Cerema

Direction territoriale Ouest

Dossier de conception

d'ISAUr

2/4/2014

Table des matières

1 Lexique

1.1 Termes métier

- **Aménagement routier :** disposition constructive routière destinée soit à réduire les vitesses, soi à améliorer les échanges, soit à réduire les bruits et la pollution.
- **Chaussée** : partie bornée d'une route ; on distingue la chaussée droite qui parcourt toute la route dans le sens des PR croissants ; la chaussée gauche est la partie distincte de la chaussée droite, que l'on parcourt lors de chaque séparation de chaussée, dans le sens inverse des PR. Une chaussée est délimitée à droite et à gauche par une ligne épaisse continue.

- **Voie** : une voie est une partie de chaussée délimitée par une ligne de la voie suivante ou de la voie en sens inverse. La BAU est une voie de service, qui peut exister à droite de la ligne de bord de la chaussée.
- **Route** : Ensemble identifié par un gestionnaire, de façon unique, allant d'un point A à un point B, en parcourant une chaussée unique et/ou droite dans le sens des bornes kilométriques croissantes.
- **Axe routier** : partie continue ou discontinue de route. Un aménagement routier est constitué d'un certain nombre d'axes routiers entre lesquels s'échangent des flux de circulation.

1.2 Termes propres au logiciel

- **Editeur graphique de page** : tout ce qui est montré dans ISAUr, à l'exception de la cartographie du bruit, se trouve dans une couche de calque d'un éditeur graphique, qui permet de placer des textes, des images, des dessins divers tels que des cadres ; lignes, flèches, etc.) . Même pour les états imprimables, les éléments sont tous des entités graphiques mémorisables. La présentation de cet éditeur graphique se trouve dans un document indépendant.
- **Fond (de carte)**: Couche géographique hébergeant des images ECW de préférence. Ces images géocodées, sont optimisées pour fournir des fond de calque « satellites » à tous les zooms d'affichage utiles.
- **Carte**: Cette couche permet l'affichage et l'édition de réseaux routiers. Elle offre des outils qui assurent l'identification et la cohérence des axes routiers.
- **Fichier .ecw** : fichiers géographiques représentant la vue en surface d'un territoire rectangulaire; un ou plusieurs fichiers .ecw forment un fond cartographique de représentation d'éléments géocodés (contours, polylignes, rectangles).
- **Magnéto** : allusion au magnétoscope, represente au minimum une flèche pour avancer et une pour reculer (d'une photographie, d'un schéma, d'un document, etc.).
-

2 Introduction

En l'absence d'une analyse fonctionnelle préalable, le présent document présente les processus d'ISAUr, et leur implémentation dans les sources.

Ce document reprend les fonctions d'ISAUr utiles dans la version ISAUr-CITEDYNE. Les fonctions omises, sont :

- la gestion des récepteurs de bruit,
- les calculs des coefficients de propagation,
- la cartographie de bruit,
- les appels à la DLL de SYMUVIA,
- les différents états de calculs statistiques et comparés sur les variantes.

Il est destiné à permettre la compréhension et la maintenance de ces fonctions. Ces fonctions sont :

- les entrées sorties d'ISAUr,
- la création et modification d'un aménagement routier dans le contexte géographique,
- les saisies dans le contexte de saisie des conditions de circulation et des trafics.

En annexe, on trouvera les modèles de données et les classes qui les implémentent.

3 L'éditeur graphique et les figures

3.1 La gestion graphique

Le document externe « Editeur graphique 2D » explique l'essentiel des classes de base d'ISAUr, à savoir :

- les classes de « document-view » de l'application (CMonApp, CMainFrame, CDocGraphique enrichie par CMonDoc, CVueGraphique enrichie par CMaVue) L'éditeur est enrichi ici par la possibilité de gérer un domaine géographique, en plus d'un domaine « papier ».
- les classes d'objets graphiques gérés dans des listes qui constituent des couches.

La classe CFigure est la classe de base de tout objet graphique, dont fait partie la carte. Les autres objets graphiques (textes, rectangles, polylignes, etc.) sont utilisées dans les procès verbaux d'essais.

L'éditeur graphique gère 3 systèmes de coordonnées : le domaine géographique, le domaine « papier » correspondant, et le domaine des pixels de l'écran .L'édition géographique est utilisée dans le contexte géographique d'ISAUr, ainsi que dans le contexte de saisie des données de trafic. Le domaine papier sert essentiellement à la gestion des ascenseurs de la vue CMaVue. ISAUr l'utilise pour l'affichage et l'impression des procès-verbaux de résultats. L'éditeur graphique ne gère pas la cartographie des bruits, qui est en mode OPENGL.

3.2 Gestion de l'édition géographique

La gestion du domaine géographique consiste à utiliser les relations entre 3 systèmes de coordonnées : le domaine géographique, le domaine « papier » correspondant, et le domaine des pixels de l'écran.

La classe CFigure, classe de base des objets graphiques de l'éditeur, contient les principales fonctions géographiques (statiques pour la plupart), et les coefficients utiles au calculs entre les différents systèmes de coordonnées (pixels, papier et géographique). C'est la classe CVueGraphique, étendue par la classe CMaVue qui gèrent les actions.

La préparation du système de coordonnées géographique est faite par la fonction ;

```
static void      FigureSetDomaineGeographique(CRect &RectDomaineXY, CRect
                                              &RectDomaineUL,
                                              long lPrecision)
```

Cette fonction installe le système de coordonnées géographique, par rapport à un domaine papier qui est quasiment aux limites du système graphique Windows (dimension de 32000). Elle affecte les variables suivantes :

Cfigure::mRectDomaineXY, qui est le domaine géographique

Cfigure::mRectDomaineUL, domaine UL correspondant au domaine géographique

CFigure::mlPrecision égal à un pour une précision de 1 mètre géographique, ou 100 pour une précision au centimètre.

Cfigure::mdRatioMXYparUL, nombre de mètre terrain pour 0.1 mm papier (papier=domaine graphique utilisateur au sens Windows).

Elle initialise la variable mbModeGeographique à TRUE. Cette variable est très utilisée dans les fonctions de dessin des figures.

Avant toute action graphique, la classe CVueGraphique, étendue par la classe CMaVue appelle la fonction suivante :

```
static void      FigureInitialiserGeographiquement(CDC *pDC,
                        CRect &RectEcranPixel,
                        etc.)
```

Cette fonction installe l'environnement graphique de l'écran en pixel et calcule le coefficient CFigure::mdRatioPixelParMXY qui est le nombre de pixels de l'écran par mètre géographique.

Ainsi, grâce aux différents systèmes de coordonnées et aux deux coefficients CFigure::mdRatioMXYparUL et CFigure::mdRatioPixelParMXY, il est possible de calculer toutes les coordonnées d'un système vers l'autre. C'est le rôle des fonctions suivantes :

```
static void          ULversXY(CRect &RectULversXY);
static void          ULversXY(CPoint &Pt);
static void          ULversXY(int &iLongueur);
static void          ULversXY(long &lLongueur);
static void          DPversXY(CRect &RectDPversXY);
static void          DPversXY(int &iLongueur);
static void          DPversXY(long &lLongueur);
static void          XYversUL(CRect &RectXYversUL);
static void          DPversUL(CRect &RectDPversUL);
static void          XYversDP(CRect &RectXYversDP);
static void          ULversDP(CRect &RectULversDP);
static void          ULversDP(long &lLongueur);
static void          DPversXY(CPoint &Pt);
static void          XYversDP(CPoint &Pt);
static void          XYversDP(int &iLongueur);
static void          XYversDP(long &lLongueur);
static void          ULversDP(CPoint &Pt);
static void          DPversUL(CPoint &Pt);
static void          XYversUL(CPoint &Pt);
void XYversDP();
```

où DP est le système en pixels écran, UL est le système papier et XY est le système géographique.

3.3 La carte

La carte hérite de la classe de base CFigure. Dans le contexte d'extraction géographique d'ISAUr, la carte est constituée d'une liste de chaussées (classes CChaussee), où chaque chaussée est une suite ordonnée de sections CSectionCarto qui sont une suite ordonnée de points.

Dans le contexte des saisies de données de trafic et de bruit, la carte renvoie directement au projet ISAUr et à sa variante active. La fonction de dessin est celle de la variante active (classe CvarianteISAUr) .

4 Les classes ISAUr

ISAUr comprend un seul projet d'aménagement routier, de classe CprojetISAUr. Un projet ISAUr contient une ou plusieurs variantes de classe CvarianteISAUr.

4.1 La classe de base CObjetSAUr

Tout élément enregistrable du projet ISAUr s'appuie sur cette classe de base, générique. Seuls le contexte d'extraction géographique et d'édition de rapports sont autonomes. Cette classe contient la plupart du temps une liste d'objets dits enfants. On trouve en annexe toutes les classes du modèle des données.

Cette classe s'appuie sur la classe CMetaDonneesObjet via l'attribut miIDtype et la variable mpMetaDonnees (qui pointe sur l'instance de classe CMetaDonneesObjet) , pour gérer la liste des attributs propres à chaque objet ISAUr.

Les principaux attributs sont :

CString **msID**, ID unique de l'objet (concaténation du type + n°)

Cstring **msLibelle**, Libellé donné par l'utilisateur

DWORD **mdwEtat**, états de l'objet (OI_ETAT_MODIFIE, OBJET_ISAUR_EST_SELECTIONNE ,OBJET_ISAUR_EST_ACTIVE, OI_ETAT_A_SUPPRIMER)

DWORD **mdwProprietes**, propriétés de l'objet (OBJET_ISAUR_SANS_LISTE, OBJET_ISAUR_EST_UNE_LISTE, OBJET_ISAUR_NE_PAS_DETRUIRE_AVEC_LISTE, OBJET_ISAUR_EST_INACTIVABLE)

CObjetISAUr ***mpObjetPere** pointe sur l'instance de classe CObjetISAUr dans lequel est contenu l'objet

CMetaDonneesObjet ***mpMetaDonnees**, pointe sur l'instance de méta-données de l'objet

CvarianteProjetISAUr ***mpVariante** pointe sur la variante du projet qui contient l'objet.

CobjetISAUr contient aussi quelques variables statiques accessibles de n'importe quel endroit du logiciel :

static CString **msCheminConfiguration** est le chemin du fichier des configurations ISAUr.cfg

static char **mcSeparateur** est le séparateur des données, utilisé dans ISAUr.cfg

static CISAUrTreeCtrl ***mpArbre** pointe sur l'arbre de présentation/Saisie

static CDataGridCtrl ***mpGrille** pointe sur la Grille de présentation/Saisiedes attributs

static CprojetISAUr ***mpProjetISAUr** pointe sur le projet (liste de variantes)

static CView ***mpVueActive** pointe sur la vue active (déterminée par le choix des onglets)

static bool **mbGarderZoom** TRUE s'il faut garder le zoom en changeant de couche

static CRect **mDernierRectangleAffichage** contient les dimensions d'affichage en gardant le zoom

static CDocument ***mpDocumentCommun** pointe sur le document commun à toutes les vues (architecture un document ; plusieurs vues)

static CView ***mpVue**[10] pointe sur chaque vue

static int **miOutilVue**[10] outil actif en cours pour les vues

static CString **msNomVues**[10] noms des vues

static CObjetISAUr ***mpObjetISAurSousMenusClicDroit** pointe sur l'objet ISAUr cliqué

static CObjetISAUr ***mpObjetISAurSurvole** pointe sur l'objet ISAUr (graphique) survolé par le souris

static CObjetISAUr ***mpObjetISAurSelectionne** pointe sur le dernier objet ISAUr (graphique) sélectionné

static CPointXY **mPointXYClicDroit** contient les coordonnées géographiques du point au clic droit de souris

static CObjetISAUr ***mpObjetISAurCopie** pointe sur le dernier objet copié

static bool **mbClicDroitSurArbre** true si le clic droit a eu lieu sur un élément de l'arbre des données

static bool **mbPropagationSurSections** true : les coeffts de propagation sont attachés aux sections, ce qui est le cas normal (évolution/optimisation d'ISAUr).

Les principales fonctions sont :

Quelques commandes statiques :

static CObjetISAUr ***pGetNewObjetISAUr**(long lIDtype, CObjetISAUr *pObjetPere ; etc.) est la fabrique d'objet ISAUr, dont le type est fourni. Cette fonction est fortement appelée à la lecture d'un aménagement depuis des fichiers.

static CMetaDonneesObjet ***pGetMetaDonnees**(long IDtype) renvoie le pointeur sur une instance des métadonnées correspondant à IDtype (liste dans objetISAUr.h et le fichier ISAUr.cfg)

static void **InitialiserLeContexte**(...) permet de mémoriser des éléments fréquemment appelés

et mis en données statiques

static bool **bCommandeEstActive**(DWORD dwCommande) false, si la commande n'est pas accessible

static void **AgirSurCommande**(DWORD dwCommande) pour une action de menu contextuel généralement

static void **DetruireObjetClicDroit**() pour détruire le dernier objet cliqué ainsi

static bool **bSaisiePossible**() false si un calcul de propagation ou une simulation est en cours, pour

interdire toute modification des données

static bool **bGetInfosFormat**(CString sFormat, int &iNoFormat, CString &sInfo, int &iInfo) fournit des informations déduites du format de saisie d'un attribut de n° d'ordre iNoFormat (ce format est ecrit à la suite de chaque attribut dans le fichier ISAUr.cfg

Fonctions enrichies et traitées par la plupart des objets :

Fonctions générales

virtual ~**CobjetISAUr**() qui permet notamment la suppression d'enfants éventuels

virtual void **vSetID**() fonction qui attribue un ID unique lors de la création d'un objet

virtual void **vInitialiser**() initialise les attributs. Obligatoire pour tous les objets

virtual void **vVider**() fonction de vidage, appelée notamment à la destruction d'un objet

virtual CObjetISAUr ***vpDupliquer**(CObjetISAUr *pObjetISAUr)

virtual void **vMarquerLaModification**(bool bModifie) met l'état de l'objet à OI_ETAT_MODIFIE

virtual bool **vbEstModifie**() renvoie true si l'état contient OI_ETAT_MODIFIE

virtual bool **vbEstUneListe**() true si l'objet contient une liste d'enfants

virtual bool **vbDetruireAvecParent**() true si l'objet est un enfant et peut être détruit lors de la destruction du parent

virtual bool **vbContenuInitialise**() true si les enfants sont initialisés

virtual long **vlGetIDtypeEnfant**(CString sTitre = _T(""), bool bExclusivite = false) renvoie le type d'enfant contenu dans le parent

virtual POSITION **AddTail**(CObjetISAUr *pObjetISAUr) ajoute un enfant en fin de liste

virtual POSITION **AddHead**(CObjetISAUr *pObjetISAUr) ajoute un enfant en début de liste

virtual CObjetISAUr ***vpGetNewObjetISAUr**(bool bDupliquer) créée un objet de même type, dupliquant this ou non (utile en copie)

virtual CObjetISAUr ***vpGetParent**() Objet parent et containeur de this

Fonctions en lien avec le référentiel (dictionnaire)

virtual bool **vbLireLaReferenceXML**() uniquement pour les objets du référentiel (véhicule type, sol type), permet de lire en fichier depuis le référentiel (=dictionnaire)

virtual bool **vbEcrireLaReferenceXML**() pendant de la précédente, pour l'écriture

virtual CObjetISAUr ***vpGetReference**() renvoie l'objet de référence pour un objet ISAUr initialisé à partir du référentiel

Fonctions concernant l'affichage sur l'arbre

virtual HTREEITEM **vhCreerLaBranche**(HTREEITEM hItemParent, bool bCreerLesBranchesEnfants, bool bOuvrir = false, HTREEITEM hInsertAfter = TVI_LAST) crée une branche pour l'objet, y ajoute éventuellement ses enfants, et en déploie éventuellement l'affichage.

virtual void **vCreerLesBranchesDesEnfants**(HTREEITEM hItemParent, bool bCreerLesBranchesEnfants) appelée pour créer les branches des enfants

virtual bool **vbEstSurArbre**() true si l'objet est affiché sur l'arbre

Fonctions pour la saisie/modification des données

bool **bEstModifie**() true si l'objet a été modifié

bool **bSelectionner**(BOOL bAfficherLesOnglets) sélectionne l'objet sur l'arbre et le dessin si l'objet est graphique

virtual void **vSelectionnerLesEnfants**(bool bSelection) sélectionne les enfants de l'objet, uniquement sur la vue graphique

virtual bool **vbSurSelection**(BOOL bAfficherLesDonnees), appelé après un selection par la souris, valide la saisie en cours et prépare l'affichage

static void **DeSelectionner**() désélectionne un objet (sur arbre et dessin)

virtual bool **vbAUnElementSelectionne**() true si l'objet ou un de ses enfants est sélectionné

virtual void **vAfficherLesOnglets**() /affiche les onglets en fonction de l'objet sélectionné

virtual bool **vbGetMetaDonneesObjet**(CString &sLibelleObjet, UINT &uNbreData,
CString &sDescription) renseigne sur les attributs

int **iGetNombreDonneesSaisies**() fournit le nombre d'attributs à saisir

virtual CString **vsGetLibelleData**(UINT uNoData) libellé de la donnée de n° d'ordre uNoData, en commençant par 1

virtual bool **vGetMetaDonneesData**(UINT uNoData, bool &bModifiable, bool &bEstUneListe,
CObjetISAUr *pObjetAssocie = NULL) renseigne sur la donnée de n° d'ordre uNoData,

virtual bool **vbGetLexique**(CString sLibelleData, CString &sLexique, bool &bProtegerLaSaisie,
char cSeparateur = _T(';')) fournit un lexique des valeurs pour l'attribut, lu dans ISAUr.cfg

virtual bool **vbGetLexique**(UINT uNoData, CString &sLexique, bool &bProtegerLaSaisie,
char cSeparateur = _T(';')) fournit un lexique des valeurs pour l'attribut, lu dans ISAUr.cfg

virtual bool **vbGetLexiqueDynamique**(UINT uNoData, CString sFormat, long lIDTypeObjetDemandeur,
CString &sLexique, bool &bProtegerLaSaisie,
char cSeparateur) fournit un lexique des valeurs pour l'attribut, ici, c'est le type d'objet qui décide du lexique à fournir

virtual bool **vbGetLexiqueDesEnfants**(CString sFormat, long lIDTypeObjetDemandeur,
CString &sLexique, bool &bProtegerLaSaisie,
char cSeparateur) fournit par défaut la liste des enfants

virtual bool **vbGetMenuSurClicDroitSouris**(bool bSurArbre, CString &sMenu, char cSeparateur)

virtual bool **vbPreparerLesDonneesDeSaisie**(int iNbreData, CstringArray &Libelles,

CStringArray &mFormats,

CStringArray &LexiquesEnCombo,

CStringArray &Aides,

CWordArray &Proprietes) renvoie toutes les informations pour organiser la saisie des attributs dans la grille

virtual bool **vbSituerSurDisque**(bool bCheminDeFichier, int iNoData, bool bLecture,

CString &sChemin) cette fonction permet de saisir un attribut qui est un chemin de fichier (ex : une image)

virtual bool **vbVerifierSurDisque**(bool bCheminDeFichier, int iNoData, bool bLecture, CString &sChemin) vérifie que le chemin de fichier correspond bien aux données attendues

virtual bool **vbGetData**(CString &sData, char cSeparateur = _T(';')) récupère les données saisies pour l'objet, modifiées ou non

virtual bool **vbSetValeur**(CString &sValeur, int iNoData, bool bValider = false,

bool bModifier = true, bool *pbModif = NULL,

bool *pbModifierLesAutresCellules = NULL) données de l'objet

virtual bool **vbValiderLesDonneesSaisies**(CString sData, bool &bModif, int &iBadData) pour un attribut de libellé sData, contrôle si la saisie est valide, et renvoie le n° de d'ordre de donnée à resaisir (qui peut être différent du n° d'ordre de la donnée à vérifier)

bool **bAffecterLesDonneesSansControle**(CString sData, bool bMarquerLaModification) affecte une donnée saisie sans vérification ; il y a des cas où des données ne peuvent être validées que par des données qui n'ont pas encore été saisies (attributs inter dépendants)

virtual void **vTransmettreAuxEnfants**(int iNoData) utile au cas où une donnée impacte les enfants (ex : la modification du nombre de voies d'une section implique des modifications dans les enfants, qui sont les voies)

virtual void **vTransmettreAuParent**(int iNoData, CString sValeur) utile, dans le cas une donnée impacte un parent ou ses enfants

virtual bool **vbSurModificationDeDonnee**(CString sValeur, int iNoData, CDataGridCtrl *pGrille) permet de faire des actions immédiates lors de la modification d'une donnée (ex : coloration sur modification, d'un sol, déplacement d'une image,,,)

virtual void **vSurModificationDeDonnees**() sert notamment à impacter une variante, suite à la modification de données (ex : invalidation des calculs de propagation pour qu'ils soient réactualisés)

bool **vbVerifierLeFormat**(CString sFormat,

CString &sValeurAverifier,

CString &sFourchette,

CString &sLibelle, CString *psMessage) vérifie la valeur sValeurAverifier par rapport au format spécifié dans le fichier ISAUr.cfg (valeur maxi par exemple)

Cstring **sMessageControleEnParametres**(int iNoData, CString sData, bool bAfficher);

Cstring sMessageControle(int iNoData, CString sData,

bool bAfficher) Formatte un message d'erreur ; ce message peut être issue du fichier ISAUr.cfg

bool **bVerifier**(CString sValeur, int iNoData, bool bObligatoire, int iNbreCarMini,
CString &sAttribut, bool bModifier, bool *pbModif = NULL,
bool bAvecMessage = true) vérifie une donnée textuelle selon son format, indique si elle a été effectivement modifiée et affiche un message en cas d'erreur, si bAvecMessage est true

bool **bVerifier**(CString sValeur, int iNoData, bool bObligatoire,
bool bTestValeurInferieure, long lValeurInferieure,
bool bTestValeurSuperieure, long lValeurSuperieure,
long &lAttribut, bool bModifier, bool *pbModif = NULL,
bool bAvecMessage = true) vérifie une donnée numérique de type long selon son format, indique si elle a été effectivement modifiée et affiche un message en cas d'erreur, si bAvecMessage est true

bool **bVerifier**(CString sValeur, int iNoData, bool bObligatoire,
bool bTestValeurInferieure, WORD wValeurInferieure,
bool bTestValeurSuperieure, WORD wValeurSuperieure,
WORD &wAttribut, bool bModifier, bool *pbModif = NULL,
bool bAvecMessage = true) vérifie une donnée numérique de type WORD selon son format, indique si elle a été effectivement modifiée et affiche un message en cas d'erreur, si bAvecMessage est true

bool **bVerifier**(CString sValeur, int iNoData, bool bObligatoire,
bool bTestValeurInferieure, int iValeurInferieure,
bool bTestValeurSuperieure, int iValeurSuperieure,
int &iAttribut, bool bModifier, bool *pbModif = NULL,
bool bAvecMessage = true) vérifie une donnée numérique de type int selon son format, indique si elle a été effectivement modifiée et affiche un message en cas d'erreur, si bAvecMessage est true

bool **bVerifier**(CString sValeur, int iNoData, bool bObligatoire,
bool bTestValeurInferieure, double dValeurInferieure,
bool bTestValeurSuperieure, double dValeurSuperieure,
double &dAttribut, bool bModifier, bool *pbModif = NULL,
bool bAvecMessage = true) vérifie une donnée numérique de type double selon son format, indique si elle a été effectivement modifiée et affiche un message en cas d'erreur, si bAvecMessage est true

Fonctions de gestion des actions sur commande de menu (en général)

virtual DWORD **vdwGetCommande**(int iPosition, CString sItemDuMenu, bool bSurArbre) renvoie l'ID de la commande, soit, par défaut, selon la position de la fonction dans le menu, soit en

fonction de sa désignation

virtual bool **vbAgirSurCommande**(DWORD dwCommande) répond à l'activation d'une commande (en général via le clic droit de souris sur un item de l'arbre, ou sur la vue graphique)

virtual bool **vbCommandeEstActive**(DWORD dwCommande) est à true si la commande est active

virtual bool **vbAgirSurFinDeCommande**(DWORD dwCommande, UINT nFlags, CPoint PointXY, CFigure *pFigure = NULL) pour certaines commandes, enclenchées dans un premier temps, attendent une position de la souris pour se terminer

bool **bSurClicDroit**(bool bSurArbre, UINT nFlags, CPointXY PointXY, CPoint Point)

virtual bool **vbSurMenuObjet1**()

virtual bool **vbSurMenuObjet2**()

virtual bool **vbSurMenuObjet3**()

virtual bool **vbSurMenuObjet4**()

virtual bool **vbSurMenuObjet5**()

virtual bool **vbSurMenuObjet6**() ces fonctions répondent à des commandes propre à chaque type d'objet ISAUr, annoncées dans le fichier ISAUr.cfg

Fonctions liées aux commandes générales (ajouter, supprimer,...)

virtual bool **vbAjoutEstPossible**(bool bObjetCopie) true si un ajout d'objet est possible, par copie d'un autre ou par création pure

virtual bool **vbSuppressionEstPossible**() true si la suppression de l'objet est possible

virtual bool **vbAjouterUnEnfant**(long lIDType = -1) ajoute un enfant ; lIDType doit être spécifié si this peut avoir plusieurs types d'enfants différents

virtual bool **vbAjouterUnEnfant**(long lType, bool bAjouterSurArbre, bool bSelectionner) ajoute un enfant en précisant s'il faut l'ajouter sur l'arbre et éventuellement le sélectionner ; lIDType doit être spécifié si this peut avoir plusieurs types d'enfants différents

virtual CObjetISAUr ***vpAjouterUnEnfant**(long lType, bool bAjouterSurArbre, bool bSelectionner) idem fonction précédente, sauf qu'elle renvoie un pointeur sur l'enfant créé

virtual bool **vbAjouterUnEnfantAvant**(CObjetISAUr *pObjetSuivant, long lType, bool bAjouterSurArbre, bool bSelectionner) idem vbAjouterUnEnfant, mais permet d'insérer l'enfant à une place donnée, dans la liste des enfants

virtual bool **vbAjouterUnObjetACopier**(CObjetISAUr *pObjetISAUr) ajoute un enfant en copiant pObjetISAUr

virtual bool **vbSupprimerUnEnfant**(CObjetISAUr *pEnfant) supprime l'enfant pEnfant

virtual bool **vbSupprimerLesEnfants**(bool bReafficher = false) supprime tous les enfants

4.2 La classe du Projet ISAUr CprojetISAUr

Le projet ISAUr contient la liste de toutes les variantes d'un même aménagement routier. Les variantes peuvent différer géométriquement ou plus simplement par les hypothèses de données. Le projet contient aussi une liste de récepteurs et le domaine géographique commun aux variantes.

Les principaux attributs sont :

```
CString msNoDuDossier;
CString msNomDuDossier;                 //! données plus techniques
CString msChemin;                       //! Chemin d'accès des données
CString msCheminSYMUBRUIT;              //! Chemin d'accès de SIMUBRUIT
int     miNature;                       //! Nature du dispositif
int     miInitialisation;               //! Mode de création du projet 0:manuel, etc.
Cstring msCheminSourceCARTO;            //! Chemin du fichier rcr (peut être créé à côté du projet)
static CString  msCheminImages;
Cstring msDesignationDeVarianteEnCreation;

//! Domaine géographique général du projet incluant normalement toutes les variantes
CRect  mDomaineGeographique;            //! Domaine maxi géographique

CRect  mDomaineCartographie;            //! Domaine calculé en propa mDomaineGeographique par défaut

CDictionnaireISAUr  *mpDictionnaire; //! Références (sols, véhicules)

CStringList     mListeDesVariantes;     //! Liste ordonnée des variantes

CRecepteurs   mListeDeRecepteurs;       //! Récepteurs fixes definis sur une variante par un outil Oreille
```

Les principales fonctions sont :

bool **bCreerUneVariante**(bool bAjouterAuProjet)

Elle est appelée par la fonction bDupliquerLaVarianteActive(), dont le rôle est de créer une varainte dont on fera varier les données.

Les fonctions **bLireLesVariantes**(CString sChemin) et **bEcrireUneVariante**(CVariante**Projet**ISAUr *pVariante**Projet**ISAUr) s'appuient sur les fonctions virtuelles **vLireLesAttributsXML**(Generic::CXmlNodePtr &pNoeud), **vEcrireLesAttributsXML**(Generic::CXmlNodePtr &pNoeud) et vbSerializeXML(Generic::CXmlNodePtr &pNoeudPere) que l'on va trouver dans toutes les classes issue ce CobjetISAUr. Ces classes servent à l'enregistrement XML du projet, de ses variantes, ainsi qu'à l'enregistrement des données du référentiel, appelé aussi dictionnaire.

Les fonctions de modification des données

```
virtual bool vbGetData(CString &sData, char cSeparateur = _T(';'));     //! données de
```

l'objet

void GetDataSaisies(CString &sData, char cSeparateur,
CString &sLibelle, int &iNature,
CString &sNoDossier, CString &sNomDossier,
CString &sChemin);

virtual bool vbValiderLesDonneesSaisies(CString sData, bool &bModif, int &iBadData);

virtual bool vbSurModificationDeDonnee(CString sData, int iNoData, CDataGridCtrl *pGrille);

virtual bool vbEstModifie();

5 Implémentation des fonctions géographiques

Dans ce contexte, la classe CMaVue, qui gère les actions de l'utilisateur, utilise la fonction HitTest pour déterminer la chaussée et la section sur laquelle l'utilisateur veut exercer une action (via les fonctions CMaVue::OnButtonDown, CMaVue::OnButtonUp, CMaVue::OnMouse Move, CmaVue::OnButtonDblClk).

6 Implémentation des processus d'entrées-sorties

Cette partie présente :

- les classes et fonctions qui lisent et écrivent les données géographiques,
- les classes qui portent les objets ISAUr, leur création, enregistrement et lecture,
- les classes qui concernent le modèle SYMYVIA, et l'enregistrement du fichier pour SYMUVIA.

La classe de base de l'objet ISAUr « CObjetISAUr »

Les principaux attributs sont :

Les principales fonctions sont :

La classe de base de l'objet graphique ISAUr « CObjetGraphiqueISAUr »

Les principaux attributs sont :

Les principales fonctions sont :

7 Implémentation des processus de création et de modifications géographiques

7.1 Passage dans le contexte géographique

7.2 Sélection des tronçons et ensembles de tronçons

Les principaux attributs sont :

Les principales fonctions sont :

7.3 Réduction de la carte à la sélection

Les principaux attributs sont :

Les principales fonctions sont :

7.4 Ajout d'un fond de carte

Les principaux attributs sont :

Les principales fonctions sont :

7.5 Identifications, attributs de tronçons

7.6 Outils de création et modification géométriques du tronçon

7.7 Outils de tronçons

7.8 Basculement sur le contexte d'aménagement routier

Les principaux attributs sont :

Les principales fonctions sont :

8 Contexte de saisie des données de circulation et de trafic de l'aménagement routier

Ce contexte commence avec ce qui a été préparé dans le contexte géographiques. Il est encore possible de faire des modifications minimes en terme de géométrie. Ce contexte permet surtout d'assurer les trafics des véhicules sur l 'aménagement. Rappelons que les fonctions de calculs de propagation et d'appel à la simulation des trafics sont omises de ce document.

8.1 Modifications géométriques

8.2 Calcul des connexions des voies et saisie des conditions de circulation d'une section à une autre

8.3 Calcul des itinéraires entre entrées et sorties et saisie des trafics