis per pantalla.

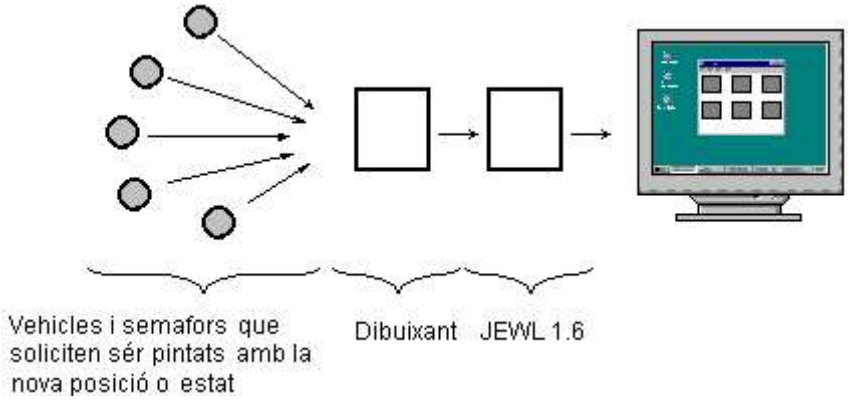
Aquest sistema és adient per a màquines lentes que no poden pintar per pantalla els esdeveniments en un temps acceptable. En aquests casos el buffer esdevé una via de sortida per a aquest problema. Aquesta tasca és auto-adaptativa en el sentit de que és ella mateixa la que regula la freqüència amb la que llegeix elements del buffer, això permet reduïr el nombre de crides al mètode *dequeue* del buffer en el cas que hi hagi pocs elements al buffer. En el cas que el buffer estigui molt carregat augmenta la freqüència de lectura.

Per altra banda, els vehicles i semàfors insereixen les seves peticions de pintar-se al buffer, sempre que s'hagi escollit el sistema gràfic basat en *buffering*, altrament aquests usen directament els mètodes de l'objecte *Dibuixant*.

Esquema mètode basat en el sistema de Buffering



Esquema mètode basat en el sistema de pintat directe



3.4. Dibuixant

Dibuixant és un tipus protegit que s'encarrega d'usar la llibreria JEWL 1.6 per pintar els elements del sistema: vehicles, semàfors i edificis. Abans d'explicar els mètodes i atributs que encapsula és convenient explicar la forma que s'ha usat per utilitzar adientment les llibreries JEWL per representar els vehicles i els semàfors. Un dels grans problemes que presenta JEWL és que cada vegada que s'efectua una operació sobre un canvas, es tornen a repintar totes les operacions efectuades anteriorment sobre aquest. Aquest problema s'agreuja quan ja si han efectuat moltes operacions, ja que aleshores decrementa el rendiment del sistema i es produeix l'efecte de parpalleig de la pantalla.

Per resoldre aquest problema s'ha usat un llistat de canvas per als vehicles i per als semàfors. D'aquesta manera cada vehcicle i semàfor té associat un canvas propi, i l'accés a aquests es fa mitjançant l'identificador dels vehicles i dels semàfors. Per a fer possible això, s'usen dos vectors de canvas:

```
Vehicles: PTLlistaVehicles;
Semafors: PTLlistaSemafors;
```

Per exemple, el vehicle amb identificador 5 té assignat el canvas apuntat per *Vehicles(5)*. Aquesta forma de tractar les imatges evita haver de redibuixar cada vegada els elements que canvien d'estat o que es mouen pel mapa. En aquest sentit, es suficient, canviar la posició d'un determinat canvas dins del seu contenidor per tal de simular el moviment d'un objecte. En el cas dels semàfors només s'ha de restaurar la imatge a una altra d'inicial o redibuixar-ne una altra de nova.

L'ús d'aquests vectors és exclusiu dels tipus protegits *Dibuixant*. Els atributs que encapçula són els següents:

```
NumVehicles: Integer; -- Nombre de canvas destinats als vehicles
NumSemafors: Integer; -- Nombre de canvas destinats als semàfors
Vehicles: PTLlistaVehicles; -- Llista de canvas dels vehicles
Semafors: PTLlistaSemafors; -- Llista de canvas dels semàfors
Finestra: Frame_Type; -- Contenidor on es troben els canvas
Superficie: Canvas_Type; -- Canvas destinat a pintar els edificis
```

El motiu d'haver usat un tipus protegit per a efectuar les operacions que es comenten a continuació és degut a que l'ús concurrent de les llibreries JEWL no ha funcionat correctament, es a dir, es produexien errors en la forma en que es pinten els objectes, es produeixen cituacions de bloqueig per culpa de JEWL, etc... Els mètodes encapçulats són els següents:

```
Procedure Inicialitzar(t: in String; num: in Integer; l: in Integer; N: in Integer;
M: in Integer; nums: in Integer);
```

Inicialitza els atributs de l'objecte. Crea la finestra de simulació i pinta els edificis del mapa. També inicialitza les matrius de *Vehicles* i *Semafors*.

```
Procedure PintarEdificis(N: in Integer; M: in Integer; ample: in Integer);
```

Pinta els edificis al mapa.

```
procedure InserirVehicle(id: in Integer; tipus: in TTipusVehicle);
```

Reserva el canvas que es troba a la posició *id* del vector *Vehicles* per al vehicle amb identificador *id*. El canvas en qüestió és colorejat en funció del tipus de vehicle definit per la variable *tipus*.

```
Procedure InserirSemafor(id: in Integer; C: in TCoordenada);
```

Reserva el canvas que es troba a la posició *id* del vector *Semafors* per al semàfor amb identificador *id*. El canvas en qüestió és posicionat a la coordenada *C* del contenidor.

```
Procedure ActualitzarImatgeVehicle(id: in integer; C: in TCoordenada; tipus: in
TTipusVehicle);
```

Canvia la posició del canvas que es troba a la posició *id* del vector *Vehicles* a la nova coordenada *C*.

```
Procedure ActualitzarImatgeSemafor(id: in integer; t: in boolean);
```

Canvia la imatge del semàfor *id* en funció del seu estat *t*.

3.5. ParellSemafor

El tipus protegit *ParellSemafor* constitueix el funcionament bàsic dels semàfors. Funciona amb concordància amb la seva corresponent tasca tipus *TimerParellSemafor*. ParellSemàfor consta dels següents mètodes:

```
procedure Inicialitzar (Ident: in Integer; Tv: in Ptram; Th: in Ptram; Dib: in
Pdibuixant; Cua: in Pqueue; Tmg: Tipusmotorgrafic);
```

Inicialitza els atributs del semàfor. *Ident* és l'identificador del semàfor, *tv* és el tram per on sortiran els vehicles que desitgin circular en sentit vertical, *th* és el tram per on sortiran els vehicles que desitgin circular en sentit horitzontal, *Dib* és l'objecte dibuixant que s'usarà en cas d'haver escollit el tipus de motor gràfic basat en el pintat directe, *cua* és el buffer que s'usarà en cas d'haver escollit el tipus de motor gràfic basant en el sistema de buffering, i *tmg* és el tipus de motor gràfic que ha escollit l'usuari.

```
procedure ActivarModeFluidesa;
```

Activa el mode de fluidesa del semàfor. Aquest procediment és usat només per la tasca de control de trànsit quan aquest s'activa.

```
procedure DesactivarModeFluidesa;
```

Desactiva el mode de fluidesa del semàfor. Aquest procediment és usat només per la tasca de control de trànsit quan aquest es desactiva.

```
procedure CanviarTorn;
```

Aquest procediment és usat exclusivament per la tasca *TimerParellSemafor* associada al semàfor en qüestió. La tasca crida aquest mètode cada cop que s'ha de canviar el torn del semàfor.

```
Procedure ObtenirNumeroVehiclesEnEspera(H: out Integer, V: out Integer) ;
```

Aquest mètode és cridat únicament per la tasca *TascaControlIndicadors* per poder mostrar per pantalla el nombre de vehicles que s'estan esperant a cada semàfor. Per tant, *H* pren per valor el nombre de vehicles que s'estan esperant en sentit horitzontal, i *V* pren per valor el nombre de vehicles que s'estan esperant en sentit vertical. Per a poder fer això es consulta el nobre de crides encuades a la entry *DemanarPasVertical* i a la entry *DemanarPasHoritzontal* respectivament, mitjançant les comandes `Demanarpasvertical'Count` i `Demanarpashoritzontal'Count` de l'Ada95.

```
entry DemanarPasVertical (Des : in Tdesicio, T : out Ptram)
when ((Torn and not Mode_Fluidesa) or
(Torn and (Demanarpasvertical'Count > 0) and Mode_Fluidesa) or
(Torn and (Demanarpasvertical'Count = 0) and (Demanarpashoritzontal'Count = 0) and
Mode_Fluidesa) or (not Torn and Demanarpasvertical'Count > 0) and
Demanarpashoritzontal'Count = 0) and (Mode_Fluidesa)));
```

L'entry *DemanarPasVertical* permet als vehicles del tram vertical encuar-se per poder seguir recte o girar, en funció del valor de la variable *Des*. Quan es satisfà la condició de la guarda, el vehicle despatxat que ha demanat el pas circularà pel tram *T*. El tram *T* dependrà de la desició *Des* que hagi escollit el vehicle.

```
entry DemanarPasHoritzontal (Des : in Tdesicio; T: out Ptram)
when((not Torn and not Mode_Fluidesa) or (not Torn and (Demanarpashoritzontal'Count > 0) and
Mode_Fluidesa) or (not Torn and (Demanarpashoritzontal'Count = 0) and
(Demanarpasvertical'Count = 0) and Mode_Fluidesa) or (Torn and (Demanarpashoritzontal'Count
> 0) and (Demanarpasvertical'Count = 0) and (Mode_Fluidesa)));
```

L'entry *DemanarPasHoritzontal* és exactament igual que en el cas anterior, excepte que en aquest cas s'hi encuen els vehicles del tram horitzontal.

Aquestes guardes tenen la funció de protegir el torn del semàfor. Aquest torn ha de canviar cada vegada que arriba el deadline (període) del semàfor corresponent. El torn també pot canviar quan està activat el control de trànsit i es satisfan les condicions necessàries. Aquestes condicions són les següents: quan el semàfor que està verd no té cap vehicle encuat i en l'altre semafor de la cruïlla (que està en roig) tenim vehicles encuats, llavors hem de canviar el torn dels semàfors de la cruïlla.

3.6. Tasca vehicle

Tots els vehicles que circulen per la ciutat tenen una distància màxima abans d'espatllar-se. Cada tipus de vehicle té una distància i també una desviació associada. Així que la tasca vehicle és un bucle finit amb el número de metres totals. Cada volta de bucle és un metre recorregut per el vehicle. Quan el vehicle es troba amb els semàfors, tenim la probabilitat de girar o seguir recte. Quan el vehicle surt del bucle significa que està espatllat i avisa al control de trànsit del seu estat i la tasca acaba.

A partir d'aquí serà la grua (quan s'activi) qui recollirà el vehicle i el portarà al taller. Un resum amb pseudocodi de la tasca vehicle és el següent:

```
tasca vehicle es
inici
      distancia_maxima_per_recorrer:= max_distancia +/- desviacio
      per i desde 0 fins distancia_maxima_per_recorrer - 1 fer
         obtenir_seguent_posicio_en_el_tram
         si fitram llavors
               si volem_anar_recte llavors
                  opcio:= recte
               sino
                  opcio:= girar
               fi si
               si tipus_tram=horitzontal llavors
                  encuar_en_el_semafor_horitzontal
               sino
                  encuar_en_el_semafor_vertical
               fi si
               tipus_tram:=tram.obtenir_tipus_tram
               obtenir_seguent_posicio_en_el_tram
               fitram:=falç
         fi si
         pintar_vehicle
         delay(velocitat)
      fi per
      eliminar_canvas
fi vehicle
```

3.7. Camió d'escombreries

Objectiu:

El camió d'escombreries és un vehicle que circularà dintre de la nostra ciutat. La seva velocitat la determina l'usuari, ja que serà la mateixa que la resta de camions, tenint en compte el UTS. Sempre estarà actiu, mai s'espatlla. La seva feina és recórrer tots els carrers de la ciutat, aturant-se en cadascun d'ells per recollir la brossa (el contenidor està situat en les voreres de la dreta, segons el sentit de circulació). Aquesta feina l'ha de fer periòdicament durant tota la durada d'execució del programa. El període del camió d'escombreries el determinarà l'usuari, si el període és massa petit i el camió no pot

recórrer tots els carrers dintre d'un període llavors ens avisarà (mitjançant un missatge de text) que no ha pogut acomplir el temps establert. En aquest cas fem que el camió torni a començar el seu recorregut amb el mateix període. En cada contenidor de brossa el camió s'atura un temps definit per l'usuari.

L'acompliment del temps dependrà de molts factors, per exemple:

- ✔ Dimensió de la ciutat.
- ✔ Mida dels carrers.
- ✔ Velocitat del camió.
- ✔ Temps que s'atura en cada contenidor.
- ✔ Número de vehicles que hi ha en la via.
- ✔ Període dels semàfors
- ✔ L'activació del control de trànsit.

Per iniciar la simulació tots aquests valors tindran valors amb els quals el camió tingui temps d'acomplir el deadline. Perquè amb aquest cas el mateix període ens marca el deadline del camió.

Implementació:

El camió d'escombreries és una tasca periòdica amb límit de temps. El límit de temps (deadline) en aquest cas serà el mateix període.
L'esquelet de la tasca camió d'escombreries amb pseudocodi serà el següent:

```
tasca camió d'escombreries és
      periode = periode que ens arriba per paràmetre
      exhaurit = false
inici
      proxim temps = ara
      loop
            delay until el meu_torn
            límit := temps actual + període (deadline de la tasca)
            select
                  delay until límit
                  exhaurit = true
            then abort
                  recorrer tots els carrers
            end select
            si temps exhaurit llavors
                  mostrar per pantalla
                  exhaurit = false
            fisi
            tornar a la posició inicial
            proxim temps = pròxim temps + període
      end loop
fi tasca camió d'escombreries
```

Anem a comentar aquest pseudocodi. La variable període és paràmetre de la tasca, tenim també la variable *límit* i *pròxim temps* que ens ajuden a calcular temps intermitjos necessaris. En primer lloc

inicialitzem la variable *pròxim temps* amb el valor del temps actual, *exhaurit* a falç (indica que la tasca està dins els terminis deadline). El primer cop el delay until no s'espera ja que el temps que hi ha a la variable és anterior a l'actual (temps absolut). A continuació assignem la suma del *temps actual* més el període a la variable *límit*. Així tenim el temps absolut màxim que hauria de trigar el camió en fer el recorregut. Llavors fem l'acció de recórrer tots els carrers fins que s'arribi al temps absolut marcat per *límit*.

Si s'exhaureix el temps deixem el que estem fent i posem *exhaurit* a cert i sortim del *select*. Fora del select mirem si hem acabat el select perquè hem avortat, si és aquest cas, ho notifiquem a la tasca que pinta els esdeveniments per pantalla. Tanmateix fem en els dos casos *tornar a la posició inicial*, això elimina el camió de la pantalla i el situa en la posició inicial. Després fem que el *pròxim temps* sigui el temps base (que havíem agafat) més el període. Així que tornem al principi del bucle i ens haurem d'esperar fins que arribi el temps absolut marcat pel període.

Tenim un algorisme per fer que el camió passi per tots els carrers. Hi ha definida una posició fixa de sortida, que serà la mateixa durant tot el programa. S'ha de complir que la sortida sigui des d'una cantonada de la ciutat, sinó l'algorisme no funciona. L'itinerari es crea a partir de les dimensions de la ciutat i es guarda com una llista canvis de sentit en els semàfors. La probabilitat de girar a cada cantonada, aquí està desactivada, el vehicle sap si vol girar o vol seguir recte a través de l'itinerari. Repeteix a questa ruta cada cop que torna a començar el període de la tasca.

Mentre s'exhaureix el deadline de tasca, el camió s'ha d'anar movent, anem a veure amb pseudocodi com ho fa.

```
ruta := obtenir_ruta_del_camio_escombraries (files, columnes)
temps_aturat := calcular_temps(Xce, UTS)
es_mig_tram := false
quedin_carrers := cert

mentre quedin_carrers fer
      si es_mig_tram llavors
            delay (temps_aturat)
            es_mig_tram := false;
      sino
            obtenir_seguent_posició_en_el_tram
            si fi_tram llavors
                  if num_carrers = total_carrers llavors
                        quedin_carrers := false
                  fisi
                  decidir_girar_recte (ruta)
            sino
                  moure_el_vehicle
            fisi
      fisi
fimentre
```

Podem apreciar que tan el temps que s'atura el camió per recollir la brossa com el període depenen del valor de UTS.

3.8. Semàfor

Un semàfor controla una cruïlla de la nostra ciutat. Dóna pas en un sentit (vertical o horitzontal) i el denega per l'altre. Simula que un semàfor està en verd i l'altre en roig. El semàfor de la cruïlla tindrà sempre un únic valor, això significa que donarà pas vertical o horitzontal, mai es pot donar el cas que ho faci a la vegada. Quan un cotxe arriba a un semàfor demana pas horitzontal (per exemple), si el semàfor està donant pas vertical, el cotxe s'espera a la cua del semàfor. En cas contrari, el cotxe decideix si girar o anar recte i continua la seva trajectòria.

En la fase d'inicialitzacions és on es creen els semàfors, un per cada cruïlla. En aquest moment es quan es crea una tasca temporitzadora per cada semàfor. Aquesta tasca s'encarrega de controlar el període del semàfor. Hem de tenir en compte que el període del semàfor pot canviar en el moment que s'engega el control de trànsit, això es controla mitjançant una entry a una guarda. Quan es compleix una certa condició, intercanviem el sentit de pas i tornem a començar un nou període.

Anem a veure quin és l'aspecte del temporitzador del semàfor. Pensem que aquesta tasca haurà de ser un bucle infinit, mirant constantment si s'ha acomplert el període. Aquest *pulling* constant feia decaure en picat el rendiment del sistema, ja que teníem aquest codi replicat a el número de semàfors de la ciutat. Això ens va fer provar de posar un *delay* molt petit per a què estigui constantment ocupant *cpu* sinó que ho faci cada 0.005 segons. Aquest valor ens dóna una precisió prou bona i fa que el sistema no estigui sobrecarregat.

```
tasca timer_semafor es
      periode := calcular_periode (periode_semafor, uts)
      temps_seguent, ara
inici
   temps_seguent := temps_ara + periode
   iterar
         Ara := temps_ara
         delay(0.005)
         si ara - temps_seguent > periode llavors
            temps_seguent := ara + periode
            Canviartorn
         fisi
   fi iterar
fi timer_semafor
```

3.9. Grua

La funció de la grua dins la nostra ciutat és la d'anar a buscar els vehicles espatllats quan l'usuari activa el control de trànsit. Els cotxes quan s'espatllen envien un missatge a la grua i aquesta els va a buscar amb ordre. Quan la grua arriba on està el vehicle espatllat, desapareix, fins que arriava al garatge, que torna a crear-se un vehicle del mateix tipus.

El vehicle grua és de tipus camió i no s'espatlla mai. La seva velocitat està condiciona al valor de UTS i a la velocitat del vehicle tipus camió que decideixi l'usuari.

Si la grua ha sortit a recollir un cotxe i es desactiva el control de trànsit, continua la seva feina fins que arriba al garatge amb el vehicle espatllat.

El vehicle grua serà una tasca que es crea al principi del programa i es queda en *standby* fins que el control de trànsit no la desperta. Mentretant els vehicles que s'espatllen poden encuarse en la seva cua. Una vegada hagi portat el vehicle al garatge, enviarà un missatge el al control de trànsit. Llavors el control de trànsit enviarà un missatge amb un nou cotxe espatllat. Tenim que la tasca que encua tots els vehicles espatllats és el control de trànsit. I mitjançant els missatges, la grua s'assabenta dels cotxes que ha d'anar a recollir. Anem a veure el pseudocodi de la tasca grua:

```
tasca grua es
      vehicle_espatllat_trobat := false
      vehicle_portat_al_taller := false
inici
     acceptar control_transit_desactiu fer
      esperar
     fi acceptar
    iterar
         acceptar hi_ha_cotxe_espatllat llavors
            guardar_parametres
         fi acceptar

         posicio_grua := obtenir_posicio_grua
         posicio_vehicle := obtenir_posicio_vehicle

         mentre no vehicle_espatllat_trobat o no vehicle_portat_al_taller fer
            si arrivem_semafor llavors
                 obtenir_desicio_grua;
                 fitram := false;
            fisi
            si (no vehicle_espatllat_trobat) i (tram_vehicle_espatllat) i
                  Pos_vehicle_espatllat llavors
                  vehicle_espatllat_trobat:=true;
            sinosi (vehicle_espatllat_trobat) i (no vehicle_portat_al_taller)
            i (he_arribat_garatge) llavors
               vehicle_portat_al_taller:=true;
            fisi
            esperar(velocitat)
         fi iterar
         avisar_a_finestra_comunicador
         vehicle_espatllat_trobat := false
         vehicle_portat_al_taller := false
         crear_vehicle_nou
         grua disponible
      fi iterar
 fi grua
```

Quan la grua rep la informació del vehicle espatllat a d'anar a buscar-lo. La ruta que farà la grua per arrivar fins el vehicle no és crea a priori, sinó que a mesura que avancem per la ciutat, s'escolleix millor per arribar-hi. Cada cop que arriben a un semàfor hem de decidir si girar o continuar recte, és en aquest punt quan fem el càlcul de la distància mínima entre la posició de la grua i el vehicle espatllat. La funció és la següent:

```
funció Obtenir_Desicio_Grua(tipustramg: in TtipusTram; posiciogrua: in
TCoordenada; tipustramv: in TtipusTram; posiciovehicle: in TCoordenada)
retorna (RECTE o GIRAR) es
opcio: (GIRAR o RECTE)
inici
      si ((tipustramg=VERTICAL) i (tipustramv=VERTICAL) i (posiciogrua.x /=
            posiciovehicle.x)) then opcio:=GIRAR;
      sinosi ((tipustramg=VERTICAL) i (tipustramv=VERTICAL) i (posiciogrua.x =
            posiciovehicle.x)) llavors opcio:=RECTE;
      elsif ((tipustramg=HORITZONTAL) i (tipustramv=HORITZONTAL) i
            (posiciogrua.y /= posiciovehicle.y)) llavors opcio:=GIRAR;
      elsif ((tipustramg=HORITZONTAL) i (tipustramv=HORITZONTAL) i
            (posiciogrua.y = posiciovehicle.y)) llavors opcio:=RECTE;
      elsif (((tipustramg=HORITZONTAL) i (tipustramv=VERTICAL) i (abs
            (posiciogrua.x-posiciovehicle.x) = 1)) o ((tipustramg=VERTICAL) i
            (tipustramv=HORITZONTAL) and (abs(posiciogrua.y-posiciovehicle.y)
            = 1))) llavors opcio:=GIRAR;
      else opcio:= RECTE;
      end si
      returna opcio
fi funcio
```

3.10. Tasca Control d'indicadors

Mentre s'executa el simulador no tenim una noció clara de quins esdeveniments estan succeint en cada moment. S'ha creat un nova tasca que permet en cada moment saber l'estat el número de vehicles aturats en cada semàfor, el nombre de vehicles total esperant en tots els semàfors i el nombre vehicles espatllats. Per aquest motiu, hi ha una finestra exclusiva que mostra aquesta informació.

Dins la finestra s'hi pot veure dos matrius, una per els semàfors verticals i els altres per els semàfors horitzontals. Els valors de la matriu són el nombre de cotxes que hi ha en el semàfor en aquest instant. Aquesta tasca s'activa cada 0.05 segons, temps suficient per a què pugéssim apreciar els canvis que s'hi produeixen.

La feina d'aquesta tasca és recórrer tots els semàfors de la ciutat i mirar les seves cues. Finalment mostrar-ho per pantalla. Per aquest motiu, hi ha una finestra exclusiva que mostra aquesta informació. El pseudocodi del programa és el següent:

```
tasca tasca_control_indicadors es
   en_espera_vertical: Integer:=0
   en_espera_horitzontal: Integer:=0
   total_vehicles_en_espera: Integer:=0
   total_vehicles_espatllats: Integer:=0
inici
Inicialitzar_matrius_pantalla
Inicialitzar_els_valor_que_mostra_la_finestra
   iterar
      totalvehiclesenespera:=0
      per x en 0 fins num_carrers_horitzontals - 1  fer
         per y en 0 fins num_carrers_verticals - 1 fer
            obtenir_numero_vehicles_en_els_semafors
```

```
                total_vehicles_en_espera:= total_vehicles_en_espera +
                            en_espera_vertical + en_espera_horitzontal
                mostrar_per_pantalla
            fi per
         fi per
         delay(0.05)
      fi iterar
fi tasca_control_indicadors
```

3.11. Tasca Control menú

Per a interactuar amb el simulador i fer proves més exhaustives, es crea una nova finestra on hi apareix uns botons. Amb aquests botons es pot insertar on-line qualsevol tipus de vehicle i activar/desactivar el control de trànsit. Tots els vehicles surten del mateix lloc, cadascun amb el seu color. Aquesta tasca s'executa i està a l'espera de que l'usuari premi algun botó.

```
tasca TascaControlMenu es
   control_activat: boolean:=false;
inici
   iterar
      cas tecla_pitjada sigui
         quan 'A' => afegir_cotxe
                     avisar_tasca_històric
         quan 'B' => afegir_camió
                     avisar_tasca_històric
         quan 'C' => afegir_ciclomotor
                     avisar_tasca_històric
         quan 'D' => afegir_bicicleta
                     avisar_tasca_històric
         quan 'E' => si no control_activat llavors
                           control_transit.activar_control
                           control_activat:=cert
                     sino
                           control_transit.desactivar_control
                           control_activat:=fals
                     fi si
         quan altres => no_fer_res
      fi cas
   fi iterar
fi tasca_control_menu
```

3.12. Tasca històric

La funció d'aquesta tasca és informar al detall de tot el que succeeix durant la simulació. Cada event que es produeix, queda reflectit en el text que apareix dins el camp de text que hi ha en una de les finestres de la simulació. El cos de la tasca és esperar a rebre un missatge i llavors mostrar el tipus d'esdeveniment que s'ha produït.

El pseudocodi és el següent:

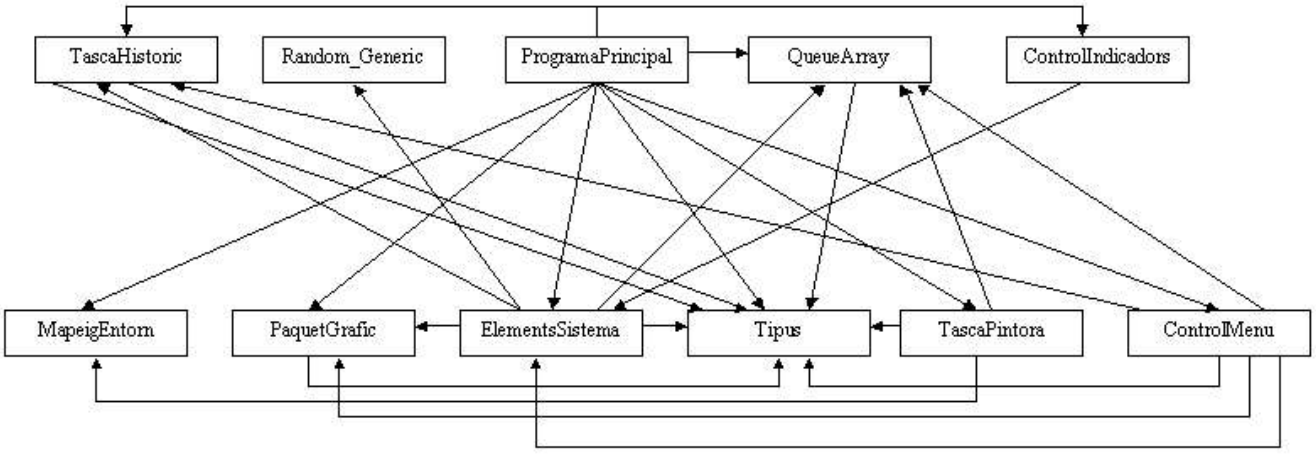
```
tasca historial es
      s : cadena (1 .. 17)
inici
```

```
iterar
   seleccionar
      acceptar avis_estat_control_transit (
            estat : boolea ) fer
         cas estat es
            quan cert =>
               escriure("Control de trànsit activat")
            quan falç =>
               escriure("Control de trànsit desactivat")
         fi cas
      fi avis_estat_control_transit
   o
      acceptar avis_insersio_vehicle (
            tipus     : tipus_vehicle
            quantitat : enter) fer
         cas tipus es
            quan cotxe =>
               escriure("cotxe/s")
            quan ciclomotor =>
               escriure("ciclomotor/s")
            quan camio =>
               escriure("camió/ns")
            quan bicicleta =>
               escriure("bicicleta/es")
            quan camio_escombraries =>
               escriure("camió/ns d'escombraries")
            quan camio_grua =>
               escriure("grua/es")
            quan altres =>
               no_fer_res
         fi cas
      fi avis_insersio_vehicle
   o
      acceptar avis_vehicle_espatllat (
            tipus : tipus_vehicle ) fer
         S:= "S'ha espatllat un"
         cas tipus es
            quan cotxe =>
               escriure(" cotxe")
            quan ciclomotor =>
               escriure(" ciclomotor")
            quan camio =>
               escriure(" camió")
            quan bicicleta =>
               escriure(" bicicleta")
            quan altres =>
               no_fer_res
         fi cas
      fi avis_vehicle_espatllat
   o
      acceptar avis_recorregut_escombriaire_iniciat fer
         escriure("L'escombriaire inicia el recorregut")
      fi avis_recorregut_escombriaire_iniciat
   o
      acceptar avis_recorregut_escombriaire_completat fer
         escriure("L'escombriaire ha completat el recorregut")
      fi avis_recorregut_escombriaire_completat;
   o
      acceptar avis_recorregut_escombriaire_no_completat fer
```

```
                escriure("L'escombriaire no ha completat el recorregut")
            fi avis_recorregut_escombriaire_no_completat
        o
            acceptar avis_grua_busca_vehicle fer
                escriure("La grua surt a buscar un vehicle espatllat")
            fi avis_grua_busca_vehicle
        o
            acceptar avis_grua_troba_vehicle fer
                escriure("La grua ha trobat el vehicle espatllat")
            fi avis_grua_troba_vehicle
        o
            acceptar avis_grua_deixa_vehicle_taller fer
                escriure("La grua ha deixat el vehicle espatllat al taller")
            fi avis_grua_deixa_vehicle_taller
        fi seleccionar
    fi iterar
fi historial
```

4. Implementació

El codi s'encapsula amb paquets, aquests paquets tenen una série de dependències entre ells. Aquestes depedències es mostren a continuació amb el següent diagrama:



4.1. ProgramaPrincipal.adb

```
with Jewl.Simple_Windows;
use  Jewl.Simple_Windows;
with ElementsSistema;
use ElementsSistema;
with PaquetGrafic;
use PaquetGrafic;
with MapeigEntorn;
use MapeigEntorn;
with QueueArray;
use QueueArray;
with Tipus;
use Tipus;
with Tascapintora;
use Tascapintora;
with Controlmenu;
use Controlmenu;
with Controlindicadors;
use Controlindicadors;
with TascaHistoric;
use TascaHistoric;

procedure ProgramaPrincipal is
   -- Menú inicial
   procedure Menuinici (
         Ncotxes  :    out Integer;
         Nmotos   :    out Integer;
         Ncamions :    out Integer;
         Nbicis   :    out Integer;
         M        :    out Integer;
         N        :    out Integer;
         Llarg    :    out Integer;
         P        :    out Integer;
         Tmg      :    out Tipusmotorgrafic;
         Xce      :    out Integer;
         Tce      :    out Integer;
         Tsem     :    out Integer;
         uts      :    out Integer) is
      F   : Frame_Type   := Frame (640, 500, "Configuració", 'Q', Font ("Arial", 9,
Bold => True));
      L1  : Label_Type   := Label (F, (10, 12), 0, 20, "Nombre de carrers
verticals:", Left);
      Eb1 : Editbox_Type := Editbox (F, (180, 10), 30, 20, "4");

      L2  : Label_Type   := Label (F, (10, 37), 0, 20, "Nombre de carrers
horitzontals:", Left);
      Eb2 : Editbox_Type := Editbox (F, (200, 35), 30, 20, "4");

      L3  : Label_Type   := Label (F, (10, 62), 0, 20, "Distància entre cruïlla i
cruïlla(en metres):", Left);
      Eb3 : Editbox_Type := Editbox (F, (257, 60), 30, 20, "10");

      P1  : Panel_Type   := Panel (F, (10, 92), 300, 120, "Dades dels cotxes", Font
("Arial", 11, Bold => True));
      L4  : Label_Type   := Label (F, (25, 112), 130, 13, "Nombre de cotxes:",
Left);
      Eb4 : Editbox_Type := Editbox (F, (150, 110), 40, 20, "5");
```

```
      L5  : Label_Type   := Label (F, (25, 137), 130, 13, "Velocitat(Km/h):",
Left);
      Eb5 : Editbox_Type := Editbox (F, (150, 135), 40, 20, "15");
      L20  : Label_Type   := Label (F, (25, 162), 225, 20, "Distància mitjana de
vida útil(m):", Left);
      Eb20 : Editbox_Type := Editbox (F, (240, 160), 55, 20, "1000");
      L21  : Label_Type   := Label (F, (25, 187), 225, 20, "Desviació màxima de la
mitjana(m):", Left);
      Eb21 : Editbox_Type := Editbox (F, (240, 185), 55, 20, "200");

      P2  : Panel_Type   := Panel (F, (10, 220), 300, 120, "Dades dels camions",
Font ("Arial", 11, Bold => True));
      L6  : Label_Type   := Label (F, (25, 240), 130, 13, "Nombre de camions:",
Left);
      Eb6 : Editbox_Type := Editbox (F, (150, 238), 40, 20, "5");
      L7  : Label_Type   := Label (F, (25, 265), 130, 13, "Velocitat(Km/h):",
Left);
      Eb7 : Editbox_Type := Editbox (F, (150, 263), 40, 20, "10");
      L24  : Label_Type   := Label (F, (25, 290), 225, 20, "Distància mitjana de
vida útil(m):", Left);
      Eb24 : Editbox_Type := Editbox (F, (240, 288), 55, 20, "1000");
      L25  : Label_Type   := Label (F, (25, 315), 225, 20, "Desviació màxima de la
mitjana(m):", Left);
      Eb25 : Editbox_Type := Editbox (F, (240, 313), 55, 20, "200");

      P3  : Panel_Type   := Panel (F, (320, 92), 300, 120, "Dades dels
ciclomotors", Font ("Arial", 11, Bold => True));
      L8  : Label_Type   := Label (F, (335, 112), 140, 13, "Nombre de
ciclomotors:", Left);
      Eb8 : Editbox_Type := Editbox (F, (485, 110), 40, 20, "5");
      L9  : Label_Type   := Label (F, (335, 137), 130, 13, "Velocitat(Km/h):",
Left);
      Eb9 : Editbox_Type := Editbox (F, (485, 135), 40, 20, "20");
      L22  : Label_Type   := Label (F, (335, 162), 225, 20, "Distància mitjana de
vida útil(m):", Left);
      Eb22 : Editbox_Type := Editbox (F, (555, 160), 55, 20, "1000");
      L23  : Label_Type   := Label (F, (335, 187), 225, 20, "Desviació màxima de la
mitjana(m):", Left);
      Eb23 : Editbox_Type := Editbox (F, (555, 185), 55, 20, "200");

      P4   : Panel_Type   := Panel (F, (320, 220), 300, 120, "Dades de les
bicicletes", Font ("Arial", 11, Bold => True));
      L10  : Label_Type   := Label (F, (335, 240), 140, 13, "Nombre de
bicicletes:", Left);
      Eb10 : Editbox_Type := Editbox (F, (470, 238), 40, 20, "5");
      L11  : Label_Type   := Label (F, (335, 265), 130, 13, "Velocitat(Km/h):",
Left);
      Eb11 : Editbox_Type := Editbox (F, (470, 263), 40, 20, "5");
      L26  : Label_Type   := Label (F, (335, 290), 225, 20, "Distància mitjana de
vida útil(m):", Left);
      Eb26 : Editbox_Type := Editbox (F, (555, 288), 55, 20, "1000");
      L27  : Label_Type   := Label (F, (335, 315), 225, 20, "Desviació màxima de la
mitjana(m):", Left);
      Eb27 : Editbox_Type := Editbox (F, (555, 313), 55, 20, "200");

      L12  : Label_Type   := Label (F, (320, 12), 0, 20, "Unitats de temps per
segon(UTS):", Left);
      Eb12 : Editbox_Type := Editbox (F, (520, 10), 40, 20, "1");
```

```
      L13  : Label_Type   := Label (F, (320, 37), 0, 20, "Probabilitat de seguir
recte en una cruïlla(%):", Left);
      Eb13 : Editbox_Type := Editbox (F, (580, 35), 40, 20, "50");

      L16  : Label_Type   := Label (F, (320, 62), 0, 20, "Píxels per metre:",
Left);
      Eb16 : Editbox_Type := Editbox (F, (430, 60), 30, 20, "5");

      P5  : Panel_Type       := Panel (F, (320, 350), 247, 75, "Motor de l'entorn
gràfic", Font ("Arial", 11, Bold => True));
      Rb1 : Radiobutton_Type := Radiobutton (F, (325, 370), 220, 20, "Esdeveniments
pintats al moment", True, Font ("Arial", 9, Bold => True));
      Rb2 : Radiobutton_Type := Radiobutton (F, (325, 395), 200, 20, "Basat en el
sistema de buffering", False, Font ("Arial", 9, Bold => True));

      L17  : Label_Type   := Label (F, (10, 350), 300, 20, "Periode del camió
d'escombraries(ms):", Left);
      Eb17 : Editbox_Type := Editbox (F, (248, 348), 60, 20, "60000");

      L18  : Label_Type   := Label (F, (10, 380), 300, 20, "Temps aturat camió
d'escombraries(ms):", Left);
      Eb18 : Editbox_Type := Editbox (F, (248, 378), 50, 20, "2000");

      L19  : Label_Type   := Label (F, (10, 410), 300, 20, "Periode dels semàfors
(ms):", Left);
      Eb19 : Editbox_Type := Editbox (F, (175, 408), 50, 20, "4000");

      A : Button_Type := Button (F, (450, 440), 80, 25, "Sobre...", 'A');
      B : Button_Type := Button (F, (540, 440), 80, 25, "Iniciar", 'X');

   begin
      Set_Origin(F,(5,5));

      loop
         case Next_Command is
            when 'A' => Show_Message ("Autors:  Albert Nadal G.  &  Xavier Estellé
L.   [ Maig de 2004 ]", "Sobre...");
            when 'X' =>
               exit;
            when others =>
               null;
         end case;
      end loop;

      M := Integer'Value(Get_Text(Eb1));
      N := Integer'Value(Get_Text(Eb2));
      Llarg := Integer'Value(Get_Text(Eb3));
      P := Integer'Value(Get_Text(Eb13));
      Ncotxes := Integer'Value(Get_Text(Eb4));
      Nmotos := Integer'Value(Get_Text(Eb8));
      Ncamions := Integer'Value(Get_Text(Eb6));
      Nbicis := Integer'Value(Get_Text(Eb10));
      Tce := Integer'Value(Get_Text(Eb17));
      Xce := Integer'Value(Get_Text(Eb18));
      Tsem := Integer'Value(Get_Text(Eb19));
      uts := Integer'Value(Get_Text(Eb12));
      metre := Integer'Value(Get_Text(Eb16));
      velocitats.cotxe := Integer'Value(Get_Text(Eb5));
      velocitats.camio := Integer'Value(Get_Text(Eb7));
```

```
      velocitats.ciclomotor := Integer'Value(Get_Text(Eb9));
      velocitats.bicicleta := Integer'Value(Get_Text(Eb11));
      velocitats.camio_escombraries := velocitats.camio;
      velocitats.camio_grua := velocitats.camio;
      Distancies.cotxe:= Integer'Value(Get_Text(Eb20));
      Desviacions.cotxe:= Integer'Value(Get_Text(Eb21));
      Distancies.camio:= Integer'Value(Get_Text(Eb24));
      Desviacions.camio:= Integer'Value(Get_Text(Eb25));
      Distancies.ciclomotor:= Integer'Value(Get_Text(Eb22));
      Desviacions.ciclomotor:= Integer'Value(Get_Text(Eb23));
      Distancies.bicicleta:= Integer'Value(Get_Text(Eb26));
      Desviacions.bicicleta:= Integer'Value(Get_Text(Eb27));

      if Get_State(Rb1) then
         Tmg:=Metode_Directe;
      else
         Tmg:=Metode_Buffering;
      end if;

   end Menuinici;

   C               : Pvehicle;
   D               : Pdibuixant;
   Paint           : Ppintor;
   Cua             : Pqueue;
   Tmg             : Tipusmotorgrafic;
   Cmenu           : Ptascacontrolmenu;
   Cindica         : Ptascacontrolindicadors;
   Ncotxes,
   Nmotos,
   Ncamions,
   Nbicis,
   M,
   N,
   Llarg,
   P,
   Id              : Integer:=0;
   Escombriaire    : Pcamioescombraries;
   G               : Pgrua;
   Control_Transit : Pcontrol;
   Xce             : Integer;
   Tce             : Integer;
   Tsem            : Integer;
   uts             : Integer;
   loger           : PHistorial;
begin
   -- En primer lloc demanem les dades de configuració
   Menuinici (Ncotxes,Nmotos,Ncamions,Nbicis,M,N,Llarg,P,Tmg,Xce,Tce, Tsem, Uts);
   -- Crea una cua de elements per pintar
   Cua:= new Queue;
   -- Crea els Trams
   Creartrams(M, N);
   -- Crea els Semàfors
   Crearsemafors(M, N, Cua, Tsem, Uts);
   -- Crea l'objecte Dibuixant que pinta esdeveniments
   D:= new Dibuixant;
   -- Crea la finestra de simulació
   D.Inicialitzar("Simulació", 200, Llarg, N, M, 100);
   -- Inicialitza els Trams i els Semàfors
```

```
   Inicialitzar_Trams_I_Semafors(M, N, Llarg, D, Cua, Tmg);
   -- Fa d'intermediari entre la cua i el Dibuixant
   Paint:= new Pintor(D, Cua);
   -- Crea la tasca que mostra el historial mitjançant una finestra
   Loger := new Historial;
   -- Reserva un Canvas per a la Grua
   D.Inserirvehicle(Id, Camio_Grua);
   -- Crea la tasca Grua
   G := new Grua(Id, D, Tramshoritzontals(0,0), Cua, Tmg, P,M, N,Loger,Uts);
   -- Crea la tasca Control de Trànsit
   Control_Transit := new Control(Semafors,M,N,G,Loger);
   -- Indica a la Grua el Control de Trànsit que té associat
   G.Indicar_Control_Transit(Control_Transit);
   -- Crea la tasca dels Indicadors de fluïdessa
   Cindica := new Tascacontrolindicadors(Semafors, M, N, control_transit);

   -- incrementem l'indentificador del vehicle
   Id:=Id+1;
   -- Reserva un Canvas per al Camió d'escombreries
   D.Inserirvehicle(Id, Camio_Escombraries);
   -- Crea el Camió d'escombreries
   Escombriaire := new Camioescombraries(Id, D, Tramsverticals(0,0), Cua, Tmg, M,
N, Tce, Xce,loger,uts);
   Id:=Id+1;
   -- Crea els vehicles de tipus cotxe 1 a 1
   for I in 1..Ncotxes loop
      D.Inserirvehicle(Id, Cotxe);
      C := new Vehicle(Id, D, Tramshoritzontals(0,0), Cotxe, Cua, Tmg, p,
Control_Transit,loger,uts);
      Id:=Id+1;
      delay(0.5/uts); -- Fem una espera per a que no surtin tots alhora
   end loop;
   -- Avisar a la tasca Historial la insersió dels vehicles
   loger.Avis_Insersio_Vehicle(Cotxe, Ncotxes);
   -- Crea els vehicles de tipus ciclomotor 1 a 1
   for I in 1..Nmotos loop
      D.Inserirvehicle(Id, ciclomotor);
      C := new Vehicle(Id, D, Tramshoritzontals(0,0), ciclomotor, Cua, Tmg, p,
Control_Transit,loger,uts);
      Id:=Id+1;
      delay(0.5/uts);
   end loop;
   -- Avisar a la tasca Historial la insersió dels vehicles
   loger.Avis_Insersio_Vehicle(ciclomotor, Nmotos);
   -- Crea els vehicles de tipus camió 1 a 1
   for I in 1..Ncamions loop
      D.Inserirvehicle(Id, camio);
      C := new Vehicle(Id, D, Tramshoritzontals(0,0), camio, Cua, Tmg, p,
Control_Transit,loger,uts);
      Id:=Id+1;
      delay(0.5/uts);
   end loop;
   -- Avisar a la tasca Historial la insersió dels vehicles
   Loger.Avis_Insersio_Vehicle(Camio, Ncamions);
   -- Crea els vehicles de tipus bicicleta 1 a 1
   for I in 1..Nbicis loop
      D.Inserirvehicle(Id, bicicleta);
      C := new Vehicle(Id, D, Tramshoritzontals(0,0), bicicleta, Cua, Tmg, p,
Control_Transit,loger,uts);
```

```
      Id:=Id+1;
      delay(0.5/uts);
   end loop;
   -- Avisar a la tasca Historial la insersió dels vehicles
   loger.Avis_Insersio_Vehicle(bicicleta, Nbicis);
   -- Crea la tasca de Control Menú
   Cmenu := new Tascacontrolmenu(Id, D, Cua, Tramshoritzontals(0,0), p,Tmg,
Control_Transit,loger,uts);
end ProgramaPrincipal;
```

4.2. Tipus.ads

```
package Tipus is

   type Tipusmotorgrafic is
         (Metode_Directe,
          Metode_Buffering);
   type Ttipusvehicle is
         (Cotxe,
          Ciclomotor,
          Camio,
          Bicicleta,
          Camio_Escombraries,
          Camio_Grua);
   type Ttipustram is
         (Horitzontal,
          Vertical);
   type Tdesicio is
         (Recte,
          Girar);
   type Ttipussentit is
         (Est_Oest,
          Oest_Est,
          Nord_Sud,
          Sud_Nord);
   type Ttipuscarril is
         (Esquerra,
          Dret);

   type Tcoordenada is
      record
         X : Integer;
         Y : Integer;
      end record;

   type TVelocitats is
      record
         cotxe : Integer;
         ciclomotor : Integer;
         camio : Integer;
         bicicleta : Integer;
         camio_escombraries : Integer;
         camio_grua : Integer;
      end record;

   type TDistancia is
      record
         cotxe : Integer;
```

```
         ciclomotor : Integer;
         camio : Integer;
         bicicleta : Integer;
      end record;

   type TDesviacio is
      record
         cotxe : Integer;
         ciclomotor : Integer;
         camio : Integer;
         bicicleta : Integer;
      end record;
   -- El tipus Item s'utilitza per a inserir elements a la cua
   type Item is
      record
         Object : Boolean;       -- True -> VEHICLE ; False -> SEMAFOR
         Id     : Integer;
         -- Identificador dins la taula de VEHICLES i de SEMAFORS
         Tipus  : Ttipusvehicle;
         -- En el cas de sér un vehicle indica de quin tipus és.
         Pas    : Boolean;
         -- En el cas de ser un semàfor Pas indicarà (True -> un sentit, False ->
l'altre)
         Coor   : Tcoordenada;
         -- coordenada nova on ha d'anar l'objecte
      end record;

   Metre : Integer := 5; -- Un metre -> 5 píxels
   BaseTemps: Duration := 1.0;  -- 20Km/h -> 1 segon de delay
   Velocitats: TVelocitats; -- Velocitats dels diferents tipus de vehicles
   Distancies: TDistancia;  -- Distàncies en metres ke delimiten el temps de vida
mitg dels vehicles
   Desviacions: TDesviacio; -- Desviacions màximes en metres de les distàncies

end Tipus;
```

4.3. Random_Generic.ads

```
generic
  type Result_Subtype is (<>);
package Random_Generic is

  -- Simple integer pseudo-random number generator package.
  -- Michael B. Feldman, The George Washington University,
  -- June 1995.

  function Random_Value return Result_Subtype;

end Random_Generic;
```

4.4. Random_Generic.adb

```
with Ada.Numerics.Discrete_Random;
package body Random_Generic is

  -- Body of random number generator package.
  -- Uses Ada 95 random number generator; hides generator parameters
  -- Michael B. Feldman, The George Washington University,
  -- June 1995.

  package Ada95_Random is new Ada.Numerics.Discrete_Random
    (Result_Subtype => Result_Subtype);

  G: Ada95_Random.Generator;

  function Random_Value return Result_Subtype is
  begin
    return Ada95_Random.Random(Gen => G);
  end Random_Value;

begin

  Ada95_Random.Reset(Gen => G);  -- time-dependent initialization

end Random_Generic;
```

4.5. QueueArray.ads

```
With Tipus; use Tipus;
package QueueArray is
    type Queue is limited private;

    type Pqueue is access Queue;

    procedure Enqueue ( X: Item; Q: in out pqueue );
    procedure Dequeue ( X: out Item; Buit: out Boolean; EnEspera: out Integer; Q:
in out pqueue );
    function  Is_Empty( Q: pqueue ) return Boolean;
    function  Is_Full ( Q: pqueue ) return Boolean;
    procedure Make_Empty( Q: in out pqueue );

    Overflow : exception;
    Underflow: exception;

    private

    type Array_Of_Element_Type is array( Positive range <> ) of Item;

    protected type Queue is

       procedure Enqueue ( X: Item);
       procedure Dequeue ( X: out Item; EnEspera: out Integer);
       function  Is_Empty return Boolean;
       function  Is_Full return Boolean;
       procedure Make_Empty;
```

```
    private
         Q_Front    : Natural := 1;
         Q_Rear     : Natural := 0;
         Q_Size     : Natural := 0;
         Q_Array    : Array_Of_Element_Type( 1..1000 );
    end Queue;

end QueueArray;
```

4.6. QueueArray.adb

```
With Tipus; use Tipus;
package body QueueArray is

   protected body Queue is
    -- Avança la posició de l'iterador
    procedure Increment( X: in out Integer) is
    begin
        if X = Q_Array'Last then
            X := Q_Array'First;
        else
            X := X + 1;
        end if;
    end Increment;

   -- Esborra un item de la cua
    procedure Dequeue( X: out Item; EnEspera: out Integer) is
    begin
        if Is_Empty then
            raise Underflow;
        end if;

        Q_Size := Q_Size-1;
        EnEspera:=Q_Size;
        X := Q_Array( Q_Front );
        Increment( Q_Front );
    end Dequeue;

    -- Afegeix un item al final de cua
    procedure Enqueue( X: Item) is
    begin
        if Is_Full then
           raise Overflow;
        else
           begin
              Q_Size := Q_Size + 1;
              Increment( Q_Rear );
              Q_Array( Q_Rear ) := X;
           end;
        end if;
    end Enqueue;

    -- Indica si la cua esta buïda o no
    function Is_Empty return Boolean is
    begin
        return Q_Size = 0;
    end Is_Empty;
```

```
    -- Indica si la cua esta plena
    function Is_Full return Boolean is
    begin
        return Q_Size = Q_Array'Length;
    end Is_Full;

    -- Inicialitza la cua
    procedure Make_Empty is
    begin
        Q_Front := Q_Array'First;
        Q_Rear  := Q_Array'First - 1;
        Q_Size  := 0;
    end Make_Empty;

    end Queue;


 procedure Enqueue ( X: Item; Q: in out pqueue) is
 begin
    Q.Enqueue(X);
 end Enqueue;

 procedure Dequeue (X: out Item; Buit: out Boolean; EnEspera: out Integer; Q: in
out pqueue) is
 begin
    Buit := Q.Is_Empty;
    if not Buit then
       begin
          Q.Dequeue(X, EnEspera);
       end;
    end if;
 end Dequeue;

 function Is_Empty (Q: pqueue) return Boolean is
 begin
    return Q.Is_Empty;
 end Is_Empty;

 function Is_Full (Q: pqueue) return Boolean is
 begin
    return Q.Is_Full;
 end Is_Full;

 procedure Make_Empty (Q: in out pqueue) is
 begin
   Q.Make_Empty;
 end Make_Empty;

end QueueArray;
```

4.7. ElemetsSistema.ads

```
with Random_Generic;
with PaquetGrafic;
use PaquetGrafic;
with QueueArray;
use QueueArray;
with Tipus;
use Tipus;
with TascaHistoric;
use TascaHistoric;

package ElementsSistema is

   type Grua;
   type Control;
   type Tram;
   type Parellsemafor;

   type Pgrua is access Grua;
   type Pcontrol is access Control;
   type Pparellsemafor is access Parellsemafor;
   type Ptram is access Tram;

   type Tcarril is array (Integer range <>) of Boolean;
   type Ptcarril is access Tcarril;

   type Truta is array (Integer range <>) of Tdesicio;
   type Ptruta is access Truta;

   type TMatriuTrams is array(Integer  range <>, Integer  range <>) of PTram;
   type PTMatriuTrams is access TMatriuTrams;
   type TMatriuParellSemafors is array(Integer  range <>, Integer  range <>) of
PParellSemafor;
   type PTMatriuParellSemafors is access TMatriuParellSemafors;

   TramsHoritzontals: PTMatriuTrams;
   TramsVerticals: PTMatriuTrams;
   Semafors: PTMatriuParellSemafors;

   subtype Trangdesicions is Positive range 1..100;
   package Presadesicio is new Random_Generic (Result_Subtype => Trangdesicions);

   subtype Trangdesviacio is Positive range 1..100;
   package Paquetdesviacio is new Random_Generic (Result_Subtype =>
Trangdesviacio);

   protected type Tram is
      procedure Inicialitzar (
            T    : in     Ttipustram;
            S    : in     Ttipussentit;
            Sem  : in     Pparellsemafor;
            Long : in     Natural;
            C    : in     Tcoordenada;
            A    : in     Tcoordenada     );
      procedure AlliberarPosicio(Pos: in Integer; Carril: in Ttipuscarril);
```

```
      function ObtenirTipusTram return TtipusTram;
      procedure Obtenirseguentposicio (
            Pos       : in out Integer;
            Carril    : in out Ttipuscarril;
            C         :    out Tcoordenada;
            Fitram    : in out Boolean;
            Migtram   :    out Boolean;
            Sem       :    out Pparellsemafor;
            Tipustram :    out Ttipustram      );
      function ObtenirAdreça return TCoordenada;
      procedure Demanarsituarsealcarrilesquerra (
            Pos    : in out Integer;
            C      :    out Tcoordenada;
            Carril : in out Ttipuscarril );
   private
      Sentit: Ttipussentit;
      Tipus: Ttipustram;
      Longitud: Natural;
      Carrildret: Ptcarril;
      Carrilesquerra: Ptcarril;
      Semafor: Pparellsemafor;
      Posicio: Tcoordenada;
      Adreça:  Tcoordenada;
   end Tram;

   protected type Parellsemafor is
      procedure Activarmodefluidesa;
      procedure Desactivarmodefluidesa;
      entry Demanarpasvertical (
            Des : in     Tdesicio;
            T :    out Ptram     );
      entry Demanarpashoritzontal (
            Des : in     Tdesicio;
            T :    out Ptram     );
      procedure Obtenirnumerovehiclesenespera (
            H :    out Integer;
            V :    out Integer  );
      procedure Inicialitzar (
            Ident : in     Integer;
            Tv    : in     Ptram;
            Th    : in     Ptram;
            Dib   : in     Pdibuixant;
            Cua   : in     Pqueue;
            Tmg   :        Tipusmotorgrafic );
      procedure Canviartorn;
   private
      C: Pqueue;
      Id: Integer;
      Torn: Boolean;
      Tramvertical: Ptram;
      Tramhoritzontal: Ptram;
      D: Pdibuixant;
      Motorgrafic: Tipusmotorgrafic;
      Mode_Fluidesa: Boolean;
   end Parellsemafor;

   task type Timerparellsemafor(S: Pparellsemafor; Tsem: Integer; uts: Integer);
   type Ptimerparellsemafor is access Timerparellsemafor;
```

```
   task type Vehicle(Id: Integer; D: Pdibuixant; T: Ptram; Tipus: Ttipusvehicle; Q:
Pqueue; Motorgrafic: Tipusmotorgrafic; probabilitat: integer; control_transit:
Pcontrol; loger: PHistorial; uts: Integer);
   type Pvehicle is access Vehicle;

   task type Camioescombraries(Id: Integer; D: Pdibuixant; T: Ptram; Q: Pqueue;
Motorgrafic: Tipusmotorgrafic; M: Integer;
      N: Integer; Tce: Integer; Xce: Integer;loger: PHistorial;uts: integer);
   type Pcamioescombraries is access Camioescombraries;

   task type Grua (Id: integer; D: Pdibuixant; T: Ptram; Q: Pqueue; tmg:
Tipusmotorgrafic; p: Integer; M: integer; N: integer; loger: PHistorial; uts:
Integer) is
      entry Indicar_Control_Transit(control_transit: Pcontrol);
      entry Anar_a_Buscar_Cotxe_Espatllat(Id: Integer; Tipus: Ttipusvehicle;  T:
Ptram; Pos: Integer; Carril: Ttipuscarril; TipusTram: TtipusTram);
   end Grua;

   task type Control (Semafors: Ptmatriuparellsemafors; M: Integer; N: Integer;
vehicle_grua: Pgrua; loger: PHistorial) is
      entry Avis_Grua_Disponible;
      entry Despatxar_Vehicle_Espatllat(Id: Integer; Tipus: Ttipusvehicle;  T:
Ptram; Pos: Integer; Carril: Ttipuscarril; TipusTram: TtipusTram);
      entry ObtenirNumeroVehiclesEspatllats(NumVehicles: out Integer);
      entry Activar_Control;
      entry Desactivar_Control;
   end Control;

end ElementsSistema;
```

4.8. ElementsSistema.adb

```
with QueueArray;
use QueueArray;
with Tipus;
use Tipus;
with Ada.Real_Time;
use Ada.Real_Time;

package body ElementsSistema is
   -- Pinta el vehicle en funció del tipus de motor gràfic escollit
   procedure PintarVehicle(id: in integer; tipus: in Ttipusvehicle; C: in
TCoordenada; D: in PDibuixant; Q: in Pqueue; it : in Item; TMG: Tipusmotorgrafic)
is
      Cua: PQueue := Q;
      I: Item :=it;
   begin
      case TMG is
         when Metode_Directe => D.Actualitzarimatgevehicle(id, C, tipus);
         when Metode_Buffering =>
              I.Coor:=C;
              Enqueue(I,Cua);
      end case;
   end PintarVehicle;

   -- Obté les distàncies mitjanes de vida útil en funció del tipus de vehicle
   function ObtenirDistancia(tipus: in Ttipusvehicle) return Integer is
```

```
      dist: Integer;
   begin
      case tipus is
         when Cotxe => dist:=distancies.cotxe;
         when Ciclomotor => dist:=distancies.ciclomotor;
         when Camio => dist:=distancies.camio;
         when Bicicleta => dist:=distancies.bicicleta;
         when others => dist:=0;
      end case;
      return dist;
   end ObtenirDistancia;

   -- Obté les desviacions de les distàncies mitjanes de vida útil en funció del
tipus de vehicle
   function ObtenirDesviacio(tipus: in Ttipusvehicle) return Integer is
      desv: Integer;
   begin
      case tipus is
         when Cotxe => desv:=desviacions.cotxe;
         when Ciclomotor => desv:=desviacions.ciclomotor;
         when Camio => desv:=desviacions.camio;
         when Bicicleta => desv:=desviacions.bicicleta;
         when others => desv:=0;
      end case;
      return desv;
   end ObtenirDesviacio;

   -- Obté la desició que ha de prendre la grua quan arriba al final d'un tram
   function ObtenirDesicioGrua(tipustramg: in TtipusTram; posiciogrua: in
TCoordenada; tipustramv: in TtipusTram; posiciovehicle: in TCoordenada) return
TDesicio is
      opcio: TDesicio;
   begin
      if ((tipustramg=VERTICAL) and (tipustramv=VERTICAL) and (posiciogrua.x /=
posiciovehicle.x)) then opcio:=GIRAR;
      elsif ((tipustramg=VERTICAL) and (tipustramv=VERTICAL) and (posiciogrua.x =
posiciovehicle.x)) then opcio:=RECTE;
      elsif ((tipustramg=HORITZONTAL) and (tipustramv=HORITZONTAL) and
(posiciogrua.y /= posiciovehicle.y)) then opcio:=GIRAR;
      elsif ((tipustramg=HORITZONTAL) and (tipustramv=HORITZONTAL) and
(posiciogrua.y = posiciovehicle.y)) then opcio:=RECTE;
      elsif (((tipustramg=HORITZONTAL) and (tipustramv=VERTICAL) and (abs
(posiciogrua.x-posiciovehicle.x) = 1)) or ((tipustramg=VERTICAL) and
(tipustramv=HORITZONTAL) and (abs(posiciogrua.y-posiciovehicle.y) = 1))) then
opcio:=GIRAR;
      else opcio:= RECTE;
      end if;
      return opcio;
   end;
   -- Depenent el tipus de vehicle i del UTS retorna el temps d'espera (velocitat
del vehicle)
   function ObtenirDelayVehicle(uts :in Integer; tipus: in Ttipusvehicle) return
duration is
      a,b,c: Duration;
      vel: Integer;
   begin
      case Tipus is
         when Cotxe => Vel:=velocitats.cotxe;
         when Ciclomotor => Vel:=velocitats.ciclomotor;
```

```
         when Camio => Vel:=velocitats.camio;
         when Bicicleta => Vel:=velocitats.bicicleta;
         when Camio_Escombraries => Vel:=velocitats.camio_escombraries;
         when Camio_Grua => Vel:=velocitats.camio_grua;
      end case;
      a:= to_duration(Milliseconds(vel*1000)) / BaseTemps;
      b:= 1.0 / a;
      c:= b / to_duration(Milliseconds(uts*1000));
      return c;
   end ObtenirDelayVehicle;

   -- Calcula la ruta del camió escombraries, forçant per a què passi per tots els
carrers
   function Obtenirrutadelcamioescombraries (
         M : in     Integer;
         N : in     Integer;
         L : in     Integer  )
     return Ptruta is
      Ruta : Ptruta;
      I    : Integer := 0;
   begin
      Ruta:= new Truta(0..(L-1));
      for A in 1..M-1 loop
         for B in 1..N-1 loop
            Ruta(I):=Recte;
            I:=I+1;
         end loop;
         for C in 1..2 loop
            Ruta(I):=Girar;
            I:=I+1;
         end loop;
      end loop;

      for D in 1..N-1 loop
         Ruta(I):=Recte;
         I:=I+1;
      end loop;
      Ruta(I):=Girar;
      I:=I+1;

      for A in 1..N-1 loop
         for B in 1..M-1 loop
            Ruta(I):=Recte;
            I:=I+1;
         end loop;
         for C in 1..2 loop
            Ruta(I):=Girar;
            I:=I+1;
         end loop;
      end loop;

      for D in 1..M-1 loop
         Ruta(I):=Recte;
         I:=I+1;
      end loop;
      Ruta(I):=Girar;
      I:=I+1;

      return Ruta;
```

```
   end Obtenirrutadelcamioescombraries;

   -- Temporitzador del semàfor, canvía en un període definit per l'usuari
   task body Timerparellsemafor is
      Period                : constant Time_Span := Milliseconds ((Tsem /2) / uts);
      Next_Time,
      Now                   :          Time;
      Control_Transit_Actiu :          Boolean   := False;
   begin
      Next_Time := Clock + Period;
      loop

         Now := Clock;
         delay(0.005);
         if Now - Next_Time > Period then
            Next_Time := Now + Period;
            S.Canviartorn;
         end if;
      end loop;
   end Timerparellsemafor;

   task body Grua is
      Ve                        : Pvehicle;
      Vehicle_Espatllat_trobat : Boolean       := false;
      vehicle_portat_al_taller : Boolean       := false;
      Tipus_Vehicle_Espatllat  : Ttipusvehicle;
      Controltransit           : Pcontrol;
      Tram_Vehicle_Espatllat   : Ptram;
      Carril_Vehicle_Espatllat : Ttipuscarril;
      Tipus_Tram_Vehicle_Espatllat : TtipusTram;
      Id_Vehicle_Espatllat,
      Pos_Vehicle_Espatllat,
      Maxd,
      Desv                      : Integer;
      Posiciogrua, Posiciovehicle              : Tcoordenada;

      Pos       : Integer        := 0;
      Carril    : Ttipuscarril   := Esquerra;
      Tram      : PTram:=T;
      Vel       : Duration;
      Sem       : Pparellsemafor;
      Tipustram : Ttipustram;
      Migtram,
      Fitram    : Boolean        := False;
      C         : Tcoordenada;
      Opcio     : Tdesicio;
      I         : Item;
      Tipus           : Ttipusvehicle  := Camio_grua;
      Cua: Pqueue:=Q;
   begin
      I.Object := True;
      I.Id:= Id;
      I.Tipus:= tipus;
      vel:= ObtenirDelayVehicle(uts, tipus);
      -- Rep el Control de Trànsit associat
      accept Indicar_Control_Transit (Control_Transit : Pcontrol) do
         Controltransit:=Control_Transit;
      end Indicar_Control_Transit;
```

```
      loop
         -- Rep el cotxe que ha d'anar a recollir
         accept Anar_A_Buscar_Cotxe_Espatllat (
               Id        : Integer;
               Tipus     : Ttipusvehicle;
               T         : Ptram;
               Pos       : Integer;
               Carril    : Ttipuscarril;
               TipusTram : TtipusTram        ) do
            Id_Vehicle_Espatllat:=Id;
            Tipus_Vehicle_Espatllat:=Tipus;
            Tram_Vehicle_Espatllat:=T;
            Pos_Vehicle_Espatllat:=Pos;
            Carril_Vehicle_Espatllat:=Carril;
            Tipus_Tram_Vehicle_Espatllat:=TipusTram;
            Maxd:=ObtenirDistancia(Tipus_Vehicle_Espatllat);
            Desv:=ObtenirDesviacio(Tipus_Vehicle_Espatllat);
            Loger.Avis_Grua_Busca_Vehicle;
         end Anar_A_Buscar_Cotxe_Espatllat;

         Posiciogrua:=Tram.Obteniradreça;
         Posiciovehicle:=Tram_Vehicle_Espatllat.Obteniradreça;
         -- recorrer els carrers fins que troba el vehicle i el porta al taller ->
Grua ocupada
         while ((not vehicle_espatllat_trobat) or (not vehicle_portat_al_taller))
loop
            Tram.Obtenirseguentposicio (Pos, Carril, C, Fitram, Migtram, Sem,
Tipustram);
            if (Fitram) then
                  Opcio:= ObtenirDesicioGrua(TipusTram, posiciogrua,
Tipus_Tram_Vehicle_Espatllat, posiciovehicle);
                  if Tipustram=Horitzontal then Sem.Demanarpashoritzontal(Opcio,
Tram);
                  else Sem.Demanarpasvertical(Opcio, Tram); end if;
                  Posiciogrua:=Tram.Obteniradreça;
                  Fitram:=False;
            else PintarVehicle(Id, Tipus, C, D, Cua, I, TMG); end if;

            if ((not vehicle_espatllat_trobat) and (Tram=Tram_vehicle_espatllat)
and (Pos=Pos_vehicle_espatllat)) then

                  PintarVehicle(Id, Tipus_vehicle_espatllat, (0-metre,0-metre), D,
Cua, I, TMG);
                  Tram_vehicle_espatllat:=Tramshoritzontals(0,0);
                  Tipus_Tram_Vehicle_Espatllat:=HORITZONTAL;
                  Posiciovehicle:=Tram_Vehicle_Espatllat.Obteniradreça;
                  vehicle_espatllat_trobat:=true;

            elsif ((vehicle_espatllat_trobat) and (not vehicle_portat_al_taller)
and (Tram=Tramshoritzontals(0,0)) and (Pos=1)) then
                  Tram.Alliberarposicio(Pos, Carril);
                  Pos:=0;
                  vehicle_portat_al_taller:=true;
            end if;

            delay(vel);
         end loop;
         Loger.Avis_Grua_Troba_Vehicle;
         Vehicle_Espatllat_trobat:=False;
```

```
         Vehicle_Portat_al_taller:=false;
         -- un cop el vehicle espatllat arriva al taller, es crea un nou vehicle
del mateix tipus que surt del mateix taller
         ve := new Vehicle(Id_Vehicle_Espatllat, D, Tramshoritzontals(0,0),
            Tipus_Vehicle_Espatllat, Cua, Tmg, p,Controltransit,
            Loger,uts);
         Loger.Avis_Grua_Deixa_Vehicle_Taller;
         Controltransit.Avis_Grua_Disponible;
      end loop;
   end Grua;

   task body Camioescombraries is
      Tram            : Ptram          := T;
      Pos             : Integer        := 0;
      L               : Integer        := (M - 1) * ((N - 1) + 2) + ((N -
        1) + 1) + (N - 1) * ((M - 1) + 2) + ((M - 1) + 1);
      Index           : Integer        := 0;
      Carril          : Ttipuscarril   := Esquerra;
      Vel             : Duration;
      Sem             : Pparellsemafor;
      Tipustram       : Ttipustram;
      Fitram          : Boolean        := False;
      Quedin_Carrers,
      Ruta_Completada,
      Migtram         : Boolean        := False;
      C               : Tcoordenada;
      Opcio           : Tdesicio;
      I               : Item;
      Cua             : Pqueue         := Q;
      Tipus           : Ttipusvehicle  := Camio_Escombraries;
      Ruta            : Ptruta;
      Period          : Time_Span      := Milliseconds (Tce/uts);
      Next_Time,
      Limit           : Time;
      Temps_Aturat    : Duration       := To_Duration (Milliseconds (Xce/uts));
   begin
      -- Obté a priori la ruta que ha de fer el camió d'escombreries en funció de
la mida de la ciutat
      Ruta := Obtenirrutadelcamioescombraries(M, N, L);
      I.Object := True;
      I.Id:= Id;
      I.Tipus:= Tipus;
      vel:= ObtenirDelayVehicle(uts, tipus);
      Next_Time:= Clock;

      loop
         -- S'espera fins que arrivi el deadline(període)
         delay until Next_Time;
         Limit := Clock + Period;
         Ruta_Completada:=False;
         Loger.Avis_Recorregut_Escombriaire_Iniciat;
         select

            delay until Limit;
            Tram.Alliberarposicio(Pos, Carril);
            if not Ruta_Completada then
               Loger.Avis_Recorregut_Escombriaire_No_Completat;
            end if;
         then
```

```
            abort
         -- Mentre quedi temps, recorre la ruta
            Tram.Alliberarposicio(Pos, Carril);
            Tram:=T;
            Pos:=0;
            Carril:=Esquerra;
            Index:=0;
            Quedin_Carrers :=True;
            Migtram:=false;
            Fitram:=false;
            while Quedin_Carrers loop
               if Migtram then

                     if Carril=Esquerra then

                           PintarVehicle(Id, Tipus, C, D, Cua, I, Motorgrafic);
                           delay(Temps_Aturat);
                           Migtram:=False;

                     else
                        Tram.Demanarsituarsealcarrilesquerra(Pos,C,Carril);
                     end if;

               else

                     Tram.Obtenirseguentposicio (Pos, Carril, C, Fitram,
                        Migtram, Sem, Tipustram);
                     if Fitram then

                           Opcio:=Ruta(Index);
                           Index:=Index+1;
                           if Index=L then
                              Index:=0;
                              Quedin_Carrers :=False;
                           end if;

                           if Tipustram=Horitzontal then
                              Sem.Demanarpashoritzontal(Opcio, Tram);
                           else
                              Sem.Demanarpasvertical(Opcio, Tram);
                           end if;

                           Fitram:=False;

                     else PintarVehicle(Id, Tipus, C, D, Cua, I, Motorgrafic); end
if;

               end if;
               delay(Vel);
            end loop;

            Ruta_Completada:=True;
            Tram.Alliberarposicio(Pos, Carril);
            Loger.Avis_Recorregut_Escombriaire_Completat;
         end select;
         Tram.Alliberarposicio(Pos, Carril);
         Next_Time:= Next_Time + Period;
      end loop;
   end Camioescombraries;
```

```
   task body Vehicle is
      Tram      : Ptram          := T;
      Pos       : Integer        := 0;
      Carril    : Ttipuscarril   := Esquerra;
      Vel       : Duration;
      Sem       : Pparellsemafor;
      Tipustram : Ttipustram;
      Migtram,
      Fitram    : Boolean        := False;
      C         : Tcoordenada;
      Opcio     : Tdesicio;
      I         : Item;
      Cua       : Pqueue         := Q;
      Max       : Integer;
      Desviacio : Integer:=ObtenirDesviacio(tipus);
      Maxdist   : Integer:=ObtenirDistancia(tipus);
   begin
      Max := Maxdist - Desviacio + (2*Desviacio*Paquetdesviacio.Random_Value /
100);
      I.Object := True;
      I.Id:= Id;
      I.Tipus:= Tipus;
      vel:= ObtenirDelayVehicle(uts, tipus);
      Tram.Alliberarposicio(Pos, Carril);
      -- Cada iteració s'avança un metre
      for E in 0..Max-1 loop
         -- Obté la següent posició dins del Tram on està situat
         Tram.Obtenirseguentposicio (Pos, Carril, C, Fitram, Migtram, Sem,
Tipustram);
         if (Fitram) then
               -- Quan s'arriva al final del Tram es decideix girar o no
               if Presadesicio.Random_Value <= probabilitat then opcio:=RECTE;
               else opcio:=GIRAR; end if;

               -- S'encua al semàfor que li correspon
               if Tipustram=Horitzontal then Sem.Demanarpashoritzontal(Opcio,
Tram);
               else Sem.Demanarpasvertical(Opcio, Tram); end if;
               TipusTram:=Tram.ObtenirTipusTram;
               Tram.Obtenirseguentposicio (Pos, Carril, C, Fitram, Migtram, Sem,
Tipustram);
               Fitram:=False;

         end if;

         PintarVehicle(Id, Tipus, C, D, Cua, I, Motorgrafic);
         delay(Vel);
      end loop;
      Tram.Alliberarposicio(Pos, Carril);
      TipusTram:=Tram.ObtenirTipusTram;
      Loger.Avis_Vehicle_Espatllat(Tipus);
      -- S'ha superat la distància de vida útil del vehicle i se li fa saber al
Control de trànsit
      Control_Transit.Despatxar_Vehicle_Espatllat
(Id,Tipus,Tram,Pos,Carril,TipusTram);
   end Vehicle;

   protected body Tram is
```

```
      -- Inicialitza el Tram
      procedure Inicialitzar (
            T    : in     Ttipustram;
            S    : in     Ttipussentit;
            Sem  : in     Pparellsemafor;
            Long : in     Natural;
            C    : in     Tcoordenada;
            A    : in     Tcoordenada     ) is
      begin
         Tipus:=T;
         Sentit:=S;
         Semafor:=Sem;
         Longitud:=Long;
         Posicio:=C;
         Adreça:=A;
         Carrildret:= new Tcarril(0..Longitud-1);
         Carrilesquerra:= new Tcarril(0..Longitud-1);
         for I in 0..Longitud-1 loop
            Carrildret(I):=False;
            Carrilesquerra(I):=False;
         end loop;
      end Inicialitzar;
      -- Quan un vehicle s'espatlla, usa aquest procediment per indicar que la
posició està lliure
      procedure Alliberarposicio (
            Pos    : in     Integer;
            Carril : in     Ttipuscarril ) is
      begin
         case Carril is
            when Esquerra =>
               Carrilesquerra(Pos):=False;
            when Dret =>
               Carrildret(Pos):=False;
         end case;
      end Alliberarposicio;
      -- Retorna la posició gràfica del Tram dins de la ciutat
      function Obteniradreça return Tcoordenada is
      begin
         return Adreça;
      end Obteniradreça;
      -- Retorna el tipus de Tram (Horitzontal o Vertical)
      function ObtenirTipusTram return TtipusTram is
      begin
         return Tipus;
      end ObtenirTipusTram;
      -- Retorna la posició gràfica del vehicle
      function Obtenircoordenadavehicle (
            Pos    : in     Integer;
            Carril : in     Ttipuscarril )
        return Tcoordenada is
         C : Tcoordenada;
      begin
         if Tipus=Horitzontal then

               case Carril is
                  when Esquerra =>
                     C.Y := Posicio.Y;
                  when Dret =>
                     C.Y := Posicio.Y + Metre;
```

```
            end case;
            case Sentit is
               when Est_Oest =>
                  C.X := Posicio.X + Longitud*Metre - Pos*Metre;
               when Oest_Est =>
                  C.X := Posicio.X + Pos*Metre;
               when others =>
                  null;
            end case;

      elsif Tipus=Vertical then

            case Carril is
               when Esquerra =>
                  C.X := Posicio.X + Metre;
               when Dret =>
                  C.X := Posicio.X;
            end case;
            case Sentit is
               when Nord_Sud =>
                  C.Y := Posicio.Y + Pos*Metre;
               when Sud_Nord =>
                  C.Y := Posicio.Y + Longitud*Metre - Pos*Metre;
               when others =>
                  null;
            end case;

      end if;
      return C;
   end Obtenircoordenadavehicle;
   -- Obté la següent posició del vehicle dins del Tram
   procedure Obtenirseguentposicio (
         Pos       : in out Integer;
         Carril    : in out Ttipuscarril;
         C         :    out Tcoordenada;
         Fitram    : in out Boolean;
         Migtram   :    out Boolean;
         Sem       :    out Pparellsemafor;
         Tipustram :    out Ttipustram      ) is
   begin
      -- Cas: vehicle arriva al final del Tram
      if Pos >=Longitud-1 then

            if Carril=Esquerra then
               Carrilesquerra(Pos):=False;
            else
               Carrildret(Pos):=False;
            end if;
            Pos:=0;
            Carril:=Esquerra;
            Fitram:=True;
            Tipustram:=Tipus;
            Sem:=Semafor;

      -- Cas: vehicle està al carril esquerra
         -- Subcasos:
         -- Cas 1: El vehicle està essent adelantat per un altre
         -- Cas 2: Quan la següent del carril esquerra està lliure
         -- Cas 3: Es posiciona al carril dret degut a que la posició del davant
```

```
està ocupada
         elsif Carril=Esquerra then

               if Carrildret(Pos) or Carrildret(Pos+1) then

                     --Si s'entra aquí dins és degut a que algú vol abançar el
vehicle,
                     --com que els conductors tenen un comportament tolerant
aleshores
                     --deixem que ens abanci i ens oblidem de fer "curses".

                     null;

               elsif not Carrilesquerra(Pos+1) then

                     Carrilesquerra(Pos):=False;
                     Pos:=Pos+1;
                     Carrilesquerra(Pos):=True;

               elsif not Carrildret(Pos+1) then

                     Carrilesquerra(Pos):=False;
                     Carril:=Dret;
                     Pos:=Pos+1;
                     Carrildret(Pos):=True;

               end if;
         -- Cas: El vehicle està al carril dret
            -- Subcasos:
            -- Cas 1: El vehicle adelantador es torna a situar a l'esquerra
            -- Cas 2: El vehicle adelantador continúa adelantant
         else
               if ((not Carrilesquerra(Pos)) and (not Carrilesquerra(Pos+1))) then

                     Carrildret(Pos):=False;
                     Carril:=Esquerra;
                     Pos:=Pos+1;
                     Carrilesquerra(Pos):=True;

               elsif not Carrildret(Pos+1) then

                     Carrildret(Pos):=False;
                     Pos:=Pos+1;
                     Carrildret(Pos):=True;

               end if;

         end if;
         -- Comprova si està al mig del Tram (només ho utilitza el camió
d'escombreries)
         Migtram:= (Pos=Longitud/2);
         C:= Obtenircoordenadavehicle(Pos, Carril);
      end Obtenirseguentposicio;
      -- Demana situar-se al carril esquerra
      procedure Demanarsituarsealcarrilesquerra (
            Pos    : in out Integer;
            C      :    out Tcoordenada;
            Carril : in out Ttipuscarril ) is
      begin
```

```
         if Carril=Esquerra then
            null; --Si ja està a l'esquerra, doncs no cal fer res...
         else

               if not Carrilesquerra(Pos+1) then

                     Carrildret(Pos):=False;
                     Pos:=Pos+1;
                     Carrilesquerra(Pos):=True;
                     Carril:=Esquerra;

               end if;

         end if;
         C:= Obtenircoordenadavehicle(Pos, Carril);
      end Demanarsituarsealcarrilesquerra;
   end Tram;

   protected body Parellsemafor is
      -- S'encuen els vehicles que volen demanar pas vertical
      entry Demanarpasvertical (
            Des : in     Tdesicio;
            T :    out Ptram     ) when ((Torn and not Mode_Fluidesa) or
                                         (Torn and (Demanarpasvertical'Count > 0)
and Mode_Fluidesa) or
                                         (Torn and (Demanarpasvertical'Count = 0)
and (Demanarpashoritzontal'Count = 0) and Mode_Fluidesa) or
                                         (not Torn and (Demanarpasvertical'Count >
0)and (Demanarpashoritzontal'Count = 0) and (Mode_Fluidesa))) is
         I : Item;
      begin
         case Motorgrafic is
            when Metode_Directe =>
               D.Actualitzarimatgesemafor(Id, true);
            when Metode_Buffering =>
               I.Object:=False;
               I.Id:=Id;
               I.Pas:=true;
               Enqueue(I,C);
         end case;

         case Des is
            when Recte =>
               T:=Tramvertical;
            when Girar =>
               T:=Tramhoritzontal;
         end case;
      end Demanarpasvertical;
      -- S'encuen els vehicles que volen demanar pas horitzontal
      entry Demanarpashoritzontal (
            Des : in     Tdesicio;
            T :    out Ptram     ) when ((not Torn and not Mode_Fluidesa) or
                                         (not Torn and (Demanarpashoritzontal'Count
> 0) and Mode_Fluidesa) or
                                         (not Torn and (Demanarpashoritzontal'Count
= 0) and (Demanarpasvertical'Count = 0) and Mode_Fluidesa) or
                                         (Torn and (Demanarpashoritzontal'Count >
0) and (Demanarpasvertical'Count = 0) and (Mode_Fluidesa))) is
         I : Item;
```

```
      begin

         case Motorgrafic is
            when Metode_Directe =>
               D.Actualitzarimatgesemafor(Id, false);
            when Metode_Buffering =>
               I.Object:=False;
               I.Id:=Id;
               I.Pas:=false;
               Enqueue(I,C);
         end case;

         case Des is
            when Recte =>
               T:=Tramhoritzontal;
            when Girar =>
               T:=Tramvertical;
         end case;
      end Demanarpashoritzontal;
      -- Retorna el número de vehicles en espera vertical i horitzontal
      procedure Obtenirnumerovehiclesenespera (
            H :    out Integer;
            V :    out Integer  ) is
      begin
         H:= Demanarpashoritzontal'Count;
         V:= Demanarpasvertical'Count;
      end Obtenirnumerovehiclesenespera;
      -- Inicialitza el semàfor
      procedure Inicialitzar (
            Ident : in     Integer;
            Tv    : in     Ptram;
            Th    : in     Ptram;
            Dib   : in     Pdibuixant;
            Cua   : in     Pqueue;
            Tmg   :        Tipusmotorgrafic ) is
      begin
         Mode_Fluidesa:= False;
         Torn:=True;
         Id:=Ident;
         Tramvertical:=Tv;
         Tramhoritzontal:=Th;
         D:=Dib;
         C:=Cua;
         Motorgrafic:=Tmg;
      end Inicialitzar;
      -- Activa el mode de fluïdesa
      procedure Activarmodefluidesa is
      begin
         Mode_Fluidesa := True;
      end Activarmodefluidesa;
      -- Desactiva el mode fluïdesa
      procedure Desactivarmodefluidesa is
      begin
         Mode_Fluidesa := False;
      end Desactivarmodefluidesa;
      -- Intercanvía el torn del semàfor (vertical -> horitzontal, horitzontal ->
vertical)
      procedure Canviartorn is
         I : Item;
```

```
      begin
         Torn := not Torn;

            case Motorgrafic is
               when Metode_Directe =>
                  D.Actualitzarimatgesemafor(Id, Torn);
               when Metode_Buffering =>
                  I.Object:=False;
                  I.Id:=Id;
                  I.Pas:=Torn;
                  Enqueue(I,C);
            end case;
      end Canviartorn;
   end Parellsemafor;

   task body Control is
      Grua_Ocupada    : Boolean := False;
      Control_Activat : Boolean := False;
   begin
      loop
         select
            -- Rep l'avís que la grua està disponible (missatge que només envia la
Grua)
            accept Avis_Grua_Disponible do
               Grua_Ocupada:=False;
            end Avis_Grua_Disponible;
         or
            -- Despatxa el següent vehicle espatllat si el control de trànsit està
activat i la Grua està lliure
            when Control_Activat and not Grua_Ocupada =>
            accept Despatxar_Vehicle_Espatllat (
                  Id        : Integer;
                  Tipus     : Ttipusvehicle;
                  T         : Ptram;
                  Pos       : Integer;
                  Carril    : Ttipuscarril;
                  TipusTram : TtipusTram ) do
               Vehicle_Grua.Anar_A_Buscar_Cotxe_Espatllat
(Id,Tipus,T,Pos,Carril,TipusTram);
               Grua_Ocupada:=True;
            end Despatxar_Vehicle_Espatllat;
         or
            -- Activa el control de trànsit i el mode de fluïdesa per cada semàfor
            accept Activar_Control do
               for X in 0..M-1 loop
                  for Y in 0..N-1 loop
                     Semafors(X,Y).Activarmodefluidesa;
                  end loop;
               end loop;
               Control_Activat:=True;
               Loger.Avis_Estat_Control_Transit(Control_Activat);
            end Activar_Control;
         or
            -- Desactiva el control de trànsit i el mode de fluïdesa per cada
semàfor
            accept Desactivar_Control do
               for X in 0..M-1 loop
                  for Y in 0..N-1 loop
                     Semafors(X,Y).Desactivarmodefluidesa;
```

```
                  end loop;
               end loop;
               Control_Activat:=False;
               Loger.Avis_Estat_Control_Transit(Control_Activat);
            end Desactivar_Control;
         or
            -- Obté el nombre de vehicles espatllats
            accept ObtenirNumeroVehiclesEspatllats(NumVehicles: out Integer) do
               NumVehicles:=Despatxar_Vehicle_Espatllat'Count;
            end ObtenirNumeroVehiclesEspatllats;
         end select;
      end loop;

   end Control;

end ElementsSistema;
```

4.9. MapeigEntorn.ads

```
with QueueArray; use QueueArray;
With PaquetGrafic; use PaquetGrafic;
with Tipus; use Tipus;

Package MapeigEntorn is

   type TCoordenada is record
      x : Integer;
      y : Integer;
   end record;

   Procedure CrearTrams(M: in Integer; N: in Integer);
   Procedure CrearSemafors(M: in Integer; N: in Integer; Cua: in Pqueue; Tsem: in
Integer; uts: in Integer);
   Procedure Inicialitzar_Trams_i_Semafors(M: in Integer; N: in Integer; llarg: in
Integer; d: in PDibuixant; Cua: in Pqueue; MotorGrafic: in TipusMotorGrafic);

End MapeigEntorn;
```

4.10. MapeigEntorn.adb

```
-- PAQUET MAPEIG ENTORN --

With ElementsSistema; use ElementsSistema;
With PaquetGrafic; use PaquetGrafic;

package body MapeigEntorn is
   -- Inicialitza els semàfors i assigna coordenades als diferents objectes de la
ciutat
   Procedure Inicialitzar_Trams_i_Semafors(M: in Integer; N: in Integer; llarg: in
Integer; d: in PDibuixant; Cua: in Pqueue; MotorGrafic: in TipusMotorGrafic) is
      SentitHoritzontal: TTipusSentit;
      SentitVertical: TTipusSentit:=NORD_SUD;
      xHoritzontal, yHoritzontal, xVertical, yVertical, id: Integer;
   begin
         id:=0;
         for x in 0..M-1 loop
```

```
            SentitHoritzontal:=OEST_EST;
            for y in 0..N-1 loop
               xHoritzontal:=x*(llarg*Metre + 2*Metre);
               yHoritzontal:=(y+1)*(llarg*Metre) + (y*2*Metre) + 1;
               xVertical:=(x+1)*(llarg*Metre) + (x*2*Metre) + 1;
               yVertical:=y*(llarg*Metre + 2*Metre);

               if SentitVertical=NORD_SUD and SentitHoritzontal=OEST_EST then

                     TramsHoritzontals(x,y).Inicialitzar(HORITZONTAL, OEST_EST,
Semafors(x,y), llarg, (xHoritzontal,yHoritzontal), (x*2, y*2 + 1));
                     TramsVerticals(x,y).Inicialitzar(VERTICAL, NORD_SUD, Semafors
(x,y), llarg, (xVertical,yVertical), (x*2 + 1, y*2));
                     Semafors(x,y).Inicialitzar(id,TramsVerticals(x,(y+1) mod N),
TramsHoritzontals((x+1) mod M,y), d, Cua, MotorGrafic);
                     SentitHoritzontal:=EST_OEST;

               elsif SentitVertical=NORD_SUD and SentitHoritzontal=EST_OEST then

                     TramsHoritzontals(x,y).Inicialitzar(HORITZONTAL, EST_OEST,
Semafors((x-1+M) mod M,y), llarg, (xHoritzontal,yHoritzontal), (x*2, y*2 + 1));
                     TramsVerticals(x,y).Inicialitzar(VERTICAL, NORD_SUD, Semafors
(x,y), llarg, (xVertical,yVertical), (x*2 + 1, y*2));
                     Semafors(x,y).Inicialitzar(id,TramsVerticals(x,(y+1) mod N),
TramsHoritzontals(x,y),d,Cua, MotorGrafic);
                     SentitHoritzontal:=OEST_EST;

               elsif SentitVertical=SUD_NORD and SentitHoritzontal=OEST_EST then

                     TramsHoritzontals(x,y).Inicialitzar(HORITZONTAL, OEST_EST,
Semafors(x,y), llarg, (xHoritzontal,yHoritzontal), (x*2, y*2 + 1));
                     TramsVerticals(x,y).Inicialitzar(VERTICAL, SUD_NORD, Semafors
(x,(y-1+N) mod N), llarg, (xVertical,yVertical), (x*2 + 1, y*2));
                     Semafors(x,y).Inicialitzar(id,TramsVerticals(x,y),
TramsHoritzontals((x+1) mod M,y),d,Cua, MotorGrafic);
                     SentitHoritzontal:=EST_OEST;

               elsif SentitVertical=SUD_NORD and SentitHoritzontal=EST_OEST then

                     TramsHoritzontals(x,y).Inicialitzar(HORITZONTAL, EST_OEST,
Semafors((x-1+M) mod M,y), llarg, (xHoritzontal,yHoritzontal), (x*2, y*2 + 1));
                     TramsVerticals(x,y).Inicialitzar(VERTICAL, SUD_NORD, Semafors
(x,(y-1+N) mod N), llarg, (xVertical,yVertical), (x*2 + 1, y*2));
                     Semafors(x,y).Inicialitzar(id,TramsVerticals(x,y),
TramsHoritzontals(x,y),d,Cua, MotorGrafic);
                     SentitHoritzontal:=OEST_EST;

               end if;
               d.InserirSemafor(id,((x+1)*(llarg*Metre) + x*2*Metre + 1, (y+1)*
(llarg*Metre) + y*2*Metre + 1));
               id:=id+1;
            end loop;

            if SentitVertical=NORD_SUD then SentitVertical:=SUD_NORD;
            else SentitVertical:=NORD_SUD; end if;
         end loop;
   end Inicialitzar_Trams_I_Semafors;

   -- Crea les tasques Timer associades a cada semàfor
```

```
   Procedure CrearSemafors(M: in Integer; N: in Integer; Cua: in Pqueue; Tsem: in
Integer; uts: in Integer) is
      timer: PTimerParellSemafor;
   begin
         Semafors:= new TMatriuParellSemafors(0..M-1,0..N-1);
         for x in 0..M-1 loop
            for y in 0..N-1 loop
               Semafors(x,y):= new ParellSemafor;
               timer := new TimerParellSemafor(Semafors(x,y), Tsem, uts);
            end loop;
         end loop;
   end CrearSemafors;
   -- Crea els Trams Horitzontals i Verticals
   procedure CrearTrams(M: in Integer; N: in Integer) is
   begin
         TramsHoritzontals:= new TMatriuTrams(0..M,0..N);
         for y in 0..N loop
            for x in 0..M loop
               TramsHoritzontals(x,y):= new Tram;
            end loop;
         end loop;

         TramsVerticals:= new TMatriuTrams(0..M,0..N);
         for x in 0..M loop
            for y in 0..N loop
               TramsVerticals(x,y):= new Tram;
            end loop;
         end loop;
   end CrearTrams;

end MapeigEntorn;
```

4.11. PaquetGrafic.ads

```
with Jewl.Windows;
with Tipus; use Tipus;

Package PaquetGrafic is

   type TComanda is (Quit, Pintar);
   package Sketch_Windows is new JEWL.Windows(TComanda); use Sketch_Windows;

   type TLlistaVehicles is array(Integer  range <>) of Canvas_Type;
   type PTLlistaVehicles is access TLlistaVehicles;
   type TLlistaSemafors is array(Integer  range <>) of Canvas_Type;
   type PTLlistaSemafors is access TLlistaSemafors;

   protected type Dibuixant is
      Procedure ActualitzarImatgeVehicle(id: in integer; C: in TCoordenada; tipus:
in TTipusVehicle);
      Procedure ActualitzarImatgeSemafor(id: in integer; t: in boolean);
      Procedure Inicialitzar(t: in String; num: in Integer; l: in Integer; N: in
Integer; M: in Integer; nums: in Integer);
      Procedure PintarEdificis(N: in Integer; M: in Integer; ample: in Integer);
      procedure InserirVehicle(id: in Integer; tipus: in TTipusVehicle);
      Procedure InserirSemafor(id: in Integer; C: in TCoordenada);
```

```
   private
      NumVehicles: Integer;
      NumSemafors: Integer;
      Vehicles: PTLlistaVehicles;
      Semafors: PTLlistaSemafors;
      Finestra: Frame_Type;
      Superficie: Canvas_Type;
   end Dibuixant;

   Type PDibuixant is access Dibuixant;

End PaquetGrafic;
```

4.12. PaquetGrafic.adb

```
-- PAQUET GRÀFIC --

with Jewl.Windows;

package body PaquetGrafic is

   protected body Dibuixant is
      -- Dibuixa els edificis al mapa
      procedure Pintaredificis (
            N     : in     Integer;
            M     : in     Integer;
            Ample : in     Integer  ) is
      begin
         Set_Fill(Superficie, Gray);
         for Y in 0..N-1 loop
            for X in 0..M-1 loop
               Draw_Rectangle(Superficie,(X*((Ample*Metre)+(2*Metre)),Y*((
                           Ample*Metre)+(2*Metre))),Ample*Metre, Ample*
                  Metre,(10,10));
            end loop;
         end loop;
         Set_Fill(Superficie, (255,128,64));
         Draw_Rectangle(Superficie, (0,(Ample*Metre) - (2*Metre)), 3*Metre,
2*metre, (5,5));
      end Pintaredificis;
      -- Actualitza la imatge d'un vehicle a la nova posició
      procedure Actualitzarimatgevehicle (
            Id    : in     Integer;
            C     : in     Tcoordenada;
            Tipus : in     Ttipusvehicle ) is
         Color : Colour_Type;
      begin
         Set_Origin(Vehicles(Id), (C.X, C.Y));
         case Tipus is
            when Cotxe =>
               Color:=Blue;
            when Ciclomotor =>
               Color:=Yellow;
            when Camio =>
               Color:=Red;
            when Bicicleta =>
               Color:=Green;
```

```
         when Camio_Escombraries =>
            Color:=Magenta;
         when Camio_Grua =>
            Color:= Black;
      end case;
      Set_Colour(Vehicles(Id), Color);
   end Actualitzarimatgevehicle;
   -- Actualitza la imatge d'un semàfor al nou estat
   procedure Actualitzarimatgesemafor (
         Id : in     Integer;
         T  : in     Boolean  ) is
   begin
      Erase(Semafors(Id));
      if T then

            Draw_Line(Semafors(Id),(1,1),(1, 2*Metre - 3));
            Draw_Line(Semafors(Id),(2*Metre - 4,1),(2*Metre - 4, 2*
                  Metre - 3));

      else

            Draw_Line(Semafors(Id),(1,1),(2*Metre - 3,1));
            Draw_Line(Semafors(Id),(1,2*Metre - 4),(2*Metre - 3, 2*
                  Metre - 4));

      end if;
   end Actualitzarimatgesemafor;
   -- Reserva un nou Canvas per al vehicle que s'insertarà
   procedure Inserirvehicle (
         Id    : in     Integer;
         Tipus : in     Ttipusvehicle ) is
      Color : Colour_Type;
   begin
      Vehicles(Id):= Canvas (Finestra, (0,0), Metre, Metre, Pintar);
      case Tipus is
         when Cotxe =>
            Color:=Blue;
         when Ciclomotor =>
            Color:=Yellow;
         when Camio =>
            Color:=Red;
         when Bicicleta =>
            Color:=Green;
         when Camio_Escombraries =>
            Color:=Magenta;
         when Camio_Grua =>
            Color:=Black;
      end case;
      Set_Colour(Vehicles(Id), Color);
      Save(Vehicles(Id));
   end Inserirvehicle;
   -- Reserva un nou Canvas per al nou semàfor
   procedure Inserirsemafor (
         Id : in     Integer;
         C  : in     Tcoordenada ) is
   begin
      Semafors(Id):= Canvas (Finestra, (C.X, C.Y), 2*Metre, 2*Metre,
         Pintar);
      Set_Colour(Semafors(Id), (212,208,200));
```

```
         Save(Semafors(Id));
         Actualitzarimatgesemafor(Id, True);
      end Inserirsemafor;
      -- Inicialitza els llistats de Canvas i crea la finestra de la simulació
      procedure Inicialitzar (
            T    : in     String;
            Num  : in     Integer;
            L    : in     Integer;
            N    : in     Integer;
            M    : in     Integer;
            Nums : in     Integer  ) is
      begin
         Numvehicles:= Num;
         Vehicles:= new Tllistavehicles(0..Numvehicles-1);
         Numsemafors:= Nums;
         Semafors:= new Tllistasemafors(0..Numsemafors-1);
         Finestra:= Frame (M*((L*Metre)+(2*Metre)) + 5 + (2*Metre), N*((L*
                  Metre)+(2*Metre)) + 25 + (2*Metre), T, Quit);
         Set_Origin(Finestra, (5,100));
         Superficie:= Canvas (Finestra, (0,0), 0, 0, Pintar);
         Set_Colour(Superficie, (212,208,200));
         Pintaredificis(N,M,L);
         Save(Superficie);
      end Inicialitzar;
   end Dibuixant;

end PaquetGrafic;
```

4.13. TascaHistoric.ads

```
with Tipus; use Tipus;

package TascaHistoric is

   task type Historial is
      entry Avis_Estat_Control_Transit(estat: boolean);
      entry Avis_Insersio_Vehicle(Tipus: Ttipusvehicle; quantitat: Integer);
      entry Avis_Vehicle_Espatllat(Tipus: Ttipusvehicle);
      entry Avis_Recorregut_Escombriaire_Iniciat;
      entry Avis_Recorregut_Escombriaire_Completat;
      entry Avis_Recorregut_Escombriaire_No_Completat;
      entry Avis_Grua_Busca_Vehicle;
      entry Avis_Grua_Troba_Vehicle;
      entry Avis_Grua_Deixa_Vehicle_Taller;
   end Historial;
   type PHistorial is access Historial;

end TascaHistoric;
```

4.14. TascaHistoric.adb

```
-- PAQUET TASCA HISTÒRIC --

with Jewl.Simple_Windows;
use Jewl.Simple_Windows;
with Tipus;
```

```
use Tipus;

package body TascaHistoric is
   -- Transforma de dígit a caràcter
   function Dig_A_Car (
         Digit : in     Integer )
     return String is
   begin
      case Digit is
         when 0 =>
            return "0";
         when 1 =>
            return "1";
         when 2 =>
            return "2";
         when 3 =>
            return "3";
         when 4 =>
            return "4";
         when 5 =>
            return "5";
         when 6 =>
            return "6";
         when 7 =>
            return "7";
         when 8 =>
            return "8";
         when 9 =>
            return "9";
         when others =>
            return "0";
      end case;
   end Dig_A_Car;
   -- Transforma de enter a cadena
   function Enter_A_Cadena (
         Num : in     Integer )
     return String is
      S      : String (1 .. 3);
      D1,
      D2,
      D3     : Integer;
      Numero : Integer         := Num;
   begin
      D1:=abs(Numero/100);
      Numero:=Numero - (abs(Numero/100))*100;
      D2:=abs(Numero/10);
      Numero:=Numero - (abs(Numero/10))*10;
      D3:=Numero;
      S:=Dig_A_Car(D1) & Dig_A_Car(D2) & Dig_A_Car(D3);
      return S;
   end Enter_A_Cadena;
   -- Mostra la informació dels diversos esdeveniments que van succeint, mitjançant
una finestra.
   task body Historial is
      F : Frame_Type       := Frame (280, 160, "Historial", 'Q', Font ("Arial", 8,
Bold => False));
      M : Memo_Type        := Memo (F, (5, 5), 260, 120);
      S : String (1 .. 17);
   begin
```

```
      Set_Origin(F, (515,5));
      loop
         select
            -- rep els missatges d'activació/desactivació del control de trànsit
            accept Avis_Estat_Control_Transit (
                  Estat : Boolean ) do
               case Estat is
                  when True =>
                     Append_Line(M,"Control de trànsit activat");
                  when False =>
                     Append_Line(M,"Control de trànsit desactivat");
               end case;
            end Avis_Estat_Control_Transit;
         or
            -- rep els missatges d'inserció de nous vehicles
            accept Avis_Insersio_Vehicle (
                  Tipus     : Ttipusvehicle;
                  Quantitat : Integer        ) do
               S:= "S'ha inserit " & Enter_A_Cadena(Quantitat) & " ";
               case Tipus is
                  when Cotxe =>
                     Append_Line(M, S & "cotxe/s");
                  when Ciclomotor =>
                     Append_Line(M,S & "ciclomotor/s");
                  when Camio =>
                     Append_Line(M,S & "camió/ns");
                  when Bicicleta =>
                     Append_Line(M,S & "bicicleta/es");
                  when Camio_Escombraries =>
                     Append_Line(M,S & "camió/ns d'escombraries");
                  when Camio_Grua =>
                     Append_Line(M,S & "grua/es");
                  when others =>
                     null;
               end case;
            end Avis_Insersio_Vehicle;
            or
            -- rep els missatges de nou vehicle espatllat
            accept Avis_Vehicle_Espatllat (
                  Tipus : Ttipusvehicle ) do
               S:= "S'ha espatllat un";
               case Tipus is
                  when Cotxe =>
                     Append_Line(M, S & " cotxe");
                  when Ciclomotor =>
                     Append_Line(M,S & " ciclomotor");
                  when Camio =>
                     Append_Line(M,S & " camió");
                  when Bicicleta =>
                     Append_Line(M,S & " bicicleta");
                  when others =>
                     null;
               end case;
            end Avis_Vehicle_Espatllat;
            or
            -- rep el missatge d'escombriaire inicia el recorregut
            accept Avis_Recorregut_Escombriaire_Iniciat do
               Append_Line(M, "L'escombriaire inicia el recorregut");
            end Avis_Recorregut_Escombriaire_Iniciat;
```

```
            or
            -- rep el missatge d'escombriaire que a completat el recorregut
            accept Avis_Recorregut_Escombriaire_Completat do
               Append_Line(M, "L'escombriaire ha completat el recorregut");
            end Avis_Recorregut_Escombriaire_Completat;
            or
            -- rep el missatge d'escombriaire que no a pogut completar el
recorregut
            accept Avis_Recorregut_Escombriaire_No_Completat do
               Append_Line(M, "L'escombriaire no ha completat el recorregut");
            end Avis_Recorregut_Escombriaire_No_Completat;
            or
             -- rep el missatge que la grua surt a buscar un vehicle
            accept Avis_Grua_Busca_Vehicle do
               Append_Line(M, "La grua surt a buscar un vehicle espatllat");
            end Avis_Grua_Busca_Vehicle;
            or
               -- rep el missatge que la grua ha trobat el vehicle
            accept Avis_Grua_Troba_Vehicle do
               Append_Line(M, "La grua ha trobat el vehicle espatllat");
            end Avis_Grua_Troba_Vehicle;
            or
               -- rep el missatge que la grua ha deixat el vehicle al taller
            accept Avis_Grua_Deixa_Vehicle_Taller do
               Append_Line(M, "La grua ha deixat el vehicle espatllat al taller");
            end Avis_Grua_Deixa_Vehicle_Taller;
         end select;
      end loop;
   end Historial;

end TascaHistoric;
```

4.15. TascaPintora.ads

```
with PaquetGrafic; use PaquetGrafic;
with QueueArray; use QueueArray;

package TascaPintora is

   task type Pintor(D: Pdibuixant; Q: Pqueue);
   type PPintor is access Pintor;

end TascaPintora;
```

4.15. TascaPintora.adb

```
-- PAQUET TASCA PINTORA --
-- Només s'utilitza quan el tipus de motor gràfic que s'escolleix és amb el que
està basat amb Sistema de Buffering
-- És una tasca autoadaptada que regula la seva freqüència de lectura en funció de
la quantitat d'elements que hi ha a la cua
with Tipus; use Tipus;

package body TascaPintora is
   -- Llegeix de la cua els esdeveniments que s'han de pintar
```

```
   task body Pintor is
      Cua: Pqueue := Q;
      I: Item;
      Buit: Boolean;
      EnEspera: Integer;
   begin
      loop
         Dequeue(I,Buit,EnEspera,Cua);
         If not Buit then

               case I.Object is
                  when true => D.ActualitzarImatgeVehicle(I.Id, I.Coor, I.Tipus);
                  when false => D.ActualitzarImatgeSemafor(I.Id, I.Pas);
               end case;

               if ((EnEspera >= 0) and (EnEspera < 5)) then  Delay (0.005);
               elsif ((EnEspera >= 5) and (EnEspera < 10)) then  Delay (0.00005);
               elsif ((EnEspera >= 10) and (EnEspera < 50)) then  Delay (0.000005);
               elsif EnEspera >= 50 then Delay (0.00000005);
               end if;
            else Delay (0.05);
         end if;

      end loop;
   end Pintor;

end TascaPintora;
```

4.17. ControlMenu.ads

```
with Tipus; use Tipus;
with ElementsSistema; use ElementsSistema;
with PaquetGrafic; use PaquetGrafic;
with QueueArray; use QueueArray;
with TascaHistoric; use TascaHistoric;

package ControlMenu is

   task type TascaControlMenu(Ident: Integer; D: Pdibuixant; Q: Pqueue; T: Ptram;
p: integer; TMG: TipusMotorGrafic; Control_Transit: Pcontrol; loger: PHistorial;
uts: integer);
   type PTascaControlMenu is access TascaControlMenu;

end ControlMenu;
```

4.18. ControlMenu.adb

```
-- PAQUET CONTROL MENÚ --

-- podem inserir vehicles durant l'execució del programa mitjançant un menú

with Jewl.Simple_Windows; use  Jewl.Simple_Windows;
with TascaHistoric; use TascaHistoric;
package body ControlMenu is
```

```
task body TascaControlMenu is
   F   : Frame_Type   := Frame (335, 86, "Menú de Control", 'Q', Font ("Arial", 8,
Bold => False));
   B1   : Button_Type := Button (F, (5, 5), 75, 20, "Inserir Cotxe", 'A');
   B2   : Button_Type := Button (F, (85, 5), 75, 20, "Inserir Camio", 'B');
   B3   : Button_Type := Button (F, (165, 5), 75, 20, "Inserir Moto", 'C');
   B4   : Button_Type := Button (F, (245, 5), 75, 20, "Inserir Bici", 'D');
   B5   : Button_Type := Button (F, (5, 30), 140, 20, "Activar control de trànsit",
'E');

   Id: Integer:=Ident;
   C: PVehicle;
   control_activat: boolean:=false;
begin
   Set_Origin(F, (5, 5));
   -- comprova la opció escollida i la porta a terme
   loop
      case Next_Command is
         when 'A' => D.Inserirvehicle(Id, COTXE);
                     C := new Vehicle(Id, D, T, COTXE, Q, TMG, p,
Control_transit,loger,uts);
                     loger.Avis_Insersio_Vehicle(COTXE, 1);
                     Id:=Id+1;
         when 'B' => D.Inserirvehicle(Id, CAMIO);
                     C := new Vehicle(Id, D, T, CAMIO, Q, TMG, p,
Control_transit,loger,uts);
                     loger.Avis_Insersio_Vehicle(CAMIO, 1);
                     Id:=Id+1;
         when 'C' => D.Inserirvehicle(Id, CICLOMOTOR);
                     C := new Vehicle(Id, D, T, CICLOMOTOR, Q, TMG, p,
Control_transit,loger,uts);
                     loger.Avis_Insersio_Vehicle(CICLOMOTOR, 1);
                     Id:=Id+1;
         when 'D' => D.Inserirvehicle(Id, BICICLETA);
                     C := new Vehicle(Id, D, T, BICICLETA, Q, TMG, p,
Control_transit,loger,uts);
                     loger.Avis_Insersio_Vehicle(BICICLETA, 1);
                     Id:=Id+1;
         when 'E' => if not control_activat then
                        begin
                           Control_transit.Activar_Control;
                           control_activat:=true;
                           Set_Text(B5, "Desactivar control de trànsit");
                        end;
                     else
                        begin
                           Control_transit.Desactivar_Control;
                           control_activat:=false;
                           Set_Text(B5, "Activar control de trànsit");
                        end;
                     end if;
         when others => null;
      end case;

   end loop;

end TascaControlMenu;

end ControlMenu;
```

4.19. ControlIndicador.ads

```
with Jewl.Simple_Windows; use  Jewl.Simple_Windows;
with ElementsSistema; use ElementsSistema;

package ControlIndicadors is

   type TMatriuEtiquetes is array(Integer  range <>, Integer  range <>) of
Label_Type;
   type PTMatriuEtiquetes is access TMatriuEtiquetes;

   task type TascaControlIndicadors(Semafors: PTMatriuParellSemafors; M: Integer;
N: Integer; control_transit: Pcontrol);
   type PTascaControlIndicadors is access TascaControlIndicadors;

end ControlIndicadors;
```

4.19. ControlIndicador.adb

```
-- PAQUET CONTROL INDICADORS --

with Jewl.Simple_Windows; use  Jewl.Simple_Windows;

package body ControlIndicadors is

task body TascaControlIndicadors is
   MatriuEtiquetesSemaforsVerticals: PTMatriuEtiquetes;
   MatriuEtiquetesSemaforsHoritzontals: PTMatriuEtiquetes;
   Etiqueta, Etiqueta2: Label_Type;
   EtiquetaMaxim, EtiquetaEspatllats: Label_Type;
   F   : Frame_Type;
   EnEsperaVertical: Integer:=0;
   EnEsperaHoritzontal: Integer:=0;
   TotalVehiclesEnEspera: Integer:=0;
   TotalVehiclesEspatllats: Integer:=0;
begin
   F:= Frame (M*35 + 25, N*15 + 95, "Indicadors", 'Q', Font ("Arial", 8, Bold =>
False));
   Set_Origin(F,(345,5));
   Etiqueta:= Label (F, (5, N*15 + 20), 120, 15, "Vehicles en espera:", Center);
   EtiquetaMaxim:= Label (F, (120, N*15 + 20), 20, 15, "0", Center);
   Etiqueta2:= Label (F, (5, N*15 + 40), 120, 15, "Vehicles espatllats:", Center);
   EtiquetaEspatllats:= Label (F, (120, N*15 + 40), 20, 15, "0", Center);
   MatriuEtiquetesSemaforsVerticals:= new TMatriuEtiquetes(0..M-1,0..N-1);
   Matriuetiquetessemaforshoritzontals:= new Tmatriuetiquetes(0..M-1,0..N-1);

   -- Inicialitza el valor inicial de les etiquetes que es mostren per pantalla
   for x in 0..M-1 loop
      for y in 0..N-1 loop
         MatriuEtiquetesSemaforsHoritzontals(x,y):= Label (F, (x*15 + 10, y*15 +
10), 15, 15, "0", Center);
         MatriuEtiquetesSemaforsVerticals(x,y):= Label (F, (x*15 + 25 + M*15, y*15
+ 10), 15, 15, "0", Center);
      end loop;
   end loop;
```

```
   loop
      Totalvehiclesenespera:=0;
      -- Recorrem tots els semàfors de la ciutat i obtenim el nombre de vehicles en
espera de cadascun
      for x in 0..M-1 loop
         for y in 0..N-1 loop
            Semafors(x,y).ObtenirNumeroVehiclesEnEspera(EnEsperaHoritzontal,
EnEsperaVertical);
            Set_Text (MatriuEtiquetesSemaforsVerticals(x,y), Integer'Image
(EnEsperaVertical));
            Set_Text (Matriuetiquetessemaforshoritzontals(X,Y), Integer'Image
(Enesperahoritzontal));
            -- suma total de vehicles en espera
            TotalVehiclesEnEspera:=TotalVehiclesEnEspera + EnEsperaVertical +
EnEsperaHoritzontal;
            Set_Text (EtiquetaMaxim, Integer'Image(TotalVehiclesEnEspera));
            control_transit.ObtenirNumeroVehiclesEspatllats
(TotalVehiclesEspatllats);
            Set_Text (EtiquetaEspatllats, Integer'Image(TotalVehiclesEspatllats));
         end loop;
      end loop;
      delay(0.05);
   end loop;

end TascaControlIndicadors;

end ControlIndicadors;
```