Aplicació de representació gràfica de GRAFS de mobilitat

Memòria
TFC - SIG
2010-2011-2on Semestre

Santi Mora Geli
Enginyeria Tècnica en Informàtica de Sistemes
Consultor: Ramon Català Pou

Vull dedicar aquest treball a la Núria, la meva dona, per animar-me a començar, continuar i acabar aquesta carrera, i al Nil, el nostre primer fill, que ha nascut fa pocs dies. Ara podré dedicar-los tot el temps que es mereixen.

Resum

La finalitat d'aquest Treball de Final De Carrera d'Enginyeria Tècnica en informàtica de Sistemes és aprofundir en el món del Sistemes d'Informació Geogràfica (SIG).

L'objectiu d'aquest projecte és desenvolupar una aplicació que ens permeti generar representacions gràfiques de grafs de mobilitat, visualitzats amb l'entorn GeoMedia Professional. A partir d'unes bases de dades d'entrada, on els trams i les cruïlles seran representades visualment com a línies i punts respectivament, dissenyarem una aplicació que generi una base de dades de sortida on les dades seran visualitzades com un polígons en el cas dels carrers i cruïlles, i la senyalització de cada tram també serà representada gràficament, de manera que tindrem una representació en dos dimensions dels carrers amb la seva senyalització corresponent. En aquest cas, l'usuari decidirà quina senyalització vol veure representada.

Per al desenvolupament del projecte, a part del GeoMedia Professional, utilitzarem el Microsoft Access per la gestió de les bases de dades i el Microsoft Visual Basic 2005 com a llenguatge de programació, a banda del diferent programari per a l'edició de les diverses tasques del projecte.

Índex de continguts

1.- Introducció....4
1.1.- Descripció i justificació del projecte....4
1.2.- Objectius....4
1.3.- Planificació....5
1.4.- Programari....7
1.5.- Pla de riscos....9
2.- Sistemes d'informació geogràfica (SIG)....9
2.1.- Introducció als SIG....9
2.2.- Components d'un SIG....10
2.3.- Bases de dades d'un SIG....11
2.3.1.- Dades geogràfiques....11
2.3.1.1.- Dades vectorials....11
2.3.1.2.- Dades raster....12
2.3.1.3.- Comparativa de models....13
2.3.2.- Dades temàtiques....13
3.- Integraph GeoMedia Professional 6.1....14
3.1.- Introducció al Geomedia Professional 6.1....14
3.2.- GeoWorkspace i finestra de mapes....14
3.3.- Sistemes de coordenades....16
3.4.- Magatzems....17
4.- Treball pràctic....18
4.1.- Anàlisi de l'estructura de dades existent....18
4.1.1.- Metadades....19
4.1.1.1.- Dades específiques de Geomedia....19
4.1.1.2.- Taules d'objectes de dades geogràfiques (GDO)....21
4.1.2.- Dades de l'usuari....21
4.2.- Anàlisi funcional de l'aplicació....23
4.2.1.- Casos d'ús....23
4.2.2.- Model de dades resultant....24
4.2.3.- Interfície d'usuari....26
4.3.- Disseny de l'aplicació....27
4.3.1.- Model d'objectes....27
4.3.1.1.- GDO....29
4.3.1.2.- Geometry Collection....30
4.3.2.- Classes i funcions a desenvolupar....31
4.3.2.1.- Classe frmTFC1....32
4.3.2.2.- Classe Tram2D....33
4.3.2.3.- Classe SenyalTram....34
4.3.2.4.- Classe Senyal....35
4.3.2.5.- Classe Linia....35
4.3.2.6.- Codi de la classe Tram2D....36
4.4.- Construcció del sistema....36
4.4.1.- Funcionament de l'aplicació....36
4.4.2.- Futures Línies de millora....38
4.4.3.- Conclusions....39
5.- Bibliografia....40
Annexe 1: Diagrama de Gantt....41
Annexe 2: Codi de la classe Tram2D....43

1.- Introducció

1.1.- Descripció i justificació del projecte

En l'actualitat, la majoria d'ajuntaments amb servei de gestió i planificació de la mobilitat, disposen de sistemes d'informació geogràfica per l'edició i manteniment de les seves dades. Aquests sistemes representen els carrers com una línia, associada a dades com el nom de carrer, capacitat, sentit, velocitat, etc...

El problemes d'aquests sistemes sorgeixen, per exemple, al fer una presentació o un tríptic, ja que la visualització de la informació que se'ns ofereix no és útil.

El nostre objectiu serà dissenyar una aplicació que, amb les dades proporcionades pels ajuntaments o altres administracions, doni com a resultat un representació gràfica més atractiva i útil.

1.2.- Objectius

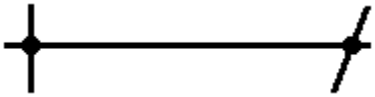
L'objectiu del projecte consisteix en desenvolupar una aplicació que ens permeti millorar la representació gràfica d'aquests sistemes, de tal forma que en comptes de línies, els carrers estiguin representats per polígons i totes les dades estiguin representades dins el carril, igual com als carrers reals. Per tant, transformarem la base cartogràfica estàndar, com les que utilitzen les diverses administracions, en una altra base de dades que doni com a resultat una millor visualització d'aquestes dades, de tal forma que l'usuari pugui elegir el tipus de dades que vulgui representar i com han de ser representades. Per aquesta visualització d'aquestes dades, utilitzarem el GeoMedia Professional.

Per aquest objectiu, tindrem com a entrada de l'aplicació una base de dades, com les que utilitzen els ajuntaments, amb una taula d'arcs, una de nodes i una tercera que els combina per representar els trams dels carrers. Com a sortida, tindrem una nova base de dades on els arcs es transformaran en polígons (que representaran els carrils), amb les seves corresponents marques segons el tipus de carril o sentit representades com a imatges inserides en els polígons.

En la base de dades d'entrada tenim les taules següents:

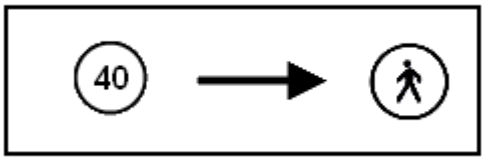
- CARRILS: Conté les dades d'un carril d'un tram de carrer.
- CRUÏLLES: Conté les dades de les cruïlles on es troben dos o més carrers.
- TRAMS: A partir d'un carril i dues cruïlles, es generarà el tram d'un carrer. Conté totes les dades del tram tals com nom, velocitat, tipus, etc...

La representació gràfica d'aquestes dades fent servir el format que hem de millorar donaria com a resultat quelcom així:



Il·lustració 1: Tram en visualització estàndar

Les línies representarien els trams dels carrers i el punts les cruïlles on es creuen. L'objectiu d'aquests projecte és dissenyar una eina en que les dades es puguin visualitzar d'una forma semblant a aquesta:



Il·lustració 2: Visualització del tram amb la nova aplicació

En aquesta nova visualització, tindrem una representació en un polígon que representarà cada tram del carrer amb els diferents senyals indicadors del tipus de carril que volem veure (sentit, velocitat, carrer peatonal, carril bici, carril bus, passos de vianants, etc...) representats dins el polígon, de tal forma que un cop tinguem la representació d'un recorregut, podrem veure els carrers d'una forma més amigable i pràctica que la que generen els sistemes actuals, amb tota la informació que volem veure representada i descartant la que no volem. Cada cop que executem l'aplicació, guardarem les dades de tal forma que la podem recuperar sempre que volem.

1.3.- Planificació del treball

Per a la planificació de l'execució del projecte, seguirem les pautes de les quatre tasques proposades en l'enunciat, les quals ens portaran de forma ordenada per les diferents fites necessàries per tal de desenvolupar el projecte d'una forma correcte. Aquestes tasques derivaran, en una sèrie de subtasques que complementaran el treball fet. L'objectiu final serà el lliurament de la memòria del projecte i la presentació virtual d'aquest. Les tasques a desenvolupar seran les següents:

1. **Anàlisi de les estructures de dades existents:**

   Anàlisi de les estructures de dades que s'utilitzen actualment per emmagatzemar dades de mobilitat.

   Inclourà:

   - **Tipologia de dades (dades necessàries per la correcta implementació de**

l'aplicació).

- **Estructures i formats de dades de bases comercials (Taules, camps i formats)**

Es procedirà a l'estudi de les dades que s'utilitzen habitualment per els SIG. S'haurà d'analitzar la tipologia de les dades i la seva representació per tal de facilitar la transformació posterior. S'analitzaran també les estructures de dades de bases comercials per tal de trobar el model que s'adeqüi millor a les nostres necessitats. Obtindrem les bases per l'anàlisi funcional de l'aplicació.

2. **Anàlisi funcional de l'aplicació:**

Anàlisi de les estructures de dades que s'utilitzen actualment per emmagatzemar dades de mobilitat.

Inclourà:

- **Casos d'ús (casuística d'ús de l'aplicació).**
- **Model de dades resultant (Estructura de dades de sortida que oferirà l'aplicació).**
- **Interfícies d'usuari (Imatges visuals de l'aplicació).**

Un cop analitzades les estructures de dades, passarem a fer l'anàlisi del sistema. En primer lloc es realitzarà el model de casos d'us de l'aplicació. Això en donarà una primera idea del funcionament de l'aplicació. Posteriorment, es dissenyarà el model de dades resultant, amb l'estructura de dades de sortida sobre la que treballarà el sistema. En darrer lloc, es dissenyaran les interfícies d'usuari, obtenint les primeres imatges visuals de com resultarà l'aplicació.

3. **Tasca 3: Disseny de l'aplicació:**

Disseny del sistema. És un document que ha d'incloure com a mínim:

- **Model d'objectes (Relació d'objectes i llibreries que s'utilitzaran en el desenvolupament).**
- **Classes i funcions a desenvolupar (Llista detallada de les funcions previstes a desenvolupar, del seu ús i de les seves relacions).**

Amb els resultats obtinguts en la tasca anterior, es procedirà al disseny de l'aplicació. Caldrà obtenir el model d'objectes, amb la relació d'objectes i llibreries que utilitzarem en el desenvolupament, així com les classes i funcions a desenvolupar. Com a resultat d'aquest procés, tindrem tot llest per començar el muntatge de l'aplicació.

4. **Construcció del sistema:**

   És la programació pròpia de l'aplicació. S'executarà el pla de proves elaborat en la fase de disseny, que servirà de control d'errors de l'aplicació.
   .
   En aquesta fase, es bolcarà tot el treball fet fins el moment en la construcció de l'aplicació objecte d'aquest projecte. Si s'han seguit correctament les pautes d'anàlisi i disseny establertes durant el procés, en la construcció del sistema es veurà si les decisions que s'han pres fins al moment són les correctes.

Tots el treball fet anteriorment quedarà reflectit en la memòria, la qual es lliurarà al final del projecte conjuntament amb la presentació virtual amb vídeo i àudio i el codi font del projecte.

Per a una planificació més detallada i exacte i el seu posterior seguiment, es disposa d'un diagrama de Gantt amb aquesta planificació. Es pot consultar el diagrama al final d'aquesta memòria. (Veure índex)

A grans trets, la planificació consta de les següents fases:

- En primer lloc, es planifica la lectura dels diferents mòduls tant del Treball Final de Carrera (TFC) com de Sistemes d'Informació Geogràfica o Geotelemàtica (SIGG) vinculant els diferents mòduls al progrés del projecte.

- Donat que no es pot garantir el seguiment d'un pla de treball estricte per motius personals, l'evolució del projecte s'encararà a complir amb el lliurament final de la memòria, la presentació virtual i el codi font dins les dates establertes. El lliurament de les PAC 2 i 3, donat que no és obligatori, no es tindrà en compte a l'hora de lliurar les tasques. Aquestes s'aniran entregant al consultor a mida que estiguin llestes per tal que aquest les pugui avaluar i fer el seguiment del procés del projecte.

- De totes maneres, s'ha fet una planificació més detallada per tal de, tot i no respectar les dates de lliurament, poder fer un seguiment lògic de les tasques i subtasques, sense deixar-ne cap. Aquesta planificació està detallada en un diagrama de Gantt. Es pot consultar el diagrama al final d'aquesta memòria. (Veure índex)

1.4.- Programari

Per a l'execució del projecte, es necessitarà la disponibilitat de programari especialitzat. S'ha facilitat l'accés gratuït a aquest programari mitjançant llicències d'estudiant o llicències temporals. El programari utilitzat durant el projecte, és el següent:

- **GeoMedia Professional 6.1[1]:** Es tracta d'una eina creada per Integraph, que ens serveix, per una banda, per a l'examen i anàlisi de dades geogràfiques, així com per presentar-les tant gràficament en pantalla, com fer impressions de mapes i dades. D'altra banda, ens serveix com a eina de captura i manteniment de dades i treballar tant amb dades vectorials

1.- http://www.intergraph.com/sgi/products/productFamily.aspx?family=10&country=

com amb *rasters*, i de forma combinada. Un dels grans avantatges, és que ens permet personalitzar-lo amb eines com el Microsoft Visual Basic o Visual C++, permetent adaptar-lo a les nostres necessitats. En un capítol posterior de la memòria aprofundirem en aquest programa.

- **Microsoft Access 2007[2]:** Aquest programa de gestió de bases de dades, forma part del paquet ofimàtic Microsoft Office 2007. Es tracta d'un sistema de gestió de bases de dades (SGBD) destinat a l'ús personal i de petites empreses. GeoMedia Professional disposa d'un ventall d'eines per utilitzar MS Acces per la gestió de les dades din el format Geomedia Access Warehouse. Tot i que Microsoft Access no és un SGBD pròpiament dit, com si ho serien plataformes com Microsoft SQL Server o Oracle, les eines que ens ofereix pel tractament de dades són suficients per a les nostres necessitats, ja que ens permet treballar amb consultes i taules amb llenguatge SQL.

- **Microsoft Visual Studio 2005:** Es tracta d'un entorn de desenvolupament integrat (IDE) que suporta diversos llenguatges de programació tals com Visual Basic, .Net o C++. GeoMedia posa a disposició del programador una serie de llibreries especialitzades, de tal forma que dissenyarem l'aplicació, en el nostre cas, utilitzat el llenguatge de programació Visual Basic 2005.

- **Microsoft Project Professional 2003:** Es tracta d'un programari d'administració de projectes. Permet planificar un projecte mitjançant els diagrames de Gantt[3]. S'ha utilitzat aquest programari en el disseny de diagrama inclòs en aquest projecte.

- **Microsoft Visio 2003:** Aquest programa de dibuix vectorial s'utilitza per realitzar tot tipus de diagrames. En el nostres cas, s'ha utilitzat per dissenyar els diferents diagrames de bases dades de la memòria, tals com el diagrama UML o el diagrama de casos d'ús.

- **OpenOffice.org 3[4]:** Es tracta d'un paquet ofimàtic de codi obert (amb llicencia LGPL[5]). En el nostre cas s'ha fet servir l'editor de text **Writer** per l'edició d'aquesta memòria. Aquest editor permet convertir el document directament en format PDF sense necessitat de programari afegit.

- **TechSmith Camtasia Studio 7[6]:** Aquest programa de captura de vídeo de pantalla. Permet capturar i editar en vídeo el que succeix en pantalla, a més d'enllaçar amb MS PowerPoint per tal d'afegir-hi presentacions. S'ha utilitzar per a l'edició de la presentació virtual.

2.- Es pot trobar més informació sobre el diversos programes del paquet ofimàtic MS Office a: http://office.microsoft.com/en-us/products/?CTT=97

3.- http://en.wikipedia.org/wiki/Gantt_chart

4.- http://ca.openoffice.org/

5.- http://ca.wikipedia.org/wiki/LGPL

6.- http://www.techsmith.com/camtasia/

1.5.- Pla de riscos

En aquest pal de riscos, entenem com a riscos els possibles imprevistos que puguin afectar l'avanç normal del projecte. S'han pres diferents decisions per tal de minimitzar els possibles riscos:

- Tal com s'esmenta en la planificació, el risc de no poder fer un seguiment de la planificació, fa que prenem la decisió de no fer el lliurament de les PAC en les seves dates, volcant tots els esforços en el lliurament final.

- En cas que el temps dedicat al projecte no fos suficient , tot i la decisió anterior, i es posés en perill el lliurament final, s'ha fet les gestions pertinents per prendre el temps necessari de les vacances personals per tal de portar el projecte a bon termini.

- Pel que fa a riscos tècnics, per evitar accidents tals com avaries en els terminals, esborrat de dades accidental o pèrdua d'informació, es farà diàriament una copia de seguretat en un disc extern per tal de poder recuperar el treball fet en cas d'alguna de les incidències abans esmentades. Així mateix, es disposa de diversos terminals de reserva, de tal forma que l'avaria d'un d'ells no afecti al projecte. Per últim, es disposa de diversos tipus de connexió a Internet, tant per ADSL com per mòbil 3G, per evitar apareguin problemes en la xarxa i que aquests puguin afectar els lliuraments.

- En darrer lloc, existeix el risc que les dades subministrades per les proves continguin errors, falsejant el pla de proves de l'aplicació. En el cas que es detectés la falta d'integritat de les bases de dades, s'intentarà fer una correcció manual, per exemple, anul·lant files incompletes. A més, es posarà en coneixement del consultor per a la corresponent correcció de la base de dades.

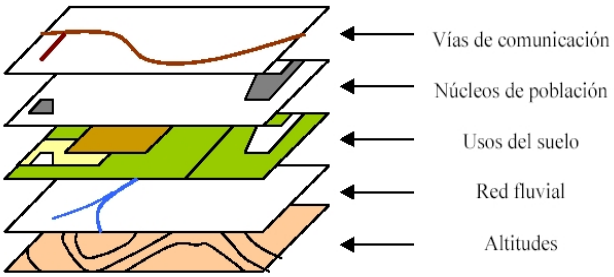
2.- Sistemes d'informació geogràfica (SIG)

2.1.- Introducció als SIG

Entenem per sistema d'informació geogràfica (SIG, o GIS en anglès) com un sistema integrat de hardware, software i dades geogràfiques dissenyat per capturar, emmagatzemar, manipular, analitzar i desplegar en totes les seves formes la informació geogràficament referenciada amb la fi de resoldre problemes complexes de planificació i gestió. Una de les millors definicions sobre els SIG és: "Un sistema de maquinari, programari, dades, persones, organitzacions i convenis institucionals per a la recopilació, emmagatzematge, anàlisi i distribució d'informació de territoris de la Terra" (Deuker; Kjerne, 1989).

Els SIG són fonamentalment sistemes de treball, la raó fonamental dels quals, és la resolució de problemes espacials. Aquests sistemes poden ser emprats per usos tant diversos com investigació científica, gestió de desastres ambientals, urbanisme, o pàgines webs. La tecnologia SIG és àmpliament emprada també en tot tipus d'organitzacions, des de institucions acadèmiques fins a agències governamentals i corporacions. Es per això , que l'ús dels SIG està molt estès a tots els nivells de la societat.

Per resumir de manera bàsica la forma de treballar dels SIG, podem ir que consisteixen en separar les diferents capes d'informació geogràfica en diverses capes temàtiques, que un cop emmagatzemades independentment, es pot accedir a cada una d'elles per separat de forma ràpida i senzilla. Això facilita l'accés per diferents tipus de personal i diferents tipus de necessitats.



Il·lustració 3: Exemple de capes temàtiques d'un SIG (Font Wikipedia)

2.2.- Components d'un SIG

Com ja s'ha comentat en l'apartat anterior, els SIG no només estan compostos per components tecnològics, com programari i equips especialitzats, sinó que també formen part dels SIG els altres components que els envolten. Procedirem a un anàlisis d'aquests components, tenint en compte que sense algun d'aquests, no podríem considerar els SIG com ho fem ara:

- **Tecnologia (programari i maquinari):** En aquest apartat, es consideren components dels SIG qualsevol programari i maquinari que serveixi per accedir, analitzar, presentar i sintetitzar les dades emmagatzemades en la base de dades, no només les dades espacials, sinó també les dades temàtiques especialitzades. Inclou aquest grup les eines d'entrada i manipulació d'informació geogràfica, els sistemes d'administració de dades, eines de consulta i anàlisi, i les interfícies gràfiques per l'accés i visualització de les dades.

- **Idees:** El cos d'idees és el conjunt de coneixements que fan progressar i desenvolupar l'ús dels SIG. Dins aquest conjunt tenim àrees tant diverses com l'enginyeria, la física, les matemàtiques, la geomàtica, la informàtica, la sociologia, etc... Totes aquestes matèries aporten la seva part per al desenvolupament continu dels SIG.

- **Personal:** Quan parlem de personal en el context dels SIG, ens referim a l'organització del personal qualificat en cada una de les àrees abans esmentades. Aquest personal qualificat té repercussió directa en la recerca, el disseny i funcionament de totes les àrees dels SIG.

- **Xarxa:** En un món globalitzat i connectat, la xarxa es transforma en un dels components essencials en els SIG, ja que permet la comunicació i informació compartida amb qualsevol part del món de manera ràpida i eficaç. Això ha permès l'aparició de portals[7] i pàgines web temàtiques que donen accés a gran quantitat d'informació especialitzada, impensable fa uns anys.

- **Dades:** Les dades són la part dels SIG que representen la realitat i permeten relacionar-la amb situacions i aplicacions específiques. Necessitarem processos per a la transferència i accés a les dades per tal de mantenir la quantitat i qualitat de les dades emmagatzemades

7.- Per exemple: http://elfar.diba.es/cil/dadeslitoral/cartograf.htm

en la base de dades. Aquestes dades es divideixen en dos subgrups, que seran les dades espacials i les temàtiques.

- **Mètodes:** Quan parlem del mètodes dels SIG, parlem dels procediments o normes per dur a terme les diferents tasques relacionades amb el disseny, creació i funcionament dels SIG. Aquests mètodes, entre d'altres, poden ser l'anàlisi espacial, la manipulació de dades, el disseny d'interfícies d'usuari i bases de dades, o la interpretació de mapes. Els mètodes, ben establerts, determinen la qualitat del resultat obtingut.

2.3.- Base de dades d'un SIG

D'acord amb el que ja s'ha esmentat en diverses ocasions, les dades utilitzades pels SIG es divideixen en dos grups. En primer lloc tenim les dades geogràfiques, que aporten tota la informació associada a la localització. Aquestes es poden emmagatzemar en dos models diferents: el model vectorial i el model *raster*. En segon lloc, tenim les dades temàtiques, que aporten tota la informació associada a les dades geogràfiques.

2.3.1.- Dades geogràfiques

Les dades geogràfiques aporten la informació associada a la localització. Aquestes dades representen objectes reals tals com edificis, rius, muntanyes o ciutats. Hi ha dos models de representació de les dades geogràfiques utilitzats, el model vectorial, basat en punts, línies i polígons i el model *raster*, basat en una malla regular de cel·les.

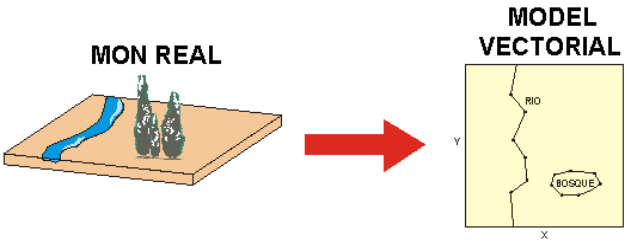
2.3.1.1.- Model vectorial

El model vectorial tracta de la representació del objectes del món real mitjançant punts, línies i polígons, representats aquests per les coordenades que els defineixen. Aquesta representació es centra en la precisió dels elements en l'espai, on els objectes representats són discrets, amb els límits definits. Tenim tres tipus de dades vectorials:

- Punts: Representen un punt concret en el territori, i es tracta de l'abstracció d'un objecte com, per exemple, un vèrtex geodèsic, una font o un arbre. Son objectes amb dimensió zero i venen representats per un parell de coordenades X,Y.

- Arc: També anomenats polilínias, representen un element lineal d'un territori com rius, carreteres o corbes de nivell. Tenen dimensió u. Es representen com un conjunt ordenat de punts successius units per segments entre cada punt i el següent.

- Àrea: També anomenats polígons, representen una àrea d'un territori, com poden ser parcel·les, edificis o comarques. Tenen dimensió dos. Venen representats per polilínies tancades amb la possibilitat de forats interiors.



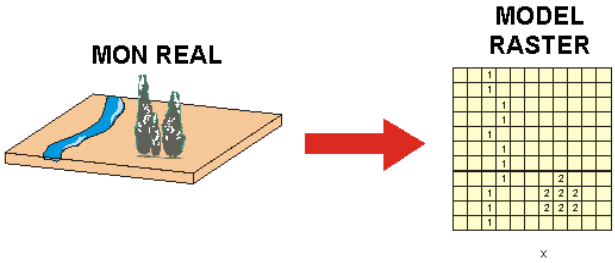
Il·lustració 4: Representació del model vectorial

2.3.1.2.- Model *raster*

En aquest mode, la informació es representa omplint l'espai representat amb una matriu de cel·les o píxels, i assignant un valor concret a cada cel·la en funció de l'objecte que representa. Cada cel·la ve representada per un parell de coordenades X, Y, i la cel·la representa un únic valor. Aquestes cel·les s'agrupen en files i columnes generant en el seu conjunt la representació de la informació. Aquestes dades s'emmagatzemen en diferents formats, com poden ser arxius d'imatge com TIFF[8] o JPEG[9], o grans objectes binaris, BLOB[10], com és el cas de Geomedia amb els seus camps *Geometry*, amb els que s'ha treballat en aquest projecte. La mida de la informació anirà estrictament lligada a la resolució, i per tant, a la qualitat. Com més resolució, més qualitat, però també més dades a emmagatzemar. Podem diferenciar dos tipus de dades *raster*:

- **Cobertures:** Es tracta d'informació geoespacial que representa fenòmens que varien en l'espai. En les cobertures, cada punt georeferenciat del territori te un valor diferent, i es pot concretar fins a cert punt. Representen, per exemple, la quantitat de pluja en una zona, la cota d'alçada o la composició del sòl.

- **Imatges raster:** Són imatges o fotografies aèries que se situen sobre el territori. Aquestes imatges es georeferencien per tal que cada punt de la imatge coicideixi amb la realitat. Tenim un exemple en Google Maps[11].



Il·lustració 5: Representació del model raster

8.- http://es.wikipedia.org/wiki/TIFF
9.-http://es.wikipedia.org/wiki/JPEG
10.- http://es.wikipedia.org/wiki/BLOB
11.- http://maps.google.es/maps?hl=ca&tab=wl

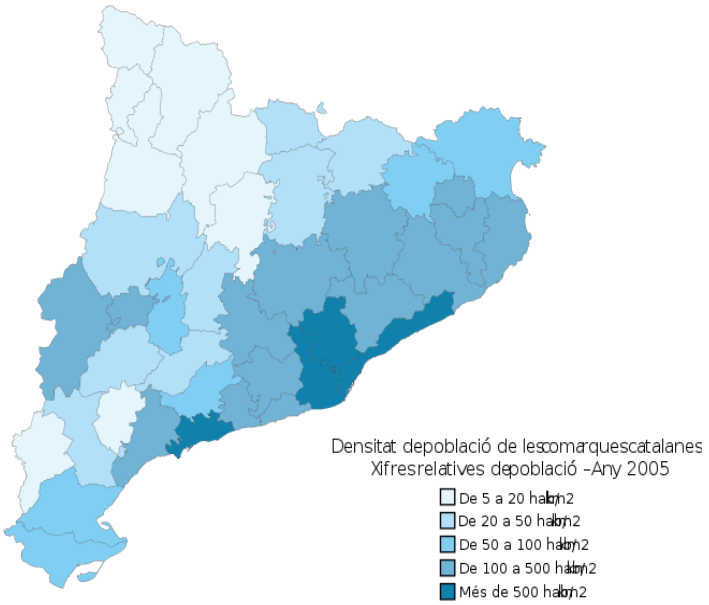
2.3.1.3.- Comparativa de models

Comparant un model amb l'altre, podem observar una serie d'avantatges i d'inconvenients en cadascun. D'una banda, el model vectorial té com a avantatges respecte al model *raster* la precisió gràfica, la integritat de les dades, l'estructura més compacta, la facilitat d'anàlisi de les dades i la facilitat de manteniment de les dades. Per contra, com a inconvenients, tenim que la quantitat de dades és major i l'estructura és més complexa, la dificultat d'actualització i la complexitat en operacions de superposició.

Normalment s'opta per una solució mixta, aprofitant els avantatges de cada model, tal com s'ha dut a terme en aquest projecte.

2.3.2.- Dades temàtiques

Les dades geogràfiques, com s'esmenta, aporten tota la informació associada a la localització. Però a més de les dades de localització, necessitem altres dades per representar més fidelment el món real. Quan necessitem conèixer la població d'una província, la quantitat de pluja en una any en una comarca o el nombre d'afectats per una malaltia en una regió, les dades geogràfiques s'esdevenen insuficients, i per tant, necessitarem unes dades complementàries a aquestes. Aquestes dades són les dades temàtiques, que es coneixen com a atributs o variables d'una base de dades. Aquestes dades, un cop analitzades, són les que ens proporcionen aquesta informació. Podem veure-ho en l'exemple següent referit a la densitat de població de Catalunya per comarques.



Il·lustració 6: Densitat de població de Catalunya per comarques(Any 2005)(Font Wikipedia)

3.- Integraph GeoMedia Professional 6.1

L'objectiu d'aquest projecte és dissenyar una aplicació que modifiqui una base de dades per ser visualitzada amb un programari SIG. En el nostre cas, aquest programa és el GeoMedia Professional 6.1. Per al disseny de l'aplicació s'ha utilitzat el model d'objectes de GeoMedia GDO, sobre el que aprofundirem en l'apartat 4.3.1. En aquest apartat, ens centrarem en una introducció al funcionament i en les diferents característiques d'aquest programari SIG.

3.1.- Introducció al Geomedia Professional 6.1

GeoMedia Professional 6.1 és un programa SIG empresarial comercialitzat per Integraph per sistemes amb plataforma Windows. Es tracta d'un producte per recollir dades SIG, omplir bases de dades i transformar la informació recollida en mapes d'acabat professional.

D'altra banda, es pot utilitzar com a eina d'anàlisis de dades, que permet combinar dades de diferents procedències, en formats diferents, tot en un únic entorn. L'entorn de treball GeoWorkspace permet realitzar consultes complexes amb dades espacials i atributs de diferents orígens, i imprimir el treball amb tot tipus d'acabats.

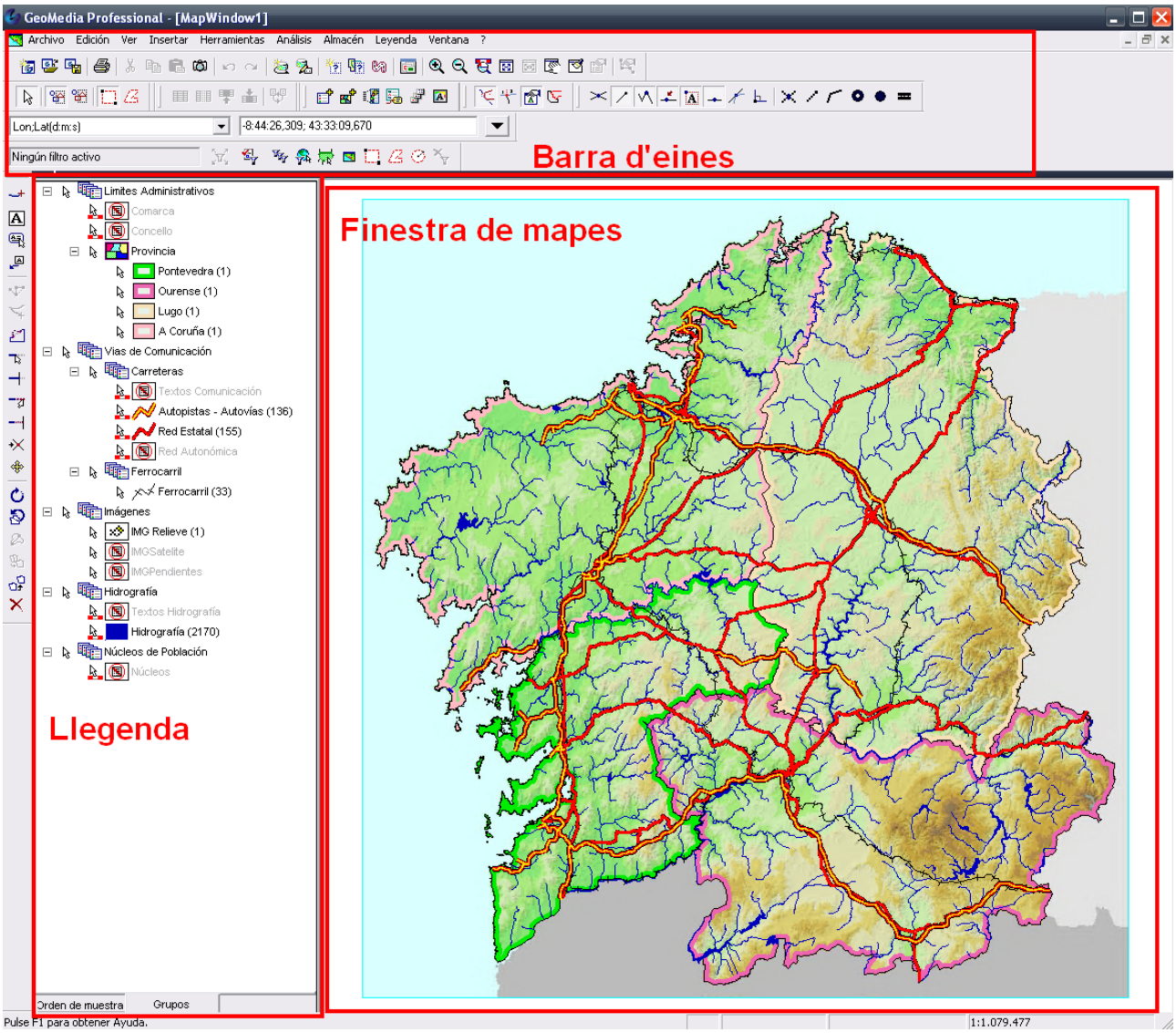
Com a eina de captura i manteniment de dades, GeoMedia permet recollir i editar dades de forma senzilla, ràpida i intel·ligent. A més, en cas d'error en la captura de dades, es disposa d'eines de detecció automàtica d'errors, així com de sistemes intel·ligents de col·locació i edició d'entitats, que permeten corregir els errors detectats de forma senzilla.

GeoMedia permet treballar de forma conjunta amb diferents sistemes de gestió de bases de dades tals com Microsoft SQL, Microsoft Access, Oracle, etc..., amb programes CAD, com AutoCAD, així com treballar conjuntament amb altres programes SIG com MapInfo o ArcInfo. GeoMedia permet també personalitzar les seves eines utilitzant els llenguatges de programació de Microsoft com el Visual Basic, .NET o C#.

3.2.- GeoWorkspace i finestra de mapes

Un GeoWorkspace és l'entorn de treball de GeoMedia. Des d'aquest entorn podem connectar-nos al magatzems de dades, a les finestres de mapa i dades, a les barres d'eines, a la informació dels sistemes de coordenades i a les consultes creades.

Els GeoWorkspace estan construïts sobre una plantilla. Es poden usar les plantilles pròpies del sistema o crear-ne de personalitzades. La plantilla per defecte, *normal.gwt*, disposa d'una finestra de mapa buida, una llegenda buida i un sistema de coordenades predefinit. Els GeoWorkspace es poden desar per tal de recuperar-los posteriorment.



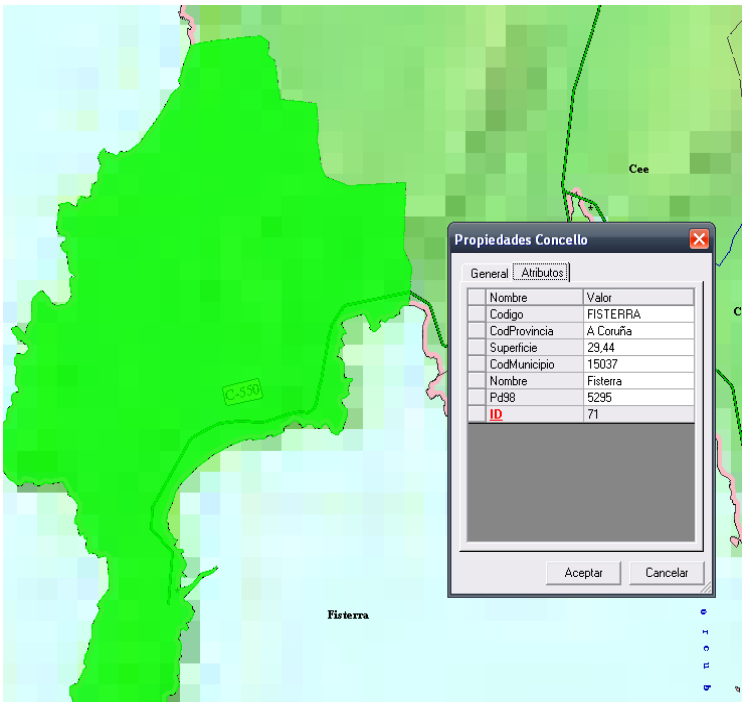
Il·lustració 7: GeoWorkspace de GeoMedia. Mapa de Galícia i llegenda (Captura GeoMedia Professional)

Com podem veure en la il·lustració 7, tenim un exemple de GeoWorkspace obert amb un mapa de Galícia amb tot tipus de dades polítiques províncies, pobles, fronteres, etc... i dades físiques com relleu, rius, carreteres, etc... En la part superior hi ha la barra d'eines, en la part central tenim la finestra de mapes, i la llegenda a la part esquerra. En aquest cas, la llegenda està fixa, però se'ns permet canviar la llegenda a la dreta o deixar-la en posició flotant, guanyant espai per a la finestra de mapes.

Dins la barra d'eines, disposem de totes les eines necessàries per a la creació, edició i visualització dels mapes i les entitats que els componen.

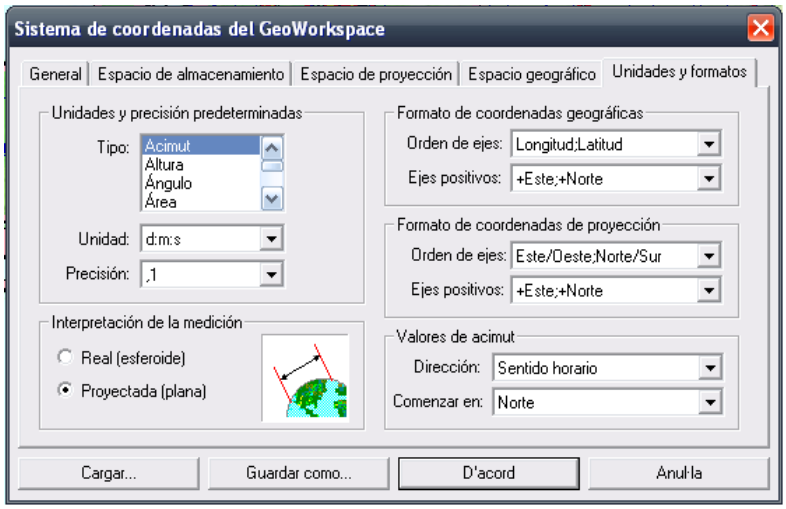
La llegenda posa a la nostra disposició totes les dades disponibles, ja sigui en forma d'entitats, estadístiques o consultes realitzades. El fet que sigui totalment editable ens permet afegir o treure entitats, modificar-ne l'ordre o configurar l'estil, de tal forma que podem personalitzar el mapa editant les entitats.

A l'hora de treballar en la finestra de mapes, una de les principals característiques és consultar els atributs d'una entitat. Deixant el ratolí sobre d'una, aquesta canvia de color, i clicant dos cop obrim la finestra d'atributs. Des de la finestra de mapes podem modificar també les propietats de visualització d'aquests.



Il·lustració 8: Atributs d'una entitat (Captura Geomedia professional)

3.3.- Sistemes de coordenades



Il·lustració 9: Sistema de coordenades GeoMedia (Captura GeoMedia Professional)

Els sistemes de coordenades proporcionen la base matemàtica per relacionar les entitats en estudi amb les seves posicions en el món real. GeoMedia admet diversos tipus de Sistemes de coordenades:

- **Geogràfic** (predeterminat): Aquest sistema de coordenades està referit a un esferoide[12], que expressa les coordenades en forma de longitud i latitud, essent la longitud la distància angular des del meridià d'origen i la latitud la distancia des de l'equador.

- **Projectat:** Aquest sistema està referit a un pla de projecció amb una relació coneguda amb l'esferoide. Aquesta relació s'expressa en coordenades en format X i Y, on X apunt a l'Est i Y al Nord.

- **Geocèntric:** Aquest sistema es refereix a un sistema cartesià amb centre a la Terra, que expressa les coordenades des d'un punt especific en format X, Y i Z respecte al centre de la Terra. En aquest sistema, l'eix X es correspon amb el meridià principal que passa per l'equador, l'eix Y passa per la intersecció de l'equador amb el pla de 90º cap a l'Est i l'eix Z correspon a l'eix polar de la Terra.

Com que la forma de la Terra no és esfèrica, la superfície varia d'un lloc a un altre. Per compensar aquestes variacions, el programa interpreta les coordenades com una xarxa de punts geodèsics anomenada *datum geodèsic*[13]. El *datum geodèsic* defineix l'el·lipsoide[14] de referencia, que és el model utilitzat per representar la superfície terrestre.

En l'espai d'emmagatzemament, podem definir les unitats d'emmagatzemament horitzontal i vertical en diversos tipus d'unitats.

En l'espai de projecció podem definir el sistema de projecció[15].

En l'espai geogràfic definim el *datum geodèsic* i l'el·lipsoide de referencia, si cal.

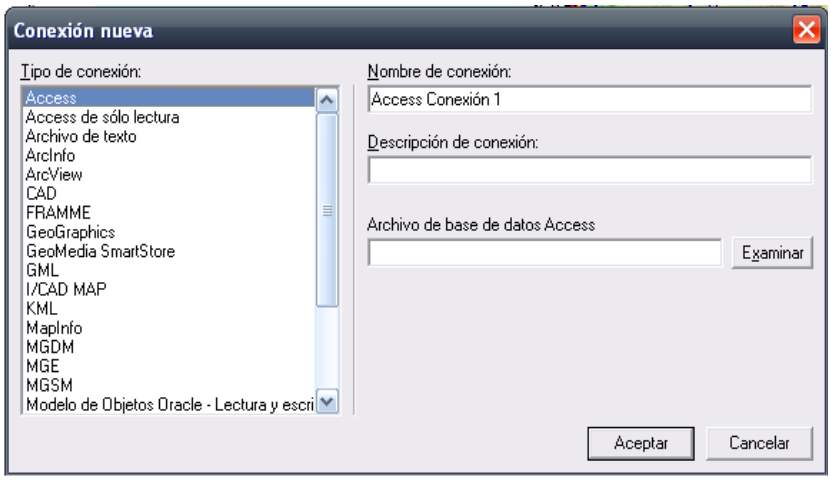
En darrer lloc, a l'apartat d'unitats i formats definim les unitats i precisió predeterminades, d'interpretació de la mesura i el format de les coordenades gràfiques.

GeoMedia permet la configuració d'aquests sistemes de coordenades, i disposa de diversos sistemes predefinits, així com la configuració dels propis sistemes de coordenades.

3.4.- Magatzems

Per treballar amb les dades, ho farem mitjançant les connexions amb magatzems. Les dades desades en aquests magatzems, són les que posteriorment seran visualitzades en el GeoWorkspace. Cada connexió de magatzem utilitza un servidor de dades per visualitzar-la. GeoMedia permet connectar amb diferents formats aliens tals com MS Acces, MS SQL Server, Oracle, MapInfo, ARC/INFO, AutoCAD, SmatrtStore Server, etc... tot i que només els tres primers són de lectura i escriptura, mentre que la resta són només de lectura. Aquesta varietat de connexions permet que GeoMedia treballi amb molts tipus diferents de dades, permetent crear diverses connexions a la vegada i que puguin ser visualitzades conjuntament.

12.- http://ca.wikipedia.org/wiki/Esferoide
13.- http://es.wikipedia.org/wiki/Datum
14.- http://es.wikipedia.org/wiki/Elipsoide
15.- http://es.wikipedia.org/wiki/Proyecci%C3%B3n_geogr%C3%A1fica



Il·lustració 10: Connexió magatzems GeoMedia (Captura GeoMedia professional)

GeoMedia permet, a més, la creació directa de magatzems en format MS Access des del WorkSpace, possibilitat utilitzada en aquest projecte per la creació de la bases de dades de sortida de l'aplicació. L'opció de crear directament els magatzems només està disponible per aquesta plataforma, mentre que per treballar amb SQL o Oracle les bases de dades s'han de configurar.

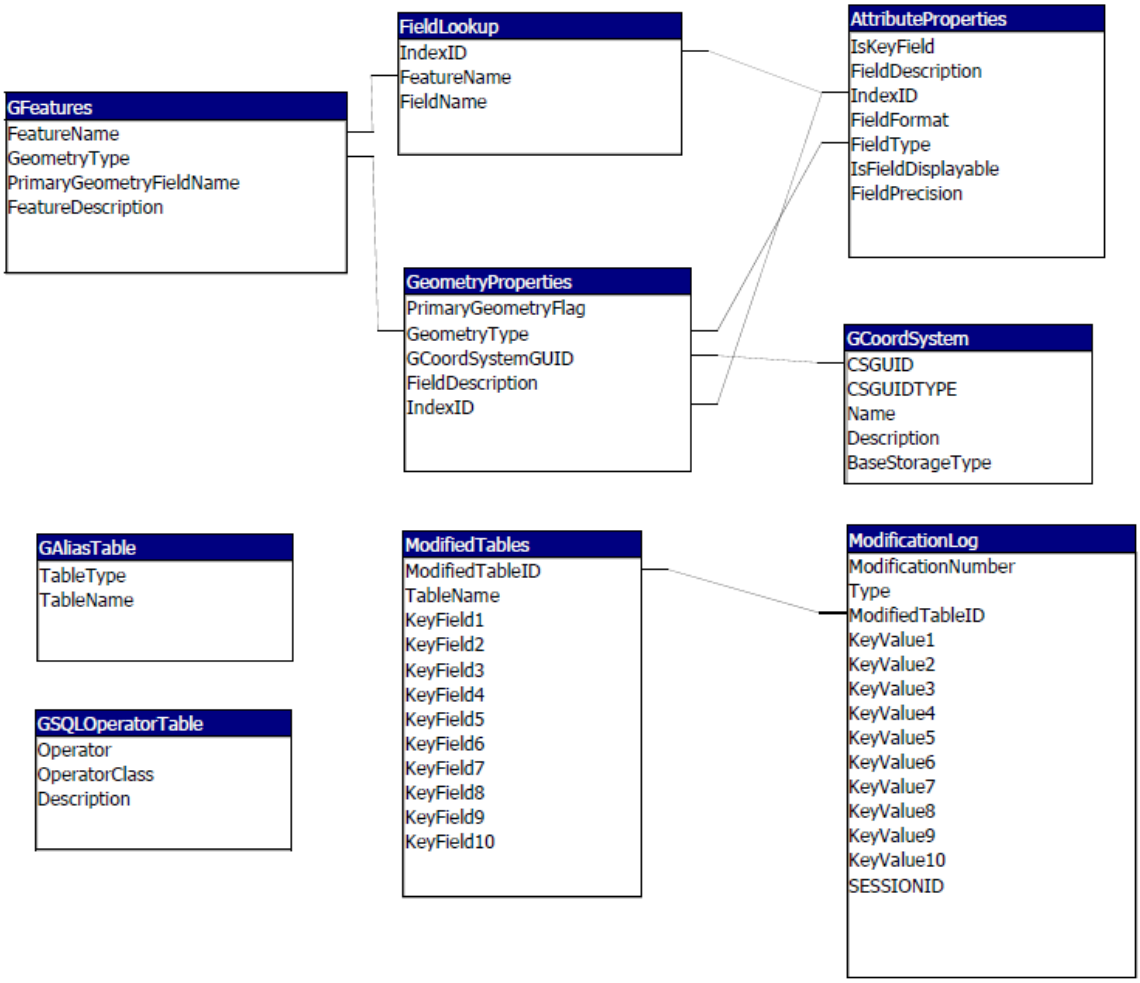
4.- Treball Pràctic

En aquest capítol es descriurà el treball pràctic fet per a l'execució del projecte. Aquest treball s'ha fet de forma seqüencial, seguint l'odre de les tasques proposades en l'enunciat. Per tant, seguirem el mateix ordre per exposar l'evolució del projecte, ja que el tenir llesta una tasca deixa la fina feta per començar la següent.

4.1.- Anàlisi de l'estructura de dades existent

En l'apartat 2.3 s'ha exposat la tipologia de dades d'un SIG. En aquest punt ens centrarem en les dades proporcionades per provar l'aplicació i sobre les que tenim que fer la transformació. Aquests dades es poden dividir en dos grups. Per una banda tenim les metadades, que es tracta d'unes dades que determinen les dades que volem estudiar. Aquestes dades són gestionades per GeoMedia i sense elles no es podrien visualitzar les dades de l'usuari. L'altre grup de dades són les dades pròpies de l'usuari. En aquest cas, tenim tres taules que ens proporcionen tota la informació per la nostra aplicació.

4.1.1.- Metadades



Il·lustració 11: Metadades de GeoMedia i relacions entre elles (Captura MS Access)

Podem definir les metadades com un conjunt de dades que determinen d'altres dades emmagatzemades, són dades sobre dades. Permeten al sistema emmagatzemar propietats sobre atributs de les dades que han d'estar necessàriament en qualsevol base de dades geogràfica. Aquestes dades es generen de forma automàtica al generar una base de dades dins del SIG, en el nostre cas, GeoMedia. Les taules de metadades i les relacions entre elles són les següents:

Procedim a descriure la funció de cada taula i els seus principals camps:

4.1.1.1.- Dades especifiques de Geomedia:

- ***AtributeProperties:*** Descriu els atributs dels camps enumerats en la taula *FieldLookUp*. Els camps de la taula son:

  - ISKEYFIELD: Ens indica si és clau primaria (-1) o no ho és (0).

  - INDEXID: Relaciona la taula AttributeProperties amb FieldLookUp. Identifica cada camp en les dues taules.

- FIELDFORMAT: Determina el format general de les dades que es mostren.
- FIELDTYPE: Determina el tipus de cada camp de la taula FieldLookUp:

1.	Booleà	8.	Date
2.	Byte	9.	Text
3.	Integer	10.	Binary
4.	Long	11.	Memo
5.	Currency	12.	GUID
6.	Single	13.	Spatial geometry
7.	Double	14.	Graphic geometry

- ISFIELDDISPLAYABLE: Determina si una columna apareix en GeoMedia. El valor serà -1 si hi apareix i 0 si està oculta.
- FIELDPRECISION: Defineix el nombre de decimals exposats a GeoMedia. En valors numèrics, el valor determinat és 6.

- ***FieldLookUp*:** Proporciona un identificador únic per a cada camp de les taules de l'usuari. Els camps d'aquesta taula són:
  - INDEXID: Identificador únic del camp.
  - FEATURENAME: Nom de la taula.
  - FIELDNAME: Nom del camp.
- ***GeometryProperties*:** Determina el sistema de coordenades que s'assigna a cada classe d'entitat. Identifica les taules amb representació geomètrica i n'emmagatzema el tipus de geometria, el CSGUID al que correspon, i l'identificador únic de la taula FieldLoookUp. Els camps d'aquesta taula són:
  - PRIMARYGEOMETRYFLAG: Si té un valor de -1, aquest camp és el camp de geometria principal (si n'hi ha), mentre que la resta tenen un valor de 0.
  - GEOMETRYTYPE: Determina el tipus de geometria de l'objecte:

1.	gdbLinear	4.	gbdAreal
2.	gdbAnySpatial	5.	gdbCoverage
3.	gdbGraphicsText	6.	gdbPoint

- GCOORDSYSTEMGUID: Conté el CSGUID de la taula CCoordSystem.

- ***GFeatures*:** Emmagatzema els noms de les taules propies dels usuaris. Els camps d'aquesta taula són:

  - FEATURENAME: Nom de les taules del usuari.
  - GEOMETRYTYPE: Determina el tipus de geometria assignat.
  - PRIMARIGEOFIELDNAME: Nom del camp de geometria primaria.

- ***GSQLOperatorTable:*** Ens proporciona els operadors que podrem fer servir per a consultes SQL.

4.1.1.2.- Taules relacionades amb els objectes de dades geogràfiques (GDO):

- ***GAliasTable***: Determina el nom de les taules pròpies del GeoMedia. No es pot modificar.

- ***GCoordSystem***: Emmagatzema les definicions del sistema de coordenades. Aquesta taula no s'ha de modificar, degut a la seva complexitat: Els principals camps de la taula són:

  - CSGUID: Identifica els paràmetres del sistema de coordenades. Associa un objecte de geometria a un sistema de coordenades.

- ***ModificationLog***: Aquesta taula fa un seguiment de totes les modificacions realitzades en les taules pròpies de Geomedia.

- ***ModifiedTables***: Es tracta d'una vista que proporciona la ID del sistema per cada taula.

4.1.2.- Dades de l'usuari

Per últim tenim les dades pròpies de l'usuari. Aquestes taules ens proporcionen totes les dades complementàries necessàries per la nostra aplicació. Ens proporcionen els objectes geogràfics que seran representats visualment des de GeoMedia i la informació no geogràfica vinculada a ells, necessària per complimentar l'aplicació. Inicialment tenim tres taules:

- ***CARRILS*:** Identifica cada carril d'un tram i ens dóna les certes dades d'aquest carril:

  - OBJECTID: Identificador únic del carril. És de tipus *AutoNumber*.
  - IDTRAM: Identifica el tram al qual pertany el carril.
  - NUM: Número de carril dins el tram:

- AMPLE: Ample del carril.

- SENTIT: Sentit del trànsit en el carril. Sentit 1: Esquerra-Dreta, Sentit 2: Dreta-Esquerra i Sentit 0: Doble sentit.

- PEATONAL: Indica si el carre és peatonal (1) o no (0)

- ***CRUILLES*:** Representa el creuament de dos o més carrers. Aquests objectes estan representats visualment per un punt, per tant, ja estem parlant de dades geogràfiques. D'aquesta taula ens interessen només alguns camps, ja que la resta no s'utilitzen i estan buits:

- ID1: Identificador únic de la cruïlla. Està en format *AutoNumber*.

- IDCRUILLA: Identificador de la cruïlla. Es tracta de l'identificador que es cridarà des d'altres taules quan es necessiti connectar una cruïlla.

- GEOMETRY: Camp que indica que es tracta d'un objecte geomètric, i per tant tindrà representació visual a GeoMedia. Les dades es deixen en format binaris llargs (BLOB[16]), qeu serà interpretat pel sistema. Podem consultar el seu tipus en les metadades en la taula GeometryProperties. En aquest cas es tracta d'un punt.

- GEOMETRY_SK: Fa referencia al sistema de coordenades utilitzat per la representació de les dades.

- ***TRAMS*:** Representa un tram de carrer que hi ha entre dos cruïlles. Està compost per un carril i dues cruïlles. A més, ens proporciona tota la informació necessària per el tram en concret. Alguns dels camps que ens interessen són:

- ID1: Identificador únic del tram. Està en format *AutoNumber*.

- IDTRAM: Identificador del tram per ser utilitzar en la crida des d'altres taules.

- CRUILLADE: Cruïlla d'origen del tram.

- CRUILLAA: Cruïlla de destí del tram.

- CARRILS: Número de carrils de circulació. Si és de doble sentit, seran dos carrils, cosa que pot ocasionar errors en el moment de la programació.

- VELOCITAT: Velocitat permesa en el tram.

16.- http://es.wikipedia.org/wiki/BLOB

- PEATONAL: Indica si el carril és peatonal (1) o no (0).
- NOM: Nom del carrer:
- GEOMETRY: Indica que es tracta d'un objecte geomètric i que tindrà representació visual des de GeoMedia. Es aquest cas està en format Linia.
- GEOMETRY_SK: Fa referencia al sistema de coordenades utilitzat per la representació de les dades.

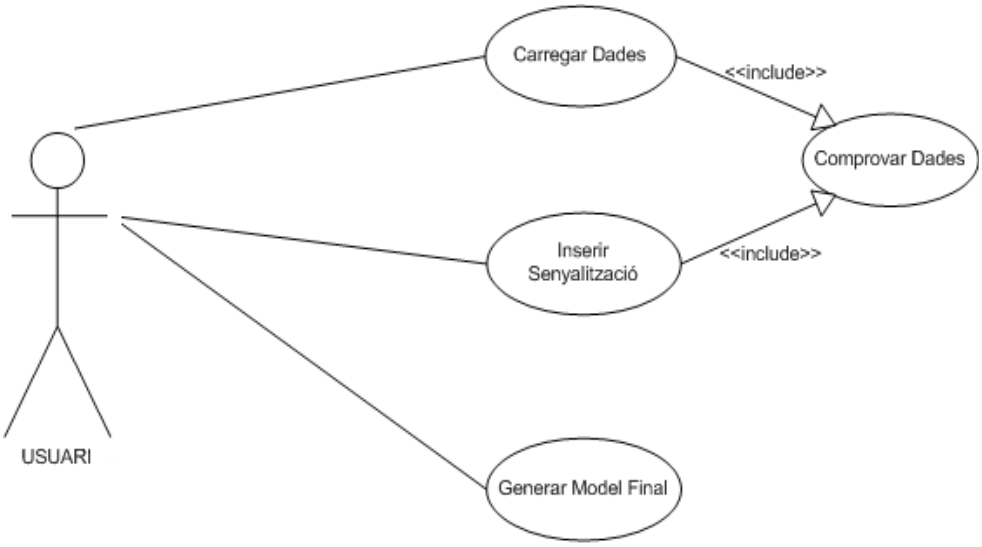
Al dissenyar l'aplicació, aplicarem els canvis que creiem convenients dins les dades de l'usuari, modificant les taules existents i afegint-ne de noves.

4.2.- Anàlisi funcional de l'aplicació

Un cop analitzades les dades existents, passem a l'anàlisi funcional de l'aplicació. Aquest apartat és molt important, ja que les decisions que es prenguin en aquest moment, afectaran a tot el desenvolupament del projecte. Una decisió mal presa o errònia pot hipotecar el desenvolupament de l'aplicació. Per tant, es tindrà especial cura en aquest apartat a l'hora de prendre les decisions.

4.2.1.- Casos d'ús

L'objectiu dels casos d'ús és documentar el comportament del sistema des del punt de vista de l'usuari. Cada cas d'ús proporciona un o diversos escenaris que indiquen cóm hauria d'interactuar el sistema amb els seus diferents usuaris, és a dir, ens mostra les relacions entre els usuaris i el sistema. Segons l'aplicació que volem dissenyar a partir dels requisits demanats, generem el següent diagrama:



Il·lustració 12: Diagrama de casos d'ús

Procedim a l'anàlisi del diagrama obtingut:

- **Actors:** Representa a una entitat externa al sistema que interactua amb ell. En la nostra aplicació tenim un sol actor, que serà l'usuari de l'aplicació.

- **Casos d'ús:** Representen les funcionalitats que ha de suportar el sistema. Hi han 4 casos d'ús:

    - **Carregar Dades:** La primera acció que ha de fer l'usuari és carregar les dades que ens proporcionen els ajuntaments. L'usuari defineix les dades i els valors per defecte que haurà de tenir el model final obtingut.

    - **Inserir senyalització:** Abans de crear el model final, l'usuari seleccionarà entre els diferents tipus de senyalització, qui vol que surti representada en el model final.

    - **Comprovar dades:** Els dos primers casos d'ús tenen una dependència o relació d'ús. Les relacions assenyalades amb <<include>>, denoten la inclusió del comportament d'un escenari a un altre. En aquest cas, el sistema comprovarà possibles errors dins les dades tant al generar els polígons del trams com al inserir la senyalització. En cas de dades errònies o defectuoses, l'aplicació llançarà un avis i aturarà l'aplicació.

    - **Generar model final:** Si tots els casos anteriors són correctes, l'usuari generarà el model que complirà les seves necessitats i el guardarà per tal de poder-lo recuperar quan vulgui.

- **Relacions:** Es tracta de les connexions entre elements. Hi haurà relacions d'associació entre l'usuari i tres dels casos d'ús (Carregar Dades, Inserir Senyalització i Generar Model Final). Les associacions indiquen que hi ha una comunicació entre l'actor i el cas d'ús mitjançant l'enviament i la recepció de missatges. La relació de dependència explicada en el tercer cas d'ús, seria l'altre tipus de relació que podem trobar en l'aplicació.

4.2.2.- Model de dades resultant

Al generar una representació gràfica, tota la informació necessària estarà representada mitjançant imatges *raster*. Per tant, el model de dades resultants serà molt simple, ja que només guardarem les dades imprescindibles. Per exemple, no caldrà, un camp Velocitat, si aquesta ja ve identificada gràficament. Per tant, el model de dades resultant és el següent:

TRAMS_2D		
ID(CLAU)	AutoNumber	Identificador del tram
IDTRAM	Integer	Identificador del tram inicial
NOM	Text	Nom del Carrer
DESCR	Text	Descripció
ESCALA	Text	Relació entre longitud i amplada del carrer
GEOMETRY	Geometry	Camp de geometria que representa el polígon
GEOMETRY_SK	Geometry	Cap propi de GeoMedia

La taula TRAMS_2D contindrà les dades de la representació en dos dimensions del tram. En principi es tractarà d'una representació poligonal del carrer, multiplicant l'amplada per l'escala per tal de facilitar la visualització. S'ha tingut en compte també l'opció de representar el tram per dues línies paral·leles en comptes d'un polígon, però la decisió es prendrà durant el muntatge de l'aplicació. Les úniques dades que tindrem disponibles, apart del IDTRAM i l' ESCALA és el NOM i la descripció (DESCR) del carrer.

SENYALS		
ID(CLAU)	AutoNumber	Identificador del senyal
IDTRAM	Integer	Identificador del tram
TIPUS	Text	Tipus del senyal
ESCALA	Integer	Relació entre longitud i amplada del carrer
GEOMETRY	Geometry	Camp de geometria que representa el raster
GEOMETRY_SK	Geometry	Cap propi de GeoMedia

La taula SENYALS contindrà les dades de tots els senyals inserits en la representació. El senyal serà una imatge *raster* col·locada dins el tram. Hi hauran diferents tipus de senyals (Velocitat, direcció, carril bici, etc...) i cada tipus tindrà una representació gràfica diferent.

Restava pendent de prendre dues decisions per al moment del muntatge de l'aplicació. Una de les decisions feia referència al la representació de les línies de separació dels carrils. Al final s'ha decidit crear dues taules de línies, LINIACONTINUA i LINIADISCONTINUA. Aquests dos objectes representaran les línies i es configurarà cada entitat en el GeoWorkspace. La configuració de les línies serà la següent:

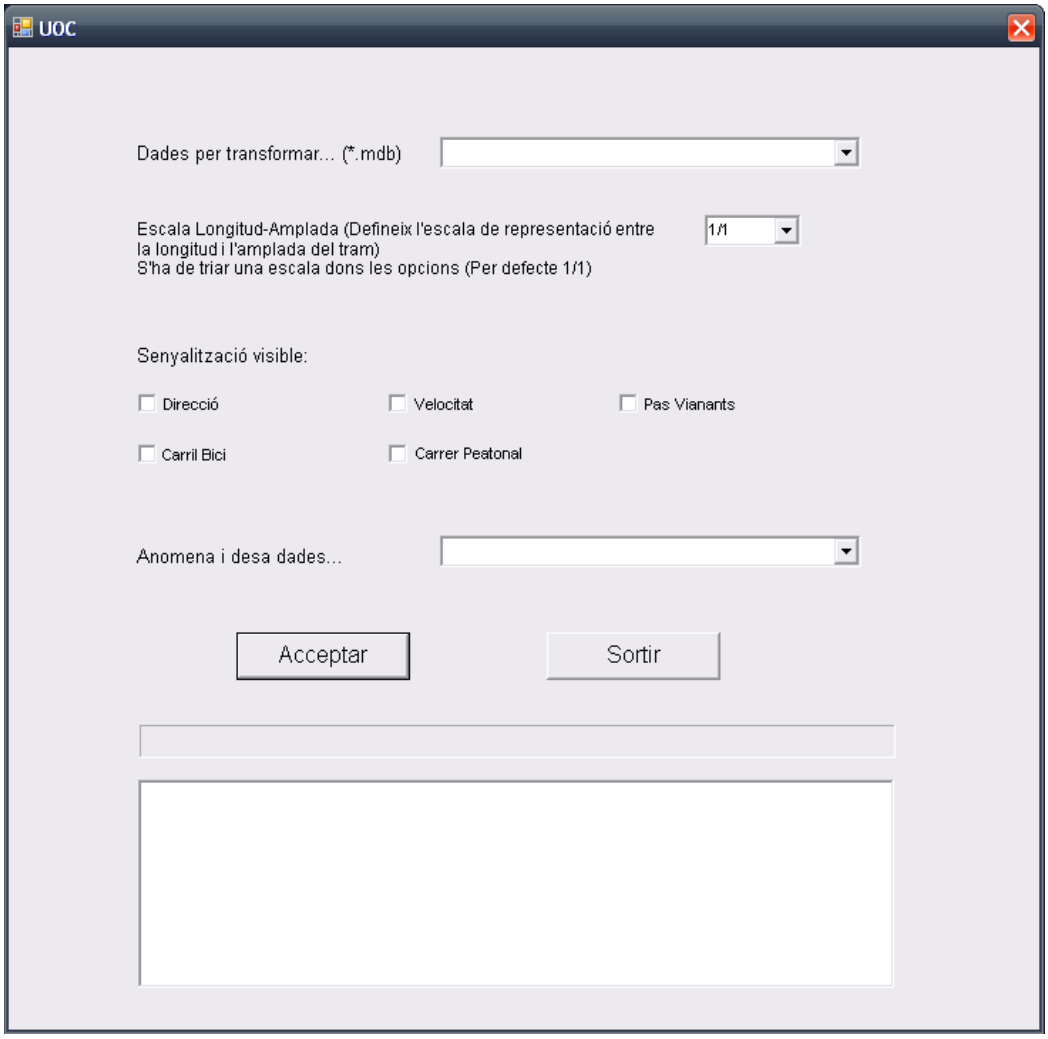
LINIACONTINUA		
ID(CLAU)	AutoNumber	Identificador del senyal
IDTRAM	Integer	Identificador del tram
CARRIL	Integer	Nº de carril dins el tram
GEOMETRY	Geometry	Camp de geometria que representa una línia
GEOMETRY_SK	Geometry	Cap propi de GeoMedia

LINIADISCONTINUA		
ID(CLAU)	AutoNumber	Identificador del senyal
IDTRAM	Integer	Identificador del tram
CARRIL	Integer	Nº de carril dins el tram
GEOMETRY	Geometry	Camp de geometria que representa una línia
GEOMETRY_SK	Geometry	Cap propi de GeoMedia

En quan a les cruïlles, aquestes haurien de quedar representades amb el model dissenyat, per tant es prescindeix de la seva representació.

4.2.3.- Interfície d'usuari

En el disseny de la interfície d'usuari s'ha cercat la senzillesa d'ús per sobre de qualsevol altre opció. La interfície dissenyada és la següent:



Il·lustració 13: Interfície d'usuari

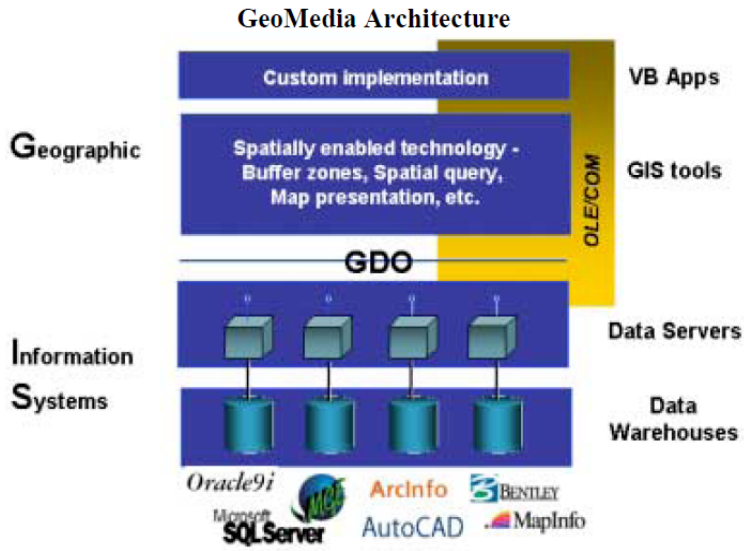
En primer lloc, es carrega la base de dades que volem transformar, sempre en format MS Acces (*.mdb). Un cop seleccionada, tenim diverses opcions per a la representació. En primer lloc, podem elegir els tipus de senyalització que volem veure representada (Velocitat, direccions, pas de vianants, etc...). Marcant els *check,* el tipus de senyal sortirà representat quan visualitzem les dades en GeoMedia. En segon lloc, escollim l'escala de la representació. Aquesta escala marca la proporció entre la longitud del tram i l'amplada d'aquest. Com major sigui l'escala, més ample serà la representació del carrer, i amb menys zoom podem veure millor la imatge. Es perdrà realitat, però es guanyarà en una millor representació gràfica. En darrer lloc, anomenem la nova base de dades per tal de poder-la recuperar quan volem i cliquem l'OK. Una barra de procés ens indicarà la situació de l'execució i un avis ens dirà quan s'ha acabat. Llavors podrem crear un nou GeoWorkspace des de Geomedia i carregar la nova base de dades per visualitzar-la. S'ha disposat un ListBox on es sistema anotarà el progrés de l'aplicació, així com una barra de progrés per controlar-ho.

4.3.- Disseny de l'aplicació

En aquest apartat ens centrem en el disseny de l'aplicació. Un cop preses les decisions en l'apartat anterior, aquestes s'han de plasmar en el disseny de cada part de l'aplicació. Un cop definit el model resultant i la interfície d'usuari, ara queda completar el disseny per poder muntar l'aplicació.

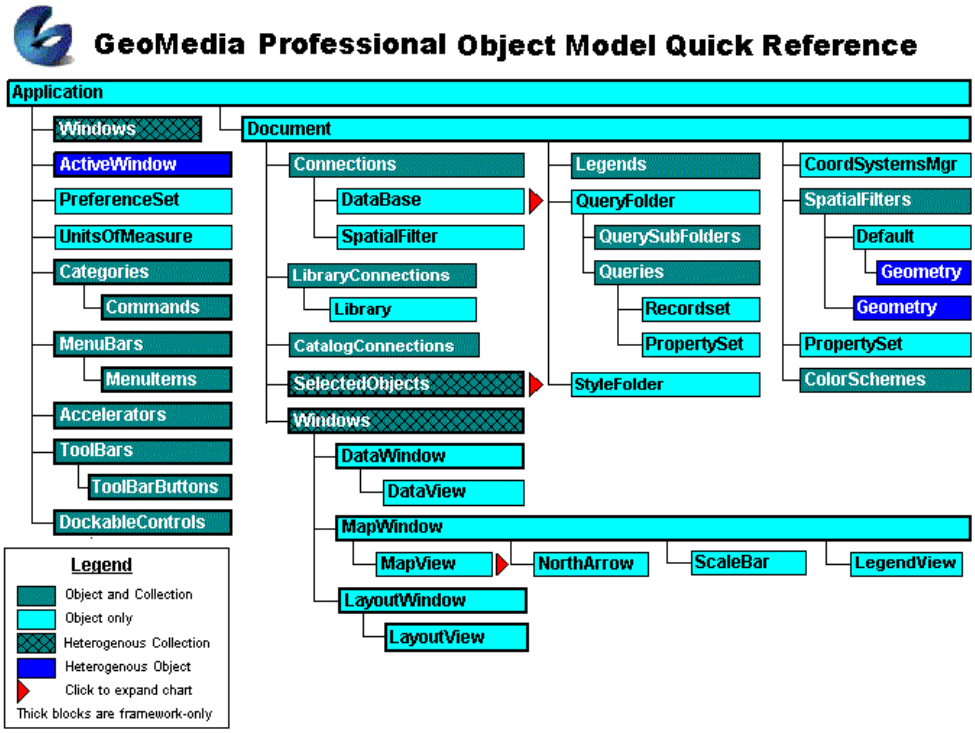
4.3.1.- Model d'objectes

Quan parlem de SIG (o GIS), estem parlant en tot moment de dades geogràfiques, totalment independents al software que usem per crear-les i pel seu manteniment. És per això que Integraph ofereix a disposició dels usuaris uns *Dataservers* a diferents formats de dades, i a disposició dels programadors un conjunt d'objectes que permeten accedir a aquestes dades i d'altres per la seva edició. Aquestes solucions permeten al programador treballar amb diferents sistemes de gestió de bases de dades (MS SQL, MS Access, Oracle, etc...), software CAD (AutoCAD), així com altres software GIS (MapInfo, ArcInfo, etc...). Per desenvolupar aquestes eines, es poden utilitzar llenguatges de programació com el MS Visual Basic, MS .NET o C#. Les solucions abans esmentades s'agrupen en el GDO (Geographic Data Object).



Il·lustració 14: The Geomedia Architecture (Font GeoMedia)

Dins del GDO disposem d'una sèrie de llibreries que van des d'eines per a la creació i manipulació de menús, barres d'eines i llegendes dins de GeoMedia, passant per eines d'enllaç amb els diversos sistemes de gestió de bases de dades, com per eines per la manipulació de les diferents formes de representació de dades dins de GeoMedia, tals com punts, línies, imatges *raster*, etc...

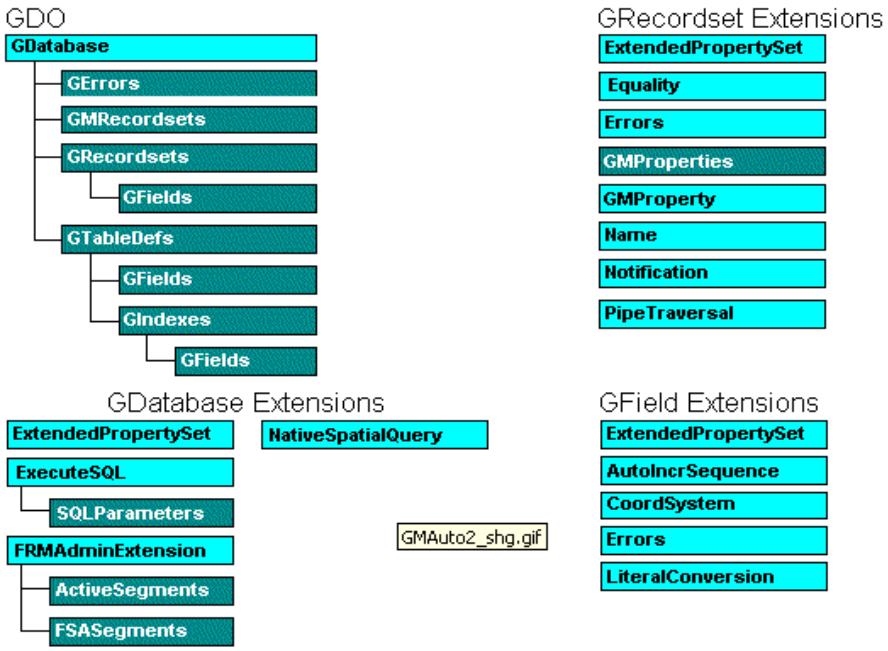


Il·lustració 15: Geomedia Professional Object Model (Font GeoMedia)

En el nostre cas, ens centrarem en les dues llibreries utilitzades per al desenvolupament de l'aplicació objecte del nostre treball, la llibreria *GDO* per a la manipulació de les bases de dades i la col·lecció *Geometry Collection*, per a la manipulació dels objectes geomètrics.

4.3.1.1.- GDO

La llibreria GDO (Geographic Data Object), està composta per una sèrie d'objectes per a la programació de bases de dades basades en les llibreries DAO (Microsoft Data Access Object) i COM (Microsoft Component Object Model). GeoMedia combina conceptes de DAO i COM amb els aspectes geogràfics dels sistemes de coordenades geodèsics, geometria i filtres espacials.



Il·lustració 16: GDO (Font Geomedia)

Per el desenvolupament del nostre projecte, s'utilitzen diferents objectes de la llibreria GDO:

- **GDataBase**: Es tracta d'un objecte que permet accedir a qualsevol dels SGDB que utilitzem, MS Access en el nostre cas. Amb els seus mètodes podem crear BBDD (***CreateDataBase()***), obrir (***OpenDataBase()***) i tancar (***Close()***) les BBDD amb les que treballem, així com crear les definicions de taules (***CreateTableDef()***), obrir registres per a la manipulació de les files i columnes (***OpenRecorSet()***) i altres mètodes per a la gestió d'una BBDD (***CommitTrans()*** i ***RollBack()***). També disposem de diferents propietats tals com ***Gdatabase.Name()***, ***GDataBase.Connect()***, etc...

- **GTableDef**: Es tracta d'un objecte on es desen les definicions de la taules de la BBDD. Permet crear columnes en les taules (***CreateField()***) i índexs (***CreateIndex()***). Disposem de diverses propietats per extreure'n les seves característiques.

- **GField/GFields**: Representa una columna/es dins una taula, així com les seves propietats. Farem servir la propietat ***GFields(FieldName).Value*** per extreure i

editar les dades de les columnes de la BBDD.

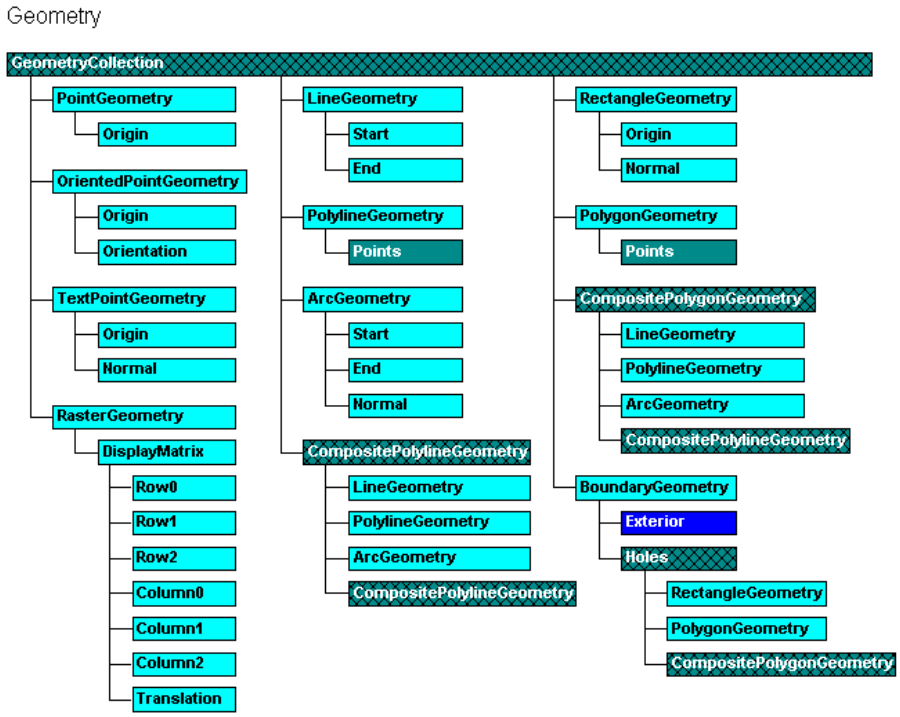
- **GRecorSet/GRecordSets**: Contenen els registres de la BBDD. Tots els objectes de ***GRecordSet*** es construeixen amb una fila i una columna, que localitzarà el registre dins la taula. Tenim mètodes per afegir un registre (***AddNew()***), editar (***Edit()***), actualitzar (***Update()***) o esborrar (***Delete()***) els registres, així com altres mètodes per moure el punter dins la columna endavant (***MoveNext()***), enrere (***MovePrevious()***), al principi (***MoveFirst()***) o al final (***MoveLast()***). També disposem de diverses propietats per extreure o editar les característiques del registre.

- **GeometryStorageService**: Aquest objecte converteix unes dades geomètriques en objectes geomètrics i en un array geomètric de bytes. Aquesta conversió es fa directament sobre un registre ***GRecorSet.GField*** del tipus ***gbdSpatial*** en una BBDD GDO ***GDataBase***.

- **GError/GErrors**: Conté els errors d'accés a la BBDD. Detecta els errors i en dóna la descripció i ajuda per solucionar-los.

4.3.1.2.- Geometry Collection

Es tracta d'una col·lecció on tenim els objectes, mètodes i propietats referents al objectes geomètrics que utilitzem per crear les representacions geogràfiques amb GeoMedia.



Il·lustració 17: Geometry Collection (Font GeoMedia)

Per al desenvolupament de la nostra aplicació treballarem amb diversos objectes d'aquesta col·lecció. Amb aquests objectes construirem la representació dels carrers:

- **Point**: Està format per les tres coordenades X, Y i Z, encara que nosaltres només utilitzem X i Y.

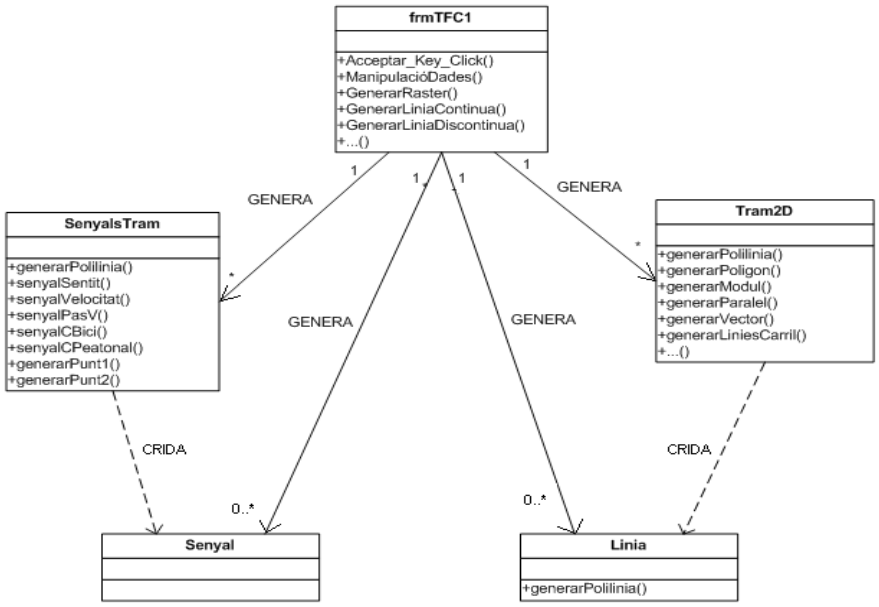
- **PolyLineGeometry**: Es tracta d'una serie de punts units per línies, formant una sola línia no tancada. Està formada per ***Points***, i tenim mètodes per poder afegir o esborrar punts, així com extreure el nombre de punts o un punt en concret.

- **PolygonGeometry**: Al igual que la ***PolylineGeometry***, es tracta d'una serie de punts units per una línia, formant una línia, però tancada. Tenim els mateixos mètodes i propietats que en la ***PolylineGeometry***.

- **RasterGeometry**: Es tracta d'una imatge georeferenciada adscrita a un polígon (***RasterPolygon***), on tenim un matriu (***DisplayMatrix***) amb les dades de georeferenciació i el nom de l'arxiu que ens proporciona la imatge (***FileName***)

4.3.2.- Classes i funcions a desenvolupar

Per al correcte funcionament d'aquesta aplicació es treballarà amb cinc classes. Per una banda tenim la classe frmTFC1, que és la classe principal i contindrà, a més de tots els mètodes referents a la interfície visual de l'aplicació, els mètodes que dirigeixen el funcionament de l'aplicació. Treballarem, a més amb les dues classes que representen els ítems de sortida de l'aplicació de les taules Tram2D i Senyal. Per altra banda, s'ha implementat la classe SenyalsTram, que a partir de les dades recollides de les taules Carril i Tram, generarà un objectes Senyal per cada tipus de senyalització del tram. En darrer lloc, hi ha la classe Linia, que representarà les línies de senyalització de l'aplicació. Amb aquests apreciacions, obtenim el següent diagrama de classes:



Il·lustració 18: Diagrama de classes

Des de la classe frmTFC1 generem els objectes tots els objectes de les classes Tram2D, SenyalTram, Senyal i Linia. Al seu torn, però, des de SenyalTram donen l'odre de generar els objectes de la classe Senyal amb les dades aportades per la classe SenyalsTram, podent crear cap o diverses senyals depenent de les necessitats de l'usuari. El mateix passa amb la classe Tram2D i Linia. Des de Tram2D s'aporten totes les dades a Linia, fent la crida de la generació de l'objecte a la classe frmTFC1.

4.3.2.1.- Classe frmTFC1

Aquesta classe és la classe principal de l'aplicació i la que, a més de controlar la interfície gràfica, conté els mètodes que generen l'intercanvi i manipulació de dades entre les dues bases de dades d'entrada i sortida, i genera tots els objectes que es desen a la base de dades de sortida. La classe té el següent disseny i mètodes:

frmTFC1
chbSentit_Checked_Changed() chbVelocitat_Checked_Changed() chbPasV_Checked_Changed() chbCBici_Checked_Changed() chbCPeatonal_Checked_Changed() cbEscala() Acceptar_Key_Click() Cancelar_Key_Click() ManipulacioDades() GenerarRaster() InserirSenyal() GenerarLiniaContinua() GenerarLiniaDiscontinua()

- Els *check box* ***chbSentit_Checked_Changed()***, ***chbVelocitat_Checked_Changed()***, ***chbPasV_Checked_Changed()***, ***chbCBici_Checked_Changed()*** i ***chbCPeatonal_Checked_Changed()*** controlen l'aparició o no dels diferents tipus de senyalització en la representació final que sortirà de l'aplicació. Si estan marcats (TRUE), els senyals corresponents sortiran representats. En cas contrari (FALSE), no sortirà en la representació.

- ***cbEscala()***: Aquest *combobox* ens indica l'escala longitud/amplada en la representació dels trams. En cas de no marcar res, tindrem una escala 1/1 per defecte.

- ***Acceptar_Key_Click()***: Aquest mètode s'activa al prémer el botó Acceptar. Això posa en marxa tota l'aplicació. Crea i carrega les base de dades d'entrada i sortida i crida al mètode *manipulacioDades()* per fer tot el procés.

- ***Cancelar_Key_Click()***: S'activa al clicar el botó Cancel·lar. Ens permet sortir de l'aplicació.

- ***ManipulacioDades()***: Aquest mètode extreu les dades de la base de dades d'entrada i després de manipular-les convenientment, carrega les dades resultants

en la BBDD de sortida. En primer lloc generarà dos objectes Tram2D i SenyalsTram. El Tram2D el carregarà en la base de dades de sortida. Des de l'objecte SenyalsTrams, cridarà el mètode *GenerarRaster()* per generar tota la senyalització i carregar-la a la base de dades de sortida.

- ***GenerarRaster()***: Aquest mètode pren un objecte SenyalsTram i genera tants objectes Senyal (senyalització dels carrers en format *raster*) com indiqui l'aplicació i els carrega a la base de dades de sortida.

- ***InserirSenyal():*** Aquest mètode rep un Senyal des de SenyalTram i l'insereix en el registre per carregar-lo a la base de dades de sortida.

- ***GenerarLiniaContinua():*** Amb aquest mètode, rebem un objecte Linia des de Tram2D i el carreguem en la base de dades de sortida. Un cop configurades seran les línies continues de la representació:

- ***GenerarLiniaDiscontinua():*** El funcionament d'aquest mètode és el mateix de l'anteriro, però es carregarà en una taula diferent, la que correspon a les línies discontinues.

4.3.2.2.- Classe Tram2D

Aquesta classe representa un tram visualitzat en 2D. Pren la polilínia de la taula *Tram* i la transforma en un polígon on l'amplada serà el total de l'amplada del tram per l'escala indicada per l'usuari. A més, generarà les línies contínues i discontínues que representen la senyalització entre carrils. Apart dels *getters* i *setters* i el constructor, tenim els següents mètodes:

TRAM2D
generarPolilinia() generarPoligon() generarVector() generarModul() generarParalel() generarLiniesCarril() ajustarExtremLinia()

- ***GenerarPolilinia()***: A partir d'una sèrie de punts donats des d'un objecte *GeometryStorageService*, genera una polilínia que representa el tram que hem de transformar posteriorment en un polígon.

- ***GenerarPoligon()***: A partir d'una polilínia i una amplada ja determinada, generem un polígon que representarà el tram indicat. Transformarem cada punt de la polilínia en dos punts paral·lels a 1/2 de l'amplada cadascun amb un vector a 90º i -90º al vector del segment dels punts als que s'ha de fer la línia paral·lela. Utilitzarem les funcions *generarModul()* i *generarParalel()* per fer els calculs.

- ***GenerarVector()***: Genera un vector entre dos punts i el desa en un altre punt.

- ***GenerarModul()***: Un cop obtingut el vector entre dos punts, utilitzant la formula del Teorema de Pitàgoras[17] ($a = \sqrt{b^2 + c^2}$) obtenim el mòdul del vector per poder calcular els punts paral·lels.

- ***GenerarParalel()***: A partir d'un vector i un mòdul, genera el paral·lel "superior" o "inferior" del punt segons la nostra demanda. Per a tal efecte utilitzarem el Teorema del Catet[18].

- ***GenerarLiniesCarril()***: Genera les línies separadores dels carrils. Pren les dues línies externes del carril dels polígon del tram i crea les línies intermitjes segons si has de ser continues (mateix sentit entre els dos carrils) o discontinua (sentits diferents entre els dos carrils). Fa una crida a a la classe *frmTFC1* per generar les polilínies.

- ***AjustarExtremsLinia():*** Al prendre les dades per crear les Linia des del polígon de Tram2D, les línies de senyalització es creuaven amb les d'altres carrils en les cruïlles. Aquesta funció, escurça les línies per tal que no es produeixi l'encreuament.

4.3.2.3.- Classe SenyalTram

La classe *SenyalsTram* funciona com una classe intermitja. Quan es genera un objecte d'aquesta classe, aquest conté tota la informació sobre els senyals en el tram. A partir d'aquest objecte, i segons la senyalització activada en la interfície d'usuari, generarem un imatge *raster* de cada tipus de senyal construint un nou objecte de la classe *Senyal* amb les dades de cada tipus de senyal en particular. Tenim els següents mètodes dins la classe, apart del constructor, els *getters* i *setters*:

SENYALSTRAM
longSentit() generarPolilinia() senyalSentit() senyalVelocitat() senyalPasV() senyalCBici() senyalCPeatonal() generarPunt1() generarPunt2()

- ***longSentit()***: Retorna la longitud de l'array on desem els sentits de circulació de cada carril de cada tram.

17.-http://es.wikipedia.org/wiki/Teorema_de_Pit%C3%A1goras
18.-http://es.wikipedia.org/wiki/Teorema_del_cateto

- ***generarPolilinia()***: A partir d'una sèrie de punts donats des d'un objecte *GeometryStorageService*, genera una polilínia que representa el tram que hem de transformar posteriorment en un polígon.

- ***senyalSentit()***, ***senyalVelocitat()***, ***senyalPasV()***, ***senyalCBici()*** i ***senyalCPeatonal()***: Genera un objecte Senyal que representarà el tipus de senyal indicat del carril, sempre i quan la opció estigui activada. Generem l'objecte indicant el directori de la imatge, el tipus de senyal, el carril on va el senyal i l'escala de representació, així com la posició on col·locarem el senyal.

- ***generarPunt1()***: Per posicionar una imatge *raster* en Geomedia, cales 2 punts. A partir de la *Linia* del carril, generarem el primer punt de posicionament de la imatge, que serà a partir de la línia del carril a una distància de l'extrem marcada per la variable *posicio*.

- ***generarPunt2():*** Genera el segon punt per posicionar la imatge del *raster*. Li donarem a la imatge una mida de 2 x 2 mts.

4.3.2.4.- Classe Senyal

La classe Senyal representa un tipus de senyal determinat col·locat en una posició determinada. Aquest senyal ve establert per la classe *SenyalsTram*. Cada senyal representada estarà formada per un únic objecte Senyal, de tal forma que de *SenyalsTram* en poden sortir diversos objectes de la classe *Senyal*.

SENYAL

- Aquesta classe no disposa de cap mètode, apart dels *getters* i *setters*. La part més important d'aquesta classe és, apart del nom del fitxer que conté la imatge, els dos punts de posicionament, *p1* i *p2*.

4.3.2.5.- Classe Linia

Aquesta classe representa les línies de separació dels carrils. A través de la classe *frmTFC1*, carregarà les línies en la taula *LINIACONTINUA* o *LINIADISCONTINUA* segons les condicions establertes en el mètode *generarLiniesCarril()* de la classe *Tram2D*. A banda dels *getters* i *setters*, aquesta classe té un sol mètode:

LINIA
generarPolilinia()

- ***generarPolilinia()***: A partir d'una sèrie de punts donats des d'un objecte *GeometryStorageService*, genera una polilínia que representa el tram que hem de transformar posteriorment en un polígon.

4.3.2.6.- Codi de la classe Tram2D

S'ha cregut convenient afegir a aquesta memòria una part del codi de programació, a banda del codi font que es subministra junt a aquesta. S'ha optat per afegir els mètodes de la classe Tram2D, donat que són els més importants pel funcionament de l'aplicació. Es pot trobar el codi en l'annexe 2.

4.4.- Construcció del sistema

Donat que junt a aquesta memòria es subministra el codi font de l'aplicació, en aquest apartat s'analitzarà el funcionament de l'aplicació, així com les possibles millores que es poden aplicar en un futur a l'aplicació.

4.4.1.- Funcionament de l'aplicació

Per el funcionament de l'aplicació s'analitzen les diverses fases dins del procés:

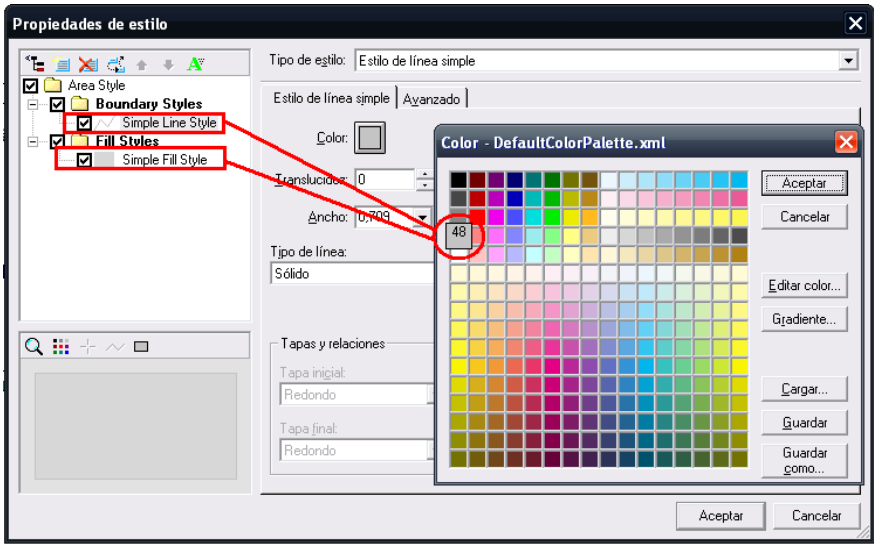
- **Base de dades d'entrada:** S'han subministrat dos exemples per fer les proves de l'aplicació. Aquestes dues bases de dades (*Exemples1.mdb* i *Exemples2.mdb*) venen amb un format predeterminat i que s'ha analitzat en l'apartat 4.1. S'ha configurat un tercera base de dades (*Exemples3.mdb*) amb només 8 carrers per tal de facilitar la càrrega de les entitats i reduir el temps d'execució, així com facilitar la detecció d'errors bàsics. S'han provat les tres bases de dades i s'ha detectat algunes incidències que es comenten posteriorment.

- **Base de dades de sortida:** Per facilitar el codi i reduir errors, s'ha optat per la configuració de la base de dades de sortida des de GeoMedia aprofitant l'opció que ofereix de crear una base de dades completa (amb metadades configurades incloses) en format MS Access. S'ha afegit les entitats segons el model resultant del apartat 4.2.2 i s'ha desat amb el nom de *outputDb.mdb*. En un principi es va voler aprofitar el sistema de coordenades dels exemples subministrats, però en la representació visual en GeoMedia, els polígons quedaven deformats, motiu pel qual s'ha fet servir el sistema de coordenades EPSG4326, subministrat pel consultor (Arxiu adjunt)

- Un cop iniciada l'aplicació, s'omplen les diferents opcions (bases de dades d'entrada i sortida, escala i senyalització) i s'executa. Podem veure el progrés en la barra i finalment, un cop acabat aquest, consultar les incidències en el ListBox.

- Un cop s'ha omplert la base de dades de sortida, iniciem el GeoMedia Professional. Seguidament, obrim un *GeoWorkspace* en blanc, obrim un magatzem amb *Almacen/Conexión nueva...* i cerquem la base de dades *outputDB.mdb* on l'hem desada.

Cliquem l'opció *Sacar clases de entidad...* i elegim totes les entitats. Aquestes surten carregades en pantalla. En aquest punt, el resultat encara no és el definitiu. Cal donar format a les entitats per millorar la visualització. Obrim les propietats de la entitat TRAM2D a la llegenda amb el botó dret i elegim l'opció *Propiedades de Estilo*. Ho configurem com en l'imatge adjunta:



Il·lustració 19: Configuració entitat TRAM2D (Captura GeoMedia)

Amb aquest format, el color dels Senyals *raster* coincidirà amb el dels TRAM2D, ja que s'ha establert d'aquesta manera anteriorment. Les línies les configurarem imitant les reals, configurant les LINIADISCONTINUA amb l'opció *Tipo de Linia: Trazo corto* a *Propiedades de Estilo* d'aquesta entitat. Configurem d'igual forma les LINIACONTINUA. En darrer lloc, configurarem les línies amb el color que creiem més convenient (blanc o negre) i tindrem el model de sortida final.

- Amb la visualització podem detectar certes incidències. La primera de totes, si s'ha marcat l'opció, és que els senyals de direcció no surten representats. El motiu és que en el disseny de l'aplicació es va decidir que les imatges *raster* es rotarien segons l'angle del tram abans de carregar-les a la base de dades de sortida. Aquesta opció no pot ser programada amb Visual Basic 2005 (s'ha consultat al consultor i en diverses webs de programació) (Si que es pot fer amb .NET). Mentre que la resta de senyals no es veuen afectades seriosament, els indicadors de direcció mal orientats donarien molt mala imatge, per lo tant, s'ha decidit anular l'opció. En el cas dels passos de vianants, ha sorgit el mateix problema. Aquest s'ha pogut solucionar inserint el senyal de pas de vianants en comptes de representar-lo gràficament sobre el tram. El sentit de circulació es pot deduir fàcilment, ja que els senyals de velocitat es posicionen al inici del carril.

- Un altre punt a tenir en compte, és el resultat de la decisió de prescindir de representar les cruïlles. Tot i que es reconeix que certs punts queden buits en les cruïlles, el resultat és força satisfactori en la majoria de la representació. L'opció d'incloure la representació de les cruïlles ho proposen com una oportunitat de futura millora.

- Un altre punt que ha donat molts problemes, sobretot en la senyalització de senyals i línies, és el fet que els trams no segueixen un esquema predefinit en la posició de les cruïlles. Tota l'aplicació s'ha dissenyat amb la pressuposició que les cruïlles CRUILLAA i

CRUILLADE dels TRAMS seguien un ordre d'esquerra a dreta i de dalt a baix. S'han detectat cassos en els que no és així, donant com a resultat una representació errònia del tram. Tot i que es pot arreglar, per facilitar la interpretació de les dades, es proposa que es tingui en compte a l'hora de generar les bases de dades d'entrada.

- S'ha detectat que GeoMedia no reconeix les coordenades de la representació. S'ha provat amb diversos sistemes de coordenades, s'han arreglat les metadades amb les utilitats de GeoMedia, però no s'ha trobat la solució. Aquest punt requerirà un posterior estudi per intentar solucionar el problema.

- En darrer lloc, l'aplicació falla al intentar llegir certs registres de les bases de dades d'exemple. Després de revisar les línies on falla l'aplicació (per exemple els trams 5163 i 5164 de *Exemples2.mdb*). S'ha detectat la presència de registres buits en la columna CARRILS_cc. Tot i que aquest registre no és necessari per l'aplicació i no es llegeix, és l'única explicació que s'ha trobat. Queda el dubte si es tracta d'un problema d'integritat de dades o per el contrari, és un problema de disseny de l'aplicació.

- No s'ha aconseguit implementar correctament el control d'errors GDO.GError i s'ha desat el codi sense el control d'errors en l'entrada de les bases de dades.

4.4.2.- Futures línies de millora

Tot i que l'aplicació funciona correctament, durant el disseny s'han detectat possibles oportunitats de millora. Aquestes millores donarien un salt de qualitat a l'aplicació, però no s'ha cregut convenient incloure-les en aquest projecte:

- **Base de dades de sortida:** En l'aplicació dissenyada, utilitzem com a base de dades de sortida un arxiu predefinit muntat des de GeoMedia. D'aquesta menara es minimitzen els errors. Un possible millora podria ser generar la base de dades de sortida des de l'aplicació, configurant les metadades des de aquesta. Així, milloraríem en flexibilitat a l'hora de generar la base de dades de sortida.

- **Format de les entitats:** No s'ha estudiat la possibilitat de donar el format de les entitats directament des de l'aplicació. En cas que GDO ho permeti, aquesta possibilitat estalviaria la "postproducció" de la representació des de GeoMedia.

- **Cruïlles:** Ja s'ha comentat anteriorment, que la decisió de prescindir de la representació de les cruïlles, de primeres semblava correcta, analitzant el resultat final, aquesta representació milloraria el resultat obtingut.

- **Explorador d'arxius en l'aplicació:** Per tal de centrar-nos en l'objectiu del projecte, certes decisions s'han descartat. Una d'elles és la possibilitat de obrir un explorador d'arxius des dels ComboBox on inserim els noms dels arxius d'entrada i sortida. Seria, doncs, una millora, afegir aquesta opció en l'aplicació.

- **Codi font:** Donat que és el primer cop que es treballa amb Visual Basic, hi ha la certesa que el codi entregat pot ser depurat i millorat.

4.4.3.- Conclusions

En primer lloc, donat la desconeixença total del món dels SIG i de GeoMedia, ha estat tot un repte desenvolupar aquest projecte des de zero. L'esforç de conèixer a fons tant aquestes matèries ha estat molt gran, però l'experiència ha estat totalment positiva. El resultat obtingut, tot i que és millorable, deixa la sensació d'haver aconseguit un resultat satisfactori.

No tant positiva, més aviat el contrari, és l'experiència de treballar amb Visual Basic. Després de treballar amb Java, queda la sensació que es tracta d'un llenguatge incomode, rígid i poc potent. Certes limitacions trobades a l'hora de muntar el codi, confirmen aquesta opinió (Personal).

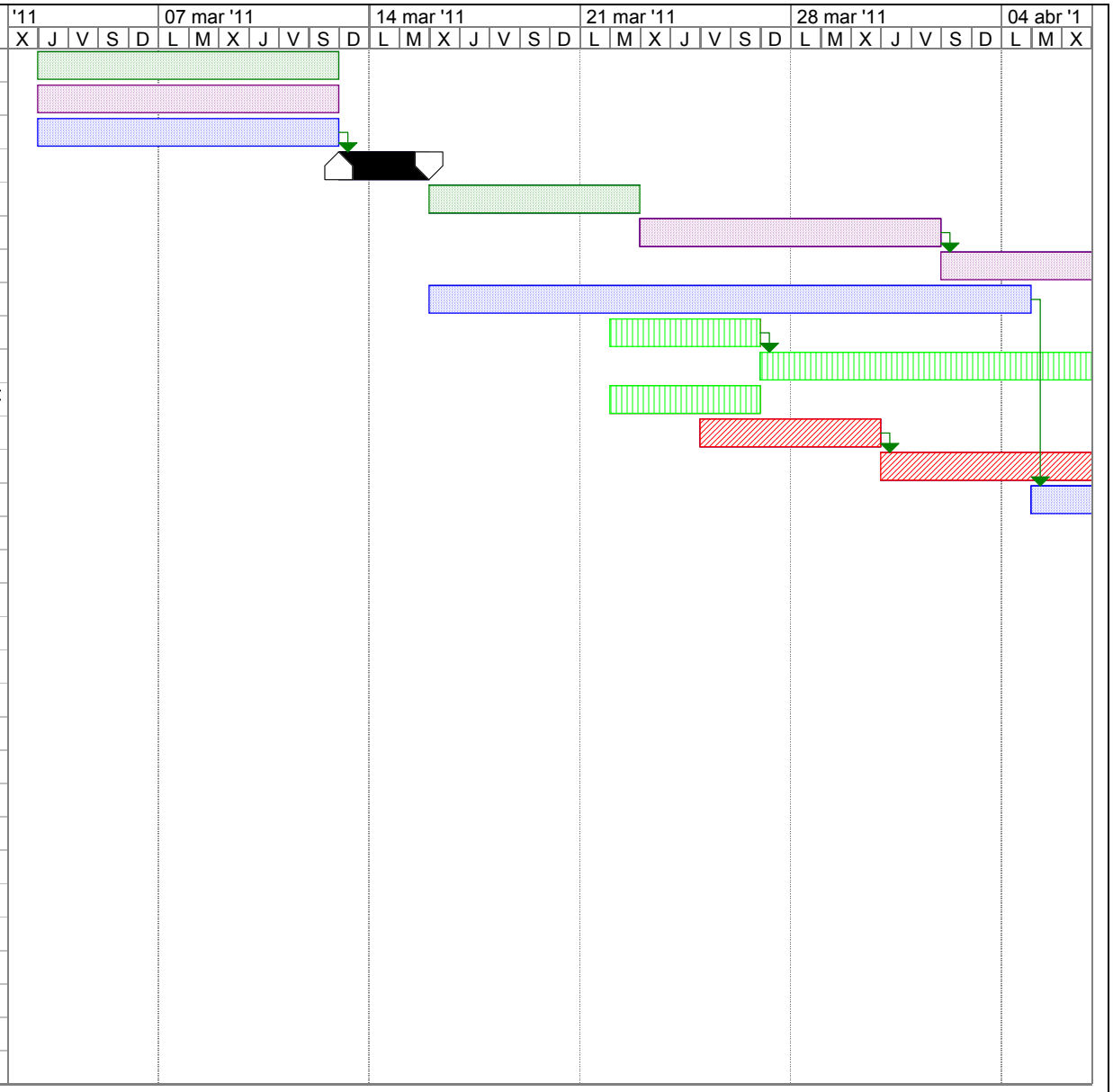
En darrer lloc, ressaltar la dificultat de combinar vida professional, vida familiar i vida d'estudiant. No sempre resulta senzill trobar temps tot i definir les prioritats.

Bibliografia

- Sistemes d'informació geogràfica i geotelemàtica (2009, UOC)
- Treball Final de Carrera (2008, UOC)
- Bases de dades I (2009, UOC)
- Bases de dades II (2009, UOC)
- Manual del Usuario de Geomedia Proffesional (1998-2004 Integraph Corporation)
- Manual del Usuario de Geomadia Viewer (1998-2004 Integraph Corporation)
- GeoMedia Professional Object Reference (2008 Integraph Corporation)
- The GeoMedia Architecture Advantage (2002 Integraph Corporation)
- Wiquipedia (Sistema_de_Informaci%C3%B3n_Geogr%C3%A1fica)
- GabrielOrtiz.com (http://www.gabrielortiz.com)
- MSDN Library (http://msdn.microsoft.com/es-es/library/ms123401.aspx)
- VB6.us (http://www.vb6.us/)
- Recursos Visual Basic (http://www.recursosvisualbasic.com.ar/)
- Aprenda Visual Basic 6.0 (Universidad de Navarra, 1999)
- Primeros pasos con Visual Basic 2005 Express (Carlos de Mora) (http://www.infosoftw.com/VS2005ManxHTML/)
- Paseo por Visual Basic 2008 (Everts Garay)
- Recursos Visual Basic (http://www.recursosvisualbasic.com.ar/)
- Wikipedia (http://es.wikipedia.org/wiki/Teorema_de_Pit%C3%A1goras)
- Wikipedia (http://es.wikipedia.org/wiki/Teorema_del_cateto)

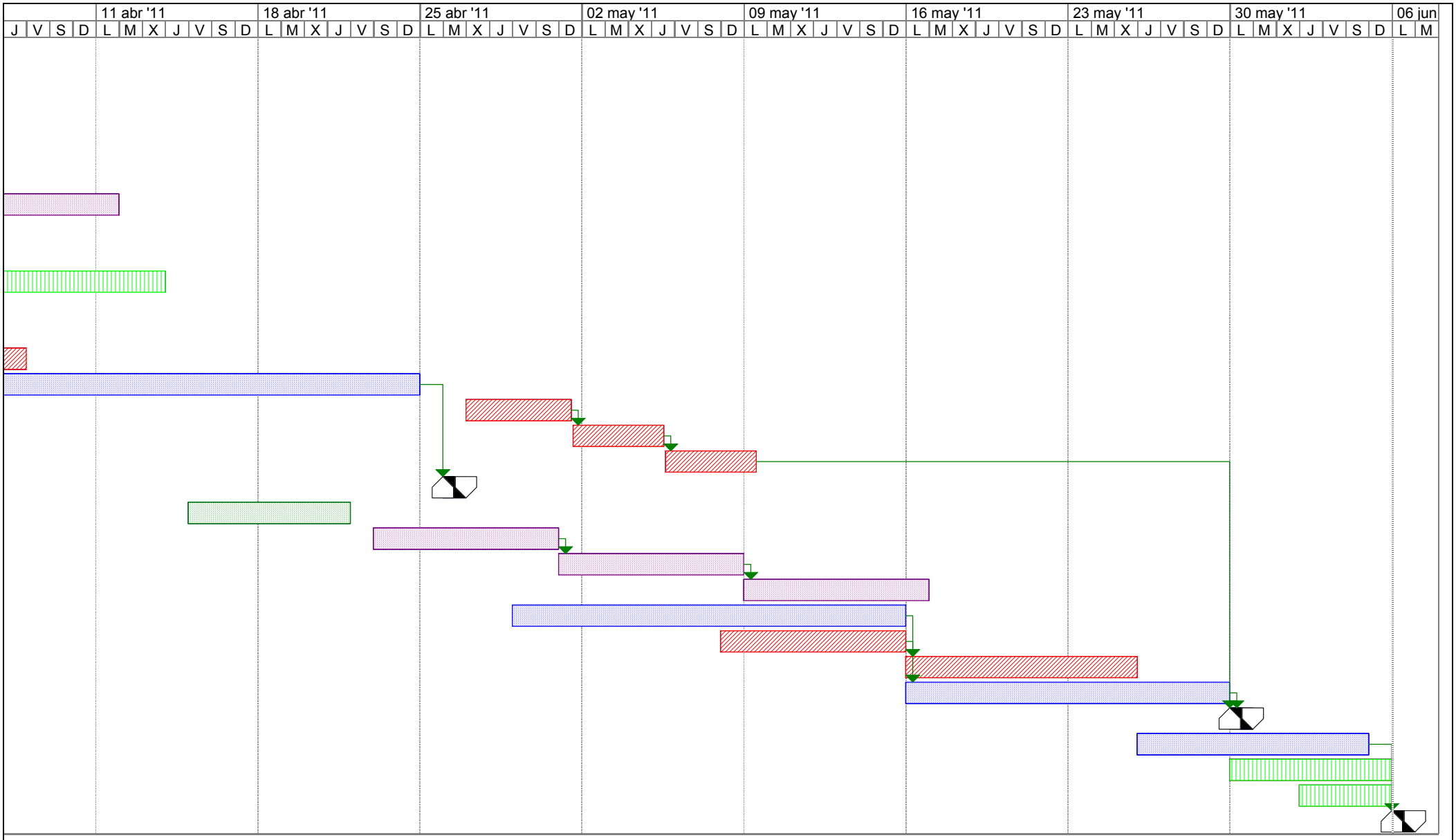
Annexe 1: Diagrama de Gantt

Id	Nombre de tarea
1	Mòdul 1 TFC
2	Mòdul 1 SIGG
3	Planificació del projecte
4	Lliurament Pla de Treball
5	Mòdul 2 TFC
6	Mòdul 2 SIGG
7	Mòdul 3 SIGG
8	Analisis de les estructures de dades existents
9	Instal·lació de programari (Geomedia)
10	Familiaritzar-se amb Geomedia
11	Decidir llenguatge programació (Visual Basic o .NET). Instal·lar editc
12	Tipologia de dades
13	Estructures i formats de dades de bases comercials
14	Anàlisi funcianal de l'aplicació
15	Casos d'us
16	Model de dades resultant
17	Interficies d'usuari
18	Lliurament Tasca 1
19	Mòdul 3 TFC
20	Mòdul 4 SIGG
21	Mòdul 5 SIGG
22	Mòdul 6 SIGG
23	Disseny de l'aplicació
24	Model d'objectes
25	Classes i funcions a desenvolupar
26	Construcció del sistema
27	Lliurament Tasca 2 i 3
28	Preparació Lliurament Final
29	Redactat memòria
30	Muntatge video i audio
31	Lliurament Final



Proyecto: Projecte FC
Fecha: lun 06/06/11

Mòdul TFC		Tasca		Altres tasques	
Mòdul SIGG		Subtasques		Lliurament	



Proyecto: Projecte FC
Fecha: lun 06/06/11

Mòdul TFC		Tasca		Altres tasques			
Mòdul SIGG		Subtasques		Lliurament			

```
'*****************************************************************************
'Classe Tram2D: genera un polígon que serà una representació en 2D del tram.
'Un cop s'hagi creat el polígon, omplirà els camps de la taula TRAMS2D de la
' BBDD de sortida outputDB
'*****************************************************************************

Imports PBasic = Intergraph.GeoMedia.PBasic

Public Class Tram2D

    'Declaració de variables de l'objecte
    Private idTram As Integer, escala As Integer, nCarrils As Integer
    Private nom As String, descr As String
    Private ampladaTram As Double
    Dim polilinia As New PolylineGeometry
    Dim poligon As New PolygonGeometry
    Dim punt As New PBasic.point
    Dim carrils() As Double
    Dim sentit() As Integer
    Dim liniesTram() As PolylineGeometry
    Dim liniaC As Linia
    Dim pLinia As New PolylineGeometry

    'Declaració de variables diverses
    Dim n As Integer, n2 As Integer, i As Integer, j As Integer, k As Integer, m As Integer
    Dim w As Integer

    '-----------------------------------------------------------------------------
    ' Constructor Tram2D
    '-----------------------------------------------------------------------------
    Public Sub New(ByVal nIdTram As Integer, ByVal nNom As String, _
    ByVal nDescr As String, ByVal nCarrils As Integer, ByVal nEscala As Integer)

        tIdTram = nIdTram
        tNom = nNom
        tDescr = nDescr
        tNCarrils = nCarrils
        tEscala = nEscala
        tAmpladaTram = 0

    End Sub

    '-----------------------------------------------------------------------------
    ' Getters i Setters
    '-----------------------------------------------------------------------------
    Public Property tIdTram() As Integer
        Get
            Return idTram
        End Get
        Set(ByVal tram As Integer)
            idTram = tram
        End Set
    End Property

    Public Property tNom() As String
        Get
            Return nom
        End Get
        Set(ByVal name As String)
            nom = name
        End Set
    End Property

    Public Property tDescr() As String
        Get
            Return descr
        End Get
        Set(ByVal description As String)
            descr = description
        End Set
    End Property

    Public Property tNCarrils() As Integer
        Get
```

```
            Return nCarrils
        End Get
        Set(ByVal nCarr As Integer)
            nCarrils = nCarr
        End Set
    End Property

    Public Property tEscala() As Integer
        Get
            Return escala
        End Get
        Set(ByVal esca As Integer)
            escala = esca
        End Set
    End Property

    Public Property tAmpladaTram() As Double
        Get
            Return ampladaTram
        End Get
        Set(ByVal aTram As Double)
            ampladaTram = aTram
        End Set
    End Property

    Public Property tPoligon() As PolygonGeometry
        Get
            Return poligon
        End Get
        Set(ByVal plg As PolygonGeometry)
            poligon = plg
        End Set
    End Property

    '-----------------------------------------------------------------------------------------↙
    -----
    'Procediment setCarril: substitueix al setter habitual ja que getter i setter funcionen↙

    'diferent.
    '-----------------------------------------------------------------------------------------↙
    -----
    Sub setCarril(ByVal ampleCarril As Double, ByVal k As Integer)

        Try
            ReDim carrils(k)
            carrils(k) = ampleCarril
            tAmpladaTram() = tAmpladaTram() + ampleCarril
        Catch ex As NullReferenceException
            MsgBox("NullreferenceException. SenyalsTram.setSentit()")
            MsgBox(ex.Message)
        End Try

    End Sub

    '-----------------------------------------------------------------------------------------↙
    -----
    'Procediment getSentit: substitueix al getter habitual ja que getter i setter funcionen↙

    'diferent.
    '-----------------------------------------------------------------------------------------↙
    -----
    Function getCarril(ByVal k As Integer) As Integer

        Dim ret As Integer

        ret = carrils(k)

        Return ret

    End Function

    '-----------------------------------------------------------------------------------------↙
    -----
    'Procediment setSentit: substitueix al setter habitual ja que getter i setter funcionen↙
```

```
    'diferent.
    '-----------------------------------------------------------------------------------------
    -----
    Sub setSentit(ByVal sent As Integer, ByVal k As Integer)

        Try
            ReDim sentit(k)
            sentit(k) = sent
        Catch ex As NullReferenceException
            MsgBox("NullreferenceException. SenyalsTram.setSentit()")
            MsgBox(ex.Message)
        End Try

    End Sub

    '-----------------------------------------------------------------------------------------
    -----
    'Procediment getSentit: substitueix al getter habitual ja que getter i setter funcionen

    'diferent.
    '-----------------------------------------------------------------------------------------
    -----
    Function getSentit(ByVal k As Integer) As Integer

        Dim ret As Integer
        ret = sentit(k)

        Return ret

    End Function

    '-----------------------------------------------------------------------------------------
    -----
    'Procediment getLiniesTram: substitueix al getter habitual
    '-----------------------------------------------------------------------------------------
    -----
    Function getLiniesTram(ByVal k As Integer) As PolylineGeometry

        Dim ret As PolylineGeometry
        ret = liniesTram(k)

        Return ret

    End Function

    '-----------------------------------------------------------------------------------------
    -------
    'Funció longLiniesTram(): Retorna la mida de l'array on desem les linies de cada tram
    '-----------------------------------------------------------------------------------------
    -------
    Function longLiniesTram() As Integer

        Dim l As Integer

        Try
            l = UBound(liniesTram)
        Catch ex As NullReferenceException
            MsgBox("NullreferenceException")
        End Try

        Return l

    End Function

    '-----------------------------------------------------------------------------------------
    -----
    'Procediment generarPolilinia(): Genera una polilinia a partir dels punts extrets.
    Aquesta
    'polilinia equival als Trams. Sobre aquesta polilinia farem els calculs per crear el
    poligon.
    '-----------------------------------------------------------------------------------------
    -----
    Sub generarPolilinia(ByVal x As Double, ByVal y As Double)
```

```
        punt.X = x
        punt.Y = y
        polilinia.Points.Add(punt)

    End Sub

    '-------------------------------------------------------------------------------------
    ------
    ' Procediment generarPoligon(): Genera un polígon a partir d'una polilínea, amb una
    amplada
    ' totals del polígon que serà l'amplada dels trams per l'escala elegida.
    '-------------------------------------------------------------------------------------
    ------
    Sub generarPoligon()

        Dim pt1 As New PBasic.point, pt2 As New PBasic.point, pt3 As New PBasic.point
        Dim vt As New PBasic.point, vect As New PBasic.point, pt As New PBasic.point
        Dim modul As Double, amp As Double

        pt1 = Nothing
        pt2 = Nothing
        pt3 = Nothing

        'Si la polilinia representa la part central del carril llavors la meitat de l'ample
     serà
        'l'amplada a afegir per cada costat
        amp = (tAmpladaTram() * escala) / 2

        n = polilinia.Points.Count
        Try ' Controlem les excepcions

            '-------------------------------------------------------------------
            ' Passem d'una polilínia a un polígon                              |
            '                                      p1------p2---...------pn      |
            '   p1---p2--...----pn    ----->        |                     |      |
            '                                      pn2----pn2-1-...-----pn+1     |
            '-------------------------------------------------------------------

            n2 = n * 2
            k = 1
            i = 1
            '-------------------------------------------------------------------------------
    -----
            'Calculem el punt "superior" del primer extrem (p1).
            'Tenim un vector entre els punts p1 i p2, i calculem un vector a 90º d'aquest
    vector
            'per obtenir el punt
            pt1 = polilinia.Points.Item(1)
            pt2 = polilinia.Points.Item(2)

            vect = generarVector(pt1, pt2)
            modul = generarModul(vect, amp)

            ' Inserim p1
            poligon.Points.Add(generarParalel(modul, vect, pt1, 1))
            i = i + 1
            '-----------------------------------------------------------------------------
            'Calculem els punts intermitjos "superiors" (de p2 fins pn-1).
            'A partir de 3 punts, obtenim el vector dels punts pi-1 i pi+1 per obtenir el
            'vector de pi, i calculem el punt amb un vector a 90º sobre el vector obtingut.
            '(En cas de polígons de 4 punts, aquest pas es salta)
            While (i < n)
                pt1 = polilinia.Points.Item(i - 1)
                pt2 = polilinia.Points.Item(i)
                pt3 = polilinia.Points.Item(i + 1)

                vect = generarVector(pt1, pt3)
                modul = generarModul(vect, amp)

                'Insertem de p2 fins pn-1
                poligon.Points.Add(generarParalel(modul, vect, pt2, 1))
                i = i + 1
```

```
            End While
            '-----------------------------------------------------------------------
            'Calculem els punts del segon extrem (pn i pn+1).
            'Sobre el vector de pn-1 i pn de la polilinia, calculem el vector a 90º
            'per obtenir el punt pn i el vector a -90º per obtenir el punt pn+1
            pt1 = polilinia.Points.Item(n - 1)
            pt2 = polilinia.Points.Item(n)

            vect = generarVector(pt1, pt2)
            modul = generarModul(vect, amp)

            'Insertem pn
            poligon.Points.Add(generarParalel(modul, vect, pt2, 1))

            'Insertem pn+1
            poligon.Points.Add(generarParalel(modul, vect, pt2, 2))
            i = i - 1
            '-------------------------------------------------------------------------------
-
            'Part "inferior" del polígon de pn+2 fins pn2-1
            'Calculem els punts intermitjos "inferiors" (de pn+2 fins pn2-1)
            'A partir de 3 punts, obtenim el vector dels punts pi-1 i pi+1 per obtenir el
            'vector de pi, i calculem el punt amb un vector a -90º sobre el vector obtingut
.
            '(En cas de polígons de 4 punts, aquest punt es salta)
            While (i > 1) 'Ja s'ha fet de p1 fins pn
                pt1 = polilinia.Points.Item(i - 1)
                pt2 = polilinia.Points.Item(i)
                pt3 = polilinia.Points.Item(i + 1)

                vect = generarVector(pt1, pt3)
                modul = generarModul(vect, amp)

                'Insertem de pn+2 fins a pn2-1
                poligon.Points.Add(generarParalel(modul, vect, pt2, 2))
                i = i - 1
            End While

            '-------------------------------------------------------------------------------
------
            'Calculem el punt "inferior" del primer extrem (pn2)
            'Tenim un vector entre els punts p1 i p2, i calculem un vector a -90º d'aquest
vector
            'per obternir el punt
            pt1 = polilinia.Points.Item(1)
            pt2 = polilinia.Points.Item(2)

            vect = generarVector(pt1, pt2)
            modul = generarModul(vect, amp)

            'Insertem pn2
            poligon.Points.Add(generarParalel(modul, vect, pt1, 2))

            'Capturem les excepcions
        Catch ex As NullReferenceException
            MsgBox("NullReferenceException: Tram2D.generarPoligon()")
            MsgBox(ex.Message)
        Catch ex As ArgumentException
            MsgBox("ArgumentException: Tram2D.generarPoligon()")
            MsgBox(ex.Message)
        Catch ex As AccessViolationException
            MsgBox("AccessViolationException: Tram2D.generarPoligon()")
            MsgBox(ex.Message)
        End Try

    End Sub

    '-----------------------------------------------------------------------------------
    ------
    ' Funció generarVector(): Genera un vector entre dos punts i el desem en un altre punt.

    '-----------------------------------------------------------------------------------
    ------
    Function generarVector(ByVal pt1 As PBasic.point, ByVal pt2 As PBasic.point) As PBasic.
```

```
    point

        Dim vt As New PBasic.point

        vt.X = pt1.X - pt2.X
        vt.Y = pt1.Y - pt2.Y

        Return vt

    End Function

    '-----------------------------------------------------------------------------------------↙
    ------
    'Funció generarModul(): Genera el mòdul del vector entre dos punts. Ample fa referencia↙
     a
    'l'escala de representació
    '-----------------------------------------------------------------------------------------↙
    ------
    Function generarModul(ByVal vt As PBasic.point, ByVal ample As Double) As Double

        Dim x2 As Double, y2 As Double, modul As Double, modRet As Double

        'Per alguna raó que no he pogut esbrinar, la formula:
        '   modul = Math.Sqrt((vt.X * vt.X) + (vt.Y * vt.Y))
        'em donava resultat negatiu en cas que el punt de vector fos negatiu ¿?
        'Per tant, he hagut d'optar per insertar pas intermedi perque no em passi.
        x2 = vt.X * vt.X
        y2 = vt.Y * vt.Y

        modul = Math.Sqrt(x2 + y2)
        modRet = modul / ample

        Return modRet

    End Function

    '-----------------------------------------------------------------------------------------↙
    ------
    'Funció generarParalel(): Genera els dos punts paralels al punt de la polilínia per     ↙
    obtenir
    'els punts del polígon. Segons si el punt es de la part "superior" del polígon o de la ↙
    part
    '"inferior" prenem l'opció 1 o 2 respectivament.
    '-----------------------------------------------------------------------------------------↙
    ------
    Function generarParalel(ByVal mod2 As Double, ByVal vec As PBasic.point, ByVal pt As    ↙
    PBasic.point, _
    ByVal tipusPunt As Integer) As PBasic.point

        Dim pRet As New PBasic.point

        Select Case tipusPunt
            Case 1
                'Punt "superior"
                pRet.X = pt.X - (vec.Y / mod2)
                pRet.Y = pt.Y + (vec.X / mod2)

            Case 2
                'Punt "inferior"
                pRet.X = pt.X + (vec.Y / mod2)
                pRet.Y = pt.Y - (vec.X / mod2)

        End Select

        Return pRet

    End Function

    '-----------------------------------------------------------------------------------------↙
    ------
    'Mètode generarLiniesCarrils(): Genera les línies separadores dels carrils. Les linies
    'externes i les que separin carrils en sentit contrari generaran linia continua, la     ↙
    resta,
    'discontinua
```

```
    '-----------------------------------------------------------------------------------------
    ------
    Sub generarLiniesCarrils()

        Dim x As Double, y As Double, modul As Double, ample As Double
        Dim pt1 As New PBasic.point, pt2 As New PBasic.point, pt3 As New PBasic.point
        Dim vect As New PBasic.point, pt As New PBasic.point
        Dim l As Integer

        Try
            'Generem les linies externes del tram
            'Generem la linia "superior" de p1 a pn
            liniaC = New Linia(tIdTram(), 1)
            l = UBound(sentit)
            ReDim liniesTram(l + 1)

            ' n ja es el nº de punts del tram original. Queda desat i el fem servir
            For m = 1 To n
                x = poligon.Points.Item(m).X
                y = poligon.Points.Item(m).Y
                liniaC.generarPolilinia(x, y)
            Next

            liniaC.lPolilinia = ajustarExtremsLinia(liniaC.lPolilinia)
            liniesTram(1) = liniaC.lPolilinia
            frmTFC1.generarLiniaContinua(liniaC)

            'Generen les línies intermitjes
            'Generem els carrils paralel de "dalt" a "baix"
            For w = 2 To l
                m = 1
                'Prenem la darrera linia creada com a base
                pLinia = liniaC.lPolilinia
                liniaC = Nothing
                liniaC = New Linia(tIdTram, w)

                'Generem paralel de p1
                pt1 = pLinia.Points.Item(1)
                pt2 = pLinia.Points.Item(2)

                ample = carrils(w) * escala

                vect = generarVector(pt1, pt2)
                modul = generarModul(vect, ample)

                pt = generarParalel(modul, vect, pt1, 2)
                liniaC.generarPolilinia(pt.X, pt.Y)

                'Generem els parallels de p1+1 fin a pn-1. Si la linia té dos punts, salta
    proces
                m = m + 1
                While (m < n)
                    pt1 = pLinia.Points.Item(m - 1)
                    pt2 = pLinia.Points.Item(m)
                    pt3 = pLinia.Points.Item(m + 1)

                    vect = generarVector(pt1, pt3)
                    modul = generarModul(vect, ample)

                    pt = generarParalel(modul, vect, pt2, 2)
                    liniaC.generarPolilinia(pt.X, pt.Y)

                    m = m + 1
                End While

                'Generem el punt palarel de pn
                pt1 = pLinia.Points.Item(n - 1)
                pt2 = pLinia.Points.Item(n)

                vect = generarVector(pt1, pt2)
                modul = generarModul(vect, ample)

                pt = generarParalel(modul, vect, pt2, 2)
                liniaC.generarPolilinia(pt.X, pt.Y)
```

```
            liniesTram(w) = liniaC.lPolilinia

            'Si els dos carrils tenen el mateix sentit de marxa, generem linia
discontinua.
            'Si son diferents, generem linia continua.
            If getSentit(w) <> getSentit(w - 1) Then
                frmTFC1.generarLiniaContinua(liniaC)
            Else
                frmTFC1.generarLiniaDiscontinua(liniaC)
            End If

        Next

        'Generem la linia "inferior" de pn+1 a pn2
        liniaC = New Linia(tIdTram(), tNCarrils())
        For m = n2 To n + 1 Step -1
            x = poligon.Points.Item(m).X
            y = poligon.Points.Item(m).Y
            liniaC.generarPolilinia(x, y)
        Next

        liniaC.lPolilinia = ajustarExtremsLinia(liniaC.lPolilinia)
        liniesTram(l + 1) = liniaC.lPolilinia
        frmTFC1.generarLiniaContinua(liniaC)


    Catch ex As NullReferenceException
        MsgBox("NullReferenceException: Tram2D.generarLiniesCarrils()")
        MsgBox(ex.Message)
    Catch ex As ArgumentException
        MsgBox("ArgumentException: Tram2D.generarLiniesCarrils()")
        MsgBox(ex.Message)
    Catch ex As AccessViolationException
        MsgBox("AccessViolationException: Tram2D.generarLiniesCarrils()")
        MsgBox(ex.Message)
    Catch ex As IndexOutOfRangeException
        MsgBox("IndexOutOfRangeException: Tram2D.generarLiniesCarrils(), Tram " &
tIdTram())
        MsgBox(ex.Message)
    End Try
End Sub

'-----------------------------------------------------------------------------------
------
'Funció ajustarExtremsLinia(): Ajusta els extrems de les linies de senyalització
'(LINIACONTINUA i LINIADISCONTINUA) per tal que no es creuin en una cruilla
'-----------------------------------------------------------------------------------
------
Function ajustarExtremsLinia(ByVal linia As PolylineGeometry) As PolylineGeometry

    Dim pt As New PBasic.point, pt1 As PBasic.point, pt2 As PBasic.point, vt As PBasic.
point
    Dim ample As Double, modul As Double
    Dim n As Integer

    ample = (3.5 * tEscala) / 2
    n = linia.Points.Count

    'Ajustem el primer extrem de la linia (pt1)
    pt1 = linia.Points.Item(1)
    pt2 = linia.Points.Item(2)

    vt = generarVector(pt1, pt2)
    modul = generarModul(vt, ample)

    pt.X = pt1.X - (vt.X / modul)
    pt.Y = pt1.Y - (vt.Y / modul)

    linia.Points.Item(1).X = pt.X
    linia.Points.Item(1).Y = pt.Y

    'Ajustem el segon extrem de la linia (ptn)
    pt1 = linia.Points.Item(n)
```

```
        pt2 = linia.Points.Item(n - 1)

        vt = generarVector(pt1, pt2)
        modul = generarModul(vt, ample)

        pt.X = pt1.X - (vt.X / modul)
        pt.Y = pt1.Y - (vt.Y / modul)

        linia.Points.Item(n).X = pt.X
        linia.Points.Item(n).Y = pt.Y

        Return linia

    End Function


End Class
```