Rapport du Projet CDI 2002

Cnam Amiens

Vincent Lefeuvre

Cnam
CONSERVATOIRE NATIONAL DES ARTS ET METIERS

Introduction

- ## Origine du jeu :

Juin 1985 : Le russe Alexei PAZHITNOV, inspiré par un de ses anciens jeux de pentaminos, crée sur l'ordinateur ELECTRONICA 60 au Centre Informatique de l'Académie des Sciences de Moscou, le jeu Tetris. Quelques jours suffisent à Vadim GERASIMOV, un collègue, pour le porter sur PC.
Tetris est un des rares jeux qui ait atteint une popularité universelle. A la fois très simple et très difficile, il a été adapté sur tous les systèmes informatiques connus de l'homme, et s'est vendu à des millions d'exemplaires à travers le monde, sous tous formats possibles.
En parallèle de son incroyable succès, Tetris a donné lieu à une des plus passionnantes batailles juridiques de l'histoire du jeu vidéo.

- ## Présentation du sujet :

Description du jeu :

Tétris est un jeu d'action dans lequel des pièces descendent de haut en bas de l'aire de jeu. Les pièces s'empilent et le jeu est terminé lorsqu'une pièce touche le haut de l'aire de jeu. Lorsque ces pièces forment une rangée de blocs horizontale et continue, cette rangée disparaît. Comme c'est la seule manière de se débarrasser des blocs, pour jouer longtemps, vous devez essayer de former des rangées continues aussi souvent que possible.
L'objectif de ce jeu est de réaliser un maximum de lignes complètes.

Règle du jeu :

Une pièce Tétris est composées de 4 petits carrés élémentaires. Il existe 7 pièces différentes.
Les pièces apparaissent une à une de manière aléatoire centrée horizontalement en haut de l'aire de jeu et commencent à descendre à une vitesse constante.
Chaque pièce continue à descendre jusqu'à atterrir sur une autre pièce ou au bas de l'aire de jeu. Vous ne pouvez manipuler une pièce que lors de leur descente.

Les mouvements possibles de la pièce courante associés à leur commande clavier sont :

Pivotement de la pièce courante d'un angle de -90°
déplacement de la pièce courante vers la gauche
déplacement de la pièce courante vers la droite
descente rapide de la pièce courante.

Chaque fois que vous formez une rangée de blocs horizontale et continue, cette rangée disparaît. Il s'agit donc d'essayer de manipuler les pièces pendant leur descente de manière à former des rangées pleines et à les faire disparaître. La vitesse de chute des pièces varie avec le temps.
Le jeu se termine lorsque les pièces sont empilées jusqu'en haut de l'aire de jeu. Il est également possible de sortir du jeu à tout moment grâce à la touche clavier Échap.

- ***Conception et mise en œuvre :***

La conception est « orientée objet », l'analyse sera réalisée en UML et le développement en C++. Une attention particulière sera apportée à l'interface graphique dans un premier temps puis à l'interface de jeu.
La conception objet nous permettra plusieurs extensions comme le choix de la taille de la grille de jeu, le choix entre 2 types de pièces, une Option sonore.
En fin de rapport nous traiterons les différentes possibilités d'extensions futures avec leur coût en développement.

- ***Outils Utilisés :***

ClassBuilder : pour l'analyse UML.
Watcom : pour le développement C++.

- ***Utilisation du programme Tetris :***

Plate-forme : Windows 9.x, 2000, XP.
Outil nécessaire : Dos4GW.exe.

- ***Connaissances de l'Auditeur :***

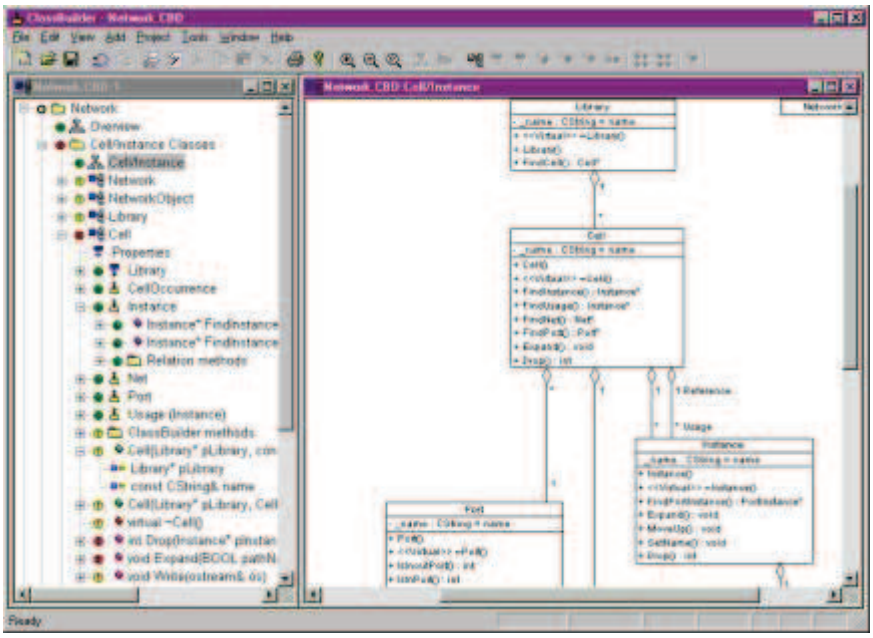
Bonne connaissance du développement Objet :
Développement d'une application « gestion commerciale » composé de 8 modules, avec l'outil Borland Delphi 6 entreprise. Développement de dll, SGDB, Client serveur, ole, basculement des journaux de facturation dans les applications de comptabilité Sage, transferts synchronisés par ole avec l'outil PcAnywhere de Symantec …
Cette application est utilisée par 50 personnes au siège social et 20 en agence. Les mises à jours de bases sont automatisées chaque nuit.

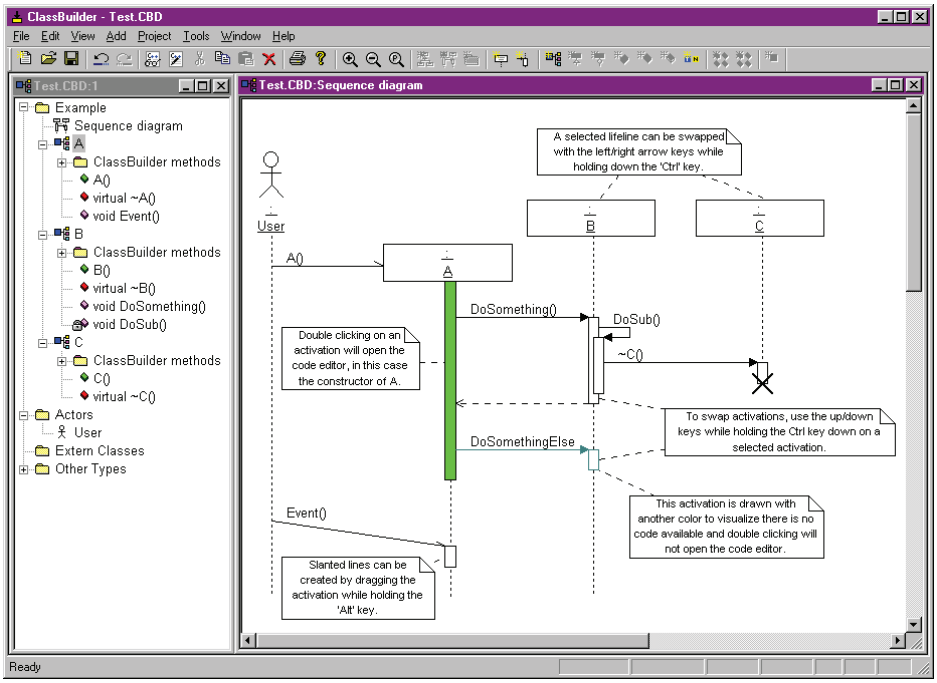
Mes connaissances en C++ sont limitées au cour de programmation du module VARI du CNAM.
Quelques connaissances d'UML mais jamais d'utilisation dans un cas concrète.

Analyse

1. Présentation de l'outil de modélisation UML "ClassBuilder ".

Classbuilder est un outil de modélisation UML puissant et performant qui vous permet de créer vos classes C++ que ce soit en travaillant depuis l'explorateur de classe ou directement à partir du diagramme de classes ou encore du diagramme de séquence.
Classbuilder vous permet donc de créer un projet en C++ de A à Z tout en le documentant, de suivre les phases du développement (analyse, design, implémentation, test, complet). Vous pouvez ensuite générer votre source C++.





ClassBuilder est disponible sous licence GNU Général Public Licence (GPL), avec sources et exemples sur les sites :

http://home.hetnet.nl/~xvenemaj/ClassBuilder.htm
https://sourceforge.net/projects/classbuilder

Une version et aussi disponible sur le cd-rom accompagnant le rapport de projet sous la rubrique outils.

2. Analyse du sujet :

Le sujet peut être traité en 2 grandes parties:

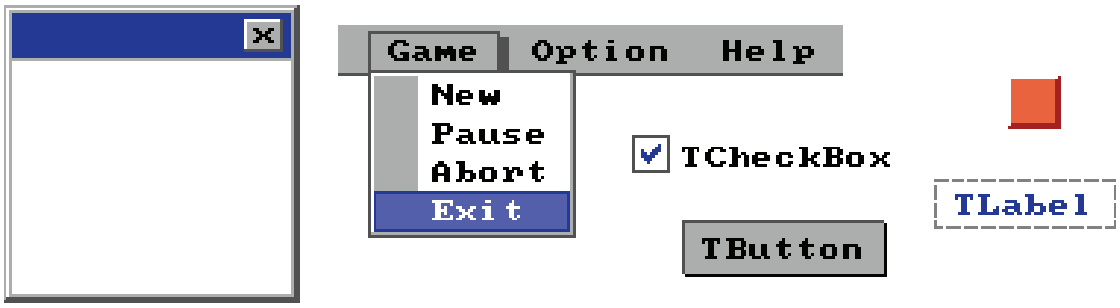
L'interface graphique : nous travaillons sur un projet développé en watcom pour dos, il faudra donc créer une interface visuel pour que l'utilisateur puisse visionner et interagir avec le jeu. (fenêtre, bouton, checkbox, Label, Brick …)

L'interface de Jeu : le jeu est constitué de plusieurs éléments qui vont réagir aux demandes d'actions du joueur et qui vont interagir entre eux. (Grille, pièce, option…).

Une classe supplémentaire ancêtre de tout les objets sera crée : "TObjet". Cette classe sera abstraite.

2.1. L'interface graphique :

Ensemble des composants visuels, c'est-à-dire que l'utilisateur peut les voir et interagir avec eux à l'exécution.



Tout ses objets ont bien entendue des points communs :

Propriétés :

- Coordonnée horizontale.
- Coordonnée verticale.
- Taille horizontale.
- Taille verticale.
- Visible ou non.

Méthodes :

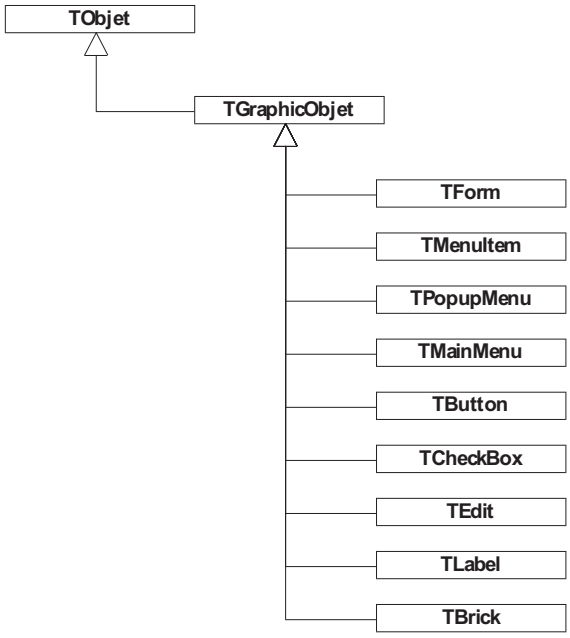
- Constructeur.
- Destructeur.
- Rend un contrôle visible.
- Rend le contrôle invisible.
- Redessine le contrôle à l'écran.

Evénements :

- Déplacement du pointeur de la souris au-dessus du contrôle.
- Clique de l'utilisateur sur le contrôle.

Une classe supplémentaire ancêtre de tout les objets Graphiques sera crée : "TGraphicObjet".

2.1.1. Diagramme de classe :



2.1.2. Etude approfondie de l'objet Menu:

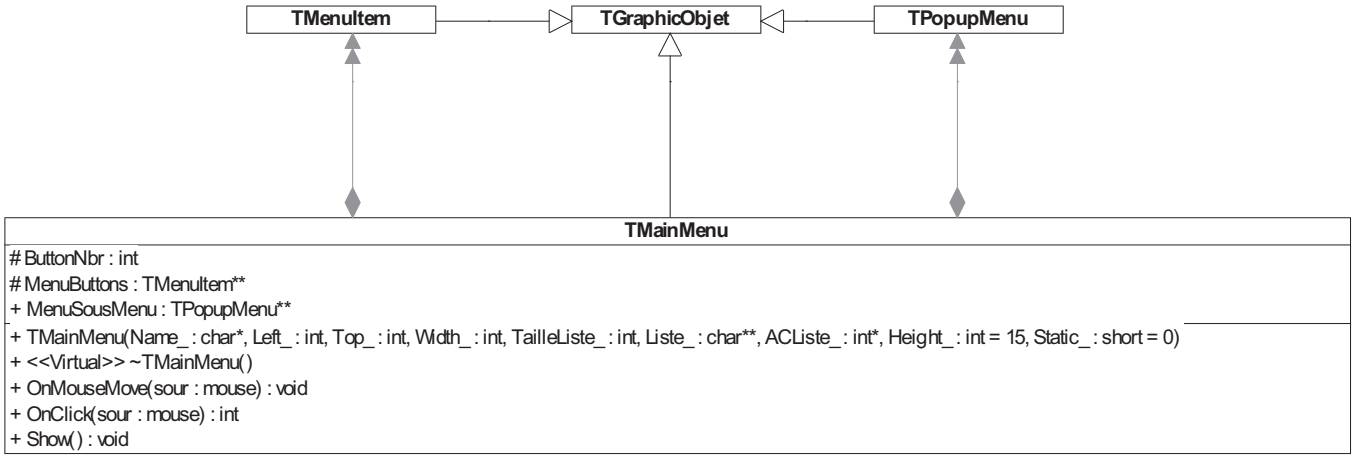
L'objet menu est un objet très complexe, il est donc possible de le décomposer en 3 éléments.

TMenuItem décrit les propriétés d'un élément de menu.

TPopupMenu est un sous menu composé de TMenuItem.

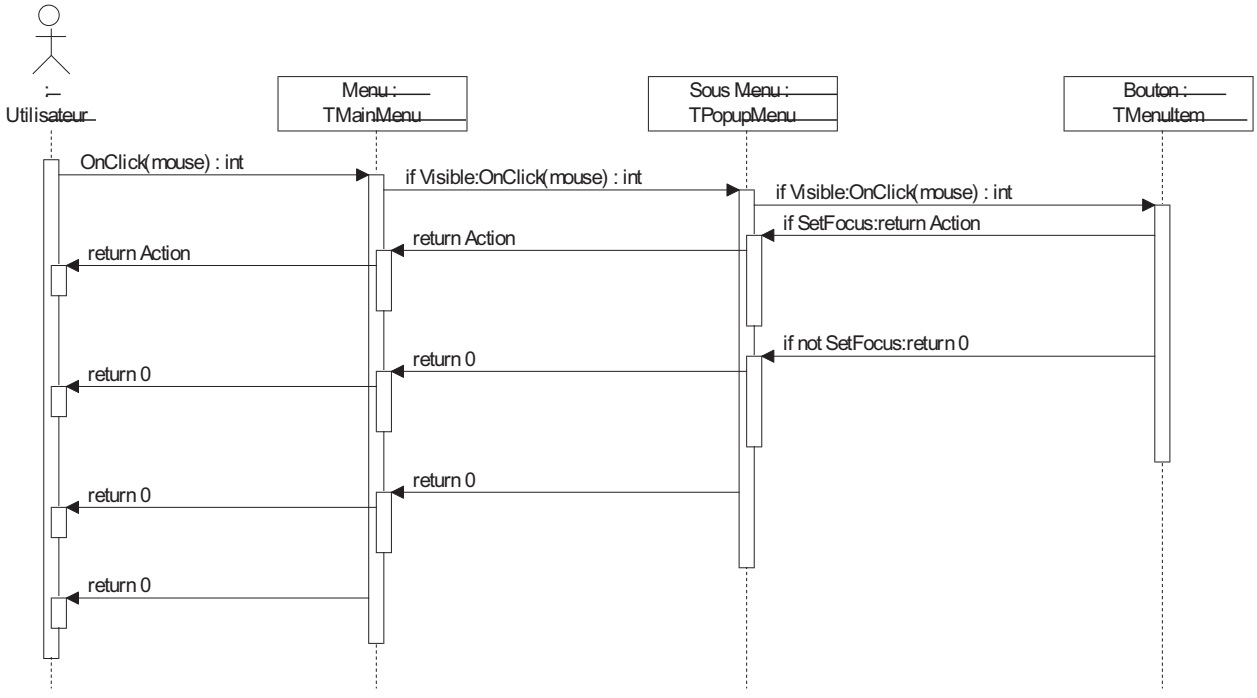
TMainMenu encapsule dans une fiche une barre des menus et ses menus déroulants.

Diagramme de classe :



Il faut donc gérer au seins de l'objet, l'enchaînement des actions de l'utilisateur. Quand l'utilisateur clique sur le menu, TmainMenu répercute l'information sur ses sous-menus. A son tour le sous-menu répercute l'information sur les boutons qui le composent. Le bouton exécute sa fonction et remonte l'information au sous-menu qui la retransmet au menu.

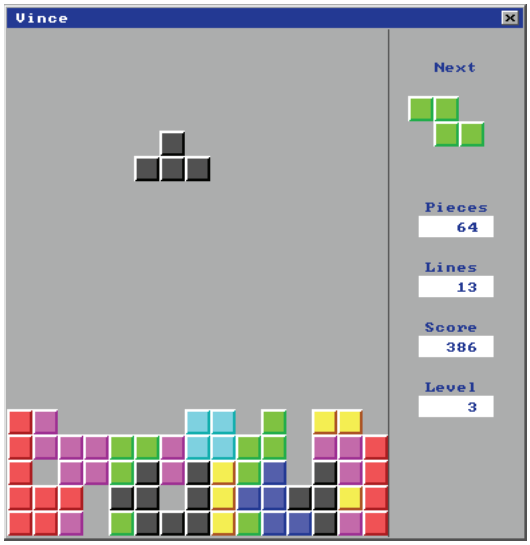
Diagramme de séquence :



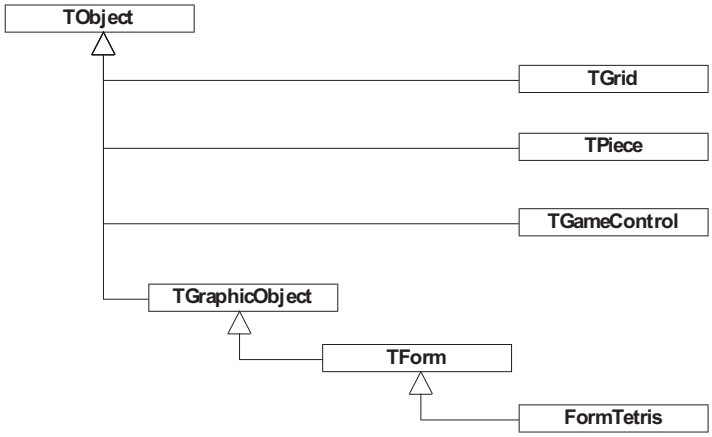
Cette analyse permet de réduire le nombre de vérifications car le traitement des informations est pyramidale.
Cette méthode sera aussi utilisée pour l'affichage du menu et pour la vérification du focus de la souris sur les différents composants du menu.

2.2. L'interface de Jeu :

L'ensemble des composants liés au jeu, déplacement des pièces, Gestion interactive avec le joueur, contrôle de collisions, détection de lignes, notion de temps contrôlé … C'est l'interface minimum pour jouer au Tetris, elle utilise néanmoins l'interface graphique pour l'affichage.



2.2.1. Diagramme de classe :



2.2.2. Etude approfondie de l'objet GameControl:

L'objet GameControl est l'objet central du jeu.
il contrôle :

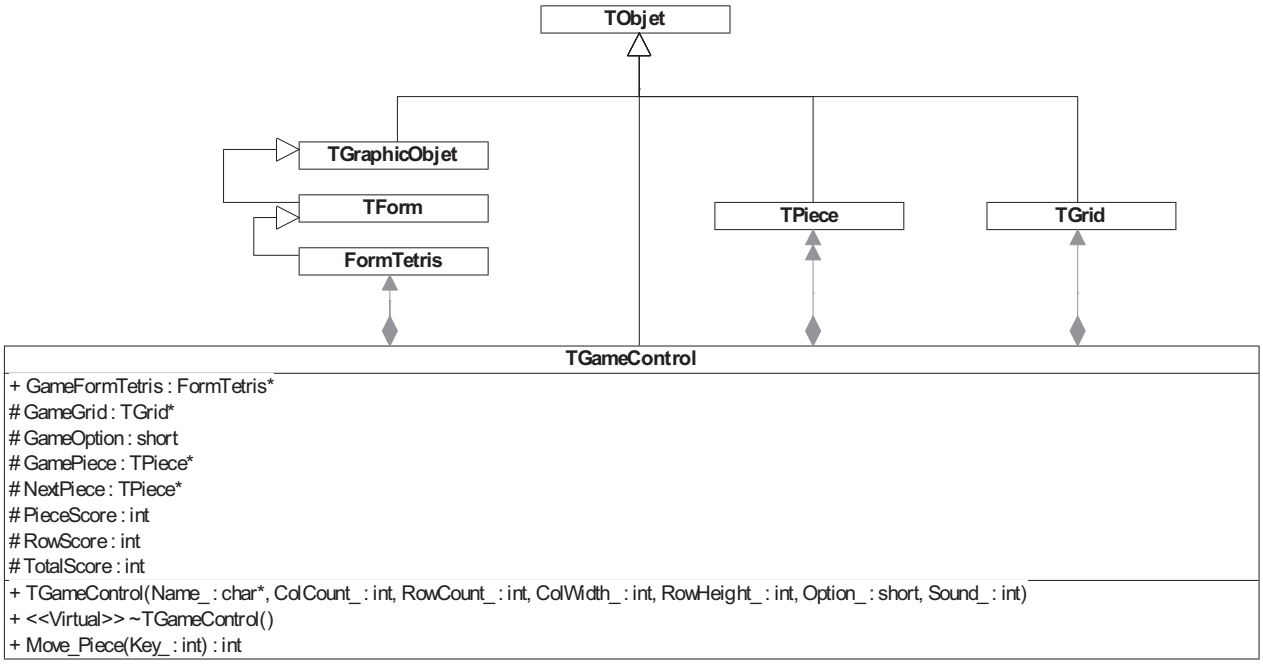
la fenêtre de jeu.
la grille.
la pièce courante, la pièce suivante, la pièce de teste.
Les scores.
La notion de temps.
L'accès clavier.
Les options.

Exemple :

quand l'utilisateur appuis sur la touche ⇑ pour faire pivoter la pièce courante :

GameControl génère une pièce teste identique à la pièce courante.
Il demande à la grille de vérifier cette pièce.
Si la réponse est positive il affecte à la pièce courante les paramètres de la pièce teste.
Si la pièce test est en collision latérale il ne change pas la pièce courante .
Si elle est en collision par le bas, il importe la pièce dans la grille de jeu, vérification de ligne complète, la pièce courante prend les paramètres de la pièce next et la pièce next est rafraîchie. Sans oublier l'incrémentation du score et la gestion de la vitesse de chute.

Diagramme de classe :



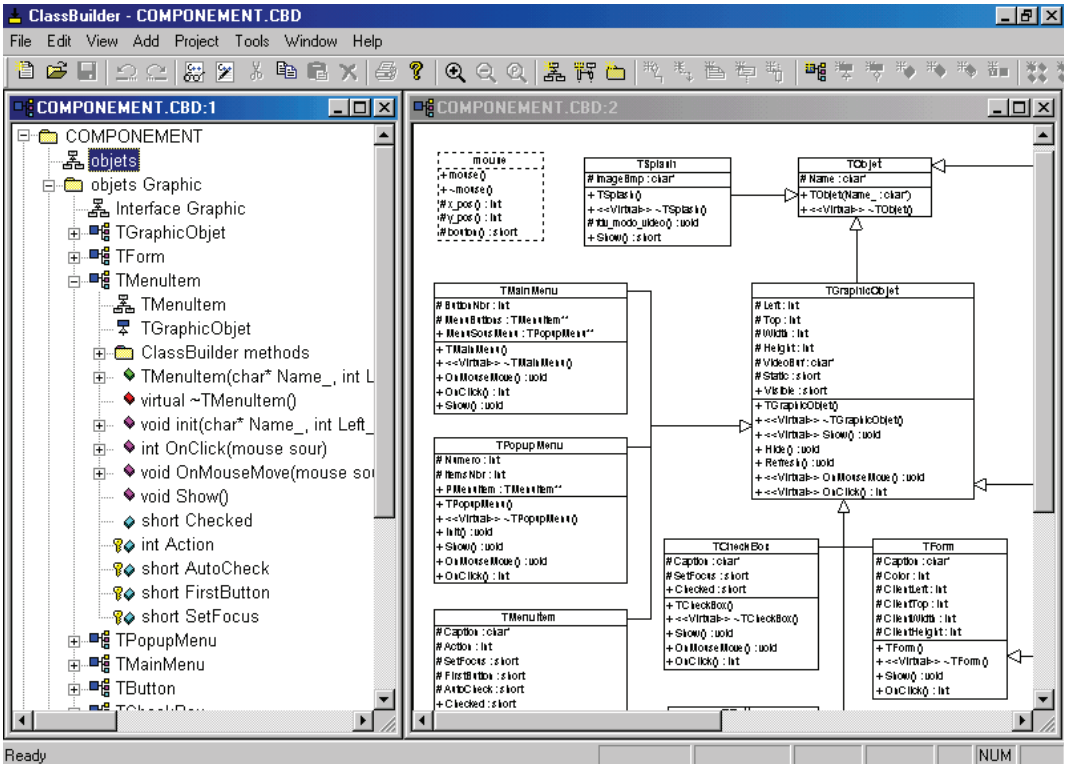
Développement

1. Solution Adoptée :

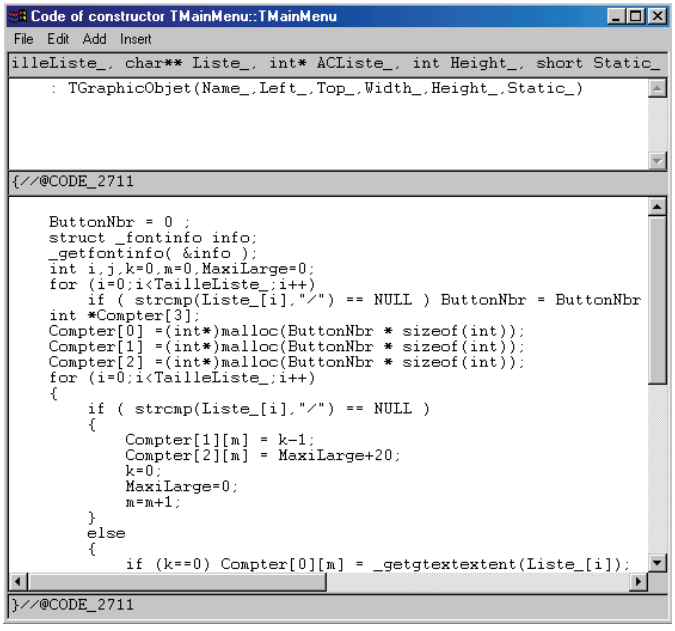
La solution adoptée découle de l'outil Classbuilder qui m'a convaincu.

La possibilité de créer un projet en C++ de A à Z tout en le documentant, de suivre les phases du développement (analyse, design, implémentation, test, complet) et ensuite générer le source C++. Pouvoir compléter et modifier le programme en faisant toutefois attention de ne pas altérer les commentaires générés par Classbuilder et lui permettent de s'y retrouver. Les mises à jour du code seront détectées par Classbuilder qui vous proposera de recharger le code modifié. Attention toutefois, une classe crée en dehors de Classbuilder ne sera pas connu de ce dernier. Il vous faudra l'ajouter dans les classes externes.

Un environnement intuitif qui nous facilite l'approche orientée Objet.



Possibilité d'implémenter directement le code.



La prise en main est d'environ une semaine et quelques règles sont à respecter comme l'implémentation externe qui doit respecter les balises Classbuilder.
Au niveau du code, un fichier (.h et .cpp) généré pour chaque objet une très bonne structure de code avec séparation des membres et des méthodes. La possibilité de créer des entêtes personnalisées …

Exemple :

```
/*****************************************************************************\
                              \\       File:    TMainMenu.h
   CCCCCC     DDDD     IIIIII  \\     Author:  Lefeuvre Vincent
  CC          DD  DD     II      \\
 CC           DD  DD     II       //   Declaration of class 'TMainMenu'
 CCCCCC   DDDDD    IIIIII     //
                              //     Copyright LVD 2001-2002, All rights reserved.
\*****************************************************************************/

#ifndef _TMAINMENU_H
#define _TMAINMENU_H

//@START_USER1
//@END_USER1

class TMainMenu
   : public TGraphicObjet
{
//@START_USER2
//@END_USER2

// Members
private:

protected:
   TMenuItem** MenuButtons;
   int ButtonNbr;

public:
   TPopupMenu** MenuSousMenu;

// Methods
private:
   void ConstructorInclude();
   void DestructorInclude();

protected:

public:
   TMainMenu(char* Name_, int Left_, int Top_, int Width_, int TailleListe_,
           char** Liste_, int* ACListe_, int Height_ = 15, short Static_ = 0);
   virtual ~TMainMenu();
   int OnClick(mouse sour);
   void OnMouseMove(mouse sour);
   void Show();
};

#endif

#ifdef CB_INLINES
#ifndef _TMAINMENU_H_INLINES
#define _TMAINMENU_H_INLINES

//@START_USER3
//@END_USER3

#endif
#endif
```

2. Classes et Objets :

TObjet

TObjet est l'ancêtre primordial de tous les objets.

Description :

Créer, gérer et détruire les instances de l'objet en allouant, initialisant et libérant la mémoire requise. Même si TObjet n'est pas une classe abstraite techniquement parlant, les objets de ce type ne sont normalement pas instanciés.

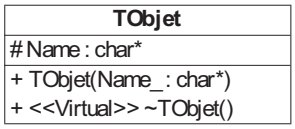
Propriétés :

Name (Contient le nom du composant.)

Méthodes :

TObjet (Crée et initialise une instance de TObjet.)
~ TObjet (Détruit une instance de TObjet.)

Diagramme de classe :



TGraphicObjet

TGraphicObjet est la classe de base de tous les composants visibles lors de l'exécution.

Description :

Les TGraphicObjet sont des composants visuels, c'est-à-dire que l'utilisateur peut les voir et interagir avec eux à l'exécution. Tous les contrôles ont des propriétés, méthodes et événements en commun qui sont spécifiques à l'aspect visuel des contrôles.

Propriétés :

Left (Détermine la coordonnée horizontale, exprimée en pixels.)
Top (Représente la coordonnée verticale , exprimée en pixels.)
Width (Détermine la taille horizontale, exprimée en pixels.)
Height (Indique la taille verticale du contrôle, exprimée en pixels.)
VideoBuf (Sauvegarde de l'affichage avant que le composant apparaît à l'écran.)
Static (Indique si le composant et statique ou dynamique.)
Visible (Détermine si le composant apparaît à l'écran.)

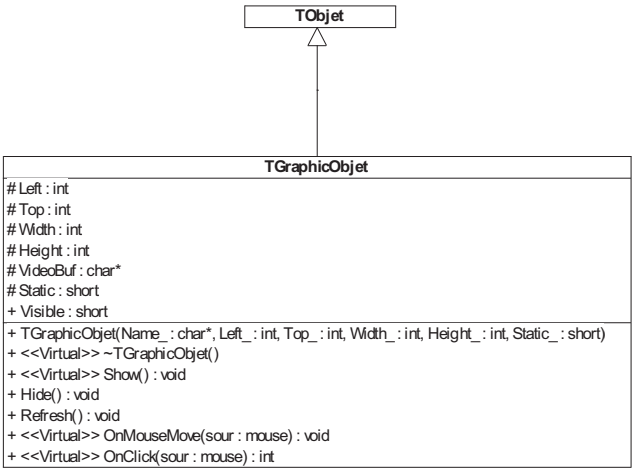
Méthodes :

TGraphicObjet (Crée une instance de TGraphicObjet.)
~ TGraphicObjet (Détruit une instance de TGraphicObjet.)
Show (Rend un contrôle visible.)
Hide (Rend le contrôle invisible.)
Refresh (Redessine le contrôle à l'écran.)

Evénements :

OnMouseMove (Se produit quand l'utilisateur déplace le pointeur de la souris au-dessus d'un contrôle.)
OnClick (Se produit quand l'utilisateur clique sur le contrôle.)

Diagramme de classe :



TForm

Description :

TForm représente une fenêtre.

Propriétés :

Caption (Spécifie le libellé qui apparaît dans la barre de titre.)
Color (Indique la couleur d'arrière-plan du contrôle.)
ClientLeft (Détermine la coordonnée horizontale, exprimée en pixels, de la zone client de la fiche.) ClientTop (Représente la coordonnée verticale, exprimée en pixels, de la zone client de la fiche.)
ClientHeight (Indique la taille, exprimée en pixels, de la zone client de la fiche.)
ClientWidth (Indique la largeur, exprimée en pixels, de la zone client de la fiche.)
Dérivées de TGraphicObjet
Left, Top, Width, Height, VideoBuf, Static, Visible.

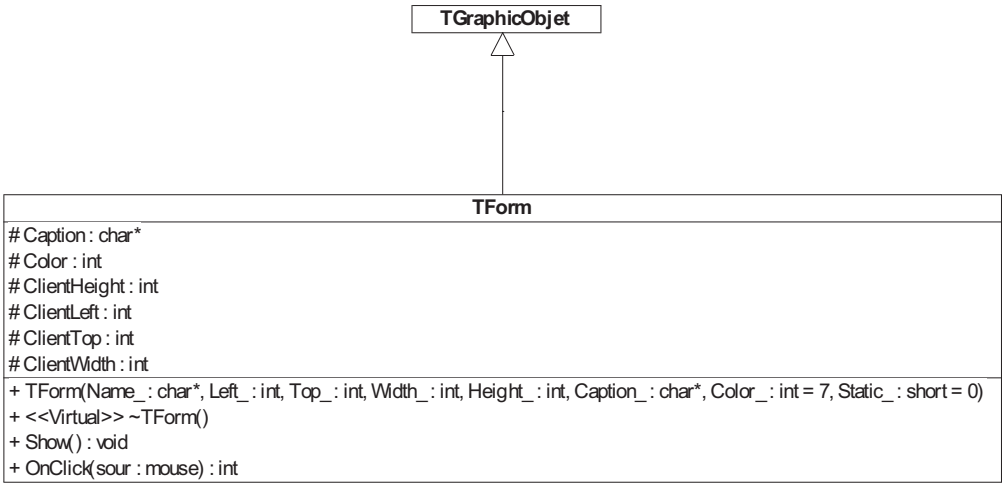
Méthodes :

TForm (Crée et initialise une instance de TForm.)
~ TForm (Détruit une instance de TForm.)
Dérivées de TGraphicObjet
Show, Hide, Refresh.

Evénements :

Dérivées de TGraphicObjet
OnMouseMove, OnClick.

Diagramme de classe :



TMenuItem

Description :

TMenuItem décrit les propriétés d'un élément de menu.

Propriétés :

Action (Désigne l'action associée à l'élément de menu.)
Caption (Spécifie le libellé qui apparaît dans le bouton.)
FirstButton (Spécifie le style du bouton.)
SetFocus (Donne la focalisation au contrôle.)
Autocheck (Contrôle si la propriété Checked bascule à l'exécution de l'action.)
Checked (Spécifie si une marque apparaît à côté de Caption.)
Dérivées de TGraphicObjet
Left, Top, Width, Height, VideoBuf, Static, Visible.

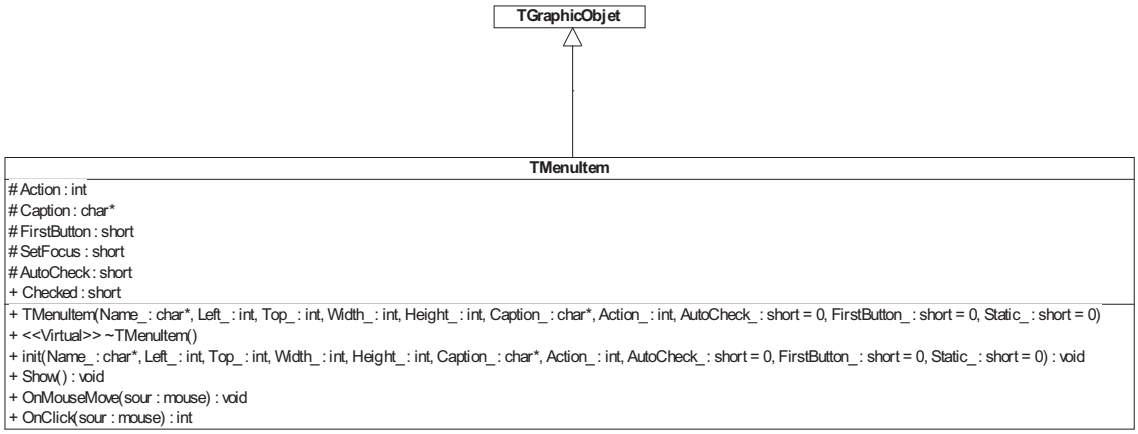
Méthodes :

TMenuItem (Crée et initialise une instance de TMenuItem.)
~ TMenuItem (Détruit une instance de TMenuItem.)
init (initialisation d'une instance avec new.)
Dérivées de TGraphicObjet
Show, Hide, Refresh.

Evénements :

Dérivées de TGraphicObjet
OnMouseMove, OnClick.

Diagramme de classe :



TPopupMenu

Description :

TPopupMenu est un sous menu composé de TMenuItem.

Propriétés :

Numero (Désigne le numero du TpopupMenu.)
Nbr (Désigne le nombre de TMenuItem associée.)
PMenuItem (Tableau de pointeur vers les différents TMenuItem.)
Dérivées de TGraphicObjet
Left, Top, Width, Height, VideoBuf, Static, Visible.

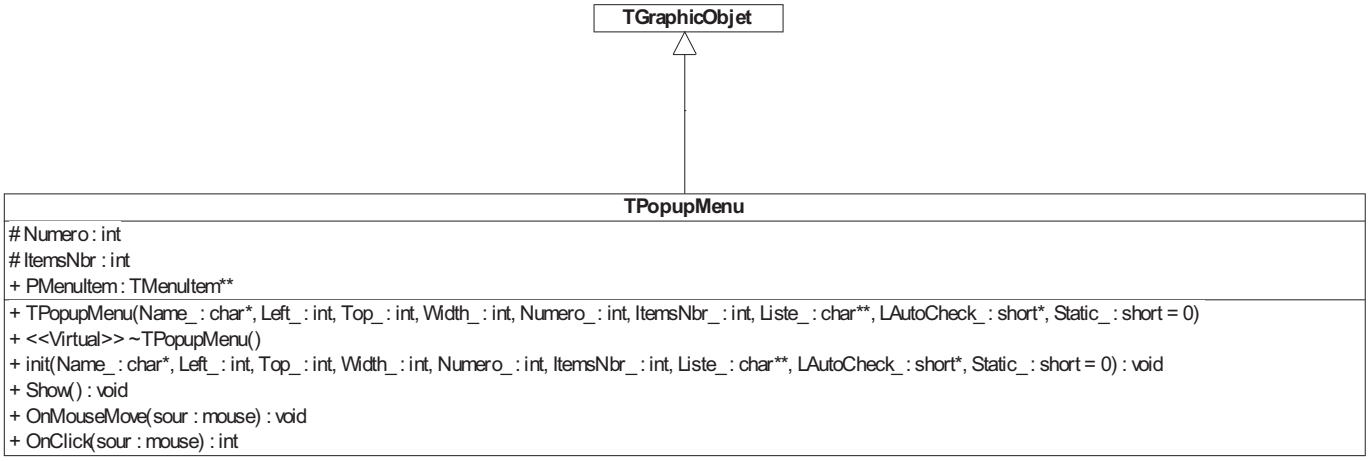
Méthodes :

TPopupMenu (Crée et initialise une instance de TPopupMenu.)
~ TPopupMenu (Détruit une instance de TPopupMenu.)
init (initialisation d'une instance avec new.)
Dérivées de TGraphicObjet
Show, Hide, Refresh.

Evénements :

Dérivées de TGraphicObjet
OnMouseMove, OnClick.

Diagramme de classe :



TMainMenu

Description :

TMainMenu encapsule dans une fiche une barre des menus et ses menus déroulants.

Propriétés :

ButtonsNbr (Désigne le nombre de TMenuItem associée .)
MenuButtons (Tableau de pointeur vers les différents TMenuItem.)
MenuSousMenu (Tableau de pointeur vers les différents TPopupMenu.)
Dérivées de TGraphicObjet
Left, Top, Width, Height, VideoBuf, Static, Visible.

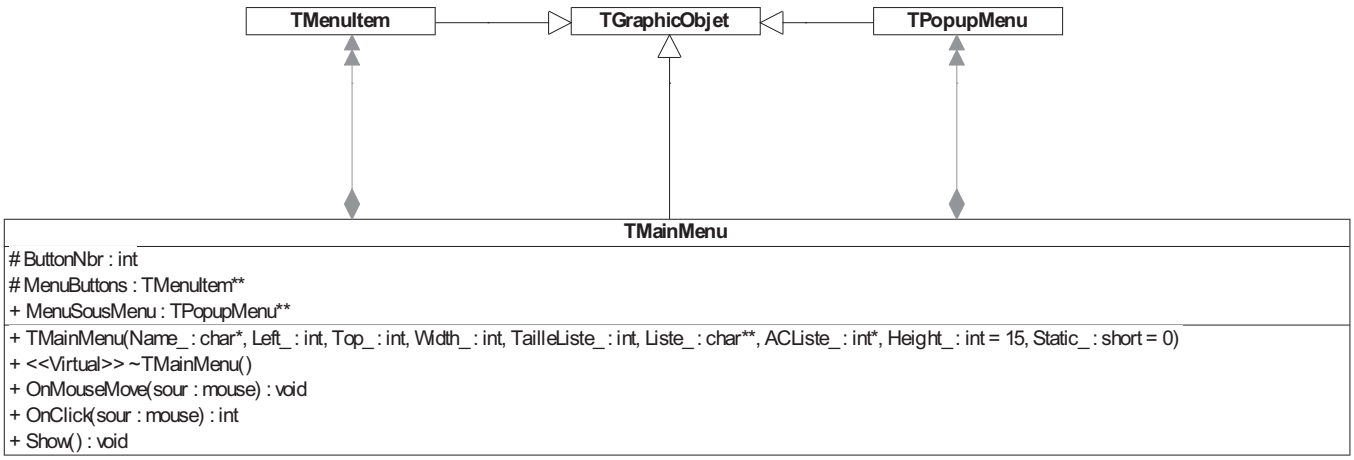
Méthodes :

TMainMenu (Crée et initialise une instance de TMainMenu.)
~ TMainMenu (Détruit une instance de TMainMenu.)
Dérivées de TGraphicObjet
Show, Hide, Refresh.

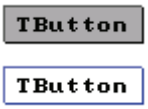
Evénements :

Dérivées de TGraphicObjet
OnMouseMove, OnClick.

Diagramme de classe :



TButton



Description :

TButton est un contrôle bouton poussoir.

Propriétés :

Caption (Spécifie le libellé qui apparaît dans le bouton.)
SetFocus (Donne la focalisation au contrôle.)
Dérivées de TGraphicObjet
Left, Top, Width, Height, VideoBuf, Static, Visible.

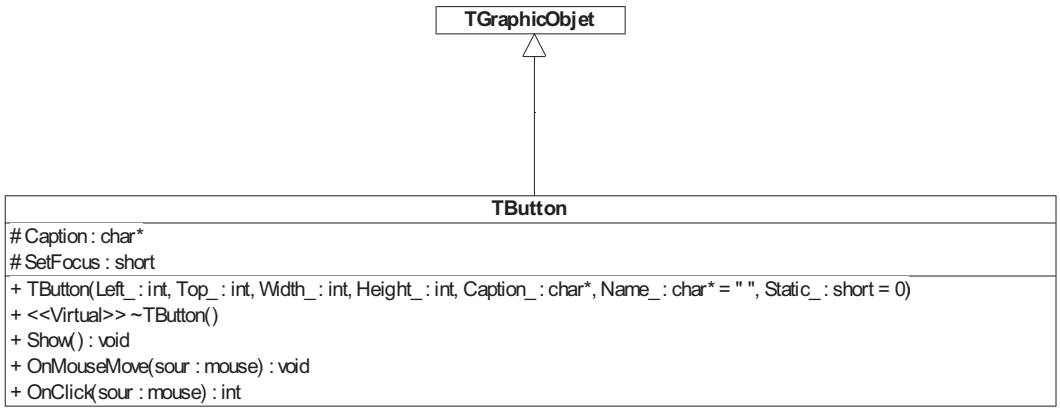
Méthodes :

TButton (Crée et initialise une instance de TButton.)
~ TButton (Détruit une instance de TButton.)
Dérivées de TGraphicObjet
Show, Hide, Refresh.

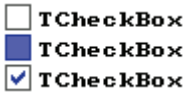
Evénements :

Dérivées de TGraphicObjet
OnMouseMove, OnClick.

Diagramme de classe :



TCheckBox



Description :

Un composant TCheckBox propose une option à l'utilisateur. L'utilisateur peut activer la case à cocher pour sélectionner l'option, ou supprimer la coche pour la désélectionner.

Propriétés :

Caption (Spécifie le libellé qui apparaît à côté du TCheckBox.)
SetFocus (Donne la focalisation au contrôle.)
Checked (Spécifie l'état de la case à cocher.)
Dérivées de TGraphicObjet
Left, Top, Width, Height, VideoBuf, Static, Visible.

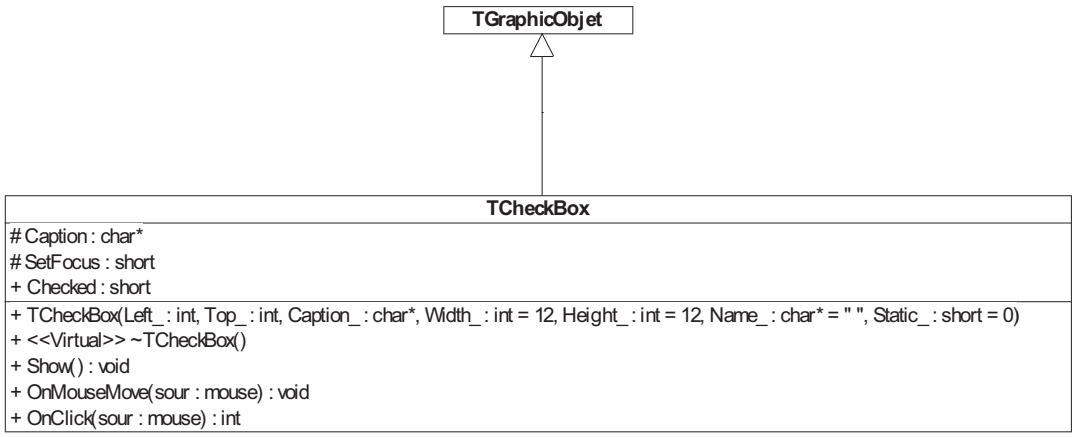
Méthodes :

TCheckBox (Crée et initialise une instance de TCheckBox.)
~ TCheckBox (Détruit une instance de TCheckBox.)
Dérivées de TGraphicObjet
Show, Hide, Refresh.

Evénements :

Dérivées de TGraphicObjet
OnMouseMove, OnClick.

Diagramme de classe :



TEdit

Description :

TEdit encapsule un contrôle de saisie monoligne.

Propriétés :

SetFocus (Donne la focalisation au contrôle.)
Text (Spécifie la chaîne de texte affichée dans la zone d'édition.)
Dérivées de TGraphicObjet
Left, Top, Width, Height, VideoBuf, Static, Visible.

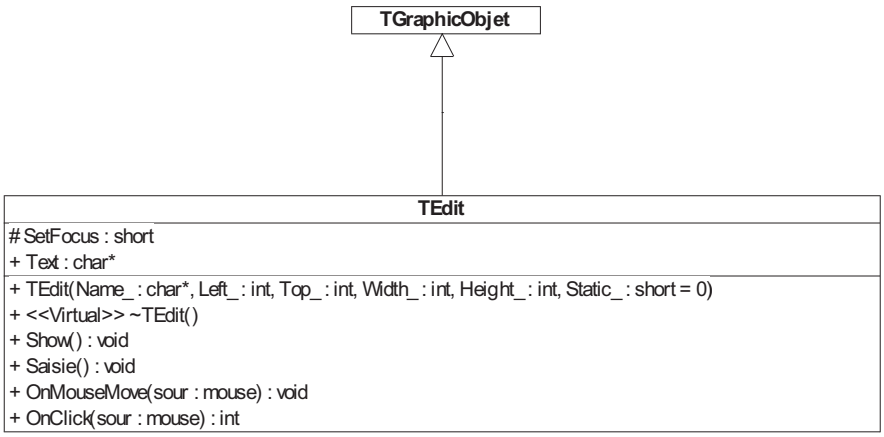
Méthodes :

TEdit (Crée et initialise une instance de TEdit.)
~ TEdit (Détruit une instance de TEdit.)
Saisie (Mode de saisie duTEdit.)
Dérivées de TGraphicObjet
Show, Hide, Refresh.

Evénements :

Dérivées de TGraphicObjet
OnMouseMove, OnClick.

Diagramme de classe :



TLabel

TLabel

Description :

TLabel est un contrôle qui affiche du texte dans une fiche.

Propriétés :

Caption (Spécifie la chaîne de texte affichée par le libellé.)
Color (La couleur de la fonte utilisée.)

Dérivées de TGraphicObjet
Left, Top, Width, Height, VideoBuf, Static, Visible.

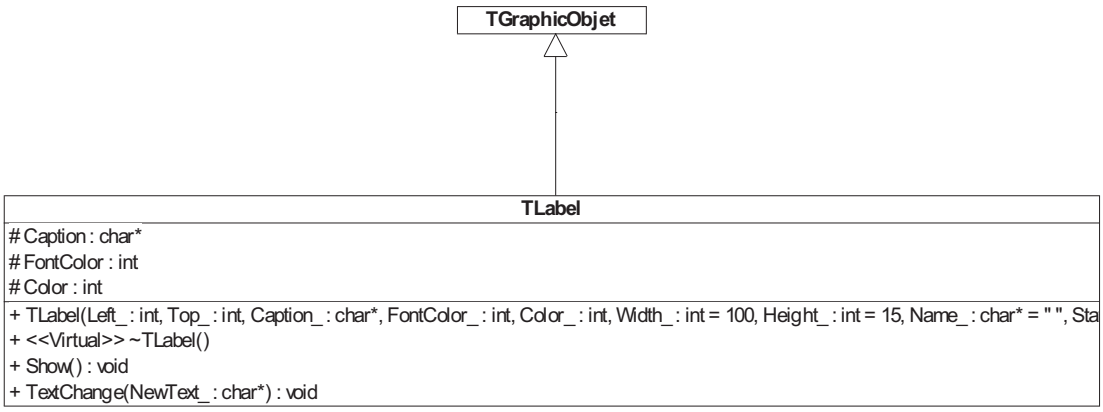
Méthodes :

TLabel (Crée et initialise une instance de TLabel.)
~ TLabel (Détruit une instance de TLabel.)
Dérivées de TGraphicObjet
Show, Hide, Refresh.

Evénements :

Dérivées de TGraphicObjet
OnMouseMove, OnClick.

Diagramme de classe :



TBrick

Description :

TBrick représente une brique de jeux.

Propriétés :

Color (Indique la couleur de la brique.)
Dérivées de TGraphicObjet
Left, Top, Width, Height, VideoBuf, Static, Visible.

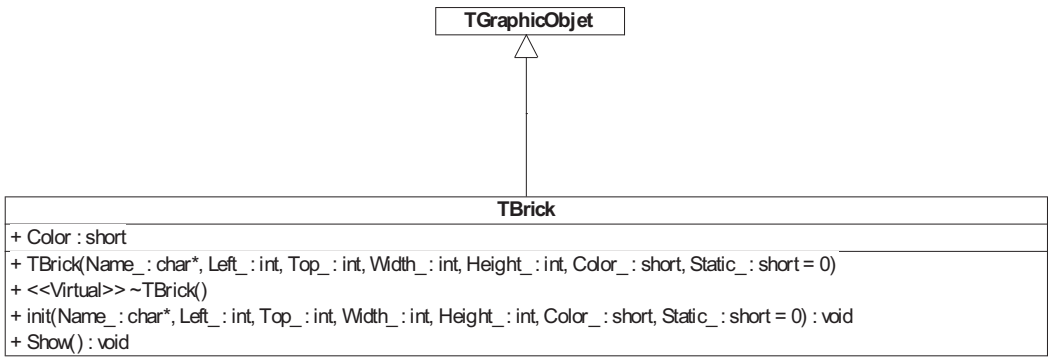
Méthodes :

TBrick (Crée et initialise une instance de TBrick.)
~ TBrick (Détruit une instance de TBrick.)
init (initialise une instance avec New.)
Dérivées de TGraphicObjet
Show, Hide, Refresh.

Evénements :

Dérivées de TGraphicObjet
OnMouseMove, OnClick.

Diagramme de classe :



TGrid

Description :

TGrid représente un tableau de Briques.

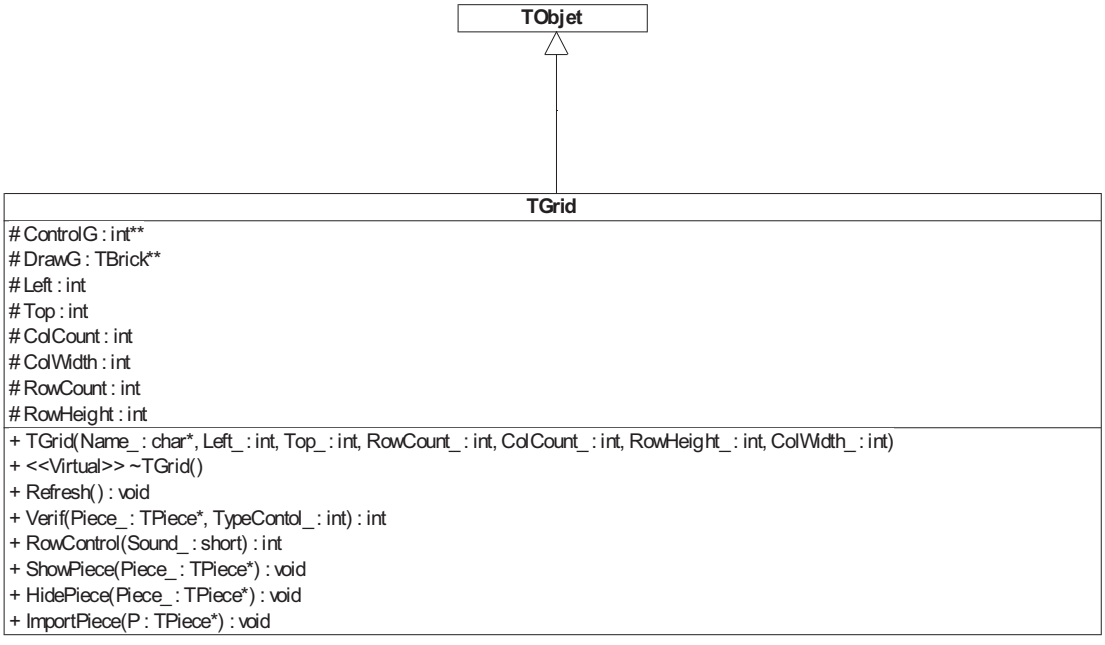
Propriétés :

ControlG (Représente un tableau de remplissage de la grille.)
DrawG (Représente un Tableau de brique.)
Left (Détermine la coordonnée horizontale, exprimée en pixels.)
Top (Représente la coordonnée verticale , exprimée en pixels.)
ColCount (Nombre de colonnes.)
ColWidth (Détermine la taille horizontale, exprimée en pixels d'une colonne.)
RowCount (Nombre de lignes.)
RowHeight (Indique la taille verticale du contrôle, exprimée en pixels, d'une ligne.)

Méthodes :

TGrid (Crée et initialise une instance de TGrid.)
~ TGrid (Détruit une instance de TGrid.)
Refresh (Redessine la grille en fonction de son remplissage.)
RowControl (Verifie si une ligne de la grille est plaine.)
ShowPiece (Affiche une piece sur la grille.)
HidePiece (effacer une piece dur la grille.)
Verif (verification de la posibilite de deplacement d'une piece sur la grille.)
ImportPiece (Importation d'une piece dans la grille.)

Diagramme de classe :



TPiece

Description :

TPiece représente une pièce de Jeu.

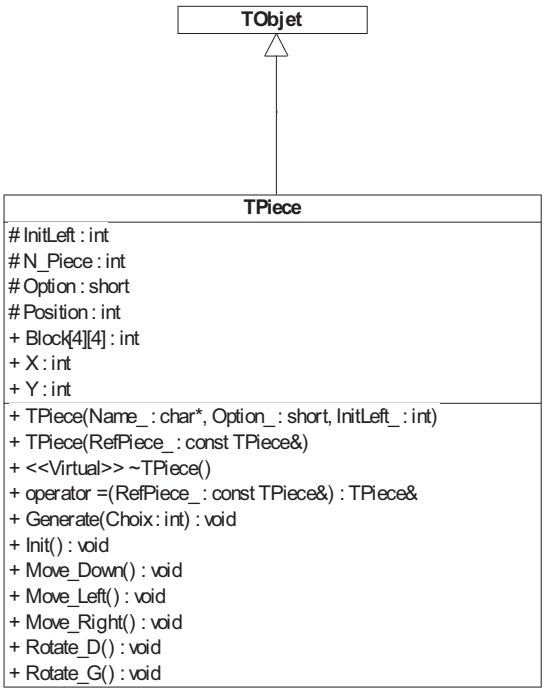
Propriétés :

InitLeft (Position au départ du jeu .)
NPiece (numéro de la pièce.)
Option (Type de pièce standard ou crazy .)
Position (numéro de rotation de la pièce.)
Block (Tableau de remplissage de la pièce.)
X (Largeur de la pièce.)
Y (Hauteur de pièce.)

Méthodes :

TPiece (Crée et initialise une instance de TPiece.)
~ TPiece (Détruit une instance de TPiece.)
TPiece (TPiece &) (Constructeur par recopie.)
Operator = (Surdefinition de l'operateur =.)
Generate(remplissage du block en fonction des differents parametres de la pièce.)
init (initialise une instance avec New.)
Move_Down, MoveLeft, Move_Right, Rotate_D, Rotate_G (Actions de deplacemant de la pièce.)

Diagramme de classe :



FormTetris

Description :

FormTetris représente le fenêtre du jeu Tetris.

Propriétés :

NextControl (Tableau de contrôle du remplissage de grille Next)
NextDraw (Tableau de briques graphiques)
LabelScore… (Différent textes d'affichage, compteurs et scores.)

Dérivées de TForm

Left, Top, Width, Height, VideoBuf, Static, Visible, Caption, Color , ClientLeft, ClientTop, ClientHeight, ClientWidth .

Méthodes :

FormTetris (Crée et initialise une instance de FormTetris.)
~ FormTetris (Détruit une instance de FormTetris.)
ShowPiece (Affichage de la piece Next.)
HidePiece (Effacer l'affichage de la piece Next.)

Dérivées de TForm

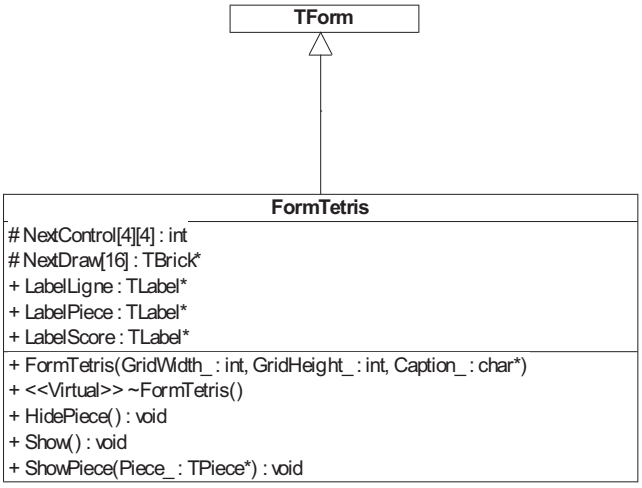
Show, Hide, Refresh.

Evénements :

Dérivées de TForm

OnMouseMove, OnClick.

Diagramme de classe :



TSplash

Description :

TSplash permet d'afficher un écran au démarrage de votre programme.

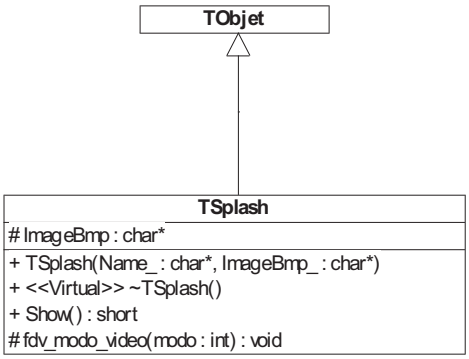
Propriétés :

ImageBmp (chemin de l'image *.bmp sur le disque.)

Méthodes :

TSplash (Crée et initialise une instance de TSplash.)
~ TSplash (Détruit une instance de TSplash.)
Show (Rend l'image visible.)
fdv_modo_video (initialise le mode video) ;

Diagramme de classe :



Exemple de fenêtre composé

Description :

FormNewGame représente le fenêtre de configuration du jeu.
Le joueur Saisie Son nom, et choisie le type de pièce ainsi que l'option de grille.

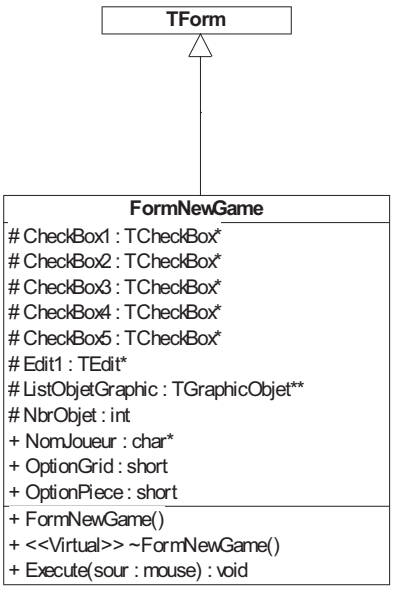
Propriétés :

ListObjetGraphic (Tableau d'objet graphique TLabel, Tbuton …)

Méthodes :

Notion très forte de polymorphisme dans les méthodes, on parcourt ListObjetGraphic en faisant appel aux méthodes de construction, Show, OnMouseMove, sans se soucier de la classe réel de l'objet.

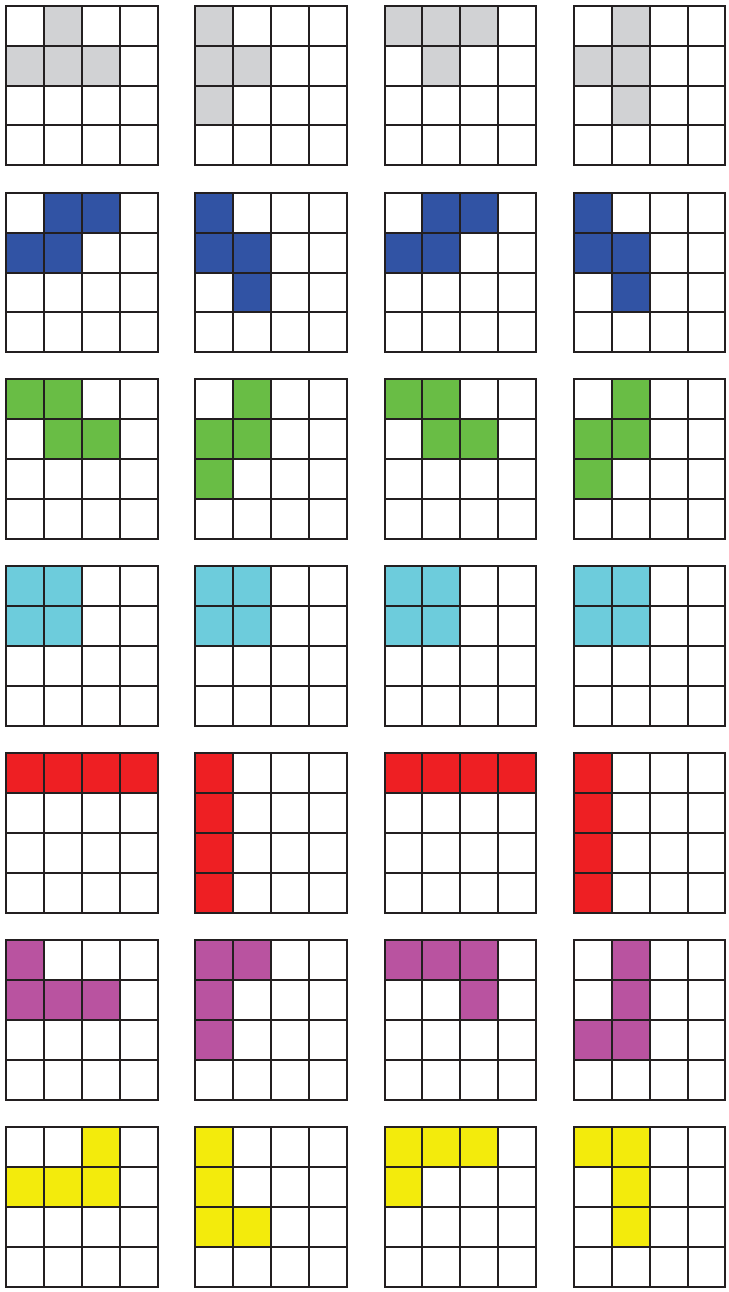
Diagramme de classe :



3. Principe utilisé pour définir les pièces:

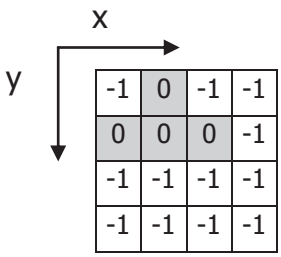
7 " Pièce " dans 4 positions de rotation :

n° Position : 1 2 3 4



décomposition d'une pièce :

n° pièce = 0 , couleur = n° pièce = 0, Position = 1;



Initialisation du block avec des (–1).

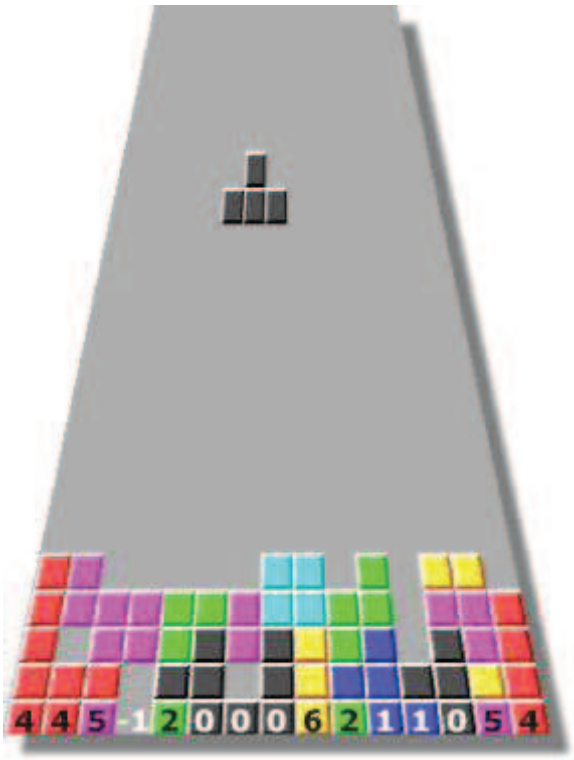
Remplissage tableau : Procédure Generate (n° pièce * 10 + Position = 1) :

Case 1 : block [1][0] = block [0][1] = block [1][1] block [1][2] = 0

A l'affichage parcours de la table et affichage de la Brick si la valeur <> -1.

La Grille :

Pour la grille on utilise le même principe de numérotation :



On peut ainsi vérifier si une case est occupée ou non lors de la descente d'une pièce.

On peut aussi voir si une ligne est complète quand tous les chiffres de la ligne sont > -1.

4. Etude approfondie du Menu:

L'élément menu est assez complexe, il est composé de 3 objets, en contre partie une utilisation très simple pour le développeur, car il est entièrement automatisé.

Menu réalisé :

Code à écrire pour un tel menu :

```
char *ButtonListe[13] =     {
                            "Game", "New", "Pause", "Abort", "Exit", "/",
                            "Option", "Sound", "High Scores", "/",
                            "Help", "About", "/"
                            };
int AutoCheckListe[13] =    {
                            1, 0, 1, 0, 0, 0,
                            1, 1, 0, 0,
                            1, 0, 0
};
MenuFG = new TMainMenu ("MenuG",ClientLeft,ClientTop,ClientWidth,13,ButtonListe,AutoCheckListe);
```

C'est Tout !!!
Le reste et géré en automatique.

1/ Le constructeur de menu calcule le nombre de sous menu délimité par "/" ici il y en à 3.

2/ il crée un tableau dynamique 3 à colonnes (Largeur du premier bouton, nombre de boutons dans le sous menu, largeur maxi du sous menu).

3/il parcourt a nouveau la liste pour remplir son tableau.

4/ il crée dynamiquement les boutons et les sous menu.

Pour les boutons a cocher (x) il utilise AutoChekListe.
Pour le code d'action retourné lorsque l'on clique sur le bouton, il utilise le système de numérotation logique suivant :

Game (1) x	Options (2) x	Help (3) x
New (11)	Sound (21) x	About (31)
Pause (12) x	Hight Scores (22)	
Abort (13)		
Exit (14)		

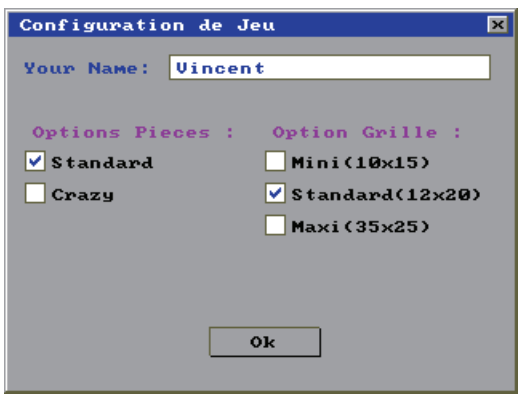
Constructeur :

```
TMainMenu::TMainMenu(char* Name_, int Left_, int Top_, int Width_,
                     int TailleListe_, char** Liste_, int* ACListe_,
                     int Height_, short Static_) //@INIT_2711
   : TGraphicObjet(Name_,Left_,Top_,Width_,Height_,Static_)
{
   ButtonNbr = 0 ;
   struct _fontinfo info;
   _getfontinfo( &info );
   int i,j,k=0,m=0,MaxiLarge=0;
   for (i=0;i<TailleListe_;i++)
      if ( strcmp(Liste_[i],"/") == NULL ) ButtonNbr = ButtonNbr +1;

   int *Compter[3];

   Compter[0] =(int*)malloc(ButtonNbr * sizeof(int));
   Compter[1] =(int*)malloc(ButtonNbr * sizeof(int));
   Compter[2] =(int*)malloc(ButtonNbr * sizeof(int));
```

```
    for (i=0;i<TailleListe_;i++)
    {
        if ( strcmp(Liste_[i],"/") == NULL )
        {
            Compter[1][m] = k-1;
            Compter[2][m] = MaxiLarge+20;
            k=0;
            MaxiLarge=0;
            m=m+1;
        }
        else
        {
            if (k==0) Compter[0][m] = _getgtextextent(Liste_[i]);
            else if (_getgtextextent(Liste_[i]) > MaxiLarge) MaxiLarge = _getgtextextent(Liste_[i]) ;
            k = k+1;
        }
    }
    MenuButtons = new TMenuItem*[ButtonNbr];
    MenuSousMenu = new TPopupMenu*[ButtonNbr];
    int CummulLeft = Left+10;
    int CummulNum = 0;
    for (i=0;i<ButtonNbr;i++)
    {
        MenuButtons[i] = new
TMenuItem("",CummulLeft,Top+1,Compter[0][i]+15,12,Liste_[CummulNum],i+1,1,1);
        char **CaptionListe = (char**)malloc(Compter[1][i] * sizeof(char*));
        short *AutoCheckListe = (short*)malloc(Compter[1][i] * sizeof(short));
        for (j=0;j<Compter[1][i];j++)
        {
            CaptionListe[j] = Liste_[CummulNum+j+1];
            AutoCheckListe[j] = ACListe_[CummulNum+j+1];
        }
        MenuSousMenu[i] = new TPopupMenu("",CummulLeft,Top+ Height-1,
Compter[2][i]+10,i+1,Compter[1][i], CaptionListe,AutoCheckListe);
        CummulNum = CummulNum + Compter[1][i] + 2;
        CummulLeft = CummulLeft+Compter[0][i]+17;
    }
}
```

OnClick : Action de 1 à 10 on affiche le sous menu correspondant et on vérifie le OnClick des boutons qui le composent.
Action > 10 on retourne à l'utilisateur le numéro d'action afin qu'il puisse gérer ses propres Actions comportementales.

5. Articulation autour des fenêtres multi-Objects :

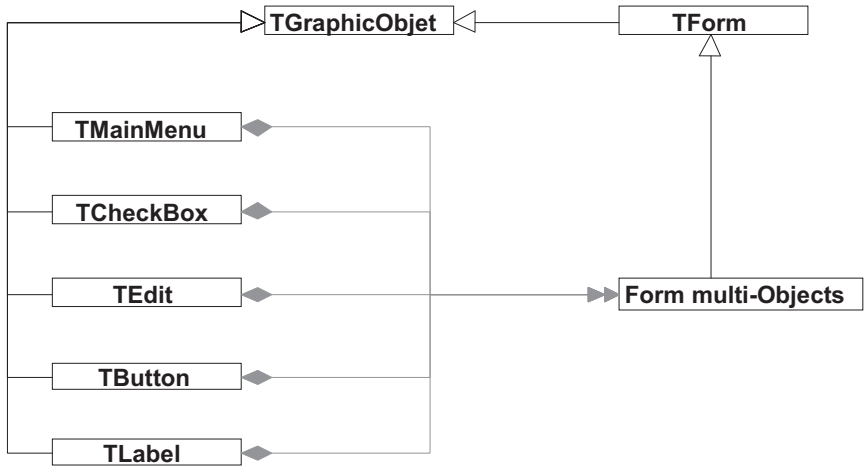


Héritage :

Les fenêtres multi-Objects héritent de l'objet Tform.
elles bénéficient des méthodes :
Show (pour l'afficher),
Hide (pour la rendre invisible et restaurer l'écran avant l'affichage),
OnClick (pour contrôler si l'utilisateur a cliqué sur le contrôle de fermeture de fenêtre).
Elles bénéficient aussi des propriétés ClientLeft, ClientTop, ClientHeight, ClientWidth , qui facilitent le positionnement des composants sur les fenêtres.

Compositions :

Les fenêtres multi-Objects peuvent contenir une multitude d'objets.



Polymorphisme :

L'avantage du polymorphisme est de gagner beaucoup de temps à l'implémentation.
En effet il suffit de créer un tableau de pointeur TGraphicObjet et de remplir ce tableau avec des objets divers tel que Tedit, Tbutton, TLabel …

```
NbrObjet = 4;
ListObjetGraphic = new TGraphicObjet*[NbrObjet];
ListObjetGraphic[0] = new TButton(ClientLeft+125,ClientTop+180,69,18," Ok ");
ListObjetGraphic[1] = new TLabel(ClientLeft+5,ClientTop+10,"Your Name:",1,7,85);
ListObjetGraphic[2] = new TLabel(ClientLeft+10,ClientTop+50,"Options Pieces :",5,7,65);
ListObjetGraphic[3] = new TCheckBox(ClientLeft+10,ClientTop+70,"Standard");
```

Après les implémentations sont simplifiées, surtout pour un grand nombre d'objets très diversifiés :

Pour afficher :

```
for (int i=0;i<NbrObjet;i++) ListObjetGraphic[i]->Show();
```

Pour vérifier les contrôles :

```
for (int j=0;j<NbrObjet;j++)
    {
      ListObjetGraphic[j]->OnMouseMove(sour);
      ListObjetGraphic[j]->OnClick(sour);
    }
```

Pour la destruction :

```
for (int i=0;i<NbrObjet;i++) delete(ListObjetGraphic[i]);
```

L'ajout d'un cinquième élément est très simple puisqu'il suffit de mettre NbrObjet à 5 et de définir ListObjetGraphic[4]. Le reste est géré automatiquement.

Evolutions du Projet

1. Evolution du jeu et de son interface

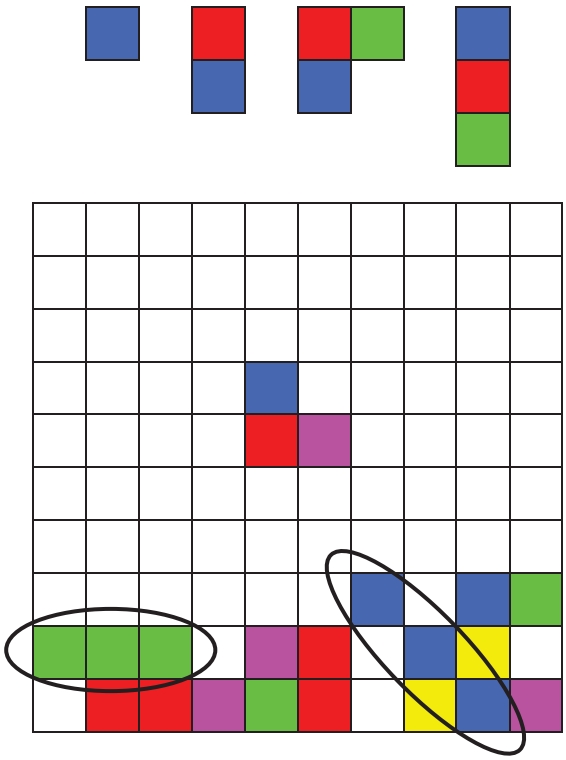
- Petites améliorations :

Gestion d'un tableau de scores : à la fin du jeu il serait intéressent de garder en mémoire le nom du joueur et son score pour le classer dans une table contenant les 5 meilleurs joueurs.

Insertion d'images : créer un Timage afin d'agrémenter le jeu.

- Nouvelle variante de jeu : option Tetris Color

Le but du jeu c'est d'aligner les couleurs, par groupe de 3 ou plus afin de supprimer les briques.



Modifications a effectuer :

TPiece : gérer la rotation des pièces par calcule vectoriel. 3 pièces de 1, 2 et 3 briques, chaque brique peut avoir l'une des 7 couleurs soit $7^1 + 7^2 + 7^3 = 399$ pièces différentes.

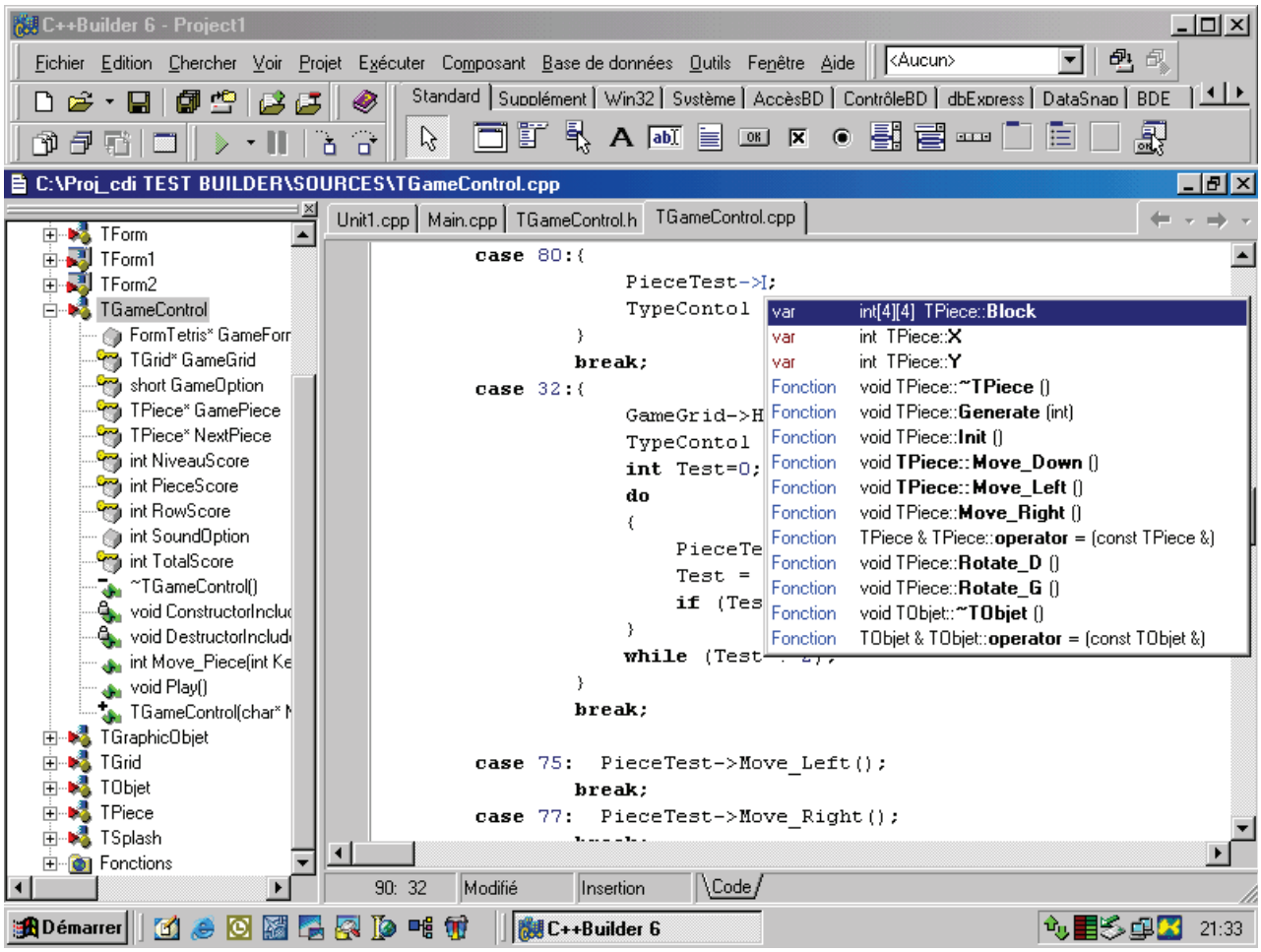
TGrid : refaire une nouvelle fonction pour la vérification de lignes (horizontales, verticales, obliques).

- Nouveau jeu :

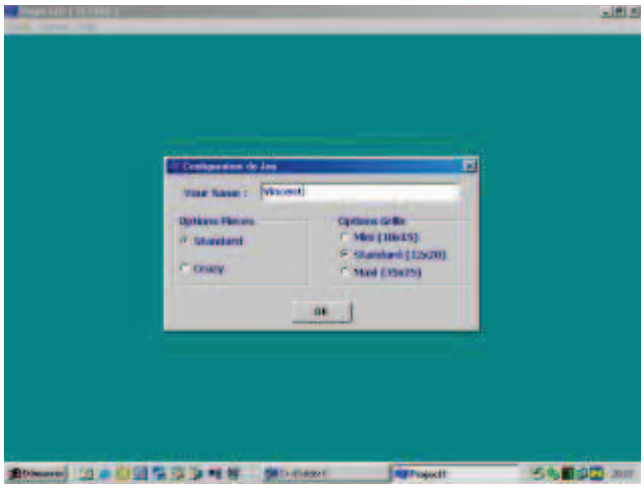
L'interface graphique étant complètement indépendante du jeu Tetris, il serait donc possible de créer d'autres tableaux de jeux comme un casse briques par exemple.

2. Evolution du Projet en C++ Builder Version 6 de chez Borland :

- Audit de code pour augmenter la rapidité et diminuer les erreurs de frappes :



- L'Interface graphique intégré de l'IDE (fenêtre générale + menu + fenêtre de configuration, bouton, radio box, édit …) réaliser en 3 minutes. Il serait même plus judicieux d'utiliser les objets CLX basés sur le X-Windows QT, recompilablent sur Kylix (plate-forme Linux de delphi et prochainement C++).



La partie de code a retravailler ne concerne que l'interface de jeu en particulier Tbrick et FormTetris. Tbrick est un objet graphique qui pourrait être traité comme une image (avec la notion de canvas). Coût de développement projet 8 Jours. Avec des perspectives très intéressantes (lecture de fichiers midi en fond sonore, couleurs 32bits …).

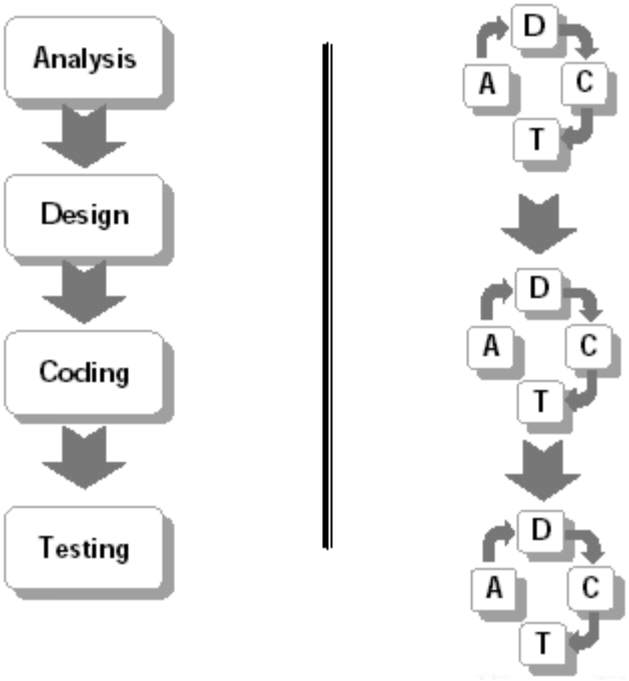
Conclusion

Les premières impressions sont plutôt positives car c'est la première fois que j'utilise un outil tel que ClassBuilder et je regrette qu'il n'existe pas d'équivalent pour Delphi.

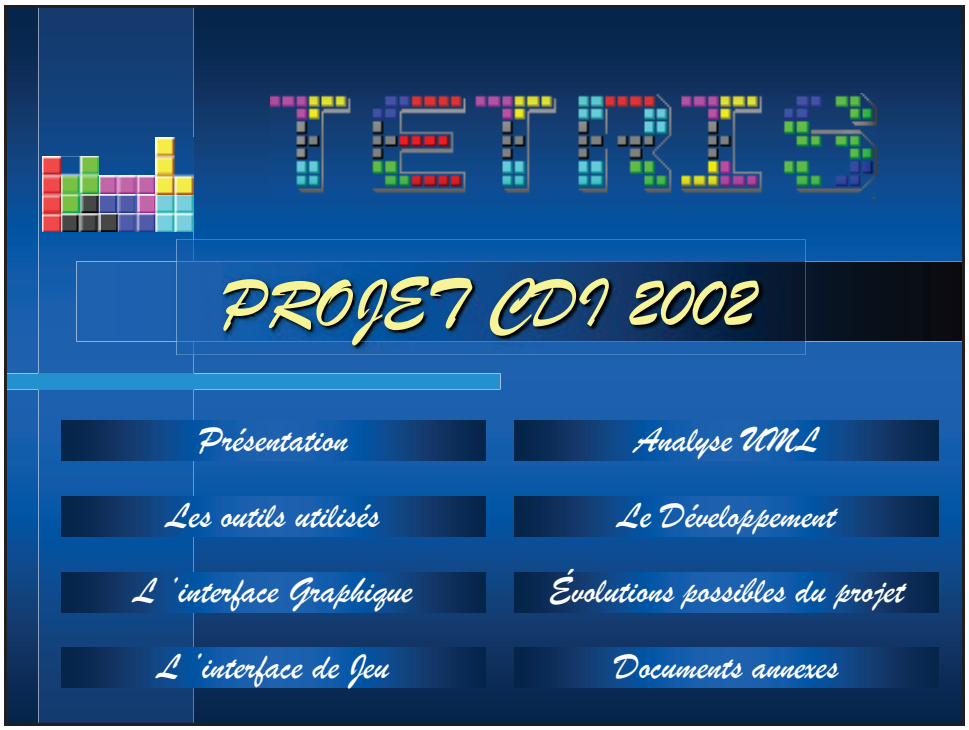
Les complications au quelles je me suis heurté son plus de l'adaptation de langage que des problèmes de fond.
Exemples : = en Delphi donne == en C++
Delphi on utilise que le point. alors qu'en C++ on utilise le point. et la flèche ->

La seule problématique c'est présenté lors de la gestion du jeu et elle a été résolue par l'objet TgameContrôle.

Il reste cependant un point qu'il aurait été intéressant à développe c'est le management de projet :



Slide Show


TETRIS
PROJET CDI 2002
Présentation
Analyse UML
Les outils utilisés
Le Développement
L 'interface Graphique
Évolutions possibles du projet
L 'interface de Jeu
Documents annexes

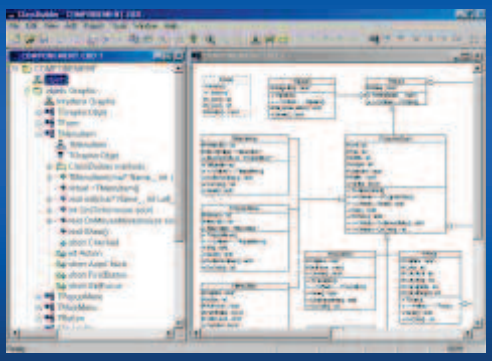

Project CDI 2002
Vincent Lefeuvre
TETRIS
Présentation
Test de Démonstration du projet
Projet CDI (TETRIS)
Game Option Help
Démonstration
Next
Pieces
65
Lines
14
Score
400
Level
3
Tetris.exe

Project CDI 2002
Vincent Lefeuvre

TETRICS

Les outils utilisés

- Modélisation UML réalisée avec ClassBuilder

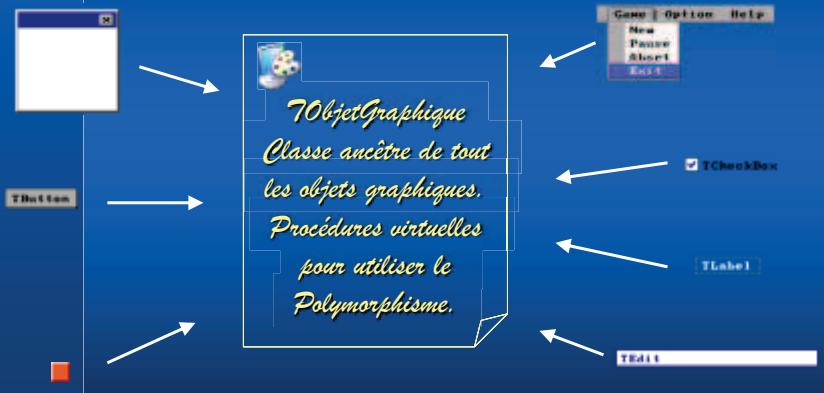


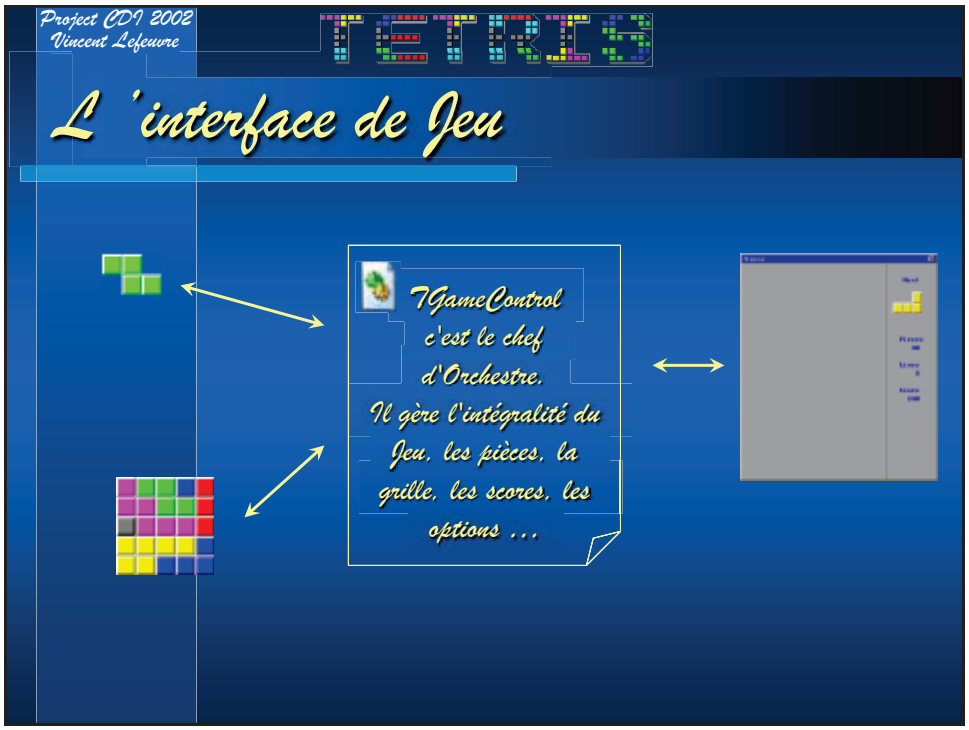
- Développement en C++ réalisé avec Watcom

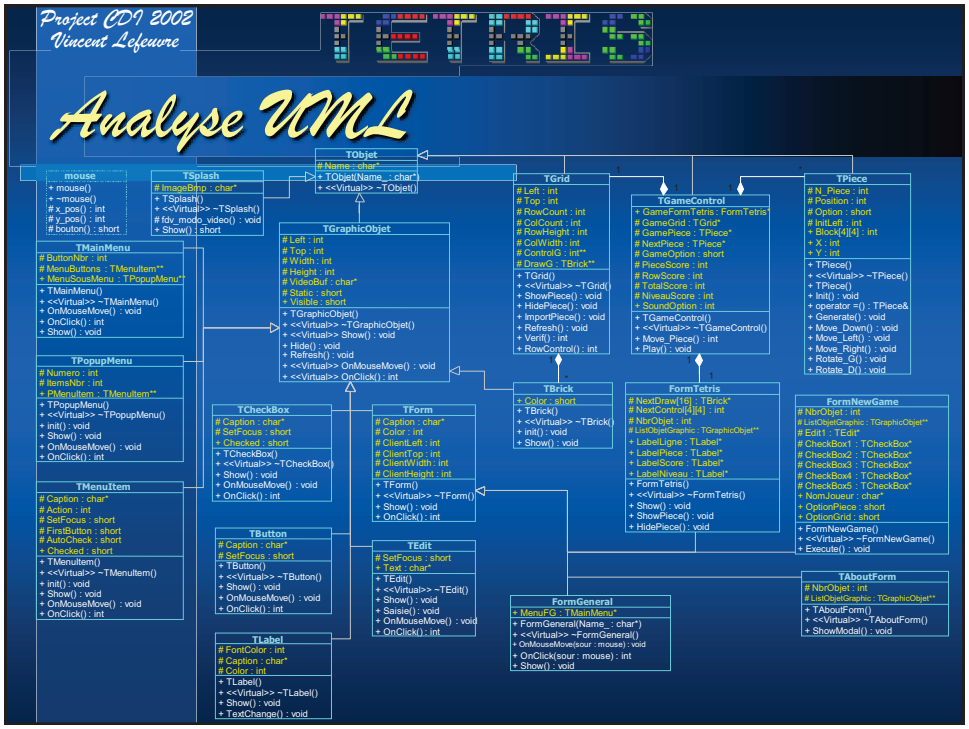
Project CDI 2002
Vincent Lefeuvre

TETRICS

L 'interface Graphique




Project CDI 2002
Vincent Lefeuvre
L 'interface de Jeu
TGameControl
c'est le chef
d'Orchestre.
Il gère l'intégralité du
Jeu, les pièces, la
grille, les scores, les
options ...


Project CDI 2002
Vincent Lefeuvre
Analyse UML
TObjet
mouse
TSplash
TGrid
TGameControl
TPiece
TMainMenu
TGraphicObjet
TPopupMenu
TCheckBox
TForm
TBrick
FormTetris
FormNewGame
TMenuItem
TButton
TEdit
TLabel
FormGeneral
TAboutForm

TETRICS

Le Développement

- Étude approfondie du Menu
  - Utilisation de l'objet Menu.
  - Constructeur.
  - Gestion des méthodes en cascades.
- Articulation autour des fenêtres Multi-Objets
  - Héritage.
  - Composition.
  - Polymorphisme.

Project CDI 2002
Vincent Lefeuvre

TETRICS

Évolutions possibles du projet

- Amélioration du Jeu
  - Gestion d'un tableau de scores.
  - Évolution graphique, insertion d'images ...
  - Nouveau jeu de Tetris basé sur des alignements de couleurs.
  - Créer de nouveaux tableaux de jeux "casse briques".
- Évolution vers Borland C++Builder
- Évolution multi plates-formes Delphi Kylix