SAGA : Système Automatisé de Gestion des Aquifères Interface Utilisateur

Ingles, J.

CEMAGREF

GROUPEMENT DE LYON
division hydrologie, hydraulique

S.A.G.A.

Système Automatisé
de Gestion des Aquifères

Interface Utilisateur

Par Jérôme Ingles
DUT informatique
Juin 1989

REMERCIEMENTS

Je tiens à remercier tous ceux qui ont permis de mener à bien ce stage :

- Mr. CHASTAN, chef de la division hydrologie-hydraulique de Lyon pour m'avoir accueilli dans son équipe.

- Mr. GIVONE, mon chef de stage, pour son aide et ses conseils.

- Mrs. LEDUC et DURBEC pour leur contribution majeure à la définition de l'interface.

- Mr. BOUCLIER pour sa sympathie et ses nombreux conseils pratiques.

Cependant, je voudrais remercier dans son ensemble toute l'équipe informatique pour son chaleureux accueil durant ses deux mois.

RESUME

Les aquifères souterrains fournissent aux activités humaines une ressource de qualité qui n'est pas inépuisable.
La nécessité de gestion en temps réel de ces eaux souterraines a conduit le **CEMAGREF** à mettre à disposition des services techniques locaux, ses modèles mathématiques de simulation d'écoulements, sous une forme utilisable par un hydrogéologue non-numéricien et non-informaticien.

Le système **SAGA** *(Système automatisé de gestion des aquifères)* associe ainsi au code numérique WATASI, un logiciel de représentation cartographique et un interface utilisateur de haut niveau.

Le but de mon stage était de réaliser l'analyse et la programmation de cet interface utilisateur qui utilise les concepts et les fonctions du logiciel WINDOWS.

INTRODUCTION

J'ai effectué mon stage de fin d'étude à la division hydrologie/hydraulique du C E M A G R E F de Lyon. Le sujet du stage s'articulait avec celui effectué l'année dernière par un autre étudiant de l'I U T.

Mon prédécesseur a réalisé l'interface graphique de **SAGA** (*Système Automatisé de Gestion des Aquifères*) afin de pouvoir représenter graphiquement l'état d'un aquifère une fois introduits les différents scénarios hydrauliques possibles. Mon travail consistait à écrire l'interface utilisateur du modèle mathématique de simulation d'écoulement afin qu'un utilisateur non-initié puisse modifier aisément et clairement les différents paramètres entrant en jeu.

1. LE CEMAGREF

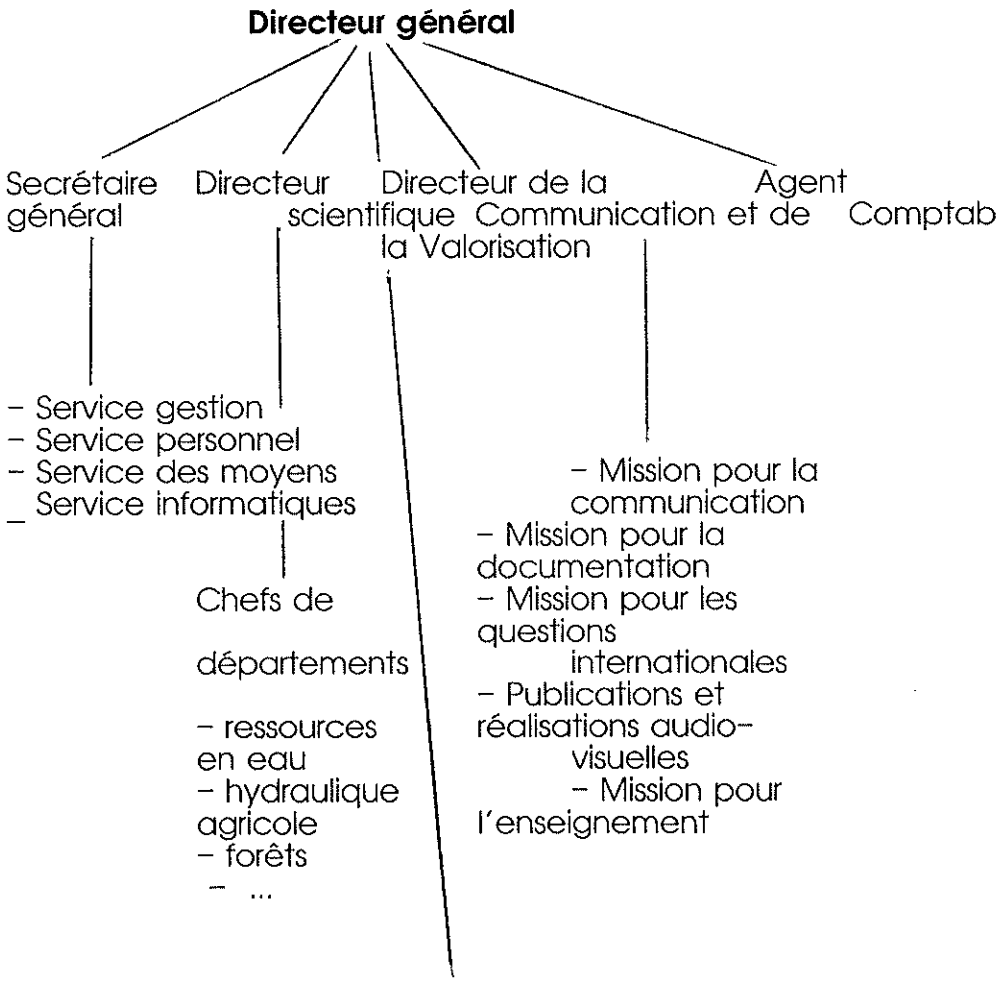
1. Le CEMAGREF

1.1. PRESENTATION GENERALE

Le **CEMAGREF** est le *CEntre National du Machinisme Agricole, du Génie Rural, des Eaux et Forêts*. C'est un établissement à caractère scientifique et technologique sous la tutelle conjointe des ministères de la recherche et de l'agriculture.

Il emploie 950 agents, dont 420 scientifiques, répartis en 10 groupements: ANTONY(siège), AIX, BORDEAUX,CLERMONT-FERRAND, GRENOBLE, LYON, MONTPELLIER, NOGENT, RENNES ,TOULOUSE.
Il conduit des recherches, des expertises, des expérimentations et des essais dans différents domaines tel que : l'eau, les risques naturels et technologiques, les zones montagneuses et défavorisées, les forêts mais aussi les équipements pour l'agriculture et les industries agro-alimentaires, la production et l'économie agricole.

L'organigramme général du **CEMAGREF** est le suivant :



Groupement et divisions

1.2. PRESENTATION DU GROUPEMENT DE LYON

Parmi les 10 groupements nationaux du **CEMAGREF**, le groupement de Lyon s'intéresse aux ressources en eau.
Il est constitué de deux divisions :

La division Qualité des eaux, pêche et pisciculture :

Elle étudie l'ensemble des problèmes de gestion du milieu aquatique d'eau douce. Ses activités de recherche, d'appui technique et de conseil aux collectivités locales et aux industriels, permettent d'améliorer principalement :

- la détection et la prévention des pollutions
- les procédés de traitement des eaux résiduaires
- les capacités de production biologique des milieux aquatiques.

Cette division regroupe une soixantaine de personnes.

La division Hydrologie-hydraulique :

Ses activités sont orientées vers l'évaluation et la gestion quantitative des ressources en eau, l'aménagement des cours d'eau et la protection contre les risques naturels et technologiques, dans un programme combinant recherche (avec des collaborations européennes), expertise et appui technique en France et à l'étranger.

Ses travaux consistent en :

- l'élaboration et l'exploitation de modèles numériques (analyse de données, prévision et prédétermination de crues, écoulements de rivières, écoulements et pollution des nappes souterraines, flux polluants, gestion d'ouvrages);
- le développement et l'exploitation de banques de données et de logiciels hydrologiques.
- le suivi de laboratoires de terrain.

Cette division, au sein de laquelle s'est déroulée mon stage, emploie une vingtaine de personnes.

2. LE PROJET SAGA

2. LE PROJET S A G A

2.1. POURQUOI ?

Les **aquifères** (nappes d'eau souterraines) sont de plus en plus sollicitées. En effet, de par leur bonne qualité, les nappes phréatiques sont systématiquement exploitées pour les besoins en eau potable de la population, besoins qui ne cessent de s'accroître.
De plus, quand la ressource est facilement accessible, l'utilisation agricole et industrielle des aquifères (par l'intermédiaire de puits ou de forages de faible profondeur) se généralise, et ceci d'autant plus quand les spéculations agricoles ou l'activité industrielle s'y prêtent. (cultures irriguées de plus en plus fréquentes).

Les régions côtières du Languedoc-Roussillon doivent ainsi faire face à ce problème d'autant plus important que :

- La ressource en eau est limitée dans ces régions méditerranéennes.

- L'utilisation en alimentation humaine est très importante en zone côtière l'été à cause du tourisme.

- L'existence d'un coin salé (partie d'eau salée provenant de la mer située sous les aquifères) risquant, s'il est atteint, de rendre inutilisable l'aquifère.

- Les aménagements de cours d'eau à buts multiples (les protections contre les crues, les aménagements touristiques, etc...) influent sur les réalimentations (ou recharges) des aquifères par le réseau hydrographique de surface.

- Les cultures irriguées sont de plus en plus fréquentes ; elles tendent à remplacer les vignes autrefois très nombreuses et ainsi augmentées les besoins en eau.

- Le développement industriel pèse sur la ressource en eau.

Dans ce contexte difficile, l'utilisation de la ressource en eau dont le stock est limité, donne lieu à des arbitrages parfois sévères entre les différents usagers.
La création d'un outil de gestion, en temps réel, des aquifères devient une nécessité et apporte une aide objective à une politique de réglementation de l'usage des eaux souterraines.

Par exemple, les autorisations pour de nouveaux prélèvements dans un aquifère, ou une intensification de prélèvements existants, ne pourront être accordées qu'après une étude précise de leur influence sur l'état général de l'aquifère. La simulation de divers scénarios climatologiques, parmi lesquels les plus sévères, et la prise en compte de la politique d'alimentation en eaux des populations permettront de conduire les arbitrages les plus efficaces.

Le système informatique SAGA propose d'étudier les conséquences de la sollicitation d'un aquifère, en traçant les cartes iso-valeurs des paramètres représentatifs de cet aquifère. Ces paramètres proviennent d'une simulation de l'état de la nappe par un modèle mathématique. Cette partie est opérationnelle et doit être complétée par un interface utilisateur.
L'interface utilisateur permettra la prise en compte par le modèle mathématique de toute modification réelle ou projetée de l'état de l'aquifère, ceci directement et sans codage de l'information.
Le système informatique complet, soit le couplage de l'interface utilisateur, du modèle mathématique et des modules de représentation graphiques, doit permettre une gestion en temps réel de la situation par un service technique local tel que le SRAE (*Service Régional d'Aménagement des Eaux*).

2.2. ORGANISATION DU SYSTEME

2.2.1. Organisation générale

Les principales fonctions du système sont les suivantes :

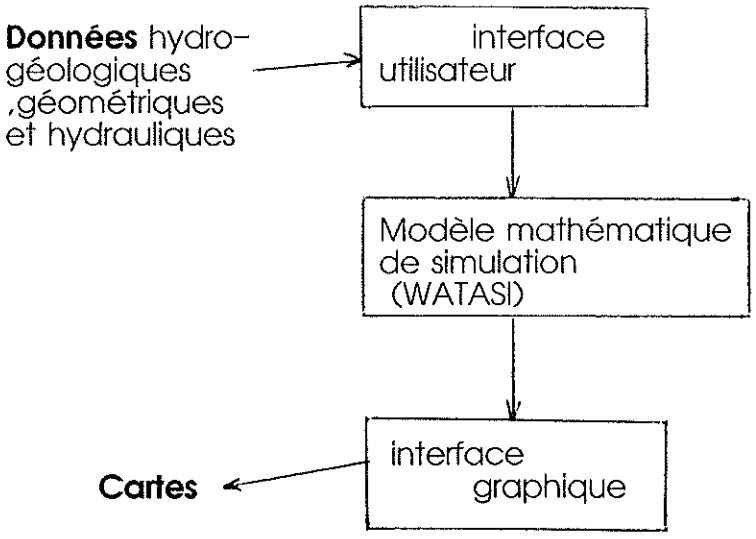
- Etre utilisable sur micro-ordinateur au niveau local.

- Prendre en compte les résultats d'un modèle mathématique (*WATASI*) qui simule le comportement d'un aquifère soumis à diverses sollicitations variables dans le temps et confrontées à divers scénarios climatologique.

- Etre facilement utilisable par un non-initié, qui doit pouvoir modifier aisément les divers scénarios climatologiques, de mettre à jour certains paramètres afin de piloter l'interface graphique.

- Représenter sous forme de cartes dessinées en temps réel l'évolution de l'aquifère (selon les différents paramètres). Ces cartes doivent être compréhensible et exploitables par un profane en hydrogéologie, même si leur interprétation en termes de projets d'aménagement ne sera réalisable qu'avec l'aide d'un spécialiste hydrogéologue.

On peut schématiser ainsi l'organisation du système :



2.2.2. Organisation interne

Le modèle mathématique, qui simule numériquement l'évolution d'un aquifère d'après différents scénarios, représente l'aquifère par différentes couches (correspondantes au couches géologiques) et un **maillage** à l'intérieur de chaque couche. Ainsi, à un aquifère donné correspond un certain nombre de **mailles**, elles-mêmes organisées en couche communicantes.

A chaque maille correspond des caractéristiques géométriques comme le numéro de couche à laquelle elle appartient, sa position (I,J) au sein de la couche, la longueur d'un de ses côtés (les mailles sont de tailles différentes mais toujours carrées).

De plus, à chacune des mailles est affectées des paramètres physiques, géométriques et hydrauliques appelés **attributs**.

Ces attributs sont au nombre de sept :

- **La côte du mur** Paramètres géométriques représentant la hauteur minimale
- **La côte du toit** et maximale de la maille

- **La conductivité** - Capacité du milieu poreux à laisse transiter de l'eau

- **L'emmagasinement** - Capacité du milieu à conserver de l'eau

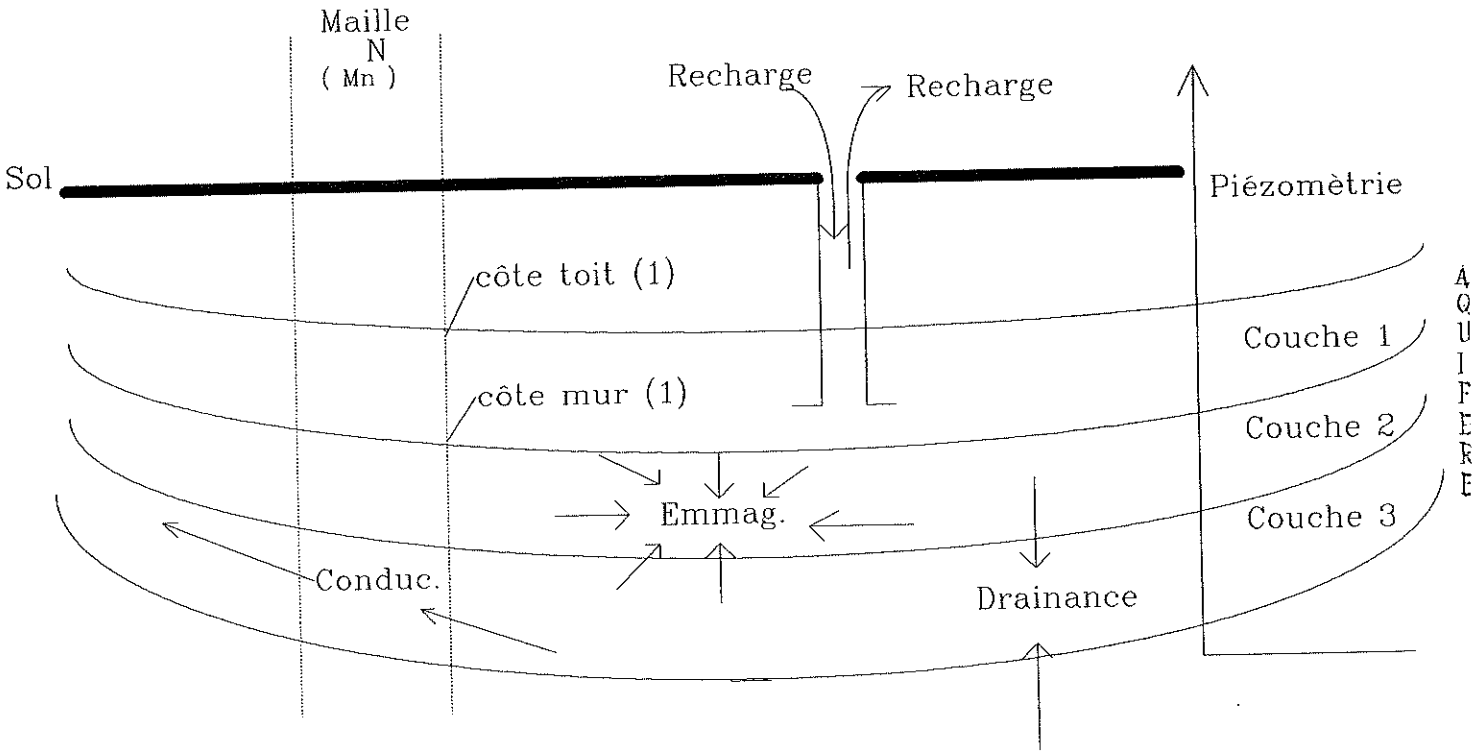
- **La drainance** - Qualifie les échanges entre deux couches

- **La recharge** - Caractérise les échanges avec le milieu extérieur à la nappe

- **La piézomètrie** - (initiale) Pression de l'eau à l'intérieur de la nappe

Ce sont les valeurs de ces sept attributs qui seront accessibles à l'interface utilisateur afin de pouvoir visualiser l'effet sur l'aquifère de divers scénarios.

Chaque maille sera considérée comme un objet (au sens des langages orientés objets) par l'interface, et les paramètres de ces objets seront les attributs.



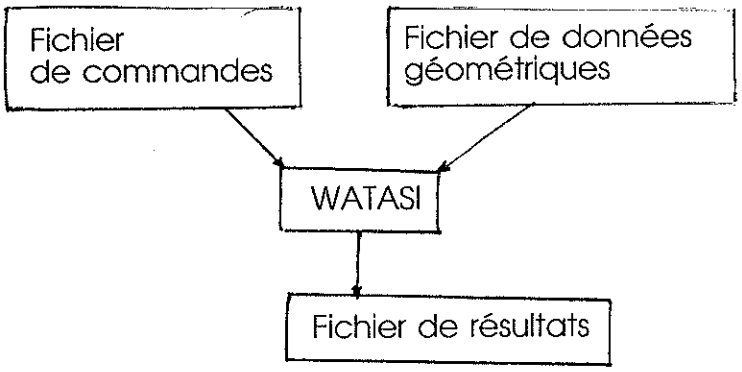
2.2.3. Les différents modules du système

Ils sont au nombre de trois :

- L'interface utilisateur (Voir chapitre 3)
- Le modèle mathématique WATASI
- L'interface graphique

Le modèle mathématique WATASI :

WATASI permet la modélisation de systèmes aquifères multi-couches connectés avec le réseau hydrographique (échanges nappes-rivières, émergence, débordements, etc...).
Ce programme FORTRAN écrit par un hydrogéologue - numéricien simule le fonctionnement hydrodynamique d'une nappe d'eau ; à partir de données (valeur des attributs de chacune des mailles) et de commandes il génère des résultats (**Piézomètrie finale**) :



Ce modèle constitue un outil de prévision de l'évolution d'un aquifère compte tenu des conditions météorologiques et d'aménagements prévus ou souhaités.
Cependant il n'est pas utilisable "tel quel" et n'est pas tout à fait déterministe en ce sens qu'il doit être au préalable **calé** par un expert en hydrogéologie. Ce calage est nécessaire car si, pour une nappe donnée, les résultats obtenus sont numériquement justes, ils ne correspondent pas forcément à la réalité physique du milieu étudié. Il est donc nécessaire de modifier les paramètres de calage jusqu'à que l'écart entre les résultats de la simulation et un état de référence soit minimal.

L'interface graphique :

Afin que l'utilisateur puisse, d'un seul coup d'oeil sur la carte, juger de l'influence de tel ou tel aménagement, il est apparu nécessaire de représenter à l'écran une carte géographique de la nappe.

Donc, dans un premier temps, il s'agit de représenter optionnellement les indications suivantes :

- Les communes (les noms,les limites)
- Le réseau hydrographique (les rivières,les étangs,la mer)
- Le réseau routier (les autoroutes,les nationales)
- Le maillage représentant la nappe (choix de la couche)

L'étape suivante consiste à superposer sur cette carte l'état d'un paramètre de la nappe. Tout en laissant la possibilité de visualiser indifféremment tous les attributs de la nappe (Drainance, Conductivité, Emmagasinement etc...), il apparait évident que le plus intéressant sera de visualiser l'état de la piézomètrie finale (résultat du modèle WATASI).

C'est sous la forme de courbes iso-valeurs que les paramètres sont représentés. En ce qui concerne la piézomètrie issue du modèle mathématique WATASI, le choix des couleurs s'est effectué en conséquence : vert pour un niveau satisfaisant de pression, rouge pour un niveau nul, jusqu'au violet pour un niveau négatif(en tout, une dizaine de couleurs)

2.3. ETAT DES LIEUX

SAGA est opérationnel sans interface utilisateur au SRAE de Montpellier sur la nappe astienne. (voir page suivante)

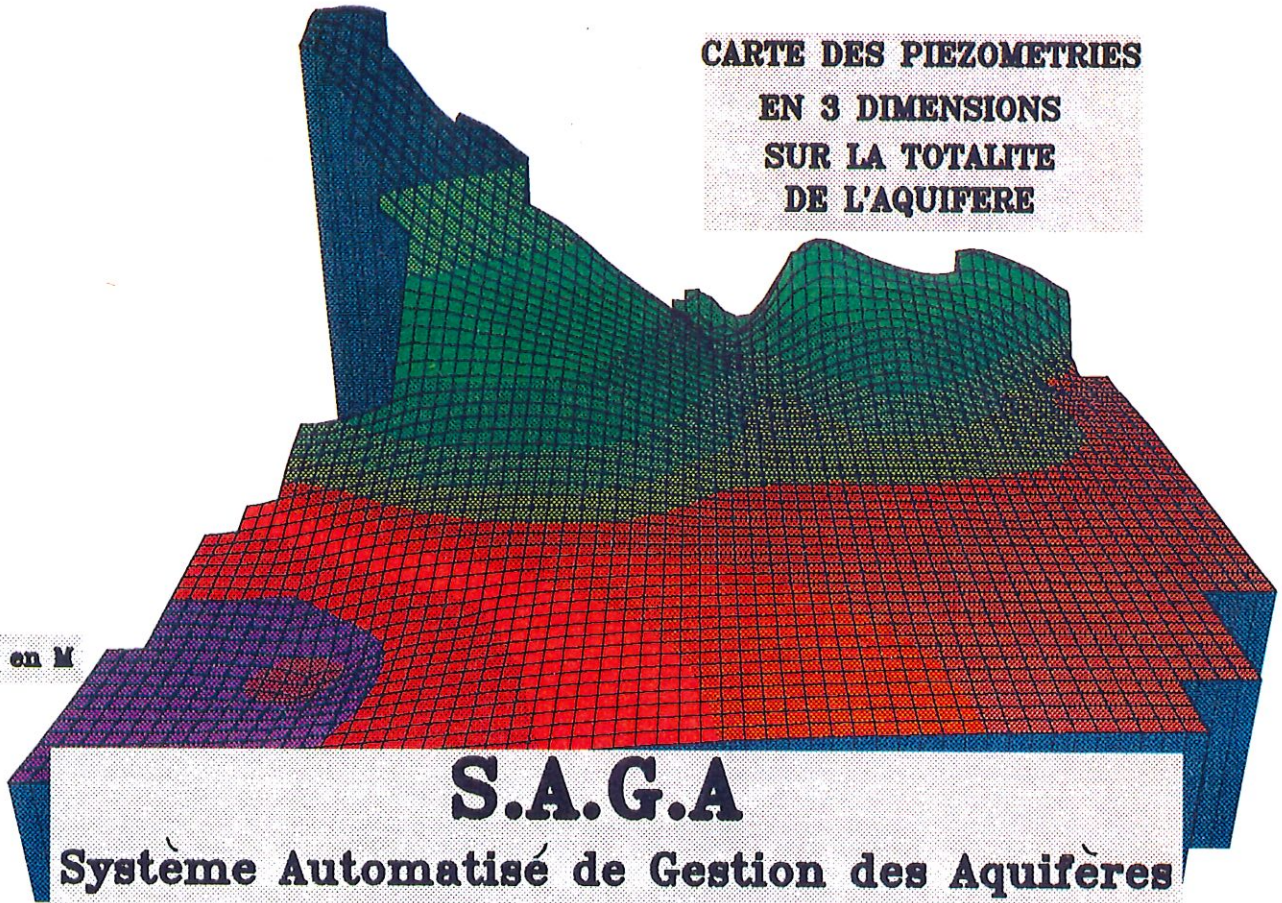
Son utilisation est cependant limitée par la complexité "informatique" qu'implique la mise à jour des fichiers de paramètres pour chaque nouvelle simulation.

CEMAGREF

FNDAE

GROUPEMENT DE LYON
Division Hydrologie, Hydraulique

CARTE DES PIEZOMETRIES
EN 3 DIMENSIONS
SUR LA TOTALITE
DE L'AQUIFERE



Profondeur en M

S.A.G.A

Système Automatisé de Gestion des Aquifères

Nappe de L'ASTIEN

3. L'INTERFACE UTILISATEUR

3. L'INTERFACE UTILISATEUR

3.1. POURQUOI ?

Le but de mon stage est donc la réalisation de l'interface utilisateur du système SAGA.

Cet interface doit permettre à un utilisateur sans compétences informatiques de maîtriser l'utilisation du système *SAGA*, en particulier de modifier à son gré les paramètres d'entrée du modèle mathématique pour simuler divers scénarios d'exploitation de l'aquifère. (ex : nouveau forage ou intensification d'un prélèvement déjà existant).

L'interface devra répondre à trois grandes fonctions :

- **Mise à jour des paramètres** : pouvoir ainsi modifier les scénarios climatologique, pouvoir supprimer ou rajouter un prélèvement dans la nappe (en temps réel).

- **Pilotage du modèle mathématique** (modification du fichier de données) **et de l'interface graphique**.

- **Etre accessible à des utilisateurs sans compétences informatiques** : son utilisation devra être facile et compréhensible, d'aspect clair et agréable.

Enfin, les modifications devront se faire sur la plus petite unité géométrique traitée, c'est à dire la maille (considérée comme l'objet de base du système). Mais les données géométriques et les paramètres communs à toutes les mailles ne pourront être modifiés sans contrôle, afin que l'utilisateur n'influe pas sur le calage du modèle mathématique.

3.2. COMMENT ?

3.2.1. Présentation du matériel

L'exploitation du système *SAGA* devra se faire sur un micro-ordinateur, dans un service technique local. Un micro-ordinateur de type **PS/2** a été utilisé pour le développement des softs.

Ce **PS/2** disposait des caractéristiques suivantes

- un micro-processeur 80286
- un coprocesseur arithmétique 80287
- un clavier AZERTY 102 touches
- deux lecteurs de disquettes 3 pouces ½ et 5 pouces ¼
- un disque dur d'une capacité de 20 méga-octets
- une carte graphique haute définition VGA
- un écran graphique VGA
- un système d'exploitation MS DOS (version 3.3)
- 640 kilo-octets de mémoire de base

Du fait du type même de l'application, le micro-ordinateur possède un écran graphique VGA essentiel pour la réalisation de l'interface graphique mais aussi pour la réalisation de l'interface utilisateur.

3.2.2. Présentation du logiciel

Depuis maintenant plusieurs années (apparition du MACINTOSH d'APPLE), les systèmes d'exploitations se sont dotés d'une interface élaborée utilisant le multi-fenêtrage et des menus déroulants. Plus récemment, IBM a conçu son nouveau OS/2 pour micro-ordinateur avec une interface utilisateur (*Presentation Manager*) qui utilise pleinement ce type de possibilités.

Des logiciels tels que WINDOWS ou GEM jouent déjà le rôle d'interfaces utilisateurs de MS-DOS sans être intégré réellement à celui-ci.
Le logiciel **WINDOWS** de *Microsoft* (version **2.0**), devenu le standard des intégrateurs graphiques sur PC, a été utilisé à la fois comme modèle et comme base opérationnelle pour la réalisation d'interface de *SAGA*.

WINDOWS est "composé" d'un noyau : *the Application Programming Interface* (**API**). L'API contient les fonctions, les structures de données, les types de données et les fichiers dont tout programme issu de WINDOWS a besoin.

Ainsi, une application peut tirer parti de bon nombre de caractéristiques générées par l'API. Parmi ces caractéristiques, citons :

- Partage de l'écran, de la mémoire, du clavier, et de la souris.

- Multi-tâche : plusieurs applications peuvent s'accomplir simultanément.

- Echange de données entre applications.o

Ces caractéristiques font apparaître la notion de **terminal virtuel**. En effet, le noyau WINDOWS (API) crée un terminal virtuel entre l'utilisateur et l'ordinateur.

De plus, WINDOWS possède une interface graphique : *the Graphics Device Interface* (**GDI**).qui contient de nombreuses fonctions graphiques. Ces ofonctions peuvent créer une grande variété de ligne, de texte, de formes et de couleurs.

En outre, le GDI est à même de générer des fenêtres, des menus, des boites de dialogues et de contrôles pouvant être utilisés dans diverses applications.
C'est ainsi que l'interface de *SAGA* est développée ; à partir d'un code source en langage C (version **5.1** du **C** de *Microsoft*) intégrant des fonctionnalités spécifiques de WINDOWS par appels à des procédures ou des primitives du *TOOL KIT*.

Mais le langage préférentiel des applications sous WINDOWS est le langage C. Le **C 5.1** de *Microsoft* a donc été choisi afin de développer l'application.

3.3. DESCRIPTION DE L'INTERFACE

L'interface développée est donc une application WINDOWS. A ce titre, sa présentation générale est standardisée. L'écran générique se compose de :

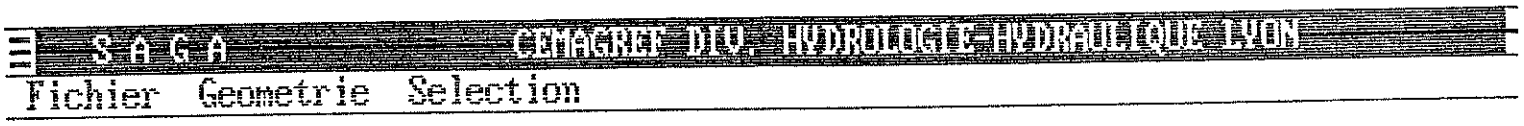
- Un titre général en partie supérieure

- Une "bande" contenant le nom des différents menus déroulants au-dessous du titre.

- Un icône en haut à droite permet la "mise en icône" de l'application, ou d'utiliser le plein écran.

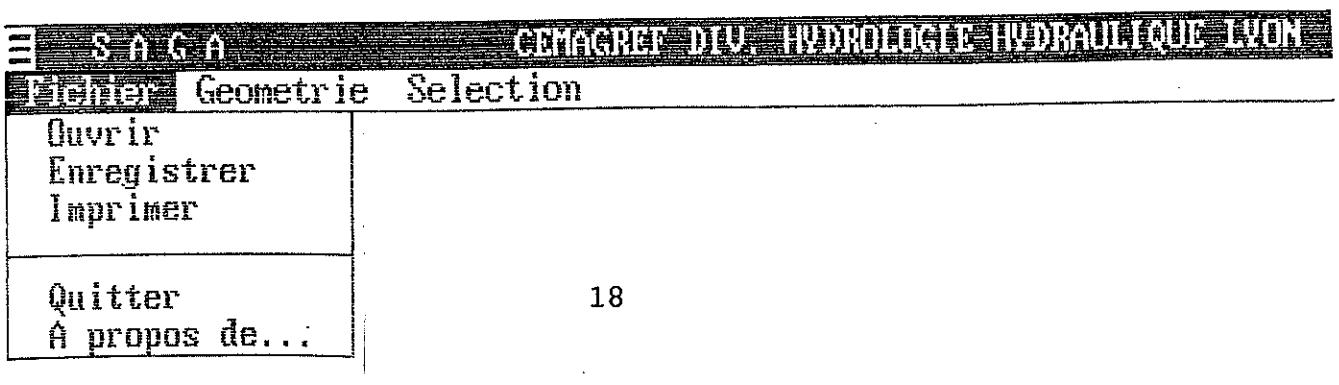
- Un icône en haut à gauche permet d'accéder à un menu système qui gère tout application WINDOWS.

Dans cet écran générique, se déroule l'application SAGA, elle même, qui s'éxecute à partir de la barre de menu.

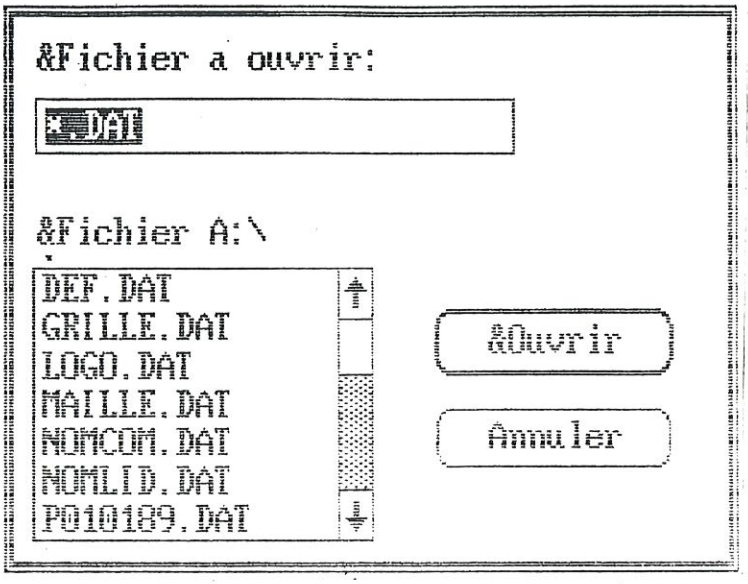


3.3.1. Le menu **FICHIER**

Le menu **FICHIER** concerne toutes les entrées-sorties. Il est constitué de cinq articles :



- L'item **OUVRIR** permet à l'utilisateur d'ouvrir un fichier de données concernant la géométrie des mailles (entraînant l'ouverture des fichiers concernant les valeurs des attributs de ces mailles). Si on clique sur cet item, une boite de dialogue apparait proposant la liste des fichiers (*.DAT) de la directory, on peut ouvrir un fichier soit en double-cliquant sur celui-ci dans la liste, soit en cliquant sur un fichier (pour le rendre actif dans la case "Fichier à ouvrir") puis en cliquant sur le bouton OUVRIR. Le bouton ANNULER permet de sortir de cette boite de dialogue sans avoir ouvert de fichiers.

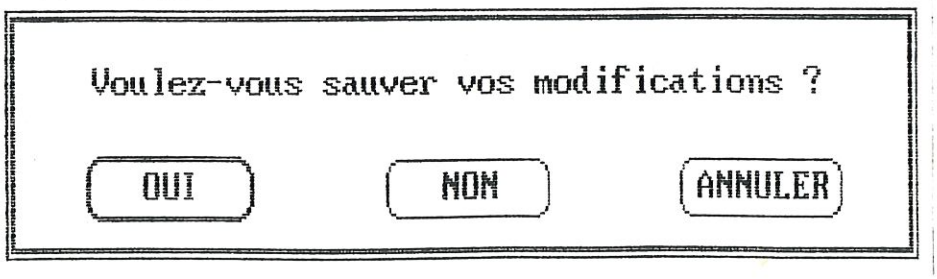


Si on ouvre un fichier de géométrie de mailles, ceci aura pour effet de faire dessiner à l'écran le squelette, avec une taille déterminée, de toutes les mailles de l'aquifère (par des rectangles à traits noirs) ainsi que la première couche de mailles (par des rectangles de couleurs). Ainsi, l'item OUVRIR est le point de départ de l'application ; si on a pas ouvert un fichier, les menus GEOMETRIE et SELECTION ne sont pas opérationnels.

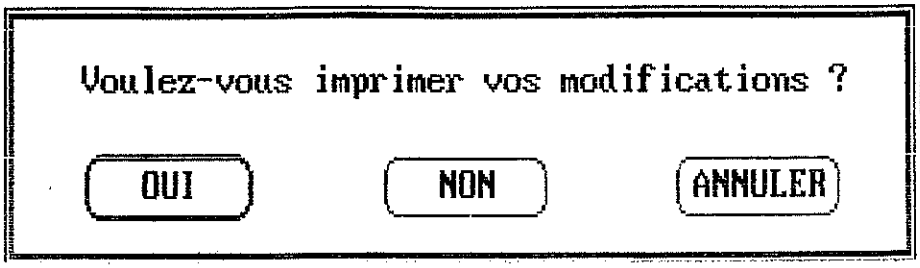
- L'item **ENREGISTRER** permet de sauver les modifications faites sur les attributs des mailles. Si on clique sur cet item, les fichiers de données des attributs sont reécrits.

- L'item **IMPRIMER** a la fonction d'imprimer la liste des modifications effectuées. En effet, l'utilisateur peut avoir besoin de savoir quelles modifications il a apportées aux attributs des mailles.

- L'item **QUITTER** permet de terminer la session de travail. Cependant, s'il y a eu des mises à jour depuis la dernière sauvegarde, une boite de dialogue apparait demandant si la sauvegarde est à effectuer :



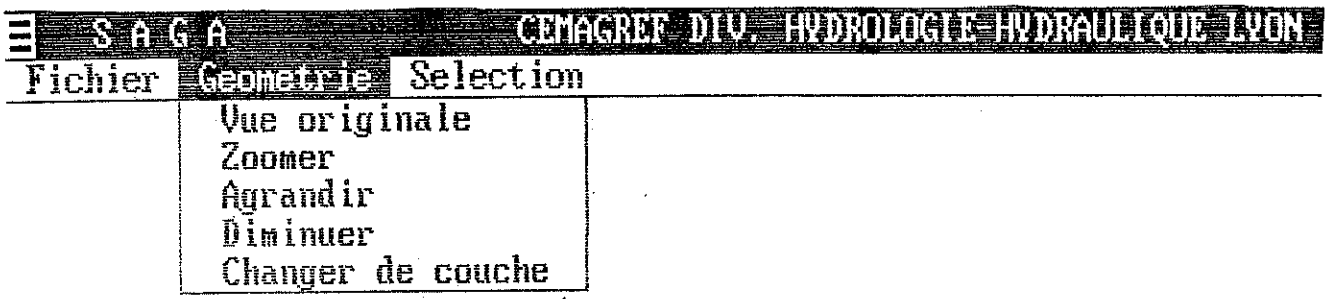
De même, si la liste des modifications n'a pas été imprimée, une boite de dialogue demandant cette impression apparait :



– L'item **A PROPOS DE...** fournit des renseignement quant au titre et à la version de l'application :

3.3.2. Le menu **GEOMETRIE**

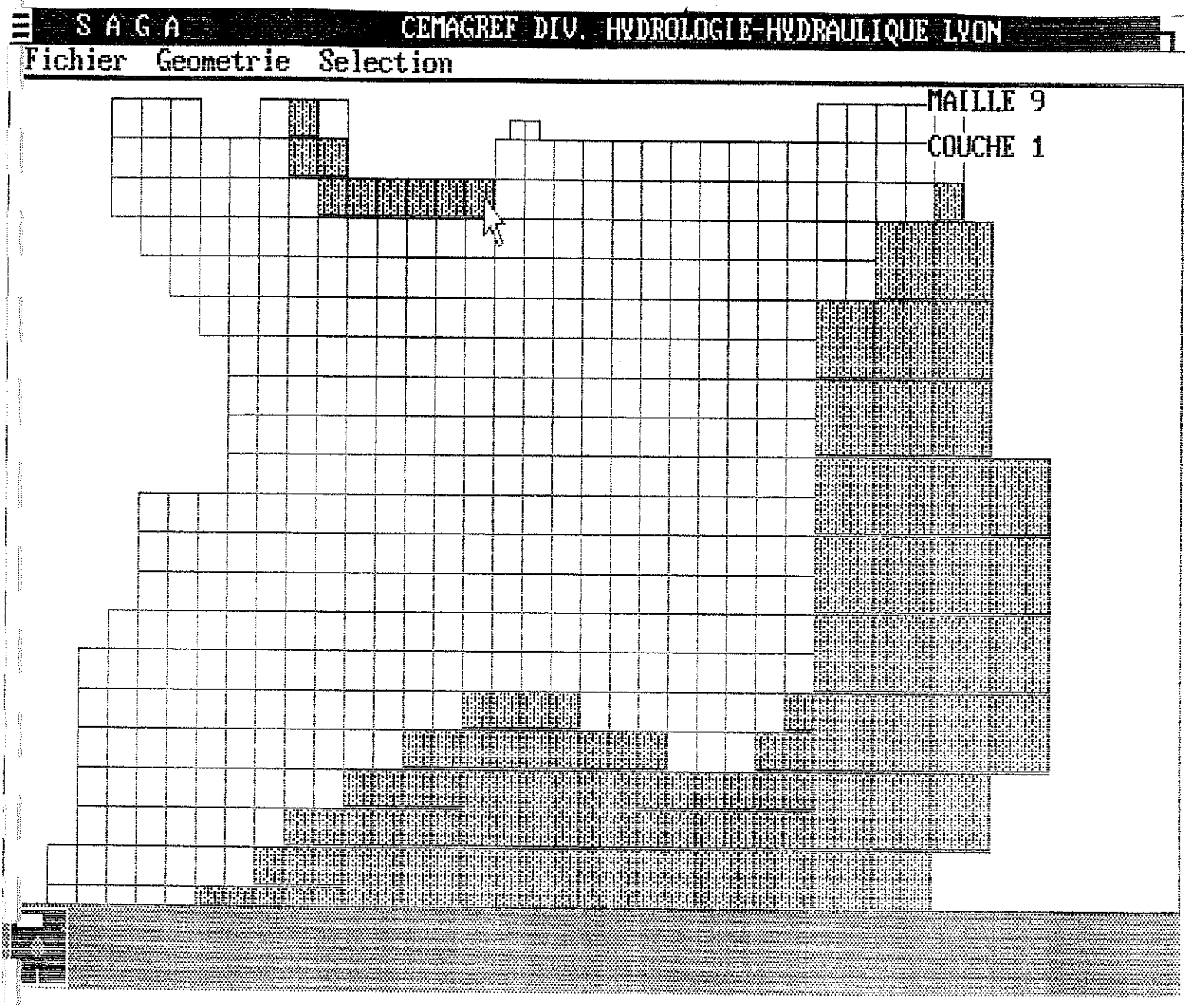
Le menu **GEOMETRIE** concerne toutes les opérations géométriques que l'on peut effectuer sur un dessin de mailles. Il est composé de cinq articles :



– L'item **VUE ORIGINALE** a pour effet de restituer la vue ORIGINALE du dessin des mailles. (Celui-ci peut en effet subir quelques modifications avec les articles suivants du menu GEOMETRIE).

– L'item **ZOOMER** permet à l'utilisateur d' agrandir à l'écran une région particulière du dessin de mailles. Si on clique sur cet item, le curseur de la souris change d' aspect jusqu'à que l'on ait choisi la région à zoomer (La flèche est remplacée par un rectangle).

– L'item **AGRANDIR** a la fonction d'agrandir la totalité du dessin de mailles se trouvant à l'écran.

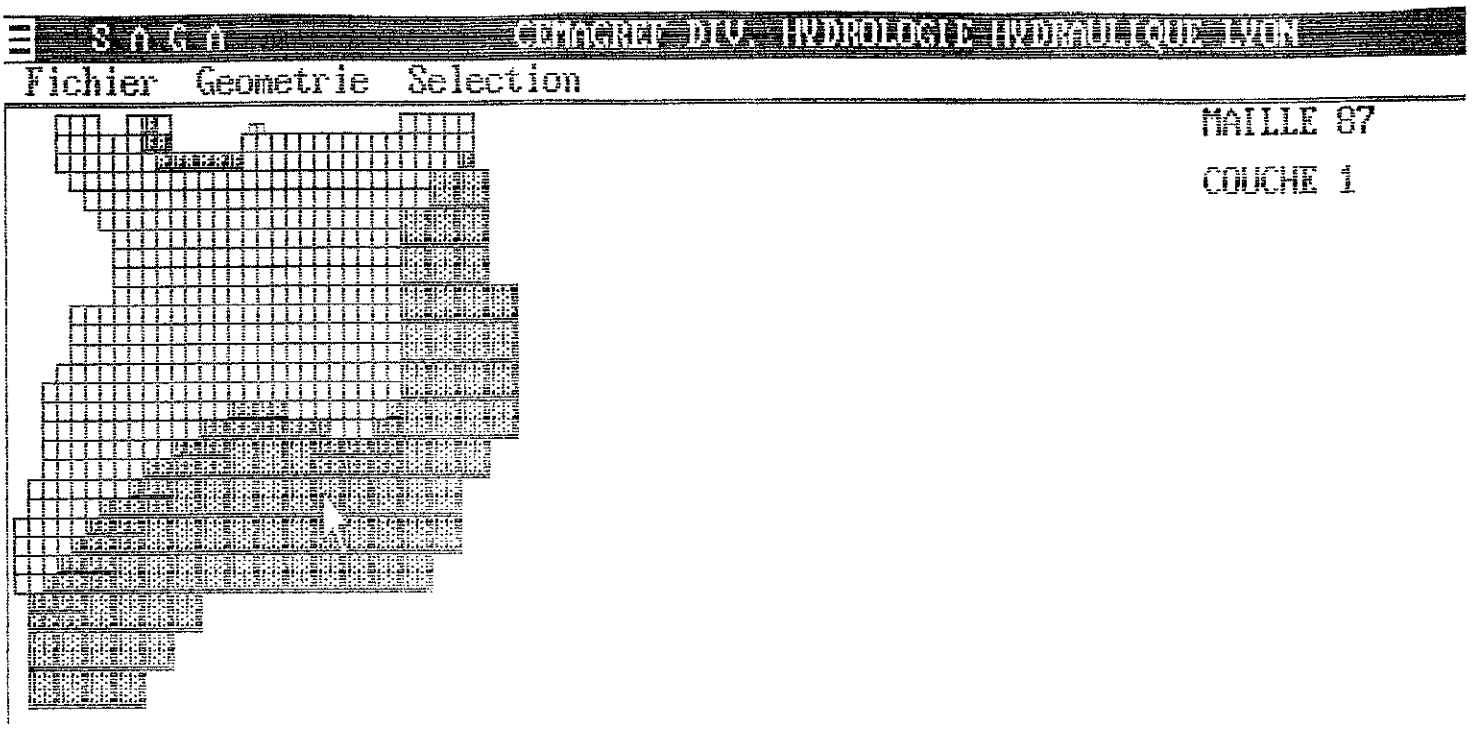


Windows
Application S A G A

Version 1.0

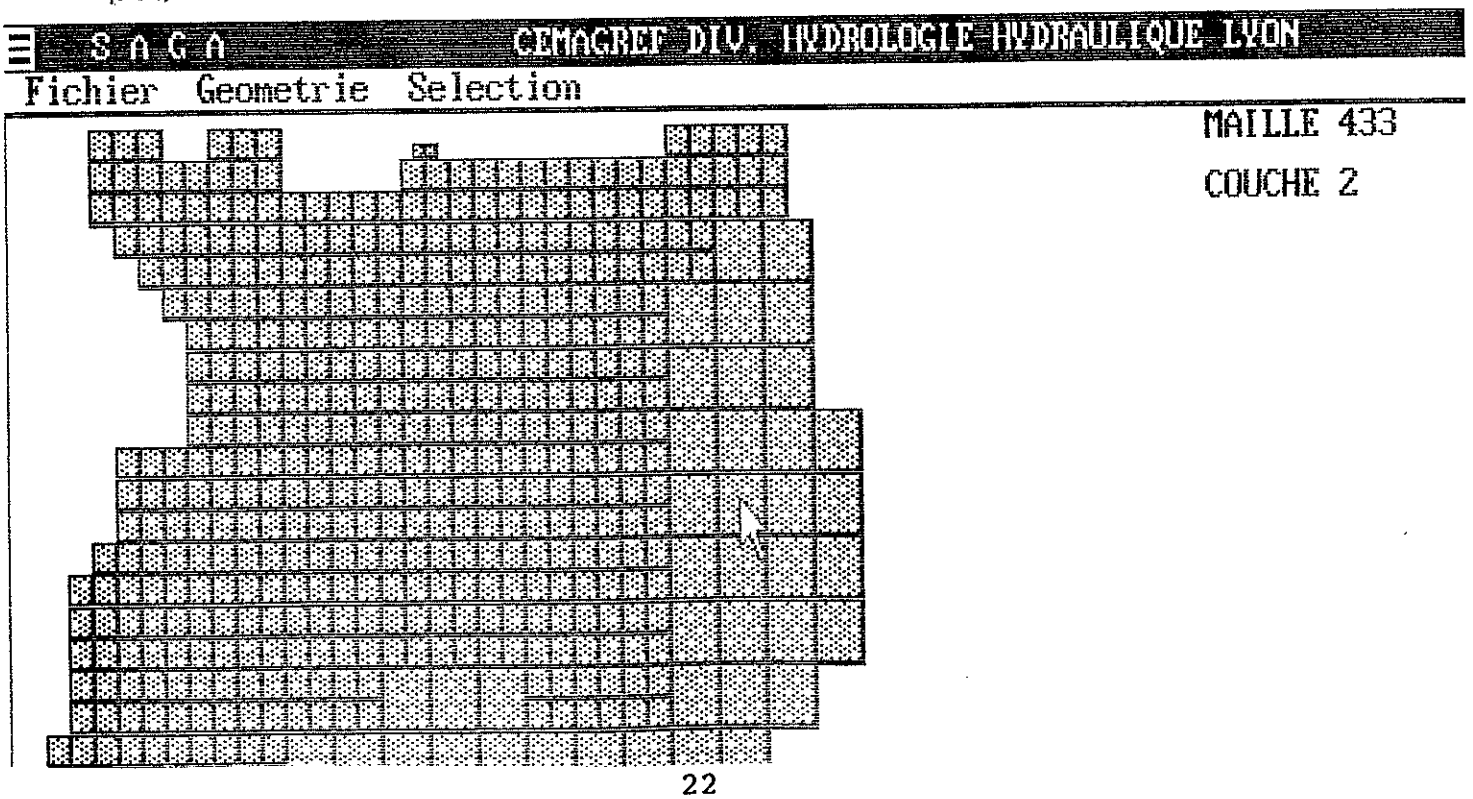
OK

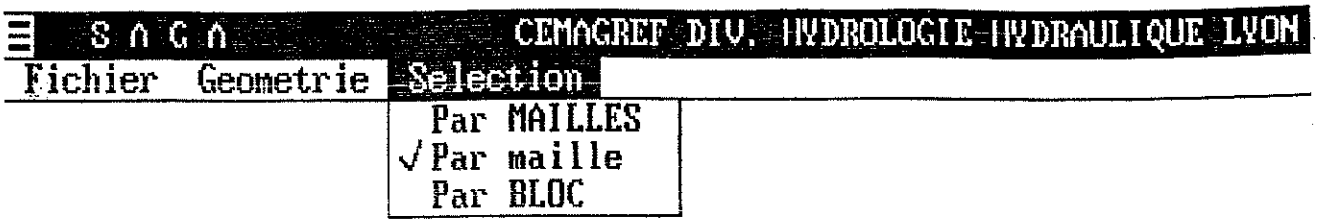
– L'item **DIMINUER** a la fonction inverse du précédent item. Il permet de diminuer la taille du dessin de mailles courant.



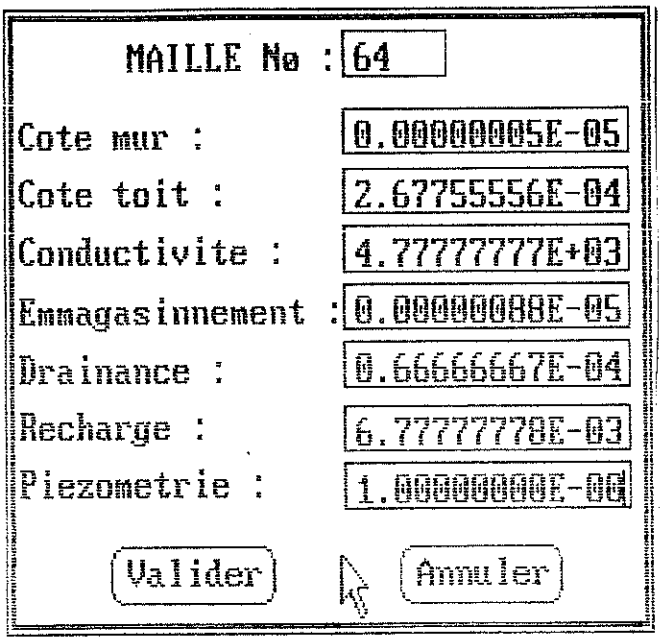
– L'item **CHANGER DE COUCHE** permet d'identifier à l'écran (par une couleur spécifique) la couche courante sur laquelle on va travailler. En effet, au départ, quand on a ouvert un fichier de géométrie, le squelette de toutes les mailles s'est dessiné avec la première couche en couleur ; si on change de couche, les mailles de la couche précédente vont devenir blanches et les mailles de la couche suivante vont devenir colorées.

L'action de changer de couche incrémente ainsi le numéro de la couche courante de 1 (On revient à la première couche si le numéro suivant n'existe pas).





Le menu **SELECTION** concerne les modes de sélection des mailles (afin de changer les attributs de celles-ci). Il est composé de trois articles, mais le résultat de ces trois modes est identiques : il entraîne l'apparition d'une boite de dialogue permettant de modifier la valeur des attributs :



– L'item **Par MAILLES** consiste à présélectionner plusieurs mailles, en cliquant successivement sur celles-ci (ce qui en modifie la couleur en rouge). Lorsqu'on estime que cette sélection est complète, on "double-clique" sur la dernière maille choisie ; une boite de dialogue apparait qui permet de modifier le(s) attribut(s) des mailles. Cette mise à jour concerne la totalité des mailles pré-sélectionnées.

– L'item **Par BLOC** permet le même type de mise à jour. Mais la pré-sélection des mailles consiste à choisir un "rectangle" de maille dans le maillage général. Ce "rectangle" est définit par ses mailles supérieure gauche et inférieure droite.
La même boite de dialogue autorise la mise à jour des attributs intéressant la totalité du rectangle pré-défini (apparaissant en rouge à l'écran).

– L'item **Par maille** est le mode de sélection initial. Il permet de ne modifier qu'une seule maille à la fois. Si on clique sur une maille, la maille choisie devient rouge et la boite de dialogue apparait. Mais si on modifie les attributs de la maille (c'est à dire si on sort de la boite de dialogue en cliquant sur VALIDER) alors la maille modifiée devient grise sinon (si on sort de la boite de dialogue en cliquant sur ANNULER) la maille reprend sa couleur normale.

3.3.4. Autres précisions

Il faut aussi préciser des caractéristiques de cet interface indispensables à la bonne compréhension et à l'utilisation de celui-ci :

– Dès que le dessin de mailles d'un aquifère se représente à l'écran, il apparait en haut à gauche de l'écran deux informations précieuses pour l'utilisateur :

* **Le numéro de couche courant**, c'est à dire la couche dont les mailles sont colorées (Ce numéro de couche est incrémente de 1 chaque fois que l'on choisit l'item CHANGER DE COUCHE du menu GEOMETRIE).

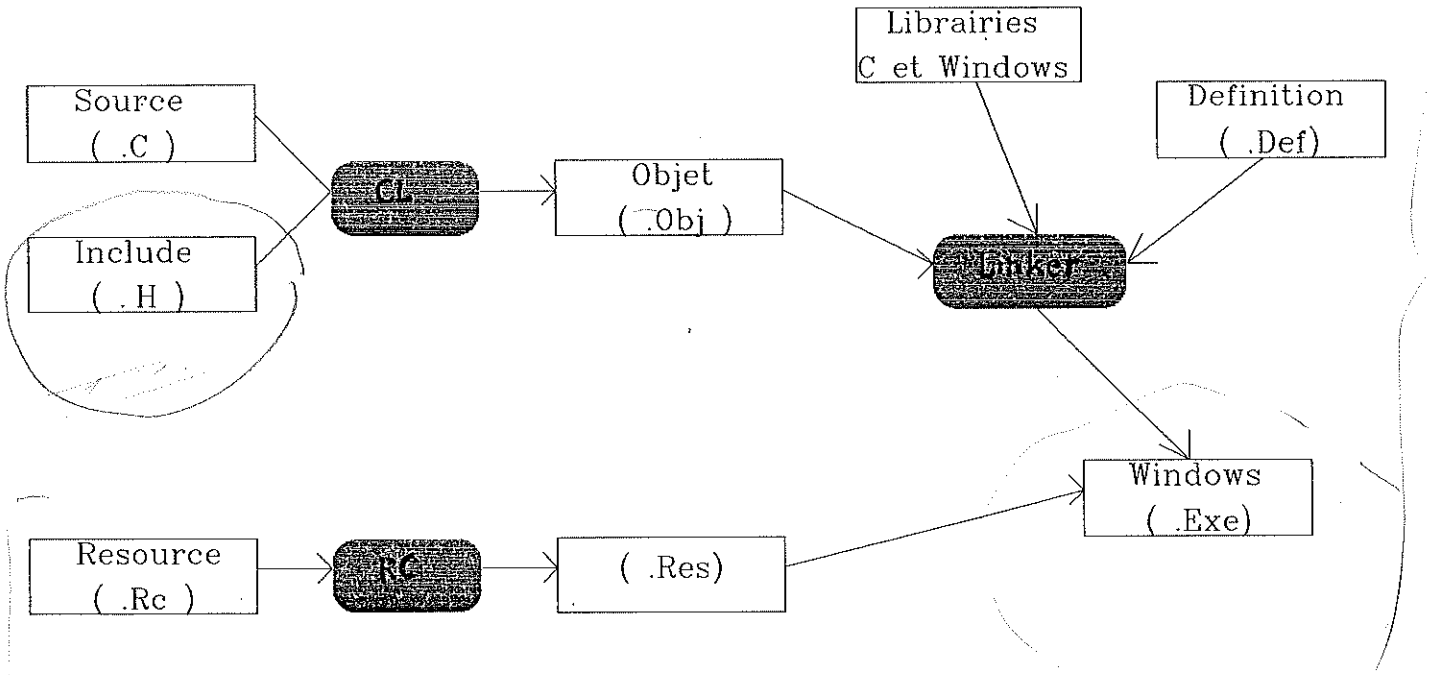
* **Le numéro de maille courant**, c'est à dire le numéro de la maille (colorée) sur laquelle se trouve la flèche de la souris. Ainsi, dès que l'utilisateur bouge la souris , il sait , *en temps réel et instantanément*, sur quelle maille il pointe.

– Dès qu'une maille est sélectionnée (Par n'importe quel mode) elle devient **rouge**. Si l'utilisateur sort de la boite de dialogue de modification des attributs par le bouton VALIDER la maille devient **grise**, si il sort par le bouton ANNULER la maille reprend sa couleur d'origine, la couleur de sa couche. Ainsi, si on ne change pas de couche, on peut voir toutes les mailles que l'on a modifiées dans la couche ; mais si on change de couche, il n'apparait, en gris, dans la nouvelle couche que la dernière maille modifiée de la couche précédente. De plus, dès qu'une maille est modifiée dans cette nouvelle couche, la maille grise (modification de la couche précédente) *reprend sa couleur normale*, et ainsi de suite.

4. REALISATION DU PROGRAMME

4. REALISATION DU PROGRAMME

La construction d'une application WINDOWS suit le schéma suivant :



Il est donc nécessaire de construire quatre fichiers pour arriver au programme exécutable (.EXE) :

- Le fichier *Definition* (.DEF)
- Le fichier *Resource* (.RC)
- Le fichier *Include* (.H)
- Le fichier *Source* (.C)

4.1. Le fichier DEFINITION (.DEF)

Toutes les applications de WINDOWS ont besoin d'un module de définition (Voir annexe A-1). Ce fichier définit le nom, les segments et les besoins en mémoire, ainsi que les fonctions "exportées" par l'application, c'est à dire toutes les procédures appelées par l'application.

Ce fichier intervient au niveau du *Linker* dans l'application elle-même.

4.2. Le fichier RESOURCE (.RC)

Le fichier RESOURCE définit toutes les ressources utilisées par l'application. Ainsi, c'est dans ce fichier que l'on déclare et définit **les menus, les boites de dialogues** ou **les icônes** (Voir annexe A-2).

Par exemple, les menus sont définies à l'aide de POPUP et chaque item par MENUITEM ; on doit cependant affecter une variable (ex : IDM_OPEN) à chaque item pour pouvoir ensuite identifier celui-ci.
La définition d'une boite de dialogue est bien plus complexe puisqu'il en existe une grande variété. On doit d'abord préciser le nom de la boite suivi de DIALOG et des données numériques correspondant à sa position dans la fenêtre et à sa taille. Ensuite, il faut définir les boutons, cases et autres textes composant la boite de dialogue (une variable est encore affectée pour chaque item de la boite ; ex : IDOK pour un bouton de validation).

Ce fichier est compilé individuellement par le *Resource Compiler* (RC) qui produit un fichier (.RES) et copie les ressources au fichier exécutable (.EXE).

4.3. Le fichier INCLUDE (.H)

Le fichier INCLUDE définit les constantes et les prototypes de fonction de l'application. En effet, une application est composée de deux fichiers source (.C et .RC) qui se partagent des constantes communes ; on a donc créée un seul fichier include (Voir annexe A-4).

Ce fichier est donc constitué d'une série de définition de constantes (ex : #define IDM_OPEN 100) ; ce sont toutes les variables affectées aux différents items des menus et autres boites de dialogues du fichier RESOURCE qui sont, ici, définies comme constantes.

De plus, toutes les fonctions du fichier SOURCE (.C) sont définies. Cela va de la définition du *WinMain* (Voir 4.4.) à tout autres fonction du .C . Il existe plusieurs types de définitions de fonctions correspondants aux types de données retournés par ces fonction :

- int : pour une valeur entière signée
- void : pour une valeur vide (spécifie que la fonction ne retourne aucune valeur)
- BOOL : pour une valeur booléenne
- HANDLE : représente une valeur indexée à un bloc de mémoire par le système

4.4. Le fichier SOURCE (.C)

Le fichier SOURCE est bien évidemment le fichier le plus important d'une application WINDOWS. Il contient toutes les définitions de la fenêtre et le programme en lui-même.

De plus, une application WINDOWS a les composants de base suivants :

- Une *WinMain function*
- Une *window function*

4.4.1. La *WinMain function*

Toutes les applications WINDOWS doivent avoir une fonction **WinMain** (Voir annexe A-7), c'est le point de départ de toutes applications (à la manière de la fonction Main d'un programme C). WinMain a les fonctionnalités suivantes :

- Déclarer la classe de la fenêtre de l'application et initialiser ses caractéristiques.

La classe de la fenêtre définie les attributs de la fenêtre, tels que la forme du curseur, le nom du menu, la couleur du fond de l'écran, etc... On déclare une classe de fenêtre en remplissant une structure de **WNDCLASS** (Voir annexe). Par des soucis de simplification du **WinMain**, mais aussi d'organisation de la mémoire, une fonction d'initialisation a été créer.

- Créer une fenêtre générale (et peut-être d'autres fenêtres utilisées par l'application).

Après avoir déclarée la classe de la fenêtre, on peut la créer.
On procède à cela à l'aide de la fonction **CreateWindow**. Cette fonction possède plusieurs paramètres tel que le nom de la classe de la fenêtre, le titre de la fenêtre ainsi que la position, le style , la hauteur, la largeur de celle-ci.

Cependant, maintenant que la fenêtre est déclarée et créée, il faut l'afficher et la mettre à jour. Cela s'effectue à l'aide des fonctions **ShowWindow** et **UpdateWindow**.

- Mettre en route l'application.

Pour débuter l'application, il faut créer un "*message loop*" (Voir annexe A-7). C'est une boucle dans lequel **WinMain** récupère et envoie différents messages, cela à l'aide des fonctions **GetMessage** et **DispatchMessage**.

- Mettre fin à l'application (message WM_QUIT).

Pour terminer l'application, il faut que WinMain reçoive, dans le *message loop*, un message **WM_QUIT**. Celui-ci peut être générer par la fonction **PostQuitMessage**.

4.4.2. La *window function*

Après la **WinMain**, la **window function** est le deuxième composant de base essentiel de toutes applications WINDOWS.

Cette fonction (Voir annexe A-9) reçoit des messages de deux types : des messages d'entrée provenant du *message loop* et des messages de WINDOWS.

Les messages d'entrée correspondent à la souris, au clavier ou à l'horloge système. On peut citer par exemple :

- WM_MOUSEMOVE : déclenché lorsque la souris se déplace
- WM_LBUTTONUP : déclenché lorsque le bouton gauche (WM_RBUTTONUP pour le droit) de la souris est clique.
- WM_LBUTTONDBCLKS : déclenché lorsque le bouton gauche de la souris est double-clique
- WM_KEYDOWN : déclenché lorsque une touche du clavier est actionnée

Il existe, cependant, une multitude d'autres types de message pouvant intervenir dans le **window function**.

Les messages de WINDOWS sont en fait des requêtes afin d'effectuer telle ou telles fonctions. Citons pour l'exemple :

- WM_CREATE : déclenché lorsque la fenêtre est créée
- WM_PAINT : déclenché lorsque une commande pour "peindre" une portion de fenêtre est appelée
- WM_COMMAND : déclenché lorsque l'utilisateur choisit un item dans un menu.
- WM_DESTROY : déclenché afin de terminer la session de travail

La **window function** (appelée SagWndProc dans mon application) teste donc tous les messages reçus. Elle est ainsi composée d'un *switch()...case* afin de répondre à chaque message.

4.4.3. Les autres fonctions

Mon application est composée d'autres fonctions. Parmi ces fonctions, on peut citer les fonctions de boites de dialogues. En effet, lorsqu'une boite de dialogue est demandée (à partir de la **window function**) une fonction de type booléen est appelée ; elle gère les différentes messages reçus par la boite de dialogue (ex : la fonction correspondant à la boite de dialogue de l'item *à propos* : BOOL FAR PASCAL About , en annexe A.19).

5. ANNEXES

```
/* SAG.DEF */
/* Fichier DEFINITION */


NAME     Sag

DESCRIPTION 'S  A  G  A'

STUB     'WINSTUB.EXE'

CODE     MOVEABLE
DATA     MOVEABLE MULTIPLE

HEAPSIZE  1024
STACKSIZE 4096

EXPORTS
     SagWndProc      @1
     About           @2
     OpenDlg         @3
     Attribut        @4
     Save            @5
     Print           @6
```

```
                                /* SAG.RC */
                            /* Fichier RESOURCE */


#include "windows.h"
#include "sag.h"

FileOpen MENU
BEGIN
    POPUP         "Fichier"
    BEGIN
 MENUITEM     "Ouvrir",                IDM_OPEN
 MENUITEM     "Enregistrer",           IDM_SAVE
 MENUITEM     "Imprimer",              IDM_PRINT
 MENUITEM     SEPARATOR
 MENUITEM     "Quitter",               IDM_EXIT
 MENUITEM     "A propos de...",        IDM_ABOUT
    END

    POPUP         "Geometrie"
    BEGIN
 MENUITEM     "Vue originale",         IDM_VUEO
 MENUITEM     "Zoomer",                IDM_ZOOM
 MENUITEM     "Agrenair",              IDM_AGRA
 MENUITEM     "Diminuer",              IDM_DIMI
 MENUITEM     "Changer de couche",     IDM_CHCO
    END

    POPUP         "Selection"
    BEGIN
 MENUITEM     "Par MAILLES",           IDM_SMUL
 MENUITEM     "Par maille",            IDM_SMON ,CHECKED
 MENUITEM     "Par BLOC",              IDM_SBLO
    END
END

Save DIALOG 69, 75, 185, 46
STYLE WS_DLGFRAME | WS_POPUP
BEGIN
    DEFPUSHBUTTON "OUI", IDOK , 14, 27, 35, 14
    PUSHBUTTON "ANNULER",IDCANCEL, 137, 27, 35, 14
    PUSHBUTTON "NON", 100, 76, 27, 35, 14
    LTEXT "Voulez-vous sauver vos modifications ?", 101, 15, 8, 155, 9
END

Print DIALOG LOADONCALL MOVEABLE DISCARDABLE 69, 75, 185, 47
STYLE WS_DLGFRAME | WS_POPUP
BEGIN
    DEFPUSHBUTTON "OUI" , IDOK, 14, 27, 35, 14
    PUSHBUTTON "ANNULER", IDCANCEL, 137, 27, 35, 14
    PUSHBUTTON "NON", 100, 76, 27, 35, 14
    LTEXT "Voulez-vous imprimer vos modifications ?",101, 13, 8, 166, 11
END
```

```
Open DIALOG 10, 10, 148, 112
STYLE WS_DLGFRAME | WS_POPUP
BEGIN
 LTEXT "&Fichier a ouvrir:", ID_FILENAME,  4,  4, 100, 10
 EDITTEXT ID_EDIT,   4, 16, 100, 12, ES_AUTOHSCROLL
 LTEXT "&Fichier dans", ID_FILES,     4, 40,  32, 10
 LISTBOX, ID_LISTBOX,   4, 52,  70, 56, WS_TABSTOP
 LTEXT "",  ID_PATH,  40, 40, 100, 10
 DEFPUSHBUTTON "&Ouvrir" , IDOK,   87, 60,  50, 14
 PUSHBUTTON "Annuler", IDCANCEL,  87, 80,  50, 14
END

Attribut DIALOG 230,60, 130, 124
STYLE WS_DLGFRAME | WS_POPUP
BEGIN
  LTEXT "MAILLE N° :",        100, 23, 3, 45, 8
  LTEXT "Cote mur : ",        101, 0, 19, 41, 8
  LTEXT "Conductivite :",     103, 0, 43, 56, 8
  LTEXT "Emmagasinnement :", 104, 0, 56, 68, 8
  LTEXT "Drainance :",        105, 0, 68, 50, 8
  LTEXT "Recharge :",         106, 0, 80, 52, 8
  LTEXT "Piezometrie :",      107, 0, 92, 55, 9
  LTEXT "Cote toit :",        109, 0, 31, 45, 8
  EDITTEXT  ID_EDCOTM , 68, 17, 60, 10
  EDITTEXT  ID_EDCOND , 68, 42, 60, 10
  EDITTEXT  ID_EDCOTT , 68, 30, 60, 10
  EDITTEXT  ID_EDDRAI , 68, 66, 60, 10
  EDITTEXT  ID_EDEMMA , 68, 54, 60, 10
  EDITTEXT  ID_EDRECH , 68, 79, 60, 10
  EDITTEXT  ID_EDPIEZ , 68, 91, 60, 10
  EDITTEXT  ID_EDITE ,  68,  1, 22, 10, ES_AUTOHSCROLL
  PUSHBUTTON "Valider", IDOK, 19, 108, 35, 14
  PUSHBUTTON "Annuler", IDCANCEL, 79, 108, 35, 14
END

AboutBox DIALOG 22, 17, 144, 75
STYLE WS_POPUP | WS_DLGFRAME
BEGIN
    CTEXT "Windows"                -1,       0,  5, 144,  8
    CTEXT "Application S A G A"    -1,       0, 14, 144,  8
    CTEXT "Version 1.0"            -1,       0, 34, 144,  8
    DEFPUSHBUTTON "OK"          IDOK,      53, 59,  32, 14,      WS_GROUP
END
```

```
                    /* SAG.H */
                /* Fichier INCLUDE */


/* Article du menu de fichier */

#define     IDM_OPEN      100
#define     IDM_SAVE      101
#define     IDM_PRINT     102
#define     IDM_EXIT      103
#define     IDM_ABOUT     104

/* Article du menu de géométrie */

#define     IDM_VUEO      200
#define     IDM_ZOOM      201
#define     IDM_AGRA      202
#define     IDM_DIMI      203
#define     IDM_CHCO      204

/* Article du menu de sélection */

#define     IDM_SMUL      300
#define     IDM_SMON      301
#define     IDM_SBLO      302

/* Article de controle */

#define     ID_FILENAME   400
#define     ID_EDIT       401
#define     ID_FILES      402
#define     ID_PATH       403
#define     ID_LISTBOX    404
#define     ID_EDITE      405
#define     ID_EDDRAI     406
#define     ID_EDPIEZ     407
#define     ID_EDCOND     408
#define     ID_EDCOTT     409
#define     ID_EDCOTM     410
#define     ID_EDRECH     411
#define     ID_EDEMMA     412

int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
BOOL SagInit(HANDLE);
long FAR PASCAL SagWndProc(HWND, unsigned, WORD, LONG);
BOOL FAR PASCAL About(HWND, unsigned, WORD, LONG);

HANDLE FAR PASCAL Save(HWND, unsigned, WORD, LONG);
HANDLE FAR PASCAL Print(HWND, unsigned, WORD, LONG);
HANDLE FAR PASCAL Attribut(HWND, unsigned, WORD, LONG);

HANDLE FAR PASCAL OpenDlg(HWND, unsigned, WORD, LONG);
void UpdateListBox(HWND);
```

```
void SeparateFile(HWND, LPSTR, LPSTR, LPSTR);
void AddExt(PSTR, PSTR);
void ChangeDefExt(PSTR, PSTR);
void _lstrcpy(LPSTR, LPSTR);
void _lstrncpy(LPSTR, LPSTR, int);
int  _lstrlen(LPSTR);
int Maille(int,int,int,int,int);
void DesMail (HWND,int,int,int);
```

```
/**********************************************************************

     PROGRAMME:      SAG.c

     AUTEUR    :     Jérôme INGLES

**********************************************************************/


#include "windows.h"
#include "sag.h"
#include "stdio.h"

#define MAXMAIL 1000
#define COEF 3

HANDLE hInst;
HANDLE hAccTable;
HWND hEditWnd;

char nb[14]= "              ";
char val;

char ButtonText[40];
char Couche[40];
int NombMail,NombC,Coef,oui,MailModif1;

int mai,i,d,REC,OP,NC = 1;
int MailModif = MAXMAIL + 1;
static char cond[MAXMAIL][14];        /* Tableau de conductivité */
static char emma[MAXMAIL][14];      /* Tableau de emmagasinement */
static char cott[MAXMAIL][14];        /* Tableau de côte du toit */
static char cotm[MAXMAIL][14];         /* Tableau de côte du mur */
static char drai[MAXMAIL][14];           /* Tableau de drainance */
static char rech[MAXMAIL][14];            /* Tableau de recharge */
static char piez[MAXMAIL][14];          /* Tableau de piézométrie */
static int tab[MAXMAIL][5];  /* Tableau des données géométriques*/
WORD wPaint  = 0;
char FileName[128];                        /* nom du fichier courant */
char PathName[128];                         /* nom du chemin courant */
char OpenName[128];                        /* nom du fichier à ouvrir */
char DefPath[128];                /* chemin par défault pour la liste*/
char DefSpec[13] = "*.DAT";      /* extension de fichier courante */
char DefExt[] = "";                          /* extension par default */
char str[255];
HBRUSH hOldBrush,hRedBrush,hWhiteBrush,hBlueBrush;
HBRUSH hBlackBrush,hRoseBrush,hJauneBrush,hVertBrush;
```

```
/*************************************************************************

    FONCTION: WinMain(HANDLE, HANDLE, LPSTR, int)

    UTILITE : appelle la fonction d'initialisation,
              génère le "message loop".

*************************************************************************/


int PASCAL WinMain(hInstance, hPrevInstance, lpCmdLine, nCmdShow)
HANDLE hInstance;
HANDLE hPrevInstance;
LPSTR lpCmdLine;
int nCmdShow;
{
    HWND hWnd;
    MSG msg;
    RECT Rect;

    if (!hPrevInstance)
 if (!SagInit(hInstance))
     return(NULL);

    hInst = hInstance;

    hWnd = CreateWindow("FileOpen",
    "S A G A                    CEMAGREF DIV. HYDROLOGIE-HYDRAULIQUE
    LYON          ",

 WS_OVERLAPPEDWINDOW,
 0,
 0,
 640,
 480,
 NULL,
 NULL,
 hInstance,
 NULL);

    if (!hWnd)
 return(NULL);

    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

                        /* "Message loop" */

    while (GetMessage(&msg, NULL, NULL, NULL)) {
     (!TranslateAccelerator(hWnd, hAccTable, &msg)) {
     TranslateMessage(&msg);
     DispatchMessage(&msg);
 }
    }
    return(msg.wParam);
```

```
/**************************************************************************

    FONCTION: EditfileInit(HANDLE)

    UTILITE : Initialise les données de la fenêtre
              et déclare la classe de la fenêtre.

**************************************************************************/

BOOL SagInit(hInstance)
HANDLE hInstance;
{
    HANDLE hMemory;
    PWNDCLASS pWndClass;
    BOOL bSuccess;

    hMemory = LocalAlloc(LPTR, sizeof(WNDCLASS));
    pWndClass = (PWNDCLASS) LocalLock(hMemory);
    pWndClass->hCursor = LoadCursor(NULL, IDC_ARROW);
    pWndClass->hIcon = LoadIcon(NULL, IDI_APPLICATION); pWndClass-
    >lpszMenuName = (LPSTR) "FileOpen";
    pWndClass->lpszClassName = (LPSTR) "FileOpen";
    pWndClass->hbrBackground = GetStockObject(WHITE_BRUSH);
    pWndClass->hInstance = hInstance;
    pWndClass->style = NULL;
    pWndClass->lpfnWndProc = SagWndProc;

    bSuccess = RegisterClass((LPWNDCLASS)pWndClass);

    LocalUnlock(hMemory);
    LocalFree(hMemory);
    return(bSuccess);
}
```

```
/*************************************************************************

    FONCTION: EditfileWndProc(HWND, unsigned, WORD, LONG)

    UTILITE:  Génère les différents messages.

    MESSAGES:

 WM_COMMAND    - Séléction de menu de l'application.
 WM_CREATE     - Création de la fen@tre.
 WM_MOUSEMOVE  - Déplacement de la souris.
 WM_LBUTTONUP  - Clic gauche.
 WM_PAINT      - Remplissage d'une région de la fen@tre.
 WM_DESTROY    - destruction de la fen@tre.

    COMMENTAIRES:

 WM_COMMAND gère les commandes des items des trois menus.

*************************************************************************/


long FAR PASCAL SagWndProc(hWnd, message, wParam, lParam)
HWND hWnd;
unsigned message;
WORD wParam;
LONG lParam;
{
    FARPROC lpProcAbout, lpOpenDlg, lpAttribut, lpSave, lpPrint

    HDC hDC;
    PAINTSTRUCT ps;

    switch (message) {

   case WM_SYSCOMMAND:
   if (wParam == IDM_ABOUT) {
     lpProcAbout = MakeProcInstance(About, hInst);/*Appel modél
     DialogBox(hInst, "AboutBox", hWnd, lpProcAbout);/*d'une
                                                      fonction*/
     FreeProcInstance(lpProcAbout);        /*de boite de dialogu
     break;
            }
            else
  return (DefWindowProc(hWnd, message, wParam, lParam));
```

```
case WM_CREATE:
      OP = 1;
      hRedBrush = CreateSolidBrush(RGB(255,0,0));     /* Création */
      hBlackBrush = CreateSolidBrush(RGB(127,127,127)); /*    des */
      hBlueBrush = CreateSolidBrush(RGB(0,255,255)); /*différents*/
      hWhiteBrush = CreateSolidBrush(RGB(255,255,255));/*stylos*/
      hRoseBrush = CreateSolidBrush(RGB(255,96,255)); /*    de */
      hJauneBrush = CreateSolidBrush(RGB(255,255,0)); /*couleurs */
      hVertBrush = CreateSolidBrush(RGB(159,255,0));
      break;

    case WM_MOUSEMOVE:
     if (OP != 1)
       {
       sprintf(Couche,"COUCHE %d",NC);
       sprintf(ButtonText,"MAILLE %d  ",
       Maille(LOWORD(lParam), HIWORD(lParam),tab,NC,Coef));
       InvalidateRect(hWnd, NULL, FALSE);
       }
            break;

    case WM_LBUTTONUP:
     hDC = GetDC(hWnd);
     mail = Maille(LOWORD(lParam), HIWORD(lParam),tab,NC,Coef);
     i = 0;
     REC = 0;
     if ( mail != 0 )
       {
       while (i<NombMail && REC == 0)
  {
  ++i;
  if (tab[i][1] == mail)
    {
    REC = 1;
    hOldBrush = SelectObject(hDC,hRedBrush);
    Rectangle(hDC,Coef*tab[i][3],Coef*tab[i][2],Coef*(tab[i][3]+ta
    b[i][5]),Coef*(tab[i][2]+tab[i][5])); }
  }
       REC = 0;
       lpAttribut = MakeProcInstance((FARPROC) Attribut, hInst);
       DialogBox(hInst, "Attribut", hWnd, lpAttribut);
       FreeProcInstance(lpAttribut);

       hOldBrush = SelectObject(hDC,hBlackBrush);
       Rectangle(hDC,Coef*tab[i][3],Coef*tab[i][2],Coef*(tab[i][3]
       +tab[i][5]),Coef*(tab[i][2]+tab[i][5]));

       if (NC == 1)
  hOldBrush = SelectObject(hDC,hBlueBrush);
  else if (NC == 2)
         hOldBrush = SelectObject(hDC,hRoseBrush);
         else hOldBrush = SelectObject(hDC,hJauneBrush);

       if (REC == 0)
  Rectangle(hDC,Coef*tab[i][3],Coef*tab[i][2],Coef*(tab[i][3]+tab[
```

```
i][5],Coef*(tab[i][2]+tab[i][5]); else {
        MailModif1= tab[i][1];
        if (tab[MailModif][4] != NC)
          {
          i = 1;
          while (tab[i][4] != NC)
     ++i;
          while (tab[MailModif][3] != tab[i][3])
     ++i;
          while (tab[MailModif][2] != tab[i][2])
     ++i;
          if (tab[i][4] != NC)
     {
     hOldBrush = SelectObject(hDC,hWhiteBrush);
    Rectangle(hDC,Coef*tab[MailModif][3],Coef*tab[MailModif][2],Coe
     f*(tab[MailModif][3]+tab[MailModif][5]),Coef*(tab[MailModif][2
     ]+tab[MailModif][5])); }
    else
Rectangle(hDC,Coef*tab[MailModif][3],Coef*tab[MailModif][2],Coef*(
tab[MailModif][3]+tab[MailModif][5]),Coef*(tab[MailModif][2]+tab[M
ail[Modif][5]));
          }
        }
        MailModif = MailModif1;
        }
        ReleaseDC(hWnd,hDC);
        break;


     case WM_PAINT:
      hDC = BeginPaint (hWnd, &ps);
      TextOut(hDC, 500, 1, ButtonText,strlen(ButtonText));
      TextOut(hDC, 500,20, Couche,strlen(Couche));
      EndPaint(hWnd, &ps);
             break;

  case WM_COMMAND:
      switch (wParam) {
   case IDM_OPEN:
     lpOpenDlg = MakeProcInstance((FARPROC) OpenDlg, hInst);
     DialogBox(hInst, "Open", hWnd, lpOpenDlg);
     FreeProcInstance(lpOpenDlg);
     if (oui == 1)
       {
       int c,i,j,ni,I,J,a,val;
       char lig[82];
       FILE *fopen(), *fp;
       MailModif = MAXMAIL;
       NC = 1;
       Coef = COEF;
       fp = fopen (OpenName,"r");
       ni = 0;
       i = I = J = 1;
       while (ni <= 3)
         {
         c = getc(fp);
```

```
      if (c == '\n')
++nl;
      )
    while ((c = getc(fp)) != 'F')
      {
      lig[i] = c;
      ++i;
      if (c == '\n')
{
a = 4;
while (a < 21)
  {
  if (lig[a-2] != 32)
    val = (lig[a-2]-48)*100 + (lig[a-1]-48)*10 + (lig[a]-48);
    else if (lig[a-1] != 32 && lig[a-1] != '-')
    val = (lig[a-1]-48)*10+(lig[a]-48);
    else val = lig[a]-48;
  tab[I][J] = val;
  ++J;
  a = a + 4;
  }
i = J = 1;
++I;
}
      )
    fclose(fp);
    NombC = tab[I-1][4];
    NombMail = I;
    i=I+1;
    while (i <= MAXMAIL)
      {
      j = 1;
      while (j<6)
{
tab[i][j] = 0;
++j;
}
      ++i;
      )
    OP = 0;
    DesMail(hWnd,NC,MailModif,Coef);
    }
nbfic = 1;
while (nbfic != 0)
  {
  int c,nl,I,J,i,a;
  char val;
  char lig[82];
  FILE *fopen(), *fp;
  fp = fopen ("donncond.dat","r");
  nl = 0;
  i = I = 1;
  while (nl <= 4)
    {
    c = getc(fp);
    if (c == '\n')
```

```
        ++nl;
      }
   while ((c = getc(fp)) != 'F')
      {
      lig[i] = c;
      ++i;
      if (c == '\n')
        {
        a = 2;
        while (a < 82)
{
J = 1;
while (J < 15)
   {
   val = lig[a+J];
   switch (val)
     {
     case 32: cond[I][J] = ' ';
       break;
     case 43: cond[I][J] = '+';
       break;
     case 45: cond[I][J] = '-';
       break;
     case 46: cond[I][J] = '.';
       break;
     case 69: cond[I][J] = 'E';
       break;
     default: cond[I][J] = val;
       break;
     }
    ++J;
   }
 ++I;
 a = a + 16;
 }
        i=1;
        ++nl;
        }
      }
      fclose(fp);
      ++nbfic;
      }

    break;

case IDM_SAVE:
     lpSave = MakeProcInstance((FARPROC) Save, hInst);
     DialogBox(hInst, "Save", hWnd, lpSave);
     FreeProcInstance(lpSave);
     break;

case IDM_PRINT:
     lpPrint = MakeProcInstance((FARPROC) Print, hInst);
     DialogBox(hInst, "Print", hWnd, lpPrint);
     FreeProcInstance(lpPrint);
     break;
```

```
        /*MessageBox(hWnd, "Commande non implémentée",
    (LPSTR) NULL, MB_OK);
        break;*/

    case IDM_EXIT:
        DestroyWindow(hWnd);
        break;

    case IDM_ABOUT:
        lpProcAbout = MakeProcInstance(About, hInst);
        DialogBox(hInst, "AboutBox", hWnd, lpProcAbout);
        FreeProcInstance(lpProcAbout);
        break;


    case IDM_SMUL:
    case IDM_ZOOM:
    case IDM_AGRA:
        ++Coef;
        DesMail(hWnd,NC,MailModif,Coef);
        break;
    case IDM_DIMI:
        if (Coef > 1)
          --Coef;
        DesMail(hWnd,NC,MailModif,Coef);
        break;
    case IDM_CHCO:
        ++NC;
        if (NC > NombC)
          NC = 1;
        sprintf(Couche,"COUCHE %d",NC);
        DesMail(hWnd,NC,MailModif,Coef);
        break;

     /*MessageBox(hWnd, "Commande non implémentée",
     (LPSTR) NULL, MB_OK);
        break;*/
       }
       break;

    case WM_DESTROY:
        PostQuitMessage(NULL);
        break;

    default:
        return(DefWindowProc(hWnd, message, wParam, lParam));

      }
      return(NULL);
}
```

```
/**************************************************************************

    FONCTION: OpenDlg(HWND, unsigned, WORD, LONG)

    UTILITE : Permet à l'utilisateur de sélectionner un fichier et
              de l'ouvrir

**************************************************************************/


HANDLE FAR PASCAL OpenDlg(hDlg, message, wParam, lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
     WORD index;
     PSTR pTptr;
     HANDLE hFile;

     switch (message) {
  case WM_COMMAND:
      switch (wParam) {

   case ID_LISTBOX:
       switch (HIWORD(lParam)) {


    case LBN_SELCHANGE:
        if (!DlgDirSelect(hDlg, str, ID_LISTBOX)) {

     SetDlgItemText(hDlg, ID_EDIT, str);
     SendDlgItemMessage(hDlg, ID_EDIT, EM_SETSEL,
         NULL, MAKELONG(0, 0x7fff));
        }
        else {
     strcat(str, DefSpec);
     DlgDirList(hDlg, str, ID_LISTBOX,
         ID_PATH, 0x4010);
        }
        break;

    case LBN_DBLCLK:
        goto openfile;
       }
       return(TRUE);

   case IDOK:

       GetDlgItemText(hDlg, ID_EDIT, OpenName, 128);
       oui = 1;

       if (strchr(OpenName, '*') || strchr(OpenName, '?')) {
```

```
        SeparateFile(hDlg, (LPSTR) str, (LPSTR) DefSpec,
      (LPSTR) OpenName);
  if (str[0])
      strcpy(DefPath, str);
  ChangeDefExt(DefExt, DefSpec);
  UpdateListBox(hDlg);
  return(TRUE);
       }

     if (!OpenName[0]) {
  MessageBox(hDlg, "pas de fichier spécifié.",
      NULL, MB_OK | MB_ICONQUESTION);
  return(TRUE);
     }

     EndDialog(hDlg, NULL);
     return(TRUE);

 case IDCANCEL:

     oui = 0;
     EndDialog(hDlg, NULL);
     return(TRUE);
    }
    break;

case WM_INITDIALOG:
     UpdateListBox(hDlg);
     SetDlgItemText(hDlg, ID_EDIT, DefSpec);
     SendDlgItemMessage(hDlg,
  ID_EDIT,
  EM_SETSEL,
  NULL,
  MAKELONG(0, 0x7fff));

     SetFocus(GetDlgItem(hDlg, ID_EDIT));
     return (FALSE);
    }
    return FALSE;
}

/**************************************************************************

    FONCTION: UpdateListBox(hDlg)

    UTILITE : Met à jour la liste de fichier proposée

**************************************************************************/


void UpdateListBox(hDlg)
HWND hDlg;
{
     strcpy(str, DefPath);
     strcat(str, DefSpec);
```

```
    DlgDirList(hDlg, str, ID_LISTBOX, ID_PATH, 0x4010);
    SetDlgItemText(hDlg, ID_EDIT, DefSpec);
}

/**************************************************************************

    FONCTION: ChangeDefExt(PSTR, PSTR);

    UTILITE : Change l'extension par défaut

**************************************************************************/

void ChangeDefExt(Ext, Name)
PSTR Ext, Name;
{
    PSTR pTptr;

    pTptr = Name;
    while (*pTptr && *pTptr != '.')
 pTptr++;
    if (*pTptr)
 if (!strchr(pTptr, '*') && !strchr(pTptr, '?'))
     strcpy(Ext, pTptr);
}

/**************************************************************************

    FONCTION: SeparateFile(HWND, LPSTR, LPSTR, LPSTR);

    UTILITE : Separe le nom du fichier de son chemin d'accès.

**************************************************************************/

void SeparateFile(hDlg, lpDestPath, lpDestFileName, lpSrcFileName)
HWND hDlg;
LPSTR lpDestPath, lpDestFileName, lpSrcFileName;
{
    LPSTR lpTmp;

    lpTmp = lpSrcFileName + (long) _lstrlen(lpSrcFileName);

    while (*lpTmp != ':' && *lpTmp != '\\' && lpTmp >
 lpSrcFileName) lpTmp = AnsiPrev(lpSrcFileName, lpTmp);

    if (*lpTmp != ':' && *lpTmp != '\\') {
 _lstrcpy(lpDestFileName, lpSrcFileName);
 lpDestPath[0] = 0;
 return;
    }

    _lstrcpy(lpDestFileName, lpTmp + 1L);
```

```
    _lstrncpy(lpDestPath, lpSrcFileName, (int) (lpTmp -
    lpSrcFileName) + 1); lpDestPath[(lpTmp - lpSrcFileName) + 1] =
    0;
}

/**************************************************************************

    FONCTION: AddExt(PSTR, PSTR);

    UTILITE: Ajoute l'extension par défaut.

**************************************************************************/

void AddExt(Name, Ext)
PSTR Name, Ext;
{
    PSTR pTptr;

    pTptr = Name;
    while (*pTptr && *pTptr != ',')
 pTptr++;
    if (*pTptr != ',')
 strcat(Name, Ext);
}

/**************************************************************************

    FONCTION: _lstrlen(LPSTR)

    UTILITE:  retourne la longueur d'une chaine.

**************************************************************************/

int _lstrlen(lpStr)
LPSTR lpStr;
{
    int i;
    for (i = 0; *lpStr++; i++);
    return(i);
}

/**************************************************************************

    FONCTION: _lstrncpy(LPSTR, LPSTR)

    UTILITE:  copie une chaine.

**************************************************************************/

void _lstrncpy(lpDest, lpSrc, n)
```

```
LPSTR lpDest, lpSrc;
int n;
{
    while (n--)
 if (!(*lpDest++ = *lpSrc++))
     return;
}

/**************************************************************************


    FONCTION: _lstrcpy(LPSTR, LPSTR)

    UTILITE:  copie une chaine.

**************************************************************************/


void _lstrcpy(lpDest, lpSrc)
LPSTR lpDest, lpSrc;
{
    while(*lpDest++ = *lpSrc++);
}

/**************************************************************************


    FONCTION: About(HWND, unsigned, WORD, LONG)

    UTILITE:  Génère les messages de la boite de dialogue About(A
              Propos...).

    MESSAGES:

 WM_INITDIALOG - initialise la boite de dialogue
 WM_COMMAND    - reçoit les entrées

**************************************************************************/


BOOL FAR PASCAL About(hDlg, message, wParam, lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
    switch (message) {
 case WM_INITDIALOG:
     return(TRUE);

 case WM_COMMAND:
     if (wParam == IDOK)
      {
      EndDialog(hDlg, TRUE);
  return(TRUE);
      }
```

```
      return(TRUE);
     }
     return(FALSE);
}


/*************************************************************************

     FONCTION: Save(hDlg, message, wParam, lParam)

     UTILITE:  Génère les messages de la boite de dialogue Save

     MESSAGES:

 WM_INITDIALOG - initialise la boite de dialogue
 WM_COMMAND    - reçoit les entrées

*************************************************************************/


HANDLE FAR PASCAL Save(hDlg, message, wParam, lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
     switch (message) {
 case WM_INITDIALOG:
      return(TRUE);

 case WM_COMMAND:
      if (wParam == IDOK)
       {
       EndDialog(hDlg, TRUE);
  return(TRUE);
       }
      return(TRUE);
     }
     return(FALSE);
}
```

```
/*****************************************************************

    FONCTION: Print(hDlg, message, wParam, lParam)

    UTILITE:  Génére les messages de la boite de dialogue Print

    MESSAGES:

 WM_INITDIALOG - initialise la boite de dialogue
 WM_COMMAND    - reçoit les entrées

*****************************************************************/


HANDLE FAR PASCAL Print(hDlg, message, wParam, lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
{
    switch (message) {
 case WM_INITDIALOG:
     return(TRUE);

 case WM_COMMAND:
     if (wParam == IDOK)
      {
      EndDialog(hDlg, TRUE);
  return(TRUE);
      }
     return(TRUE);
    }
    return(FALSE);
}


/*****************************************************************

    FONCTION: Attribut(hDlg, message, wParam, lParam)

    UTILITE:  Génére les messages de la boite de dialogue Attribut

    MESSAGES:

 WM_INITDIALOG - initialise la boite de dialogue
 WM_COMMAND    - reçoit les entrées

*****************************************************************/


HANDLE FAR PASCAL Attribut(hDlg, message, wParam, lParam)
HWND hDlg;
unsigned message;
WORD wParam;
LONG lParam;
```

```
{
  WORD index;
  PSTR pTptr;
  HANDLE hFile;

  switch (message) {

 case WM_COMMAND:
      if (wParam == IDOK)
       {
       REC = 1;
       EndDialog(hDlg, TRUE);
       return(TRUE);
       }
      if (wParam == IDCANCEL)
       {
       EndDialog(hDlg, TRUE);
       return(TRUE);
       }
      return(TRUE);

 case WM_INITDIALOG:
     d = 1;
     while (d<=14)
     {
     nb[d] = cond[mail][d];
     ++d;
     }

      SetDlgItemText(hDlg, ID_EDCOND, nb);
      SendDlgItemMessage(hDlg,ID_EDCOND,
           EM_SETSEL,NULL,MAKELONG(0,0x7fff));
      SetFocus(GetDlgItem(hDlg, ID_EDCOND));

      SetDlgItemInt(hDlg, ID_EDITE, mail, NULL);
      SendDlgItemMessage(hDlg,ID_EDITE,
           EM_SETSEL,NULL,MAKELONG(0,0x7fff));
      SetFocus(GetDlgItem(hDlg, ID_EDITE));

      return(TRUE);

     }
     return(FALSE);
}
```

```
/**************************************************************************

     FONCTION: Maille(int,int,int,int,int)

     UTILITE : recherche d'un numéro de maille dans le fichier
               GRILLE.DAT selon 2 coordonnées courante de la
               souris.

**************************************************************************/


int Maille(is,js,tab,NumC,Coef)

int is,js,NumC,Coef;
int tab[MAXMAIL][5];

{
int i,c,n1,REC,I,J,NM;
REC = 0;
i = 1;
while (i<NombMail && REC == 0)
   {
   if (tab[i][4] == NumC)
     if ((Coef*tab[i][2] + Coef*tab[i][5]) > js && (js )=
Coef*tab[i][2]))
       if ((Coef*tab[i][3] + Coef*tab[i][5]) > is && (is )=
Coef*tab[i][3]))
  {
  NM = tab[i][1];
  REC = 1;
  }
    ++i;
   }
if (REC == 0)
   NM = 0;
return(NM);
}


/**************************************************************************

     FONCTION: DesMail(hWnd, int, int, int)

     UTILITE: Dessine à l'écran le squelette des mailles
              et colore la couche courante

**************************************************************************/


void DesMail(hWnd,NumC,MailModif,Coef)

int NumC,Coef;
HWND hWnd;
{
```

```
HDC hDC;
int i;
hDC = GetDC(hWnd);

hOldBrush = SelectObject(hDC,hWhiteBrush);
Rectangle(hDC,0,0,640,480);

i = 1;
while (i<NombMail)
  {
  MoveTo(hDC,Coef*tab[i][3],Coef*tab[i][2]);
  LineTo(hDC,Coef*tab[i][3],(Coef*tab[i][2]+(Coef*tab[i][5])));
  LineTo(hDC,(Coef*tab[i][3]+(Coef*tab[i][5])),(Coef*tab[i][2]+(Co
  ef*tab[i][5])));
  LineTo(hDC,(Coef*tab[i][3]+(Coef*tab[i][5])),Coef*tab[i][2]);
  LineTo(hDC,Coef*tab[i][3],Coef*tab[i][2]);
  ++i;
  }

if (NumC == 1)
    hOldBrush = SelectObject(hDC,hBlueBrush);
    else if (NumC == 2)
    hOldBrush = SelectObject(hDC,hRoseBrush);
    else hOldBrush = SelectObject(hDC,hJauneBrush);

i = 1;
while (i<NombMail)
  {
  if (tab[i][4] == NumC)

  Rectangle(hDC,Coef*tab[i][3],Coef*tab[i][2],Coef*(tab[i][3]+tab[
  i][5]),Coef*(tab[i][2]+tab[i][5])); ++i;
  }

i = 1;
while (i<NombMail)
  {
  if (tab[i][1] == MailModif)
    {
    hOldBrush = SelectObject(hDC,hBlackBrush);
    Rectangle(hDC,Coef*tab[i][3],Coef*tab[i][2],Coef*(tab[i][3]+ta
    b[i][5]),Coef*(tab[i][2]+tab[i][5])); }
  ++i;
  }
ReleaseDC(hWnd, hDC);
}
```

CONCLUSION

Le logiciel SAGA *(Système automatisé de gestion des aquifères)* comprendra dans sa forme définitive, quatre modèles conceptuellement distincts.

- un modèle mathématique de simulation d'écoulements (WATASI)

- Un module graphique de représentation cartographique

- Un interface utilisateur

- Une base de données de paramètres hydrodynamiques mise à jour en temps réel et couplée à WATASI

Les trois premiers modules sont désormais opérationnels (au moins en partie). L'interface utilisateur, dont le développement constituait le but de ce stage, est le module qui rend le mieux compte du besoin de mise à disposition des outils scientifiques, qui sont à l'origine du projet SAGA.

De fait, le module WATASI, sous sa forme actuelle de module de SAGA n'est pas plus précis ou plus pertinent qu'auparavant, mais il est devenu un outil de gestion capable d'expliciter ses résultats graphiquement, et sa maîtrise (hors calage qui reste l'apanage des spécialistes), est assurée par un service technique local capable de répondre en temps réel à la demande sociale.

Cette maîtrise par des non-spécialistes n'est possible que grâce à un interface utilisateur convivial et d'apprentissage immédiat. Cet interface est aussi un outil scientifique à part entière puisqu'il permet d'énoncer clairement ce que l' on était capable que de concevoir bien.

BIBLIOGRAPHIE

– Microsoft Windows Software Development Kit	*Microsoft*
– Microsoft C 5.1 Optimizing Compiler	*Microsoft*
– S.A.G.A. Rapport de stage	Hervé PELLA
– Le langage C	B.W. KERNIGHAN D.M. RITCHIE Masson
– Programme WATASI Version 5	J. WOLSACK