Софийски университет "Св. Климент Охридски"

Факултет по математика и информатика

Катедра "Информационни технологии"

Тема: *Разработка на графичен модул за GPS устройство с ОС Windows CE*

Дипломант: Николай Людмилов Николов

Специалност: Информатика

Магистърска програма: Разпределени системи и мобилни технологии(РСМТ)

Факултетен No: М21878

Научен ръководител: доц. Антоний Тодоров Попов

гр.София 2007

Съдържание

1 Увод

Картографията и първите географски карти датират от преди хиляди години, но само през последните няколко десетилетия съвременната технология успява да обедини география, компютърна графика и БД (бази от данни) за създаването на различни картографски системи и GIS (географски информационни системи). Тази връзка между данни и география е основата, която прави GIS толкова мощни: картите могат да бъдат генерирани въз основа на БД, а към данните могат да се правят обръщения посредством карти.

През последните няколко години технологията даде тласък на разработването на сравнително комплексно програмно осигуряване работещо върху джобни компютри и други мобилни устройства. Разработването на професионални приложения за мобилни устройства заедно с GPS (Global Positioning System) технологиите, които автоматизират работата на геодезисти и други потребители довежда до опростяване процеса на измерване на площи и картографиране.

В картографските системи се използват карти, които представляват слоеве от географски обекти и атрибутна информация към тях. Чрез добавянето и премахването на даден слой се управляват различните категории от обекти върху картата. Върху всеки слой е предоставена възможност да се изобразяват от потребителя различни графични изображения представящи топографска информация.

Глобалната спътникова система за определяне на местоположението GPS е едно от най-важните научни и приложни постижения през последните 35 години на 20 век.

Чрез нея може да се определи по всяко време местоположението, посоката и скоростта на дижение на обекти, разположени на сушата или водата, във въздуха и околоземното космическо пространство в геоцентрична координатна система.

GPS технологиите намират приложение в множество области от науката и практиката и са в динамичен процес на обновление и развитие. Различните видове приложение на тези технологии са ограничени единствено от човешкото въобръжение.

Настоящата работа има за цел да предостави реализация на метод (програмно решение), който посредством съвременните GPS технологии предоставя възможност за създаването на слоеве от графично и атрибутно описани географски обекти, които в последствие да могат да се импортират в съвременната картографска система "MapInfo Professional". Други задачи, които следват са пренасяне на част от функционалността на приложението "MapInfo Professional", както и добавяне на функционалност за чертаене на графика посредством GPS приемник. Програмното решение трябва да се интегрира в мобилно устройство работещо под "Windows CE" операционна система с вграден GPS приемник, като част от професионално GPS приложение или самостоятелно приложение.

1.1 Стуктура на работата

Теоритична обосновка на предметната област – подробно са изложени основните принципи на работа на GPS системата, интерпретирането и използването на GPS данните, NMEA стандарт за GPS данни и тяхното интерпретиране. Разгледано е професионалното картографско приложение "MapInfo Professional" и неговия стандарт за

данни. Описана е структурата на MIF/MID файловия формат на MapInfo, който трябва да се поддържа от разработвания модул.

Проектиране и реализация – изложено е описание на дизайна и имплементацията на парсера за NMEA протокола, програмния модел и реализацията на основните графични примитиви точка, линия, полилиния и регион поддържани от MapInfo, и реализирани в програмния модул. Изложен е дизайна на функционалността за четене и запис на MIF/MID данни и тяхното изчертаване на екрана. Управлението на изобразяваните обекти на екрана изразяващо се в трансформациите от географска координатна система в екранни координати и обратно също са обект на тази точка. По-долу са описани потребителския интерфейс и разработения слой за четене на серийни данни (от сериен порт или файл).

Тестове и анализ на получените резултати – описание на тестове за производителност и изводите от тях.

Заключение – обобщение на извършената работа и насоки за бъдещо развитие и подобрение.

Използвана литература – изреждане на използваната литература.

Приложение I – екрани изобразяващи реализираната функционалност.

2 Теоритична обосновка на предметната област

Създаването на програмен продукт, който да осъществява събиране на географски данни с помощта на GPS приемник е задача, която предполага изседване на някои основни принципи заложени в работата на глобалната система за позициониране, интерпретирането и използването на GPS даните за описване на графични изображения, и съхраняването на тези данни в определен формат.

За целите на заданието е разгледано накратко и професионалното картографско приложение "MapInfo Professional", и използвания от него MIF/MID (MapInfo Interchange Format) файлов формат за представяне на данни. Описани са GPS стандарти за дани и използваните софтуерни технологии за реализиране на заданието.

2.1 Преглед на GPS

Глобалната система за определяне на местоположение е замислена като система, която да отговаря на въпросите "Колко е часът, какво е местоположението и каква е скоростта?" бързо, точно и евтино, навсякъде по земното кълбо, по всяко време.

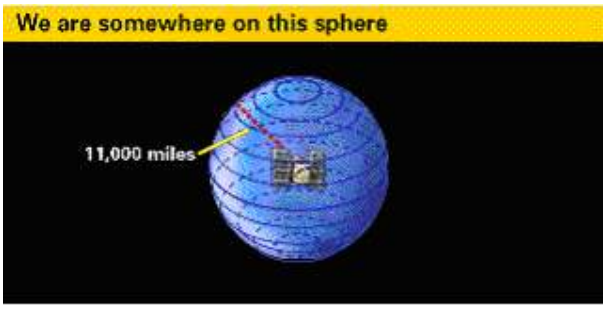
2.1.1 Определяне на местоположението с GPS

За да се осигури възможност за непрекъснато глобално определяне на положение, е създадена схема за наблюдение на достатъчен брой спътници, за да може поне четири от тях да са видими по-всяко време. Оказва се, че 24 равномерно размоложени спътника с кръгови 12-часови орбити, наклонени на 55 градуса към екваториалната равнина, осигуряват необходимото покритие при най-ниска цена. Във всички случаи тази конфигурация осигурява

минимум 4 спътника в добро геометрично положение 24 часа на ден, навсякъде по Земята. GPS спътниците са конфигурирани така, че да може ползвателя да определи положението си, изразено например в географска дължина, географска ширина и височина. Това става чрез проста пространствена засечка с разстоянията, измерени до спътниците.

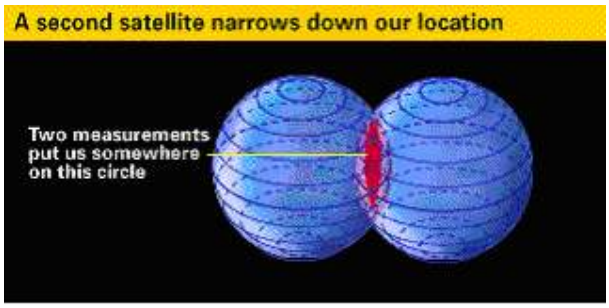
Да предположим че сме определили разстоянието (r1) от нас до един сателит. Знаейки това означава, че се намираме някъде върху сфера с център сателита и радиус r1.

(2.1)



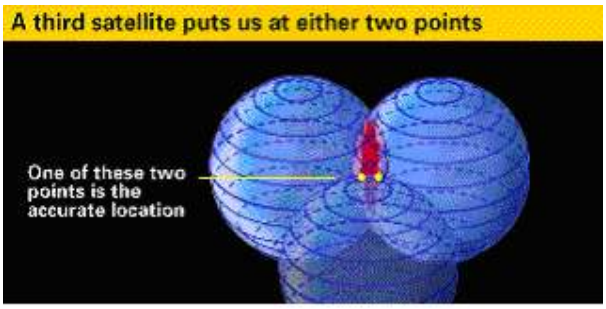
Същата процедура се повтаря за втори сателит, от който се установява, че се намираме също и върху сфера с център втория сателит и радиус разстоянието до втория сателит – r2. С второто измерване сме стеснили множеството от възможности за нашата позиция до окръжността, която се явява сечение на двете сфери.

(2.2)



След измерване на трето разстояние r3 до трети сателит свеждаме възможностите за нашата позиция до две точки (сечението на окръжността с третата сфера).

(2.3)



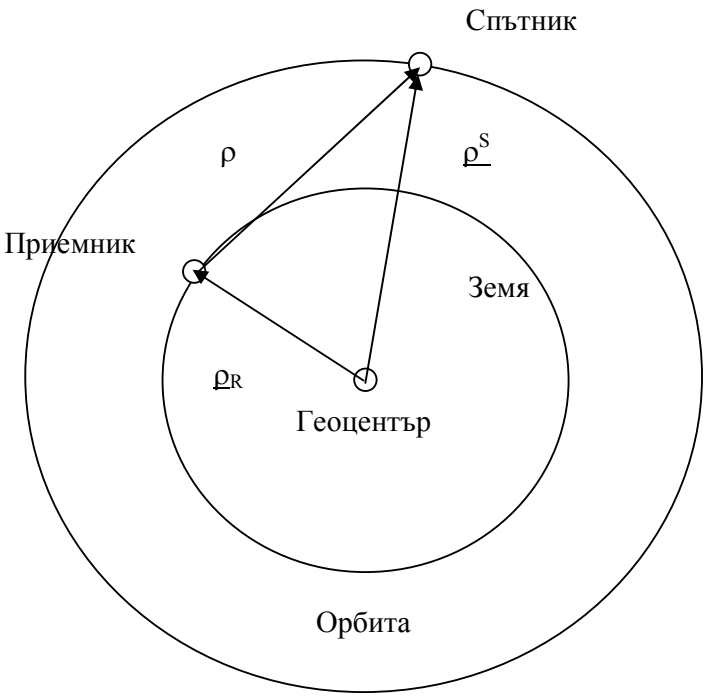
За да решим коя от двете точки е нашата позиция можем да направим четвърто измерване. Обикновенно една от двете точки кандидати за решението е очевидно невъзможна (прекаленно отдалечена е от земята или се движи с невъзможна скорост) и може да бъде лесно отхвърлена. Четвърто измерване се прави с друга цел, но това ще бъде описано. По - долу е представен елементарно математически метод за определяне на местоположението.

Нека се допусне, че в даден момент спътниците са неподвижни в пространството. Пространствените координати $\underline{\rho}^S$ на центъра на всеки спътник (фиг. 2.4) могат да се изчислят от ефемеридите, излъчени от спътника чрез определен алгоритъм. Ако приемникът на Земята, определен от вектора на геоцентричното му положение $\underline{\rho}_R$, използва часовник настроен точно на времето на GPS, действителното разстояние ρ до всеки спътник може да се измери точно, като се отчете времето, за което сигналът (кодиран) от спътника достига до приемника. Всяко разстояние определя една сфера (по-точно повърхнина на сфера), чийто център е спътникът. Затова при този метод са необходими разстоянията само до три спътника и сечението на трите

сфери дава стойностите на трите неизвестни (например ширина, дължина и височина), които могат да се определят от трите уравнения за разстояния:

$$\rho = || \underline{\rho}^{s} - \underline{\rho}_{R} ||$$

(2.4)



ρ - вектор на търсеното разстояние от приемника до спътника.

$\underline{\rho}^{S}$ - вектор на разстояние от геоцентъра до спътника.

$\underline{\rho}_{R}$ - вектор на разстояние от геоцентъра до приемника ***([1])***.

GPS приемниците използват малко по-различен метод. Те използват евтин кварцов часовник, настроен приблизително на GPS време. При това положение часовникът на приемника се отклонява от истинското GPS време и затова разстоянието до спътника се различава от "действителното" разстояние.

Тези разстояния се наричат псевдоразстояния R, тъй като те са истинското разстояние плюс корекцията на разстоянието $\Delta\rho$, получена от грешката в часовника на приемника или систематичната грешка δ. Един прост модел за псевдоразстянието е

$$R = \rho + \Delta\rho = \rho + c\delta, \qquad (2.5)$$

където c е скоростта на светлината.

Необходимо е да се измерят едновременно четири псевдоразстояния, за да се получат четирите неизвестни – това са трите компонента на положението плюс систематичната грешка на часовника δ. Геометрически решението се получава от една сфера, тангираща на четирите сфери, определени от псевдорастоянията. Центърът на тази сфера отговаря на неизвестното положение, а нейният радиус е равен на корекцията на разстоянието.

Трябва да се отбележи че грешката в разстоянието $\Delta\rho$ може да се елиминира предварително. Това става като се образува разликата в псевдоразстоянията, измерени от една станция до два спътника или две различни положения на един спътник.

Като се има предвид основното уравнение (1.1), може да се направи изводът, че точността в положението, определено с един приемник, се влияе от следните фактори:

- точността в положението на всеки спътник;

- точноста на измерените псевдоразстояния;
- геометрията.

Систематичните грешки в положението на спътниците и евентуалните систематични грешки в часовниците им могат да се намалят или елиминират, като се образуват разликите в псевдоразстоянията, измерени от две станции до спътника. Този подход е основен при измерването с GPS. Но никакъв режим на разлики в разстоянията не може да компенсира лошата геометрия на спътниците.

2.1.2 Определяне на скоростта

Определянето на моментната скорост на движещо се превозно средство е другата цел на навигацията. Това може да се извърши, като се използва Доплровия принцип при радиосигналите. Заради относителното движение на GPS спътниците спрямо движещото се превозно средство честотата на излъчения от спътниците сигнал се променя, когато сигналът стигне до него. Това Доплерово изменение е пропорционално на относителната радиална скорост. Тъй като радиалната скорост на спътниците е известна, от Доплеровите наблюдения може да се получи радиалната скорост на превозното средство. Необходими са минимум четири Доплерови наблюдения, за да се получат трите компонента на вектора на скоростта и по възможност и едно честотно изменение.

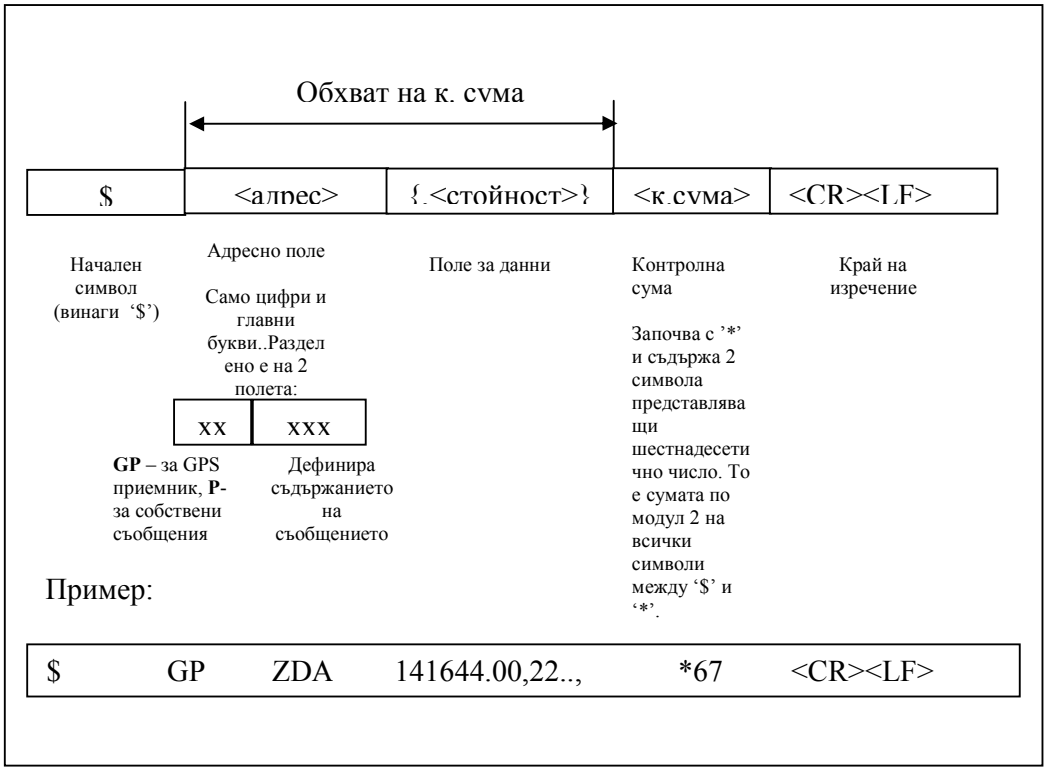
Същевствуват още много аспекти от принципа на работа и приложението на GPS системата, разглеждането на които не са предмет на настоящата разработка.

2.2 Формат на GPS данни

Както бе посочено в мобилното устройство, върху което ще работи програмния модул е интегриран GPS приемник

(receiver) .Спътниковите данни, получавани от приемника (във вид на кодиран електромагнитен сигнал) по някакъв начин е нужно да се интерпретират от програмното осигуряване, което се интересува от тях. Необходима е комуникация между приемника и мобилното устройство. Най - често използвания (а понякога и единствения) комуникационнен механизъм е посредством предаване на данни през сериен порт. Стуктурата на данните, които приемника интерпретира се дефинират от стандарта NMEA (National Marine Electronics Association) 0183 ([2]). Повечето компютърни програми, които предоставят информация за местоположението в реално време очакват данните да бъдат в NMEA формат. Тези данни включват пълното PVT (position,velocity,time) решение изчислено от GPS приемника. Идеята на NMEA е да изпрати поредица от данни наречена изречение, което е самостоятелно и напълно независимо от други изречения. Всички стандартни изречения имат двубуквен префикс, който дефинира вида на устройството, за което е предназначено изречението (за GPS приемниците префика е “GP”). Префиксът е последван от три символа, които дефинират съдържанието на изречението. Всяко изречение започва с “$” и завършва със символи за нов ред и връщане на каретката, като максималната дължина е 80 символа. Различните типове данни в изречението са разделени с ”,”. Самото изречение представлява ASCII текст с променлива дължина. Изречението завършва с разделител “*” последван от контролна сума. Края на изречението се маркира със символите <CR><LF>(<връщане на каретката><нов ред>). На следващата фигура е изобразена структурата на NMEA протоколно съобщение.

(2.6) NMEA протокол



Не всички съобщенията, които дефинира NMEA 0183 стандарта са свързани с GPS навигацията.Основните съобщения, които съдържат такива данни са следното подмножество:

- **GGA – GPS fix data** – съдържа информация на позицията.
- **GLL** – **Latitude and longitude, with time of position fix and status** -информация за географска ширина/дължина.
- **GSA** – **GPS DOP and Active satellites** - изчислена грешка в точността и активни сателити.
- **GSV – GPS Satellites in View –** видими сателити.
- **RMC – Recommended Minimum data** – препоръчан минимум от специфични данни.

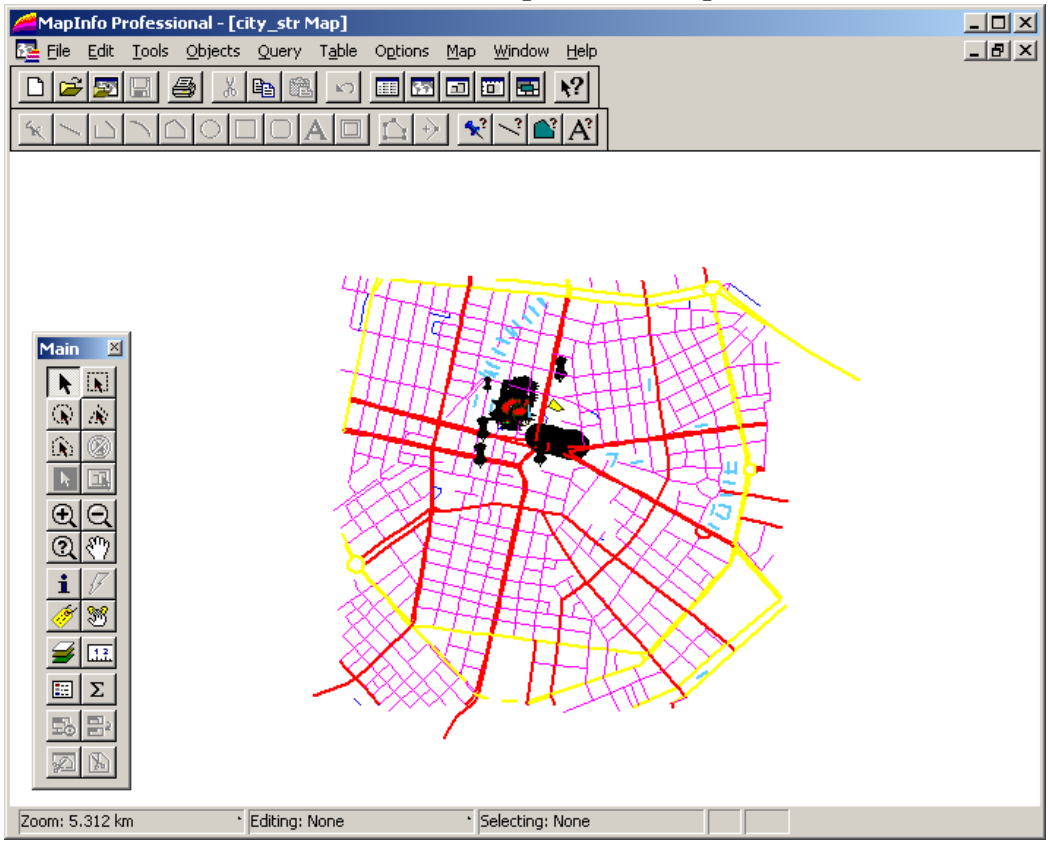
- **VTG – Course over ground and Ground speed** – хоризонтално направление и скорост.

2.3 Преглед на "MapInfo Professional"

MapInfo Professional е мощното Microsoft Windows-базирано настолно картографско приложение, което подпомага GIS професионалисти и аналитици лесно да визуализират връзките между данни и география. С *MapInfo Professional* е възможно да се извърши детайлен анализ на данни, като се използва знанието за местоположение. Включвайки местоположението в операциите по вземане на важни решения спомага за увеличение на ефективността, подобряване на доставяни услуги и намаляване на разходи ([3]).

Една от функционалните възможности на "MapInfo Professional" е да създава карти. Картите са структурирани във вид на слоеве от графични обекти заедно с атрибутна информация за тях (виж. "Увод"). Има възможност за изчертаване на множество различни геометрични примитиви и символи, които изобразяват отделни топографски обекти. Към всеки обект има асоциирана атрибутна (таблична) структурирана информация, която може да се извлича и да бъде манипулирана посредством заявки. Приложението предоставя файлов интерфейс (MIF/MID), чрез който се осъществява експорт и импорт на данни от и към други системи. На фигура 2.7 е показан изглед на работния екран на MapInfo.

(2.7) Екран от "MapInfo Professional"



2.4 "Mapinfo Interchange Format"(MIF) формат от данни

MIF файловият формат служи за импорт/експорт на данни от/към MapInfo софтуерния продукт. Данните са представени в два файла – графичната информация за обектите се намира във файла с разширение "*.MIF", а атрибутната информация във файл с разширение "*.MID" ([4]).

2.4.1 Структура на "*.MIF" файл.

MIF файла съдържа две области – "header"(заглавна част) и "data" секция. Информацията за структурата на атрибутния файл (MID) се съдържа в заглавната част (информация за имена на полета, тип на полета, разделител между полетата, кодова таблица за символите). Информацията за графичните обекти е дефинирана в "data"

секцията. По – долу е дадено описание на заглавната част (MIF header).

2.4.1.1 Описание на "MIF header" секция.

Незадължителните параметри са в квадратни скоби.

VERSION n
Charset "*characterSetName*"
[DELIMITER "<c>"]
[UNIQUE n, n...]
[INDEX n, n...]
[COORDSYS...]
[TRANSFORM...]
COLUMNS n
<name> <type>
<name> <type>

.
.

DATA

Version

Тази клауза указва дали се използва VERSION 1, VERSION 2, VERSION 300 или VERSION 450 на формата.

CharSet

Указва каква кодова таблица е използвана за създаването на текста в атрибутния файл (MID).

Пример: **Charset "WindowsCyrillic"**

Delimiter

Указва какъв разделител е използван в MID файла за разделител между отделните колони. Ако не е посочено по подразбиране се използва табулация.

Пример: **Delimiter ","**

Unique

Указва кои колони от таблицата (MID файла) са уникални (unique constraint). Колоната е реферирана с поредния и номер, като ако са повече от една са разделени със запетая (“,”).

Index

Указва кои колони в таблицата са индексирани. Колоните отново се реферират с поредния си номер.

Пример: **Index 3,7,11,17,18**

CoordSys

Използва се за да укаже, че координатите не са представени във вид на георграфска ширина/дължина. Когато не е използвана клаузата се счита, че координатите са в географска ширина/дължина.

Синтаксис:

CoordSys Earth
[Projection type,
datum,
unitname
[, origin_longitude]
[, origin_latitude]
[, standard_parallel_1 [, standard_parallel_2]]
[, azimuth]
[, scale_factor]
[, false_easting]
[, false_northing]
[, range]]
[Affine Units unitname, A, B C, D, E, F]

[Bounds (minx, miny) (maxx, maxy)

Пример: **CoordSys Earth Projection 1, 104**

Забележка: Използваните проекции(Projection type) и координатни системи (datumn) в MapInfo са много и поддръжката на всички изисква по задълбочено запознаване с начините за преобразуване от една координатна система към друга. В настоящия проект такава функционалност се

предвижда за в бъдеще. За сега се подържа една координатна система WGS84 (World Geodetic System 1984), тъй като и данните от GPS приемника пристигат в същата координатна система.

Transform

Използва се когато координатите не са представени спрямо първи квадрант. За да се преобразуват координатите спрямо първи квадрант се използва трансформация със следния синтаксис:

TRANSFORM *Xmultiplier, Ymultiplier, Xdisplaceme,Ydisplacement*

Пример:

TRANSFORM -1,0,0,0

Columns

Указва броя колони в таблицата. За всяка колона има ред, който съдържа име на колона и тип на колона.

Пример:

```
COLUMNS 3
STATE char (15)
POPULATION integer
AREA decimal (8,4)
```

2.4.1.2 Описание на "MIF Data" секция.

Секцията за данни следва след заглавната част, като задължително започва с ключовата дума "DATA" на отделен ред:

DATA

Тук могат да бъдат описани множество от графични примитиви, по един за всеки графичен обект. Връзката между графичния обект в MIF файла и неговата съответна атрибутна информация в MID файла се осъществява на

следния принцип. Първия обект в MIF файла съответства на първия ред от MID файла, втория обект с втория ред и т.н.

Когато няма графичен обект за съответния ред, в табличния файл в "DATA" секцията се поставя ред (empty, празен), който има следния вид:

NONE

2.4.1.3 Графичните примитиви, които се ползват в MapInfo са:

Point	-	точка
Line	-	линия
Polyline	-	полилиния
Region	-	регион (полигон)
Arc	-	дъга
Text	-	текст
Rectangle	-	правоъгълник
rounded rectangle	-	заоблен правоъгълник
ellipse	-	елипса
multipoint	-	масив от точки
collection	-	колекция

В настоящата дипломна работа е реализирана поддръжката само на първите четири примитива – точка, линия, полилиния и регион.

Точката е обект с два параметъра – x координата и y координата. По избор може да се укаже символ, който визуално представя точката на екрана. Ако не се укаже символ се използва такъв по подразбиране.

Синтаксис:

POINT x y

[SYMBOL (*shape*, *color*, *size*)]

Линията очаква четири параметъра – x и y координати за всеки от крайщата. По избор може да се укаже писалка (PEN), с която да се начертае линията.

Синтаксис:

LINE x1 y1 x2 y2
[PEN (*width*, *pattern*, *color*)]

Полилинията се състои от една или няколко секции. Ако има повече от една секция, към дефиницията се включва запазената дума "MULTILINE". По избор се посочва писалка и плавен стил (SMOOTH).

Синтаксис:

PLINE [MULTIPLE *numsections]*
Numpts1
x1 y1
x2 y2
:
[Numpts2
x1 y1
x2 y2]
:
[PEN (*width*, *pattern*, *color*)]

[SMOOTH]

Регионът представлява съвкупност от един или повече полигона. Броят полигони се посочва в началото след ключовата дума "REGION". За всеки полигон се указва броя точки, които го определят. Всяка точка се описва с двойка координати.

Синтаксис:

REGION *numpolygons*
Numpts1
x1 y1
x2 y2
:
[Numpts2

x1 y1
x2 y2 **]**
:
[PEN (*width*, *pattern*, *color*)]
[BRUSH (*pattern*, *for color*, *backcolor*)]
[CENTER *x y]*

2.4.2 Структура на "*.MID" файл.

MID файла съдържа атрибутната информация – за всеки графичен обект на отделен ред. Разделителят между отделните полета е посочен в заглавната част на MIF файла. Ако разделителят се включва като част от стойността на полето цялата стойност се загражда в ‘ ” ’. MID файла е опционален, т.е може и да не съществува. В този случай атрибутна информация няма.

3 Проектиране и реализация

При проектиране проблема се разделя на отделни подзадачи, така че да могат да бъдат решени. Решаването на отделните задачи ще доведе до решението и на цялостния проблем. Водещи фактори и критерии са използваемостта на по-малко ресурси на компютърната система - оперативна памет и процесорно време. Технологиите за разработка са също от голямо значение, като за настоящата работа е избран **Visual Studio 2003 – C#.**

Идеята е първо да се реализира настолно приложение (улеснен е процеса на дебъгване и отстраняване на проблемите свързани с презентационната логика), след което тази функционалност да се прехвърли на мобилното устройство, като част от професионално приложение или самостоятелно приложение.

Проекта на приложението минава през няколко основни етапа:

1. Дизайн на NMEA парсер за GPS данните.

2. Програмен модел на основните графични примитиви (точка, линия, полилиния, регион) поддържани от формата MIF/MID.
3. Дизайн четец /писец (MIFFile reader/writer) на MIF/MID данни.
4. Дизайн на графичен потребителски интерфейс (GUI).

3.1 Дизайн на NMEA парсер

Повечето протоколи реализират абстрактна машина на състоянията (краен автомат), който описва възможните състояния и грешките, които могат да настъпят по време на обработката на данните.

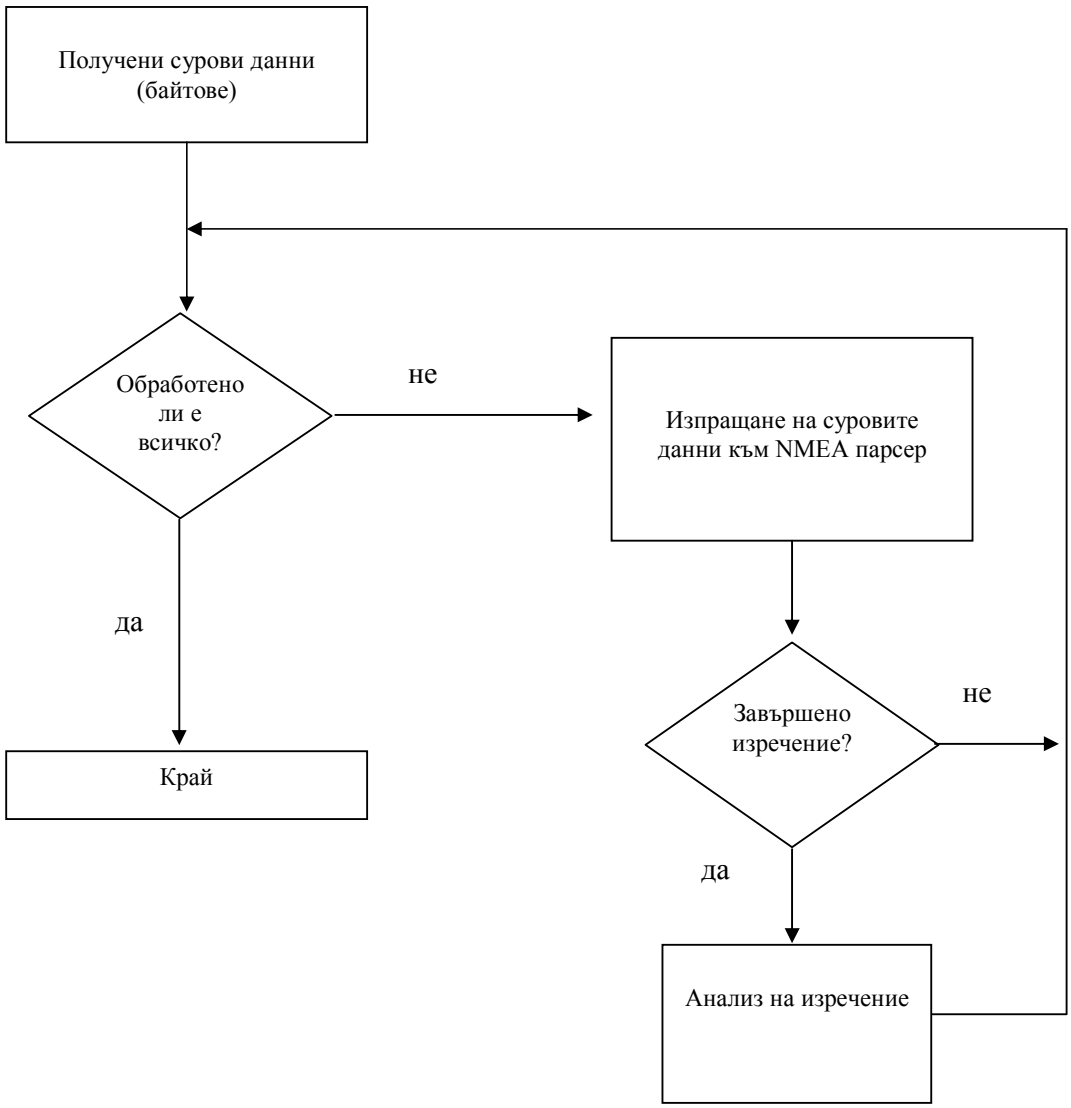
За да се реализира обработката на NMEA данните постъпили през серийня порт на устройството е необходимо да се реализира парсер. Неговата задача ще е да "сглоби" поредицата от постъпващи байтове в поредица от NMEA съобщения и да обработи тези съобщения. Парсерът, които е реализиран се базира на опростена машина на състоянията. Използвайки машина на състоянията компютърът може лесно да следи къде се намира в процеса на анализ на протокола, както и да се възстанови от всякакви грешки като таймаут, и грешни контролни суми.

Дизайнът е направен, така че прочетения буфер да може да бъде изпратен на парсера заедно с неговата големина, за да се максимизира ефикасността на компютърния процесор. Това ще позволи на компютъра да парсне данните, когато бъдат получени. Тъй като скоростта на предаване на NMEA данните стандартно е 4800 baud, на компютъра му се налага често да чака. С помощта на подходящ автомат частично или пълно множество от NMEA данни може да бъде анализирано. Ако само частично множество е получено и предадено на парсера, то когато

останалата част се получи парсерът ще завърши съответното започнато изречение.

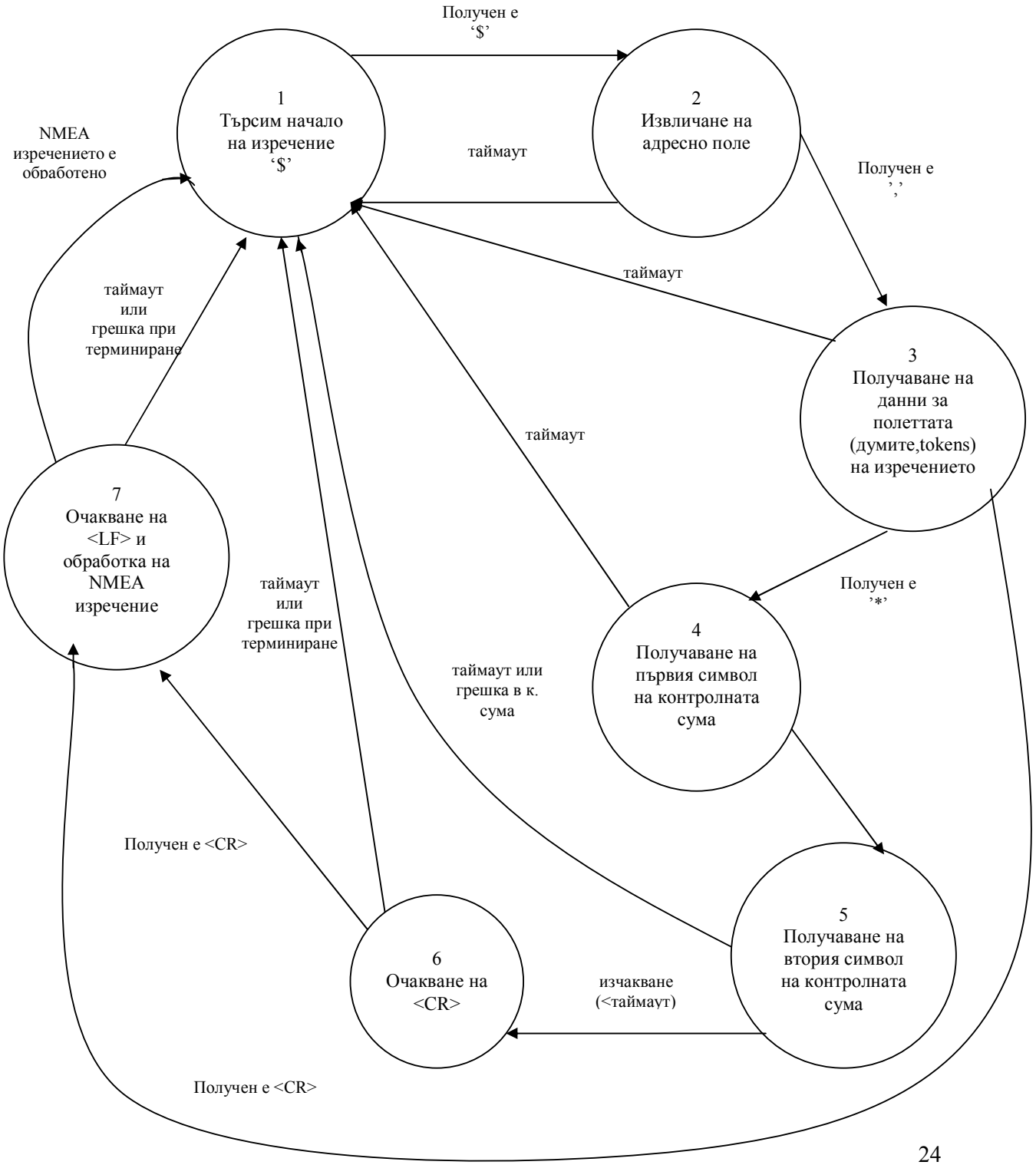
На фигурата е показан потокът от данни през NMEA парсера. Дизайна предполага два абстрактни слоя - NMEA протокол парсер и обработчик на NMEA изреченията. Броя получени байтове е без значение, както и дали изречението е частично изградено. Парсерът ще анализира данните коректно и ще извлече отделните съобщения ([5]).

(3.1) поток от данни през NMEA парсер



Всеки път, когато компютъра получи данни ги праща на NMEA парсера. По - долу ще бъде обяснено какво се случва всеки път, когато е получено валидно съобщение.

(3.2) Парсер на NMEA протокол (автоматен граф)



Фигурата илюстрира всички състояния на парсера и прехода между тях. Таймаут прехода е добавен към всяко състояние, за да може автомата да се синхронизира когато няма данни за определен период (дефиниран от приложението).

Дефиници на състоянията на крайния автомат за NMEA протокола:

- Търсене на начало на изречението. Това състояние търси символа '$'. Когато началото е открито, тогава остава прехода към извличане на адресното поле.
- Извличане на адресно поле. Парсерът ще събира символи докато не достигне до ',' (запетая). Адресното поле се предвижда да бъде с променлива дължина.
- Извличане на данните за изречението. Данните асоцииране с NMEA адресното поле се събират за по-късен анализ. Състоянието ще продължи да събира данни и да изчислява контролна сума, докато не получи ограничителя за контролна сума '*' или край на изречение <CR><LF>.
- Получаване на първия символ на контролната сума. Състоянието просто чака за първия символ на контролната сума.
- Получаване на втория символ на контролната сума. Когато втория символ на контролната сума е получен, трябва да е налична достатъчно информация, за да се верифицира получената сума с изчислената сума от състояние 3. Ако сумите съвпадат, единственото което остава е да се провери за край на изречението.
- Очакване на <CR>. Тук просто се чака за първия от двата символа за край на изречение.
- Очакване на <LF>. Когато вторият символ за край на изречението е получен ще се извърши обръщение към обработчика на NMEA изречения (който представлява

функция за обработка на вече построено синтактично валидно изречение).

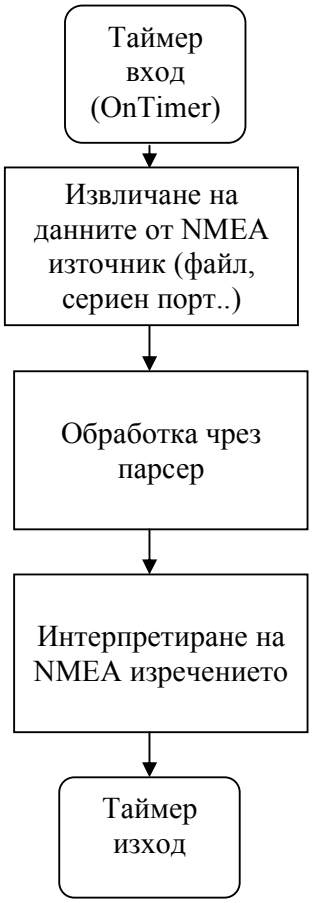
3.1.1 Обработка на NMEA изреченията.

Функцията, която обработва NMEA съобщенията е последната стъпка от действителното парсване на данните.

Има няколко избора, които могат да бъдат направени при извличането на специфични данни от символен низ. Възможен метод е да се извлекат данните и да се отбележи чрез флаг или брояч, факта че данните са се променили. Друг подход е да се използва "callback" техника, където да се дефинира функция, извикваща се всеки път, когато се получи съобщение. В настоящата работа е избран първия подход, като по-универсален.

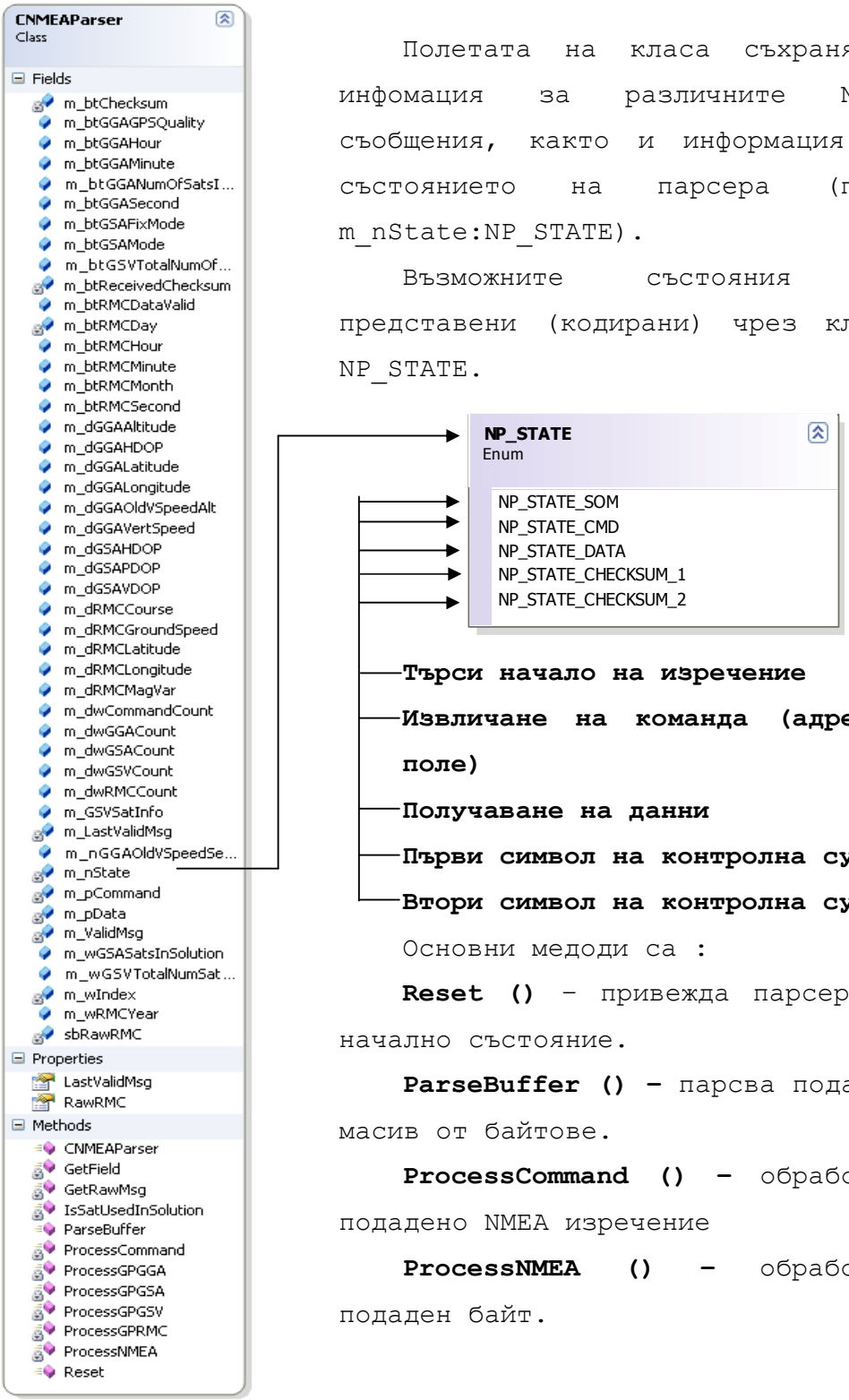
Конфигуриран е таймер на 100ms, който да се обръща за данни към RS-232 серийния порт и да изпраща тези данни към парсера. На следващата фигура е показано схематично идеята.

(3.3) Обработка на NMEA изречение



Програмния модел, който представя NMEA парсер ще се реализира от клас показан на фигурата:

(3.4) NMEA парсер клас



Полетата на класа съхраняват инфомация за различните NMEA съобщения, както и информация за състоянието на парсера (поле m_nState:NP_STATE).

Възможните състояния са представени (кодирани) чрез класа NP_STATE.

Основни медоди са :

Reset () – привежда парсера в начално състояние.

ParseBuffer () – парсва подаден масив от байтове.

ProcessCommand () – обработва подадено NMEA изречение

ProcessNMEA () – обработва подаден байт.

ProcessGPMMM () – методи обработващи конкретно NMEA изречение “MMM”(GGA, GSV, GSA, RMC).

3.2 Програмен модел на основните графични примитиви (точка, линия, полилиния, регион) поддържани от формат MIF/MID.

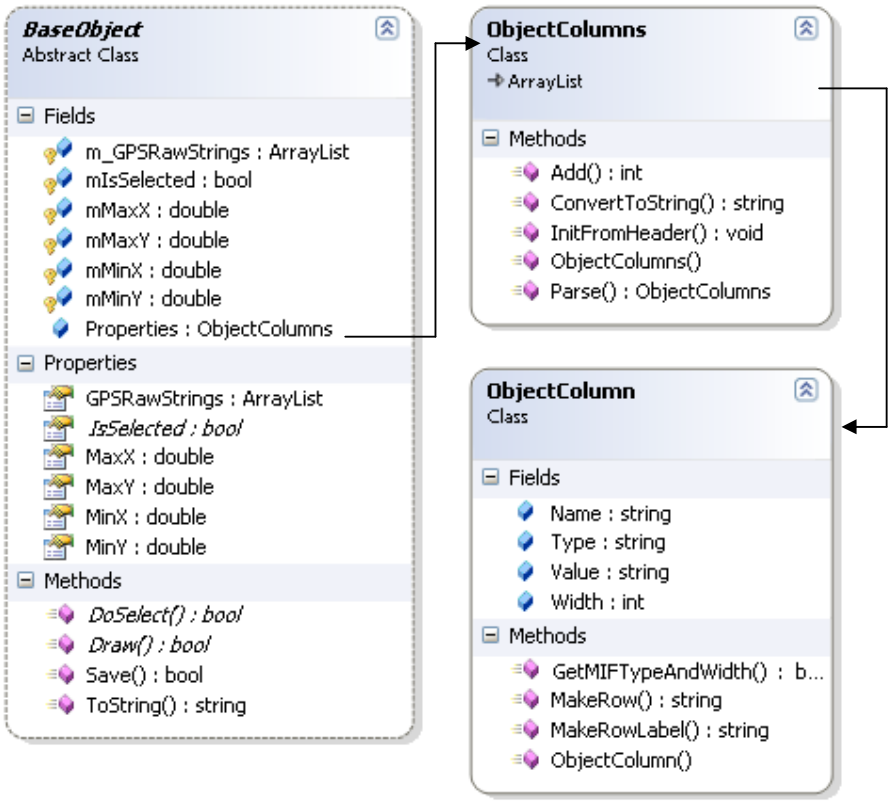
Програмния модул ще подържа основните графични примитиви на MapInfo – точка, линия, полилиния и регион. За целта е необходимо да се осигури функционалност за графичното изобразяване на прочетените примитиви от текстовия файл (MIF файла), както и функционалност за изчертаването на същите от страна на потребителя (посредством GPS или без GPS). Всеки от тези примититиви притежава стилови характеристики като цвят и дебелина на контур (линия, полилиния, регион), големина на символ (графичното изображение служещо за изобразяване на точката), цвят на запълване (регион).

За да се реализират изброените функционалности са проектирани класове, всеки от които представлява програмен модел на съответен графичен примитив. На фигурата е показана схема на тези класове .

(3.5) Йерархия от класове на графични примитиви

Абстрактния клас "BaseObject" дефинира базовите състояния и поведения, които трябва да притежава всеки един от класовете примитиви (наследниците). На следващите фигури са представени по отделно всеки един от класовете.

(3.6) Базов абстрактен примитивен клас "BaseObject"



Характеристиките, които трябва да притежава всеки графичен примитив са:

- **m_GPSRawString** – колекция от низове, които съхраняват суровите NMEA изречения, довели до дефинирането на точки от примитива.
- **mIsSelected** - булево поле за определяне дали един изобразен примитив е в състояние селектиран или не.

- **mMinX, mMinY, mMaxX, mMaxY –** минимална и максимална стойност на всяка координата (географска координата във WGS84 координатна система) с цел определяне на мащаб за изобразяване на обектите.
- **Properties –** списък с атрибутната (табличната) информация (от MID файла) за съответния обект.

Поведението, което трвбва да притежава всеки графичен примитив се представя със следните базови методи:

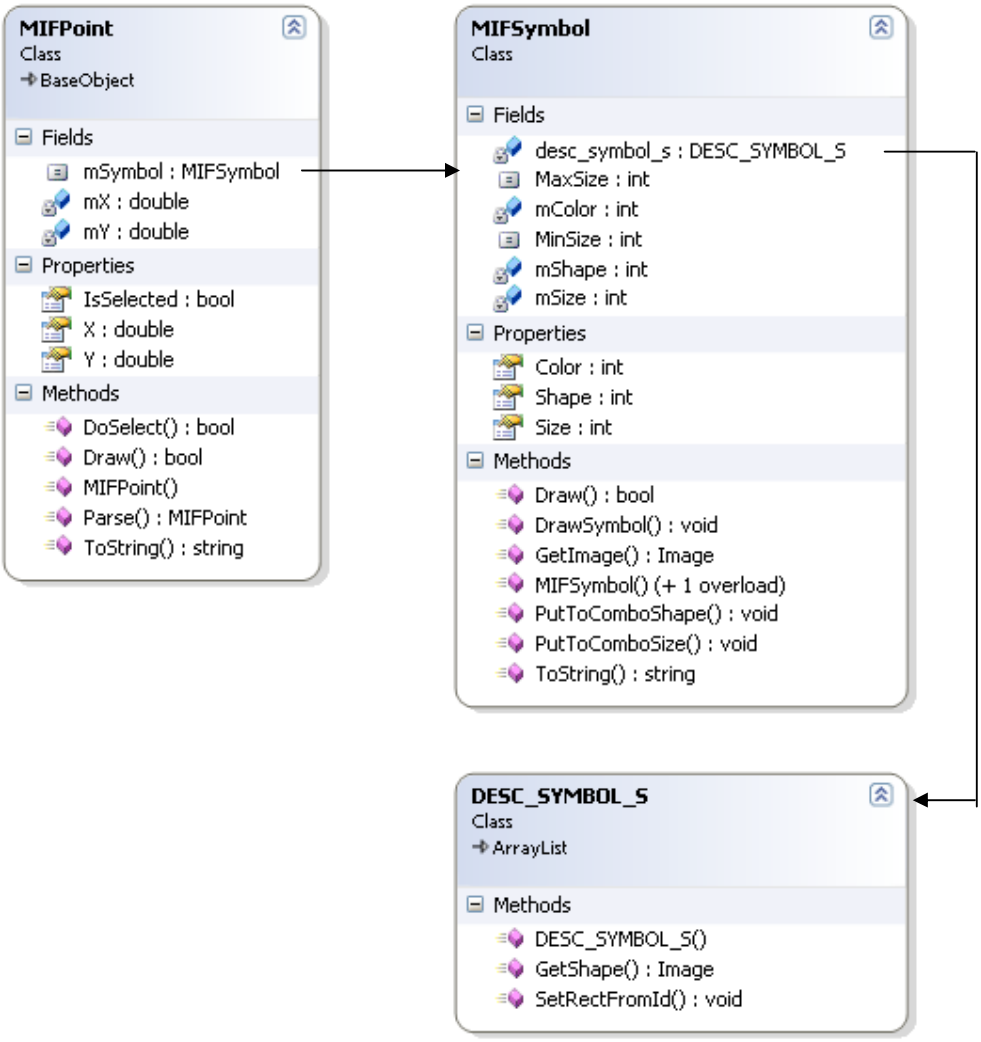
- **DoSelect()-** абстрактен метод, който определя дали потребителя е стъпил върху даден обект (чрез конкретно имплементиран алгоритъм за определяне дали точката с координати позицията при настъпил "MouseClick"(събитието при натискане ляв бутон на мишка) се намира в достатъчна близост до обекта (за точка, линия полилиния) или е вътре в него (за регион). Ако резултатът е положителен се извършва инвертиране на mIsSelect флага. Чрез този механизъм се реализира селектирането и деселектирането на изобразен графичен обект, като всеки от класовете наследници на базовия предостави припокриване на този метод със собствена вътрешна реализация.
- **Draw()-** абстрактен метод, който реализира изчертаването на графичния обект върху екрана. Отново всеки клас наследник припокрива този метод.
- **Save()-** метод, който реализира запис на графичния обект в MIF файла.
- **ToString()-** припокрива базовия метод на класа **Object.** Всеки от примитивните класове го припокрива и реализира в подходящо низово представяне на обекта, поддържано от MapInfo.

Класа “MIFPoint” е програмния модел на обекта от “MapInfo” point (*виж 2.4.1.2. Описание на “MIF Data” секция*). Точката, която се изобразява на екрана се означава с един от символите заложени в таблицата със символи на “MapInfo”, която е показана на следната фигура:

(3.7)Таблица със символи от MapInfo

Всеки един от тези символи се определя еднозначно от идентификатор (цяло число в интервала (31-67)). MapInfo професионалното приложение включва поддръжка на по-голямо множество от символи, но не трябва да се забравя, че настолните приложения обикновенно разполагат с по-голяма изчислителна мощ и оперативна памет в сравнение с приложенията за мобилни устройства и позволяват бърза работа с по-големи масиви от изображения. На следващата фигура е дадена по-подробна информация за класа MIFpoint и връзката му със символите.

(3.8) MIFPoint клас и връзката му със символите



Методите **DoSelect()** и **Draw()** вече бяха обяснени.

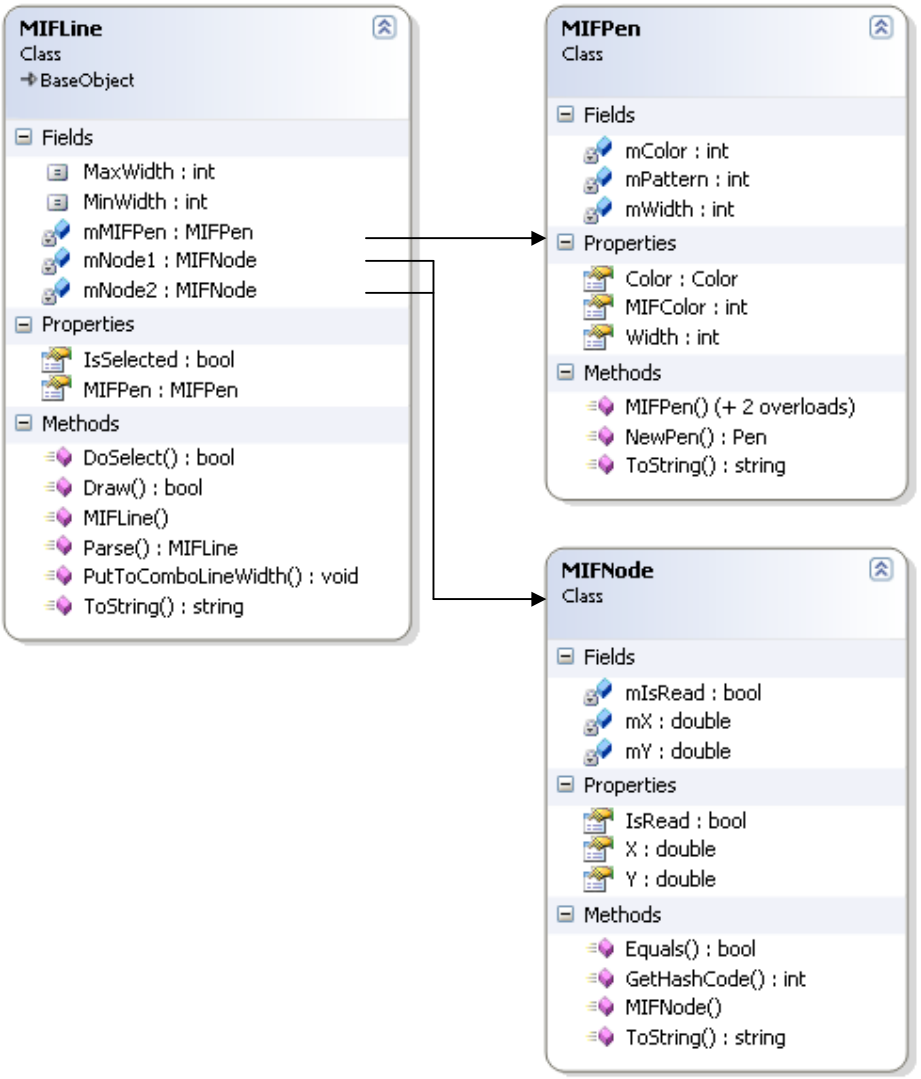
- **Parse() –** статичен метод, който връща инстанция на MIFPoint конструирана от символен низ.
- **ToString()** - извършва обратното действие на **Parse()**, т.е извършва представяне на MIFpoint обект към символен низ в MIF формат.
- **MIFSymbol** класа е модел на "SYMBOL" структурата от "MapInfo". Съхранява информация за изображението на

символа от таблицата със символи (mShape – цяло число от 31 до 67, тъй като това са идентификаторите на символи от таблицата със символи), неговия размер (mSize) и цвят (mColor).

- **DrawSymbol() –** статичен метод, който рисува върху екрана символ с определен идентификатор.
- **Draw() –**припокрива базовия абстрактен **Draw()**, като рисува определен символ (чрез **DrawSymbol**), но на конкретна позиция (mX, mY).
- **PutToComboSize() –** поставя предефинираните размери на символите в ComboBox контрола.
- **PutToComboShape() –** поставя символите в ComboBox контрола.
- **GetImage() –** връща "Bitmap" със съответния символ.
- **DESC_SYMBOL_S -** клас списък от обекти, всеки от които описва съответен символ от таблицата със символи.
- **GetShape() –** метод, които връща битмап представяне на символа по зададен идентификатор на символ.
- **SetRectFromID() –** статичен метод, който определя обграждащ правоъгълник за символ с конкретен идентификатор.

Класа "MIFLine" е програмния модел на обекта "line" от MapInfo. Чрез него се изобразява на екрана прочета от MIF файла линия, както и се съхранява в MIF файла изчертна на екрана линия. Диаграмата и основните методи са показани на фигура 3.9.

(3.9) MIFLine клас



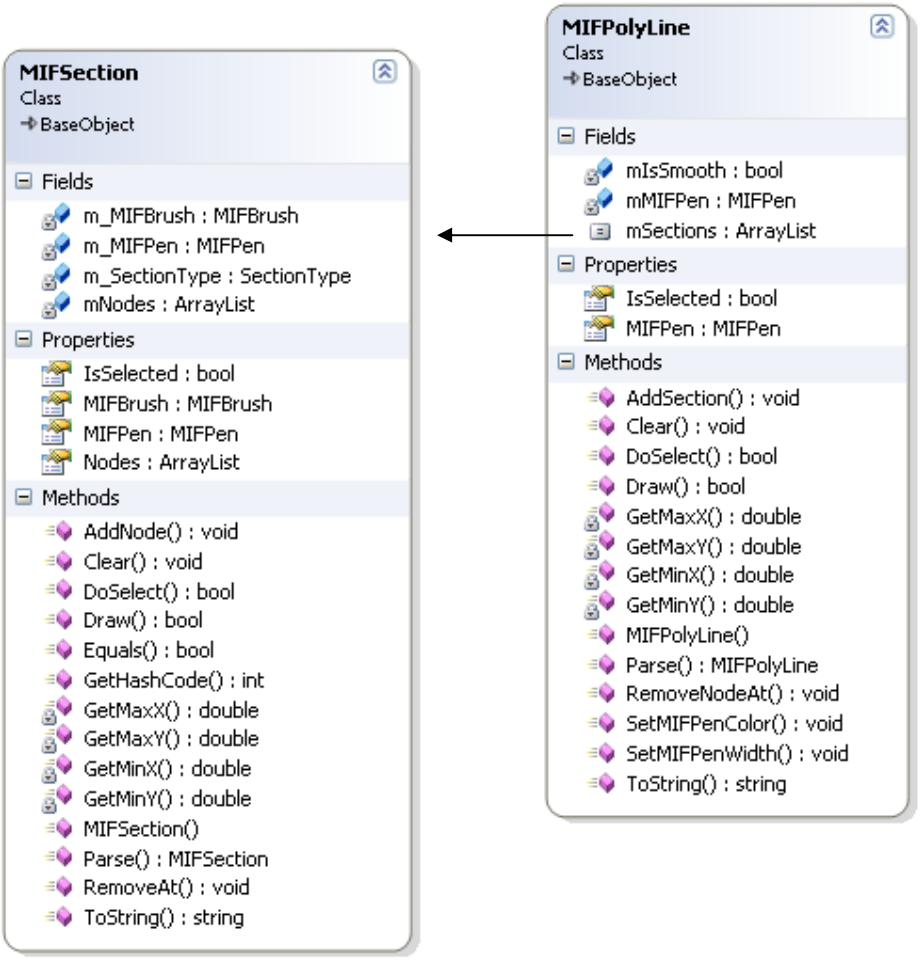
- **MIFPen** класа моделира "PEN" структурата от MapInfo, като съдържа информация съответно за цвят (mColor, mMIFColor), стил (mPattern) и дебелина (mWidth) на линията. NewPen() методът връща конструирана писалка, с която да се чертае.

- **MaxWidth**, **MinWidth** – максималната и минималната дебелина на линията.
- **NewPen()** методът връща инстанция на стандартна GDI+ (Graphic Device Interface) писалка - Pen.
- **Equals()** – предефиниран метод, който служи за стравнение на два обекта MIFNode.
- **GetHashCode()** – предефиниран метод, който връща хеш код на обекта.
- **mNode1**, **mNode2** – точките определящи линията.
- **mIsSelected** – указва дали линията е селектирана или не.
- **mMIFPen** – инстанция на класа MIFPen.
- **PutToComboGetLineWidth()** – инициализира "падащ списък" контола със стойности за дебелина на линия.
- **Parse()** – статичен метод, който връща инстанция на MIFLine конструирана на базата на подаден низ.
- **Doselect()** – метод, който извършва проверка дали точката, върху която е натиснал с левия бутон на мишката е достатъчно близо до линията и ако е така, инвертира състоянието на mIsSelected.
- **Draw()** – изчертава линията върху графичния конекст.

MIFNode класа точка, като част от графичен обект (line, polyline, region). Представя се с координати (географски) **mX** и **mY**.

Следващия клас представлява модел на структурата “polyline”.

(3.10) MIFPolyLine клас



По своята същност MIFPyline представлява списък от секции. Всяка секция (класа MIFSection) пък от своя страна представлява списък от точки, които определят краищата на отсечките формиращи полилинията (mNodes). Чрез изчертаването на отсечките от всички секции се формира цялата полилиния.

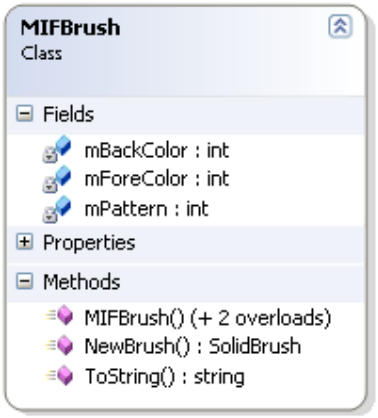
По важни полета и методи на класа MIFPolyLine са:

- **mSections** – инстанционна променлива - списък от обекти (секции) инстанции на **MIFSection.** Както е посочено в описанието на примитива "polyline" секцията е определена от списъка с точки.
- **MMIFPen –** писалката, с която е начертана полилинията.
- **AddSection() –** метод, който добавя секция към списъка със секции.
- **Clear() -** метод, който изчиства списъка със секции.
- **Parse() –** статичен метод, който връща инстатция на MIFPolyline от подаден символен низ.
- **RemoveNodeAt() –** метод, който премахва точка от списъка с точки на определена секция.
- **SetMIFPenColor() –** метод, който задава цвят на писалка.
- **SetMIFPenWidth() –** метод, който задава дебелина на писалка.
- **ToString() –** припокрит метод за низово представяне на обекта полиния.

Структурата секция, като обект съставен от отсечки изгражда както полилинията така и региона.

По важни полета и методи на класа MIFSection са:

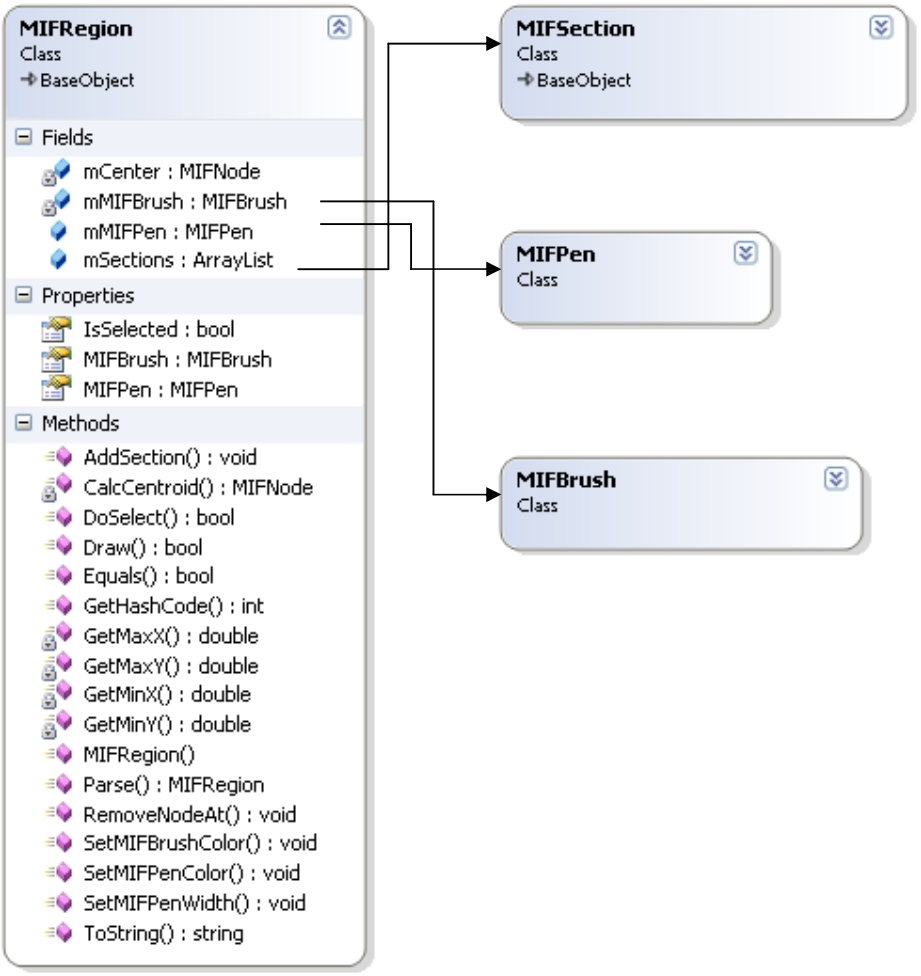
- **m_MIFBrush –** модел на четка, с която се извършва запълване на регион. Полетата mBackColor, mForeColor, mPattern съхраняват стойности за цвят на запълване, цвят на контур и текстура. Методът NewBrush() – връща констуирана четка от тип SolidBrush().



- **m_SectionType** – изброен тип, който показва дали секцията е съставна част на регион или на полилиния.
- **mNodes** – списък от точки съставящи секцията.
- **AddNode()** – метод, който добавя точка към секцията.

Поведението и състоянията на региона наподобяват тези на полилинията, като има и наякои разлики при ичертаването селектирането и представянето като символен низ. "Region" структурата се реализира чрез класа на следващата диаграма.

(3.11) MIFRegion

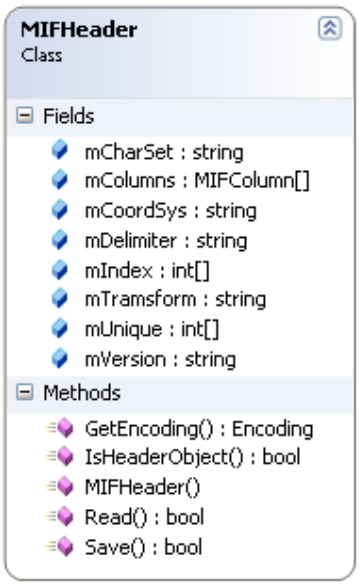


Третирането на графичните обекти като едно цяло изисква дефинирането на обединяваща структура, която да осигури универсални манипулации с графичните обекти, тяхното предстаявне в оперативната памет и тяхното съхранение на магнитен или друг носител в MIF файлов формат.

За решението на този проблем е проектиран и релизиран модел на MIF формата нпредоставен от MapInfo.

3.3 Дизайн на четец/писец (MIFFile reader/writer) на MIF/MID данни.

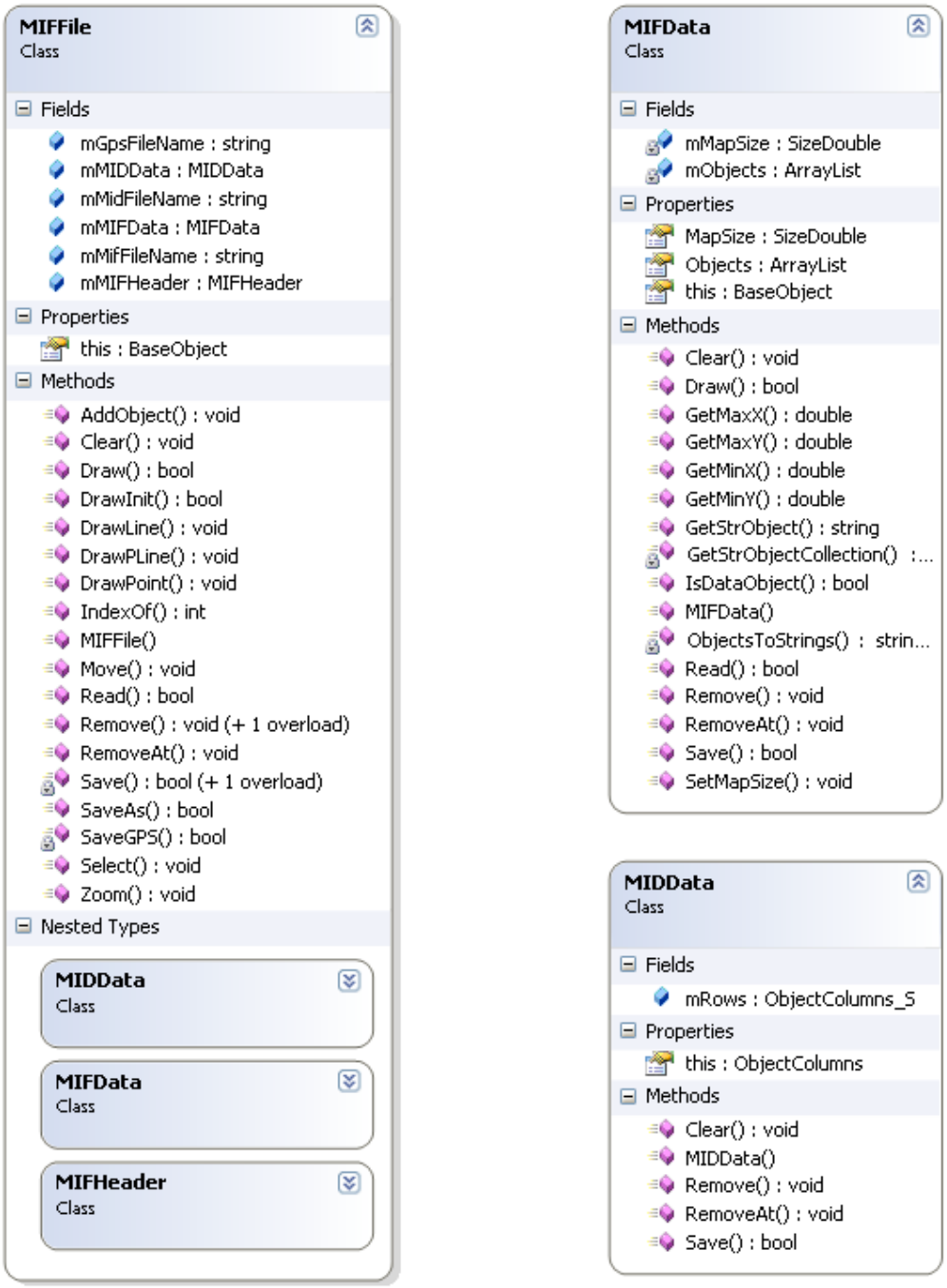
За да се осигури съхраняването на информацията за графичните обекти е необходимо да се реализира механизъм за сериализацията на тези обекти. За постигането на целта е проектиран класа MIFFile. Идеята е всеки път, когато се зарежда нов MIF файл да се създава инстанция на MIFFile в паметта и да се работи с нея. При запис да се извършва сериализиране на MIFFile във вид на текстов mif файл. Класът дефинира вътре в себе си "inline" (вътрешни) класове, които моделират отделните структури, от които е изграден MIF файла и съответния MID файл за атрибутната информация към обекта. Това са съответно класовете MIFHeader, MIFData и MIDData.



Обект от клас MIFHeader първи се инициализира при четенето от MIF файла. От него се определя символното кодиране използвано в MID файла (mCharSet), неговата структура (mColumns[]), използвания разделител между отделните стойностите (mDelimiter) и други параметри характеризиращи двата файла съответно с графичната и атрибутна информация.

Реализацията на класа MIFFile е представена на следващата диаграма.

(3.12) MIFFile клас



Класа MIFFile е проектиран да бъде модел на структурирания текстов файл за съхраняване на графична информация предоставен от MapInfo (*2.4.1. Структура на "*.MIF" файл*). Като вложени класове са предвидени отделните структури на файла –MIFHeader, MIFData, както и вложен клас моделиращ файла с атрибутната информация

(*.MID файл) – MIDData. Главната идея е MIFFile да представлява слоя от обекти, който ще се редактира. Всеки път при отваряне на mif файл се създава инстанция на MIFFile и нейната стуктура се инициализира чрез парсване на структурирания текстов файл.

Някой по-важни полета, свойства и методи са следните:

- **m_GpsFileName –** име на текстов файл, в който ще се съхраняват NMEA съобщенията, чрез които са начертани съответни графични обекти. Съобщенията се съграняват в суров вид, т.е както са получени от приемника и парснати от NMEA парсера;
- **m_MIDData** – инстанционна променлива (инстанция на класа MIDData). През нея ще минават обръщенията към файла с атрибутна информация;
- **m_MidFileName –** инстанционна низова променлива, съдържаща пълния път до MIF файла;
- **M_MidFileName –** инстанционна низова променлива, съдържаща пълния път до MID файла;
- **m_MIFData** – инстанционна променлива (инстанция на MIFData), съдържаща списък с графичните обекти в MIF формат (DATA секцията на MIF файла);
- **m_MIFHeader -** инстанционна променлива (инстанция на MIFHeader), съдържаща заглавната част на MIF файла;
- **this[index] –** предефиниран индексатор за достъп до графичен обект по неговия индекс;
- **AddObject() –** добавя графичен обект към списъка с обекти;
- **Clear() –** изчиства списъка с обекти;
- **Draw() –** изчертава всички обекти от списъка с обекти върху грфичния контекст;

- **DrawInit() –** първоначално изчертаване на обектите. **DrawLine() –** изчертава линия върху грфичния контекст;
- **DrawLine() –** изчертава линия върху грфичния контекст. Извиква се, когато подребителя чертае върху екрана.
- **DrawPoint() -** рисува точка върху графичния контекст. Извиква се, когато подребителя чертае върху екрана.
- **DrawPline() –** изчертава полилиния върху грфичния контекст.
- **IndexOf() –** метод, който връща индекса на конкретен обект от списъка с обекти.
- **Move() –** извършва преместване на екрана посредством мишката с определен вектор образуван от координатите на мишкта, където е натиснат левия бутон до координатите на мишката, където е отпуснат.
- **Read() –** прочита файла и инциализира класа.
- **Remove** - премахва последния обект от списъка с обекти.
- **RemoveAt** - премахва обект на определена позиция.
- **Save() –** метод, който презаписва информацията от инстанцията на MIFFile във файл.
- **SaveAs() –** метод, който записва информацията от инстанцията на MIFFile, като файл с определено име.
- **SaveGPS() –** метод, който записва в текстов файл суровите GPS съобщения, чрез които са начертани графични примитиви.
- **Select() –** метод, чрез който бива селектиран определен графичен обект.
- **Zoom() –** метод, чрез който се осъществява ефект на лупа.

MIFData вграден клас:

- **mMapSize** – размер на правоъгълна област (в географски координати), която се изобразява.
- **mObjects** - списък с графични обекти (MIF обекти).

Останалите полета и методи на класа MIFData са с поведение подобно на досега описаните и няма нужда от повторно описание.

Класът MIDData е втория вграден клас във MIFFile и моделира достъпа до MID файла. Той е по-прост от MIF файла (продиктувано от това, че в общия случай е CVS (Comma Value Separated)). Всеки ред е представен, като списък от стойности, а списъка с редовете е представен чрез следната член променлива:

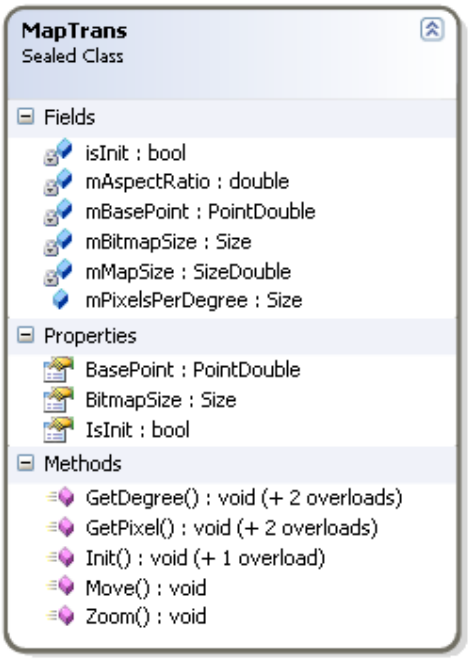
- **mRows** - инстанционна член променлива, която представлява списък от редове, като всеки ред е списък от стойности.
- **This[index]** – е предефиниран, за да може достъпа до отделните редове от файла да се осъществява във вида (<инстанция на MIDData >[<номер на ред>]).

Останалите методи на MIDData не се различават съществено от гореописаните.

3.4 Управление на изобразяваните обекти

За да се осъществи работата с географските данни (координатите на точките описващи графичните примитиви от MIF файла и получените координати от GPS приемника) и преобразуването на географски координати към пиксели от екрана е проектиран следния клас:

(3.13) клас MapTrans



Всякакви операции изискващи преобразуване на градус към пиксел или обратно преминават през този клас. Всички методи и полета на класа са статични за да се осигури бързодействие и класа е запечатан (невъзможно е да се наследи). По долу е описана неговата функционалност.

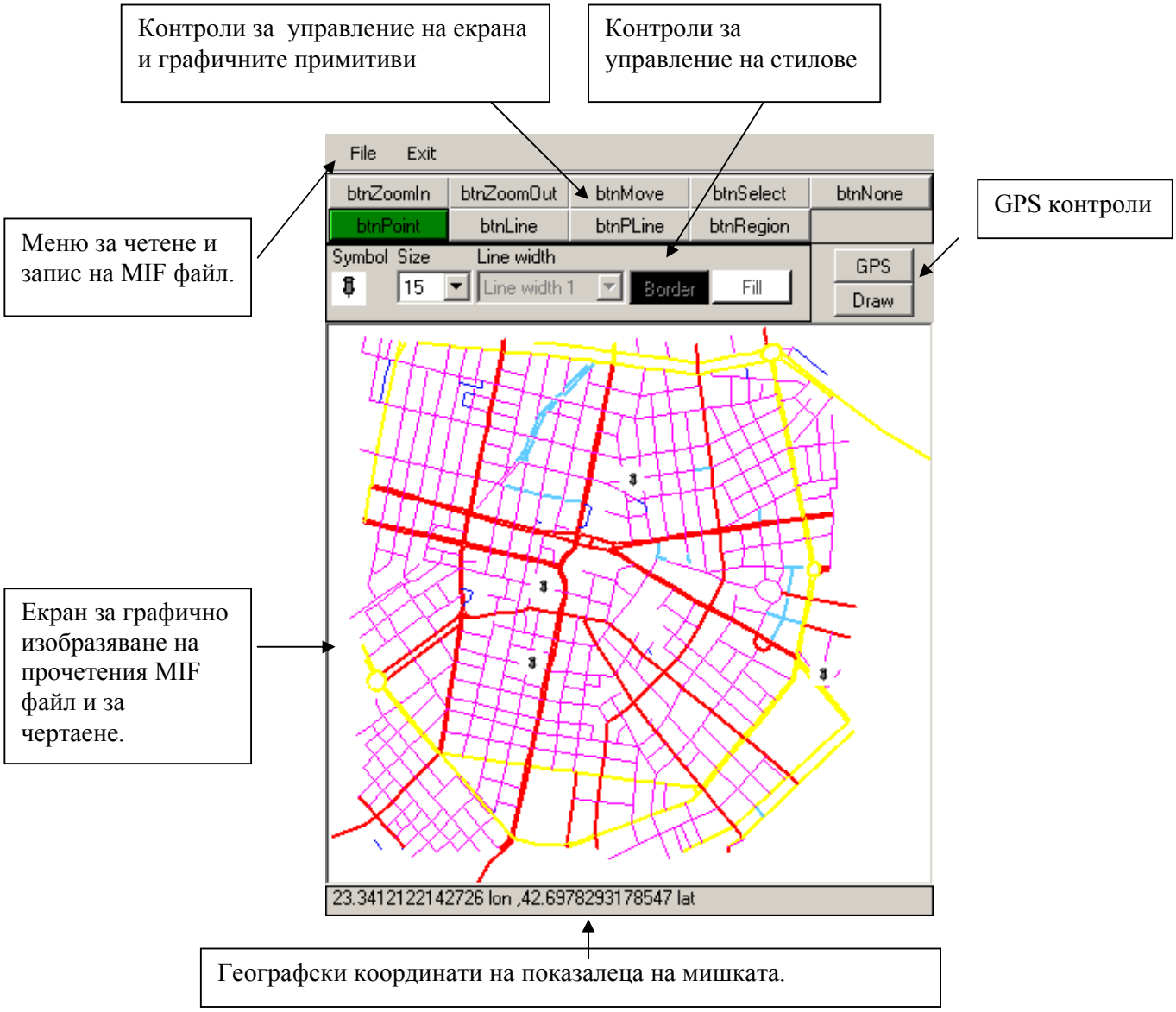
- **isInit** – булева статична променлива, която показва дали класа е инициализиран успешно.
- **mAspectRatio** – статична променлива, която съхранява отношението на максималната изобраявана географска ширина и максималната изобразявана географска дължина.
- **mBasePoint** - съхранява централната точка, спрямо която се извършва всякакво изчертаване.
- **mBitmapSize** – размера на областта за чертаене (в пиксели)
- **mMapSize** - размера на правоъгълната област от геогравски координати, която ще се изобразява.
- **mPixelsPerDegree** - съхранява отношението пиксел/градус.

- **GetDegree() –** метод, който връща географски координати от екранни координати.
- **GetPixel() –** метод, който връща екранни координати от географки такива.
- **Init() –** методът, който инициализира класа и вдига флага IsInit.
- **Move() –** извършва необходимите пресмятания по преместване на изображението.
- **Zoom() –** извършва необходимите пресмятания по реализирането на ефекта на лупата.

Реализирането на такъв клас чрез, който централизирано да се извършват всякакви манипулации и преобразувания от градус към пиксел (и обратно) от едно място осигурява възможност за всякакви по-късни подобрения и разширявания на функционалността, а и не на последно място отделя логигата по-смятането на координати от останалата логика и потребителски интерфейс. Не се дава възможност за наследяване на класа, защото няма да има особено голяма полза от това.

3.5 Дизайн на графичен потребителски интерфейс (GUI).

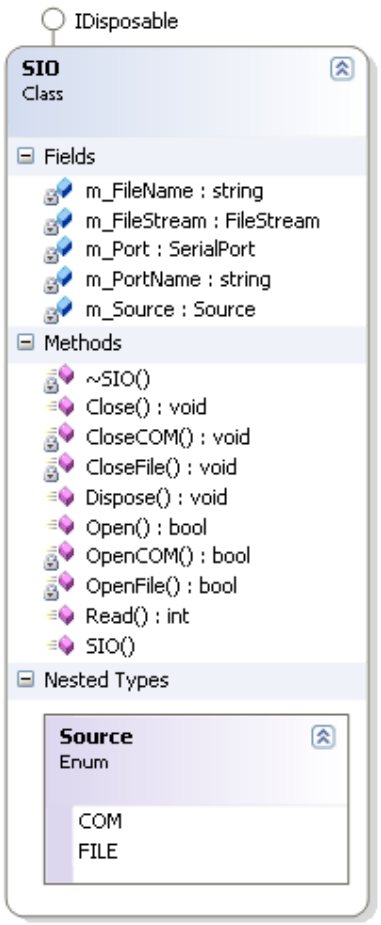
Основната задача на потребителския интерфейс е да осигури лесно и удобно средство за управление и работа с реализираната функционалност. За настоящата разработка е необходимо да се реализират четири основни компоненти на потребителския интерфейс – за чете и запис на MIF файл, инструменти за определяне на графичния примитив, който ще се изобразява на екрана, инструменти за стиловете на графичните примитиви, и изобразяване на примитиви чрез GPS управление. На следващата фигура е показан прототип на работния екран на приложението.



3.6 Осигуряване на абстрактно ниво на четене на серийни данни

Създадена е възможност лсесно и бързо да се променя източника на NMEA съобщеия. Класа SIO (Serial Input Output) е замислен да скрива (капсулира) достъпа до източика на съобщения (сериен порт или файл). По този начин NMEA парсера не се интересува дали чете от сериен порт или файл. На схемата е показан класа.

(3.14) SIO клас



- **mFileName** – абсолютно име на файл или порт, от който ще се чете.
- **m_FileStream** – инстанционна променлива за четене от файлов поток.
- **m_Port** – инстанционна променлива за четене от сериен порт.
- **m_PortName** – име на серийния порт.
- **mSource** – източник на съобщенията (COM или FILE)

SIO реализира интерфейса Idisposable. По-важни методи и свойства са :

- **SIO()** –конструира обект от клас SIO за четене от порт или файл.
- **Open()** – метод, който отговаря за четене от порт или файл (в зависимост от m_Source).
- **Read()** – метод, който чете байтове от съответния източник.

- **Close() –** затваря манипулатора към източника за четене.

С формалното описание на последния клас завършва дефинирането на основните градивни елементи (класове) на модула. С оглед постигане на една по–отворена функционалност в заключителната част са добавени някои възгледи и виждания за подобрение на дизайна.

За софтуерната реализация са използвани следните технологии **Visual Studio 2003 – C#, .NET Framework, .NET Compact Framework**. Решението за използването на тези технологии е съобразено и с първоначалното условие модулът да работи върху Windows CE платформа на мобилно устройство.

Към реализацията е подходено по следния начин. Първо са имплементирани класовете и функционалността за четене и запис на MapInfo MIF/MID данни, тъй като това е основната функционалност. След това са имплементирани методите за изчертаване на прочетената информация на екрана. В последствие е реализирана функционалността по прочитането на NMEA съобщенията от GPS приемника и тяхното интерпретиране. Като последен етап от реализацията е имплементирането на потребителския интерфейс и реализирането на функционалността по изчертаване на графичните примитиви с и без посредничеството на GPS.

Този ред на имплементация може да бъде и друг, но водещото съобръжение е да се реализират поетапно сравнително независими подзадачи, след което да се интегрират за решението на първоначално поставения проблем.

4 Тестове и анализ на получени резултати

При тественето на приложението водещи критерии са времевите и пространствените му характеристики. Основните следени показатели са следните:

- Време за зареждане на MIF/MID файл (t_{load}) и необходима оперативна памет за съхранението на прочетената информация.
- Време на пречертаване на екрана – t_{draw}.
- Време за парсване на NMEA съобщение от GPS приемника - t_{NMEA}.

Тестовете са проведени на мобилно устройство MobileMapper CE показано на фигурата във фирма DATECS.

(4.1) Мобилно устройство, на което са провеждани тестовете.

MobileMapper CE основни характеристики на устройството

- 20 MB оперативна памет свободна за приложни програми.
- 200MH процесорна мощ.
- Microsoft Windows CE .NET 4.2
- ARM920T based processor

След първоначалните тестове са получени следните стойности за наблюдаваните времена при прочитане на mif файл с големина от 130 KB :

(4.2) Отчетени времена при първоначални тестове

	t_{load}	t_{draw}	t_{NMEA}
ms	~9500	~1700	~1

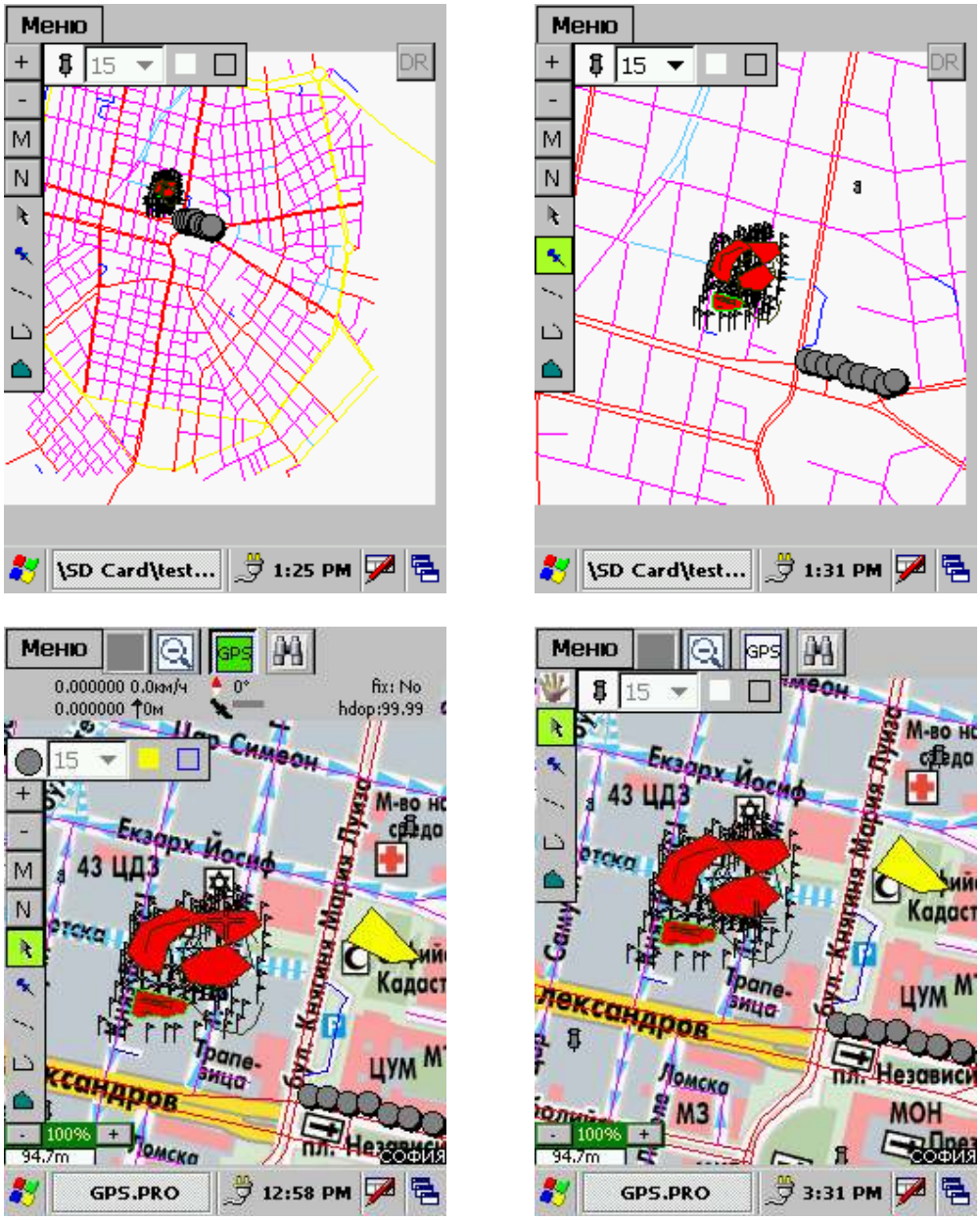
Първите две времена отразени в таблицата се оказват незадоволителни. Една от причините за големите им стойности (освен хардуерните ограничения на устройството) е неоптимизирания програмен код при парсването на mif файла (t_{load}) и при изчертаването на заложените графични примитиви в него (t_{draw}). След проучване и анализ на този код се установява, че най-тесните са честото и излишно констуиране и конкатениране на низове, които както е известно в .NET средата за изпълнение са динамични неизменни (immutable) обекти и работата с тях е изключително бавна. Тясно място се оказва и многократното констуиране на обекти, което довежда до по-честото принуждаване на системата да се самопочиства (garbage collect) и до фрагментиране на паметта. След отстраняването на тесните места и направените оптимизации горната таблица изглежда по следния начин:

(4.3) Отчетени времена след направени оптимизации

	t_{load}	t_{draw}	t_{NMEA}
ms	~4800	~900	~1

След направените тестове е започнат процеса на интегриране на модула в професионално GPS приложение. В процеса са наложени някои промени в потребителския интерфейс на приложението, но като цяло функционалната логика е запазена. На фигурата е показана част от променения изглед на приложението стартирано на мобилното устройство, както и след интегриране в професионално приложение.

(4.4) Екран на приложението върху мобилното устройство преди и след интегриране с професионално GPS приложение.



5 Заключение

Представения модул в настоящата дипломна работа реализира следните основни функционалности:

1. Четене и запис на MIF/MID данни.
2. Графично изобразвяване на MIF/MID данни според спецификациите на MapInfo.
3. Представяне на прочетените данни във вид на слой, който може да бъде редактиран и направените промени съхранени.
4. Чертаене на графични изображения ръчно чрез основни графични примитиви (точка, линия, полилиния и регион).
5. Добавена е функционалност за анализиране на GPS данни и автоматично чертаене посредством получената информация за текущата позиция.
6. Реализираните функционалности са интегрирани в професионално GPS приложение на фирма DATECS.

Чрез реализиране на посочените точки първоначално поставената задача е изпълнена.

Основните направления, по които може да се продължи да се работи са няколко основни:

- подобряване бързодействието при четене и парсване на данните, както и при изчертаването на графичните примитиви.
- добавяне на поддръжка на други графични примитиви като дъга (arc), елипса (ellipse), колекция (collection), които се поддържат от MapInfo .
- визуализиране и управление на множество слоеве(layers) от MIF/MID изображения.
- добавяне на функционалност за търсене на графичен обект по зададена атрибутна (MID) информация.
- подобряване на GPS функционалността.

6 Използвана литература

[1] Global Positioning System: Theory and Practice (1997)
by B. Hofmann-Wellenhof (Author), Herbert Lichtenegger (Author), James Collins (Author), Springer-Wien New York.

[2] www.nmea.org **- страница на NMEA (National Marine Electronic Association). Съдържа основна информация за GPS съобщенията от NMEA протокола.**

[3] www.mapinfo.com **- страница на MapInfo. Съдържа описание на основните функционалности на "MapInfo Professonal".**

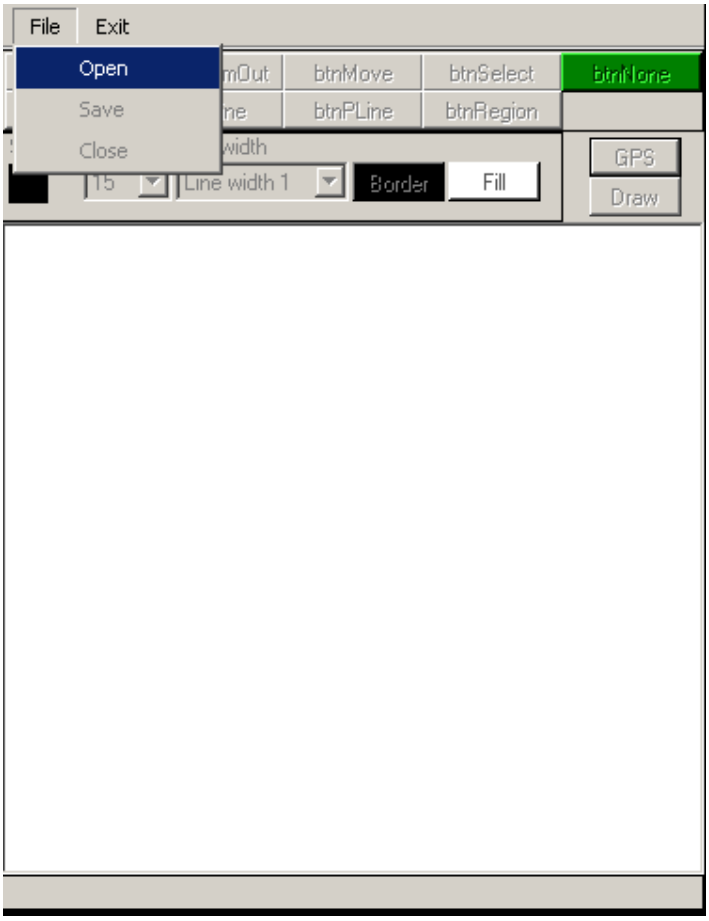
[4] http://www.directionsmag.com/mapinfo-l/mif/AppJ.pdf **- спецификация за формат (MIF/MID) на MapInfo за обмен на данни.**

[5] http://www.visualgps.net/white papers.htm **- дизайн на NMEA парсер.**

7 Приложение I

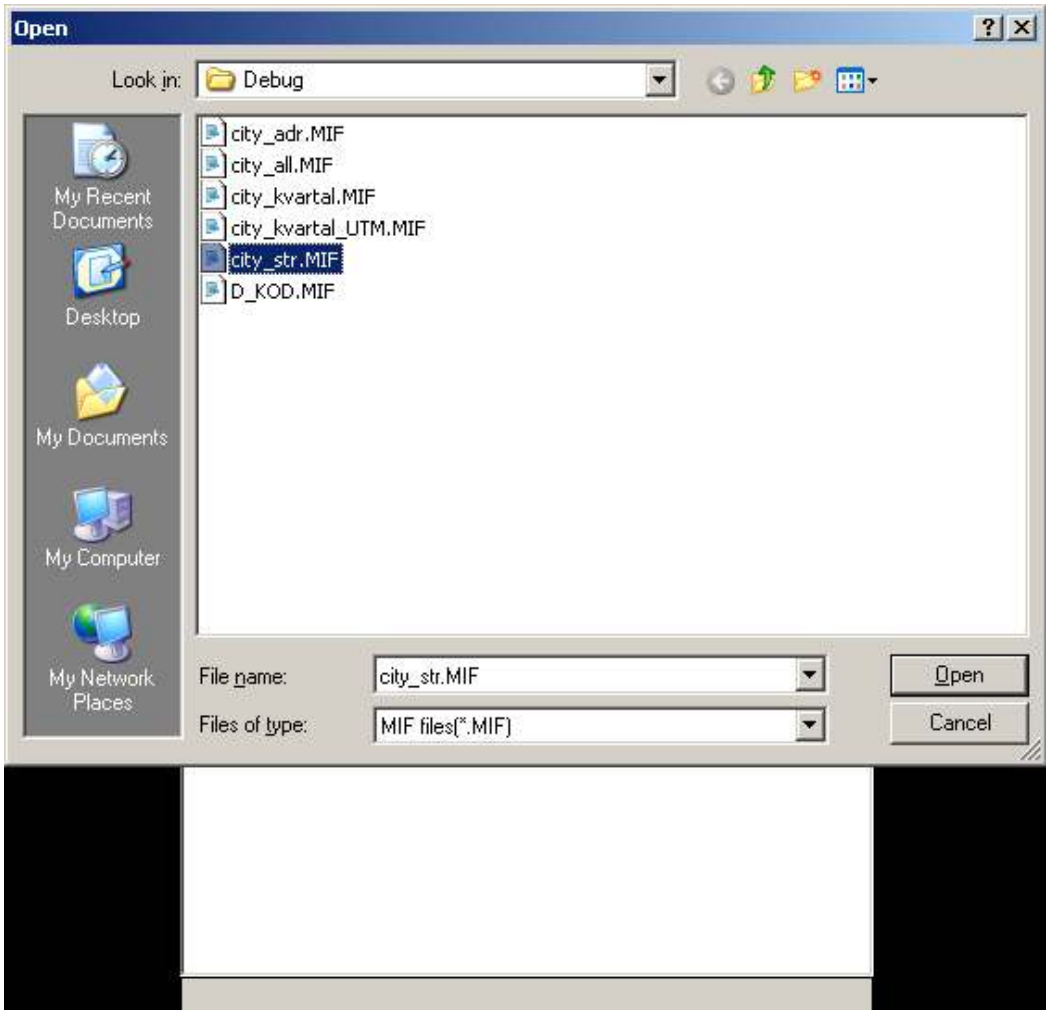
Екрани описващи реализираната функционалност в модула.

(7.1) Отваряне на MIF файл.

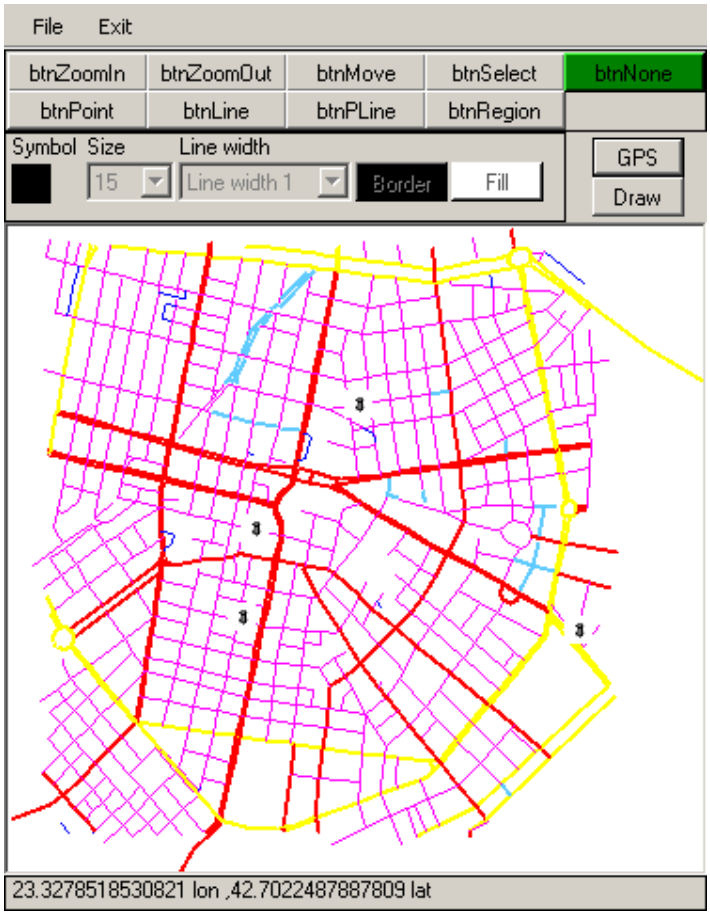


От главното меню се управлява зареждането, запазването и затварянето на MIF/MID данни от /във файл.

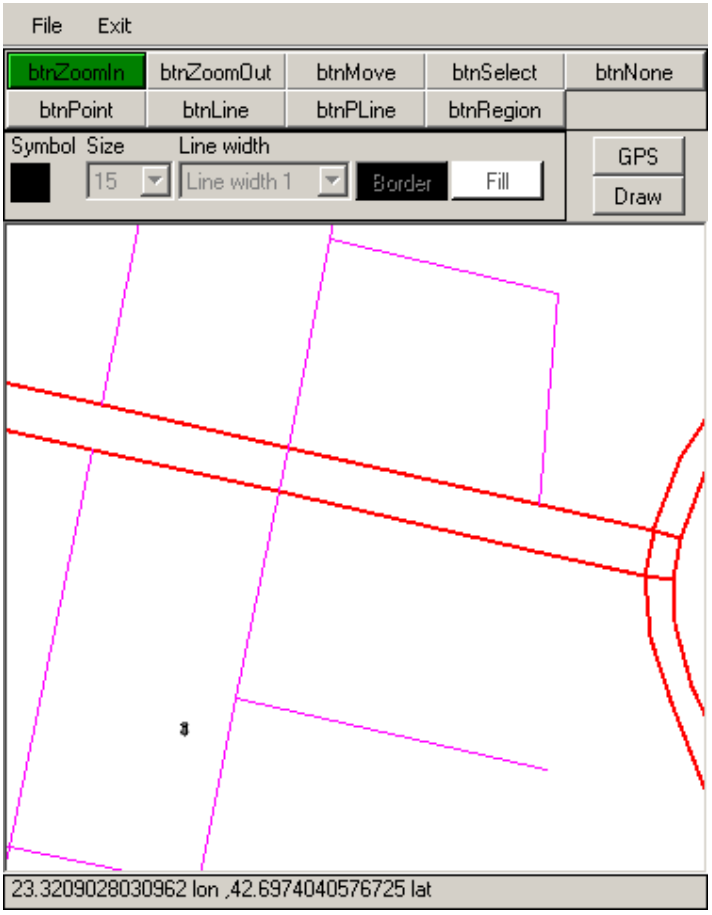
(7.2) Диалог за избор на файл, с който ще се работи.



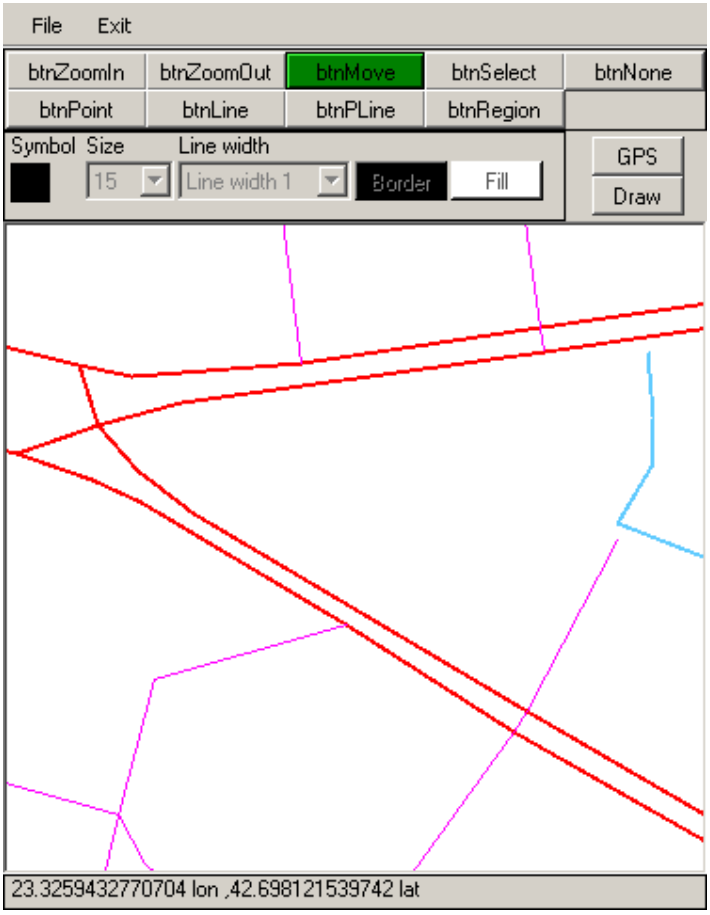
(7.3) Изобразен MIF файл след успешното отваряне.



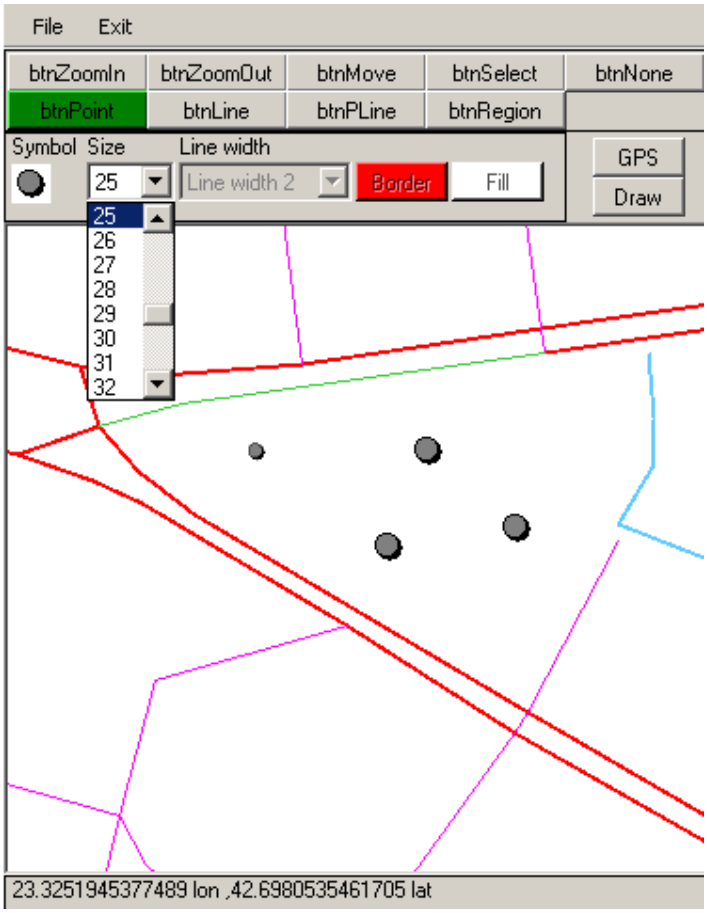
(7.4) Zoomin/Zommout функционалност.



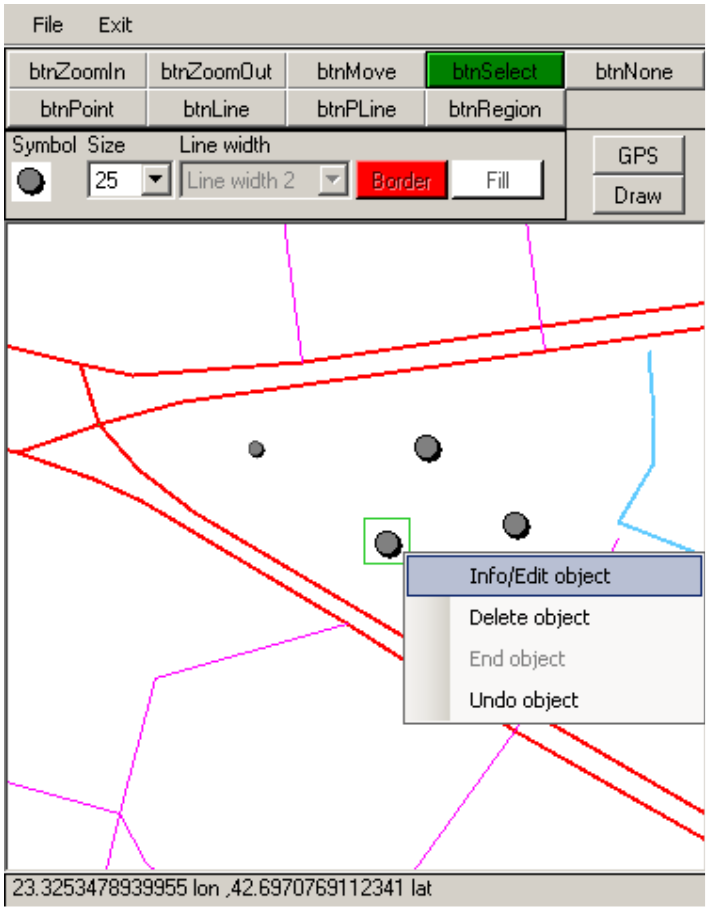
(7.5) Move (преместване на картата) функционалност.



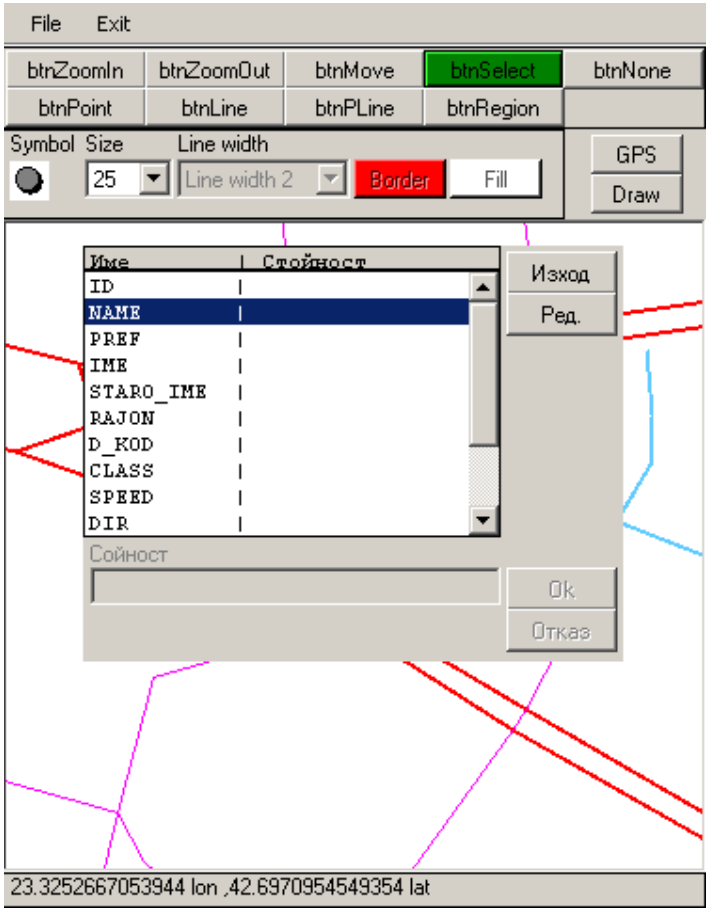
(7.6) Поставяне на символ (точка) върху картата.



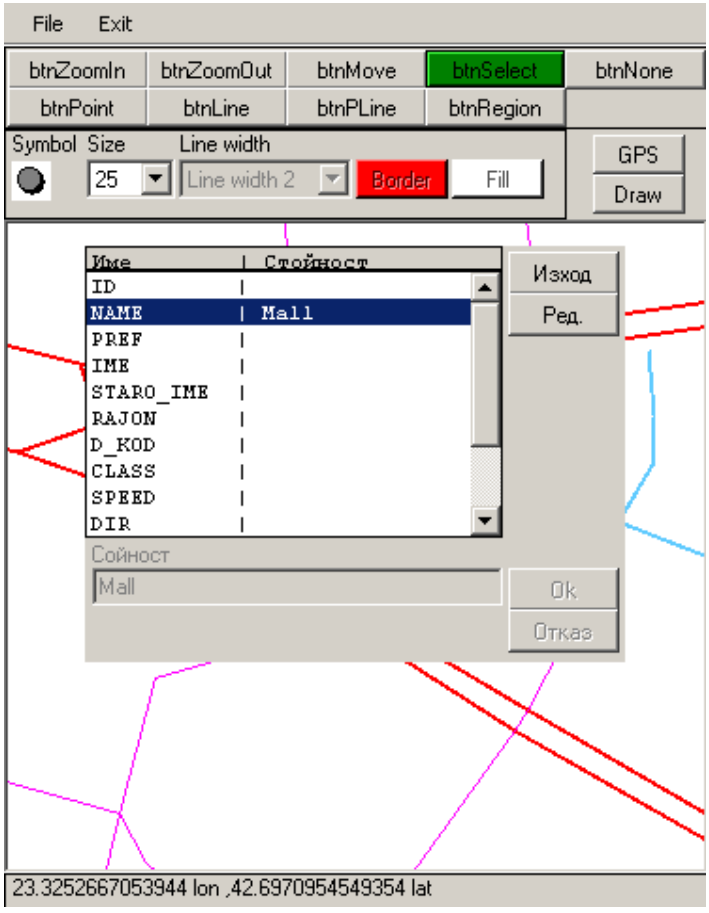
(7.7) Селектиране на обект и извикване на форма за редактиране на атрибутната информация за обекта.



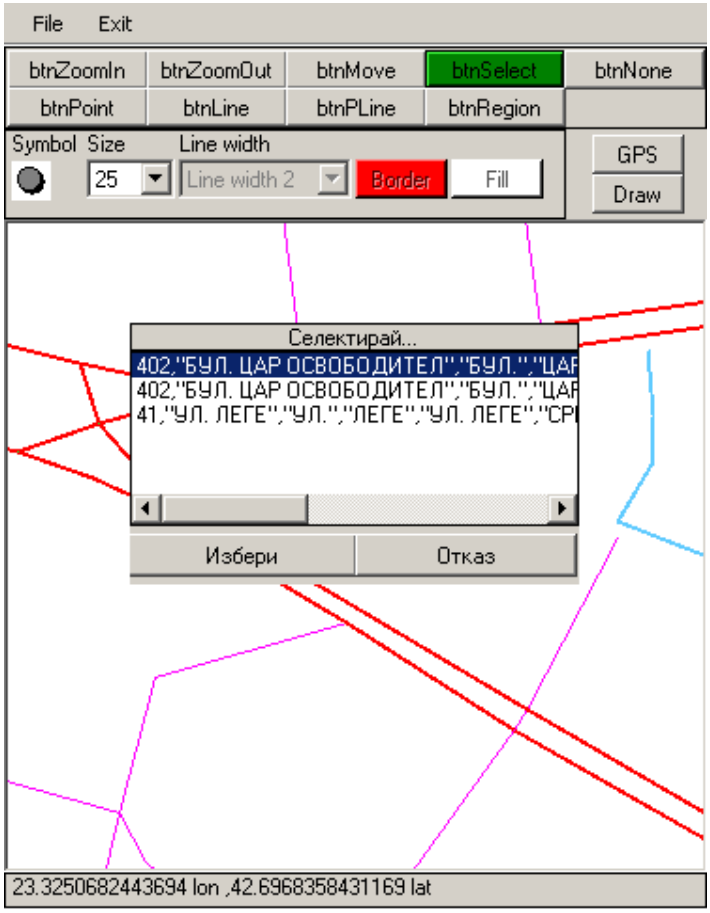
(7.8-1) Форма за редактиране на атрибутната информация на обекта.



(7.8-2) Форма за редактиране на атрибутната информация на обекта.



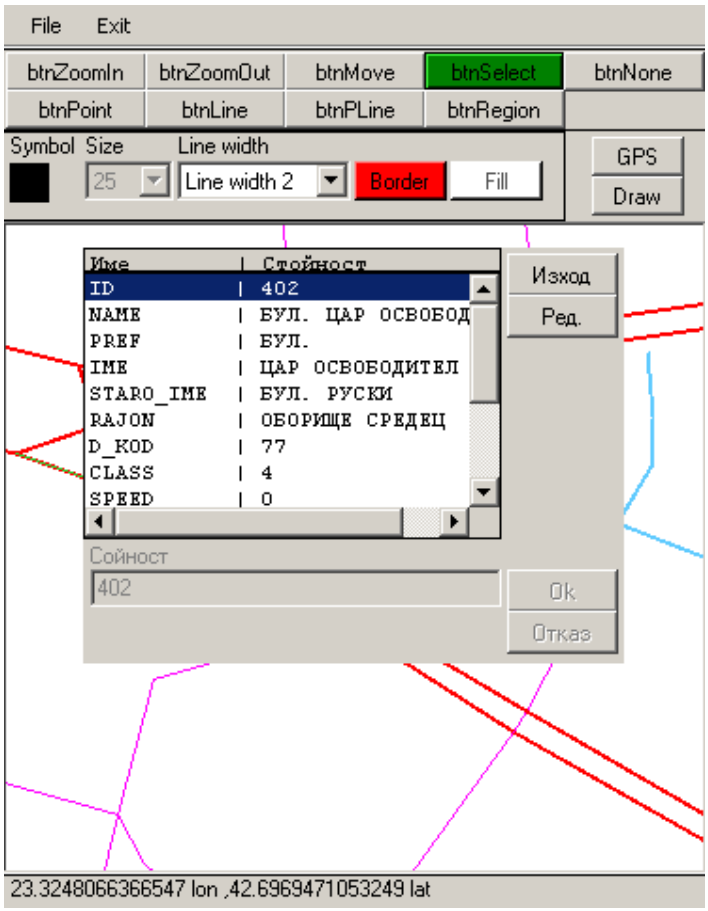
(7.9) Форма за селектиране на обект измежду множество от обекти.



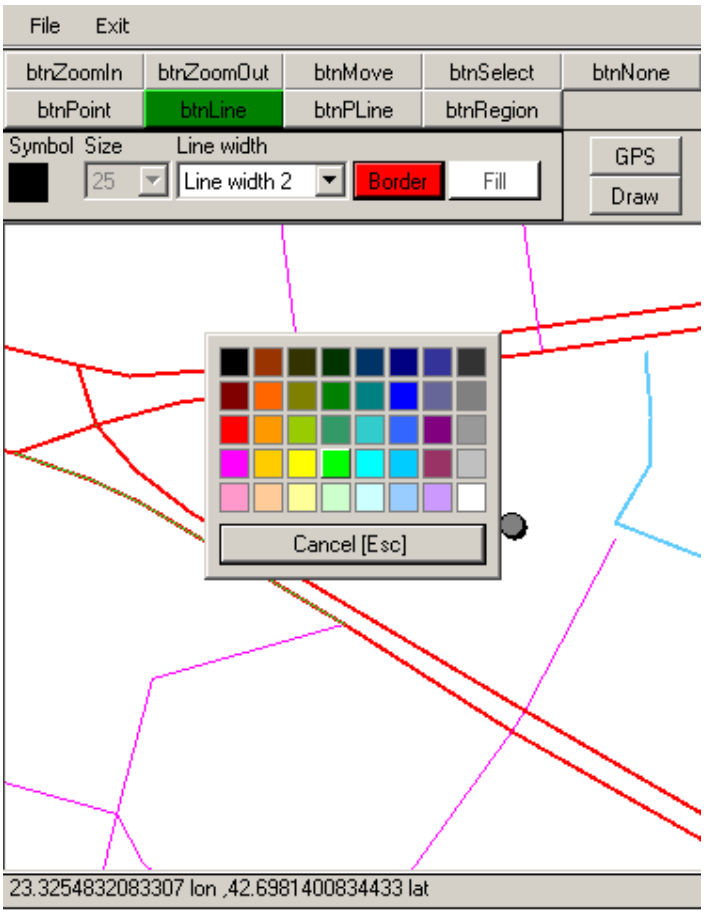
Когато потребителят се опита да селектира обект, но няколко обекта се намират на разстояние от позицията на натискане на бутона на мишката по-малко от предварително дефинирано разстояние тогава програмата не може сама да реши кой обект трябва да бъде селектиран и предлага

списъка с възможостите на потребителя, за да вземе решение.

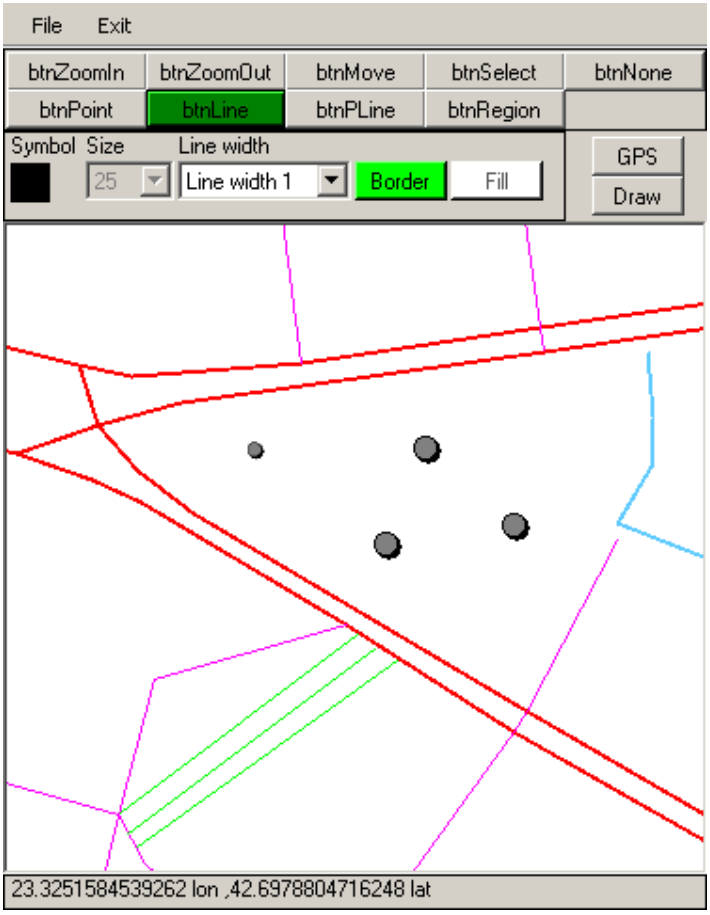
(8.0) Инфо/Редакция на селектирания обект.



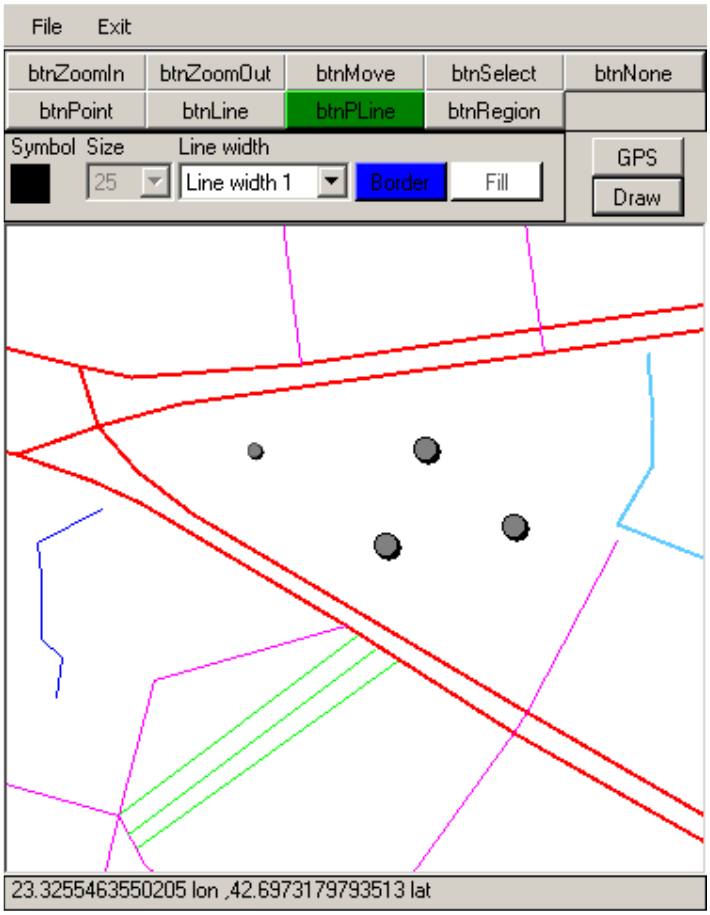
(8.0) Избор на цвят за рисуване.



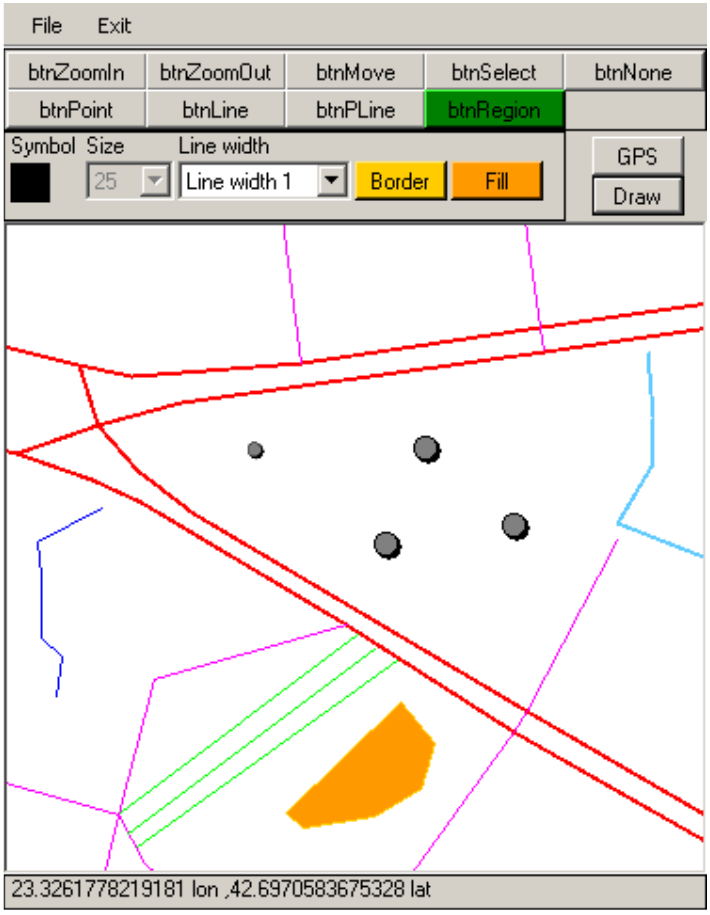
(8.1) Изчертаване на линия.



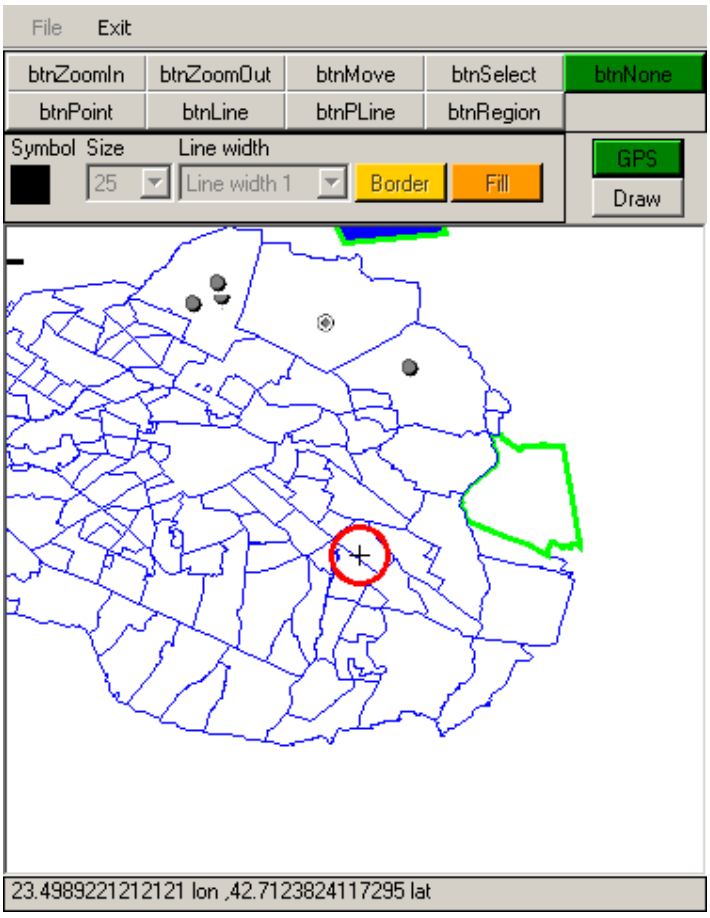
(8.2) Изчертаване на полилиния.



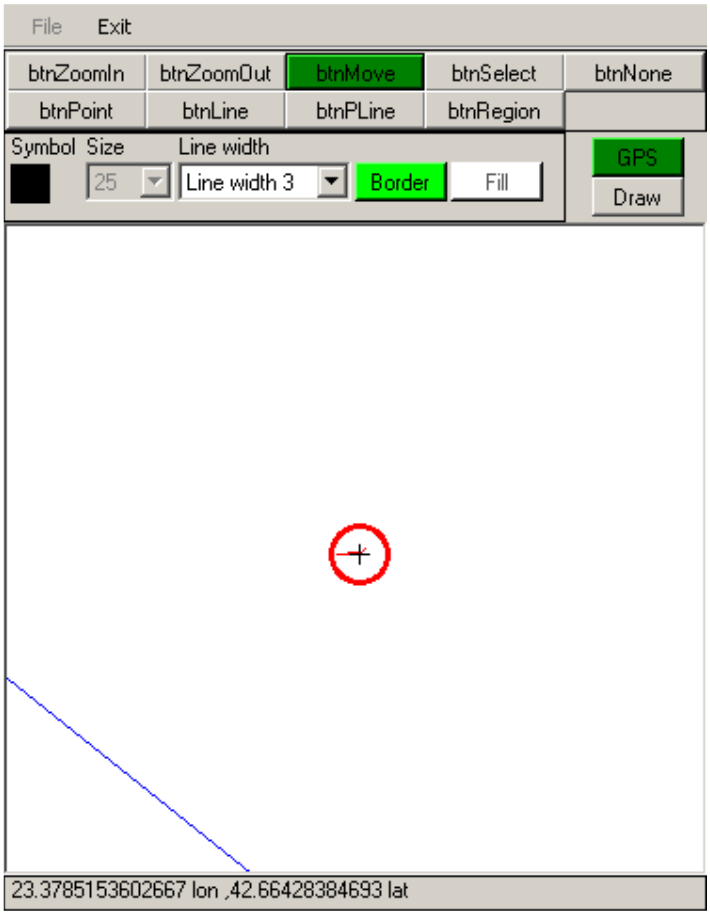
(8.3) Изчертаване на регион.



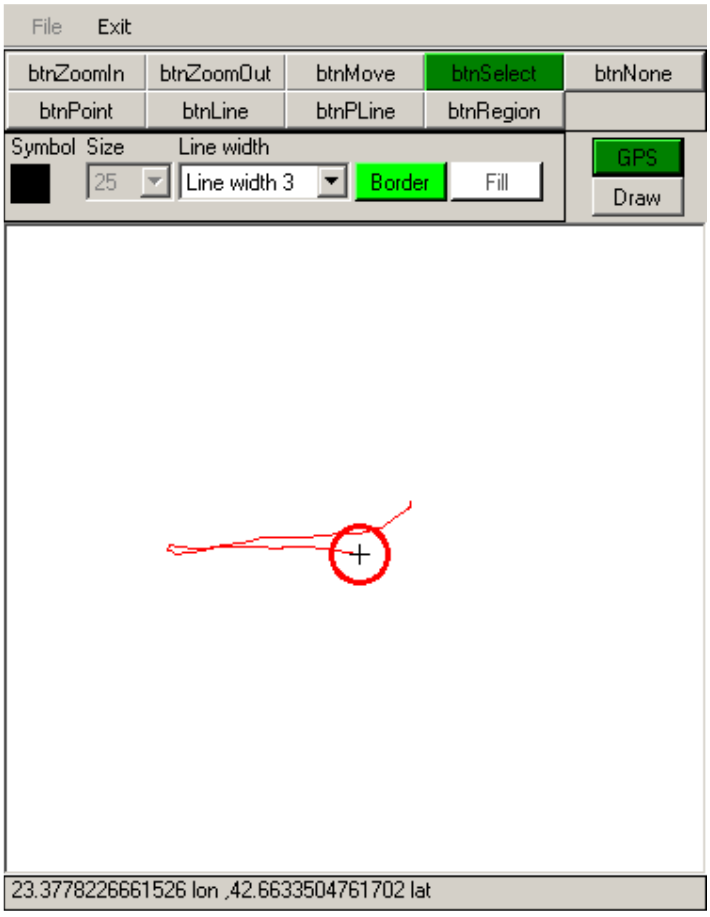
(8.4) Акривиране на GPS функционалност.



(8.5) Изчертаване чрез GPS.



(8.6) Инфо/Редакция на селектирания обект.



(8.7) Запис на редактираната информация.