GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

PROJET ALGO-ADA

Sujet n°2 : Hécatombe

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

SOMMAIRE

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

I. Introduction

Le sujet étant la réalisation d'un jeu, cela nous à motiver à réaliser une application fonctionnelle et amusante.

Nous avons choisi d'utiliser la librairie GtkAda pour réaliser l'interface graphique de l'application. Il a été difficile de trouver de la documentation. Mais des exemples fournis avec la libraire nous a permis de nous familiariser avec Gkt.

La conception avait déjà été faite lors du premier rapport donc nous allons nous intéressé plus précisément l'aspect programmation de l'application.

Pour compiler sous Linux:
Taper dans le terminale : *gnatmake hecatombe.adb `gtkada-config`*
Pour compiler sous Windows :
Taper dans la console MS-DOS : *gnatmake hecatombe.adb -Ic:\gtkada\include\gtkada*

Attention : La dernière version de GtkAda doit être installé pour pouvoir compiler.

II. Le Programme

L'analyse de l'application nous a mener à diviser l'application en trois parties distinctes, chacune de ces parties constitue un paquetage:

- ***p_interface***: Ce paquetage gère l'interface graphique (librairie gtkada) et la liaison entre l'interface graphique et les divers traitements de données (gestion des fichiers de sauvegarde, traitement de la grille du jeu, ...)
- ***p_traitement***: Ce paquetage gère les modifications de la grille de jeu (lors d'une sélection, lors d'un clique), Détection de la fin de partie, comptage des points - score et enfin la gestion de la triche.
- ***p_fichier***: Ce paquetage gère la sauvegarde et le chargement d'un partie, ainsi qui l'enregistrement et le chargement de highscores.

Les tâches se sont réparties de la manière suivantes:

- Le paquetage ***p_interface*** a été réalisé par Richard GUERCI.
- Le paquetage ***p_traitement*** a été réalisé par Richard GUERCI, Matthieu LAUER et Yann LEGUILLE.
- Le paquetage ***p_fichiers*** a été réalisé par Nicolas FERRARI.

De plus Richard a réalisé la procédure principale ***hecatombe.adb*** et la spécification des structures ***structures.ads*** .

Tous les fichiers commentés se trouve a la fin du rapport (voir Code Source dans le sommaire page 3).

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

A. La procédure principale : hecatombe.adb (Richard GUERCI)

Hecatombe.adb est le fichier/procédure principale du programme. C'est lui que l'on doit spécifier pour compiler le projet.

La procédure principale contient deux procédures pour lancer Gkt (Init et Main du paquetage gtk.main) et une autre pour lancer la premier fenêtre (Debut du paquetage p_inteface)

B. Les paquetages

1) Structrure.ads (Richard GUERCI)

Structure.ads est une spécification qui permet de déclarer les principaux types qui sont utilisés dans tous les autres paquetages (p_interface, p_traitement et p_fichier).

La grille du jeu est un tableau (T_grille) de tableau (T_colonne) de couleur (T_couleur).

Type énuméré des couleurs:

```
Type T_Couleur is (r,b,j,x);
```

r pour rouge, b pour bleu, j pour jaune. x représente le vide

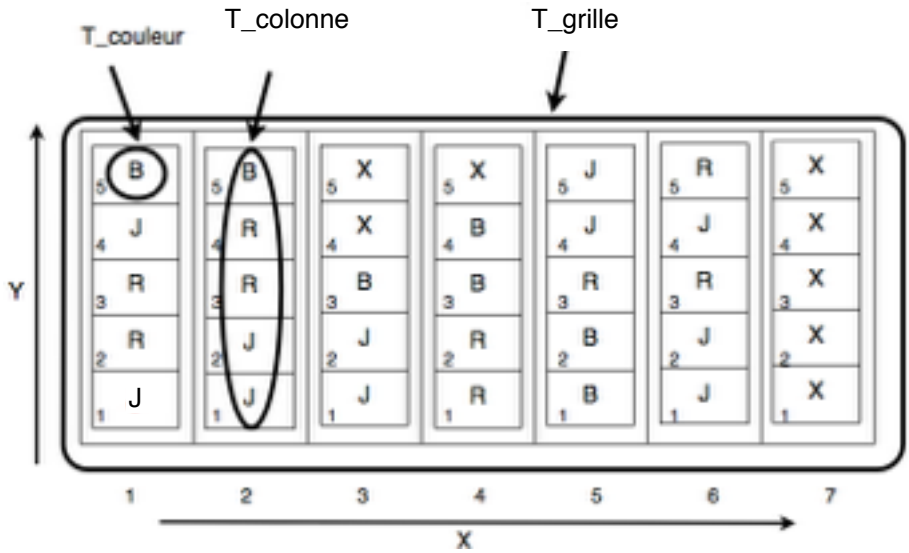
Tableau qui représente un colonne de type T_couleur:

```
Type T_colonne is array (1..Taille_Max_Y) of T_couleur;
```

Tableau de T_colonne:

```
Type T_grille is array (1..Taille_Max_X) of T_colonne;
```

Illustration par un schéma:



GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

Une seconde structure est nécessaire pour gérer des éléments sélectionnés (lors du passage du curseur sur un élément par exemple).
C'est une structure comme ci-dessus à la différence que le tableau de tableau contient des booléens (à vrai si l'élément est sélectionné et à faux sinon):

```
Type T_colonne_bool is array (1..Taille_Max_Y) of boolean;

Type T_grille_bool is array (1..Taille_Max_X) of T_colonne_bool;
```

Une dernière structure est déclaré dans cette spécification pour les scores.
Un tableau de 10 emplacement Tab_score contient une structure T_score qui contient le nom du joueur et son meilleur score

```
type T_score is record
        points:integer:=0;
        nom:string(1..20):=(others=>' ');
end record;

type Tab_score is array (1..10) of T_score;
```

2) P_interface (Richard GUERCI)

J'ai été tout particulièrement intéressé par la réalisation de ce paquetage gérant l'interface graphique du projet car c'est la première fois que je le fais avec un langage évolué (J'avais déjà fait une interface graphique au premier semestre en assembleur).

Comme dans structure.ads j'ai défini un tableau de tableau contenant le type T_bille

```
type T_bille is record
        bouton:Gtk_button;
        image:Gtk_image; --contiendra l'image associé a T_couleur
        Couleur:Gtk_label;--contiendra T_couleur sous forme de caractère
end record;

type T_colonne_graph is array (1..Taille_Max_Y) of T_bille;

type T_grille_graph is array (1..Taille_Max_X) of T_colonne_graph;
```

Le passage de variable d'une fenêtre à l'autre est très compliqué avec GtkAda car l'on ne peut que passer des variables en mode **in** (hors objet gtk) et l'on ne peut que passer qu'une seule variable a la fois.

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

Il a donc fallu créer des structures pour passer plusieurs variables d'une fenêtre à l'autre:

- Structure **parametre_dimensions**
- Structure **parametre_ouvir**
- Structure **parametre_grille**
- Structure **parametre_sauve**
- Structure **parametre_sauve2**
- Structure **parametre_score**
- Structure **parametre_triche**

Pour créer ces variables structures en fonction des variables, j'utilise les fonctions suivantes:

- function Initialisation_structure_parametre_dimensions(Fenetre:Gtk_window;X,Y:Gtk_spin_button)
  return **parametre_dimensions**;

- function Initialisation_structure_prarametre_ouvrir(Fenetre:Gtk_window;Edit:Gtk_Entry)
  return **parametre_ouvrir**;

- function Initialisation_structure_parametre_grille(Grille:T_struct_grille;J,I:integer;Selection:Gtk_label)
  return **parametre_grille**;
-
- function Initialisation_structure_prarametre_sauve(G:T_struct_grille;opt:boolean)
  return **parametre_sauve**;

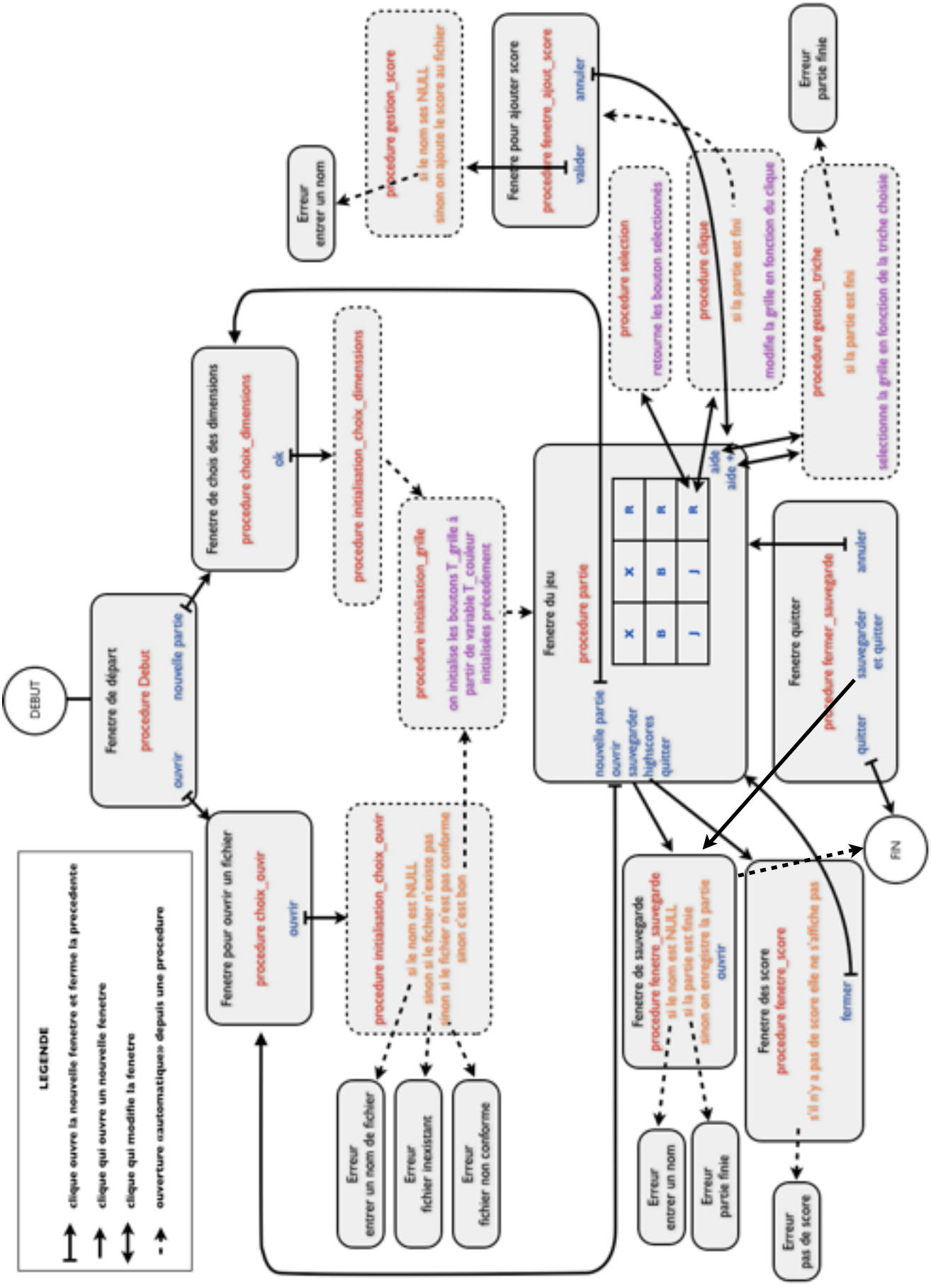
- function Initialisation_structure_prarametre_sauve_2
  (Fenetre:Gtk_window;Struct:parametre_sauve;Edit:Gtk_Entry)
  return **parametre_sauve_2**;

- function Initialisation_structure_prarametre_triche(G:T_struct_grille;opt:boolean;selection:gtk_label)
  return **parametre_triche**;

- function Initialisation_structure_prarametre_score(Fenetre:Gtk_window;Edit:Gtk_Entry;pts:integer)
  return **parametre_score**;

Le paquetage étant assez complexe, j'ai réalisé un schéma détaillant le relation entre les différente fenêtres (page suivante) :

GUERCI Richard
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu
Groupe 1



GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

LEGENDE:
- un rectangle a bordure pleine représente une fenêtre
- un rectangle a bordure discontinue représente une procédure de gestion (qui affiche rien)
- en noir le nom de la fenêtre
- en rouge la procédure qui est associé à la fenêtre
- en orange les conditions
- en bleu les boutons
- en violet les commentaires

La procédure importante est Graph_to_couleur:

```
function Graph_to_couleur(G:T_struct_grille) return T_grille;
```

Elle permet de convertir un grille de type graphique (T_grille_graph) en T_couleur
Cette conversion est nécessaire pour pouvoir effectuer des traitements (enregistrer, sélection, aide, ...)

Enfin, je termine l'explication de ce paquetage par l'explication de la gestion du clique sur un bouton.
Lorsque l'on clique su un bouton, celui-ci envoie un signal: «clicked».
Pour faire le lien entre le signal émit par le bouton et la procédure à appeler, il faut utiliser un paquetage générique de Gtk:

Callback pour appeler une procédure sans paramètre:
```
package Traitement is new Callback(Gtk_Widget_Record);
```
User_Callback pour appelere un procédure avec un paramètre:
```
package Traitement_fenetre is new User_Callback(Gtk_Widget_Record,Gtk_window);
```

On peut alors déclarer le lien:

```
Traitement_fenetre.Connect(
                           Bouton,
                           "clicked",
                           Traitement_fenetre.to_Marshaller(Annuler'Access),
                           Fenetre
                           );
```

Cette ligne permet d'appeler la procédure annuler avec la variable «Fenetre» lorsque l'on clique sur le bouton «Bouton».

```
procedure Annuler(Element_declancheur:access Gtk_Widget_Record'Class;fenetre:Gtk_window);
```

On pourra alors cacher la fenêtre passée en paramètre dans la procédure annule.

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

3) P_traitement (Richard, Yann et Mathieu)

P_traitment 'travaille' avec les types de structure.ads.

a. Fonctions et procédures réalisées Richard GUERCI

-function Traitement_selection(grille:T_grille;x,y,i,h:integer) return T_grille_bool;

Cette fonction initialise deux grille (de dimension x,y) de booleen à faux.
Si l'élément est sélectionnable, c'est à dire si c'est une couleur, alors on appel la procédure récursive Parcours_grille qui retourne la sélection du groupe de bille dans la variable res.

-procedure Parcours_grille(Elt:T_couleur;grille:T_grille;Tab,check:in out T_grille_bool;x,y,w,z:integer);

C'est une procédure récursive qui compare si les cases voisines sont de la même couleurs.
Si la case voisine est de la même couleur alors on mes cette case à vrai (variable Tab). Dans tous le cas, si on va sur un case on met a vrai check, de cette façon si un case a déjà été vérifié, on arrête la procédure récursive.

-function Aide_1(grille:T_grille;x,y:integer) return T_grille_bool; -- fait par Richard Guerci

Cette fonction récupère les coordonnées de la case qui donne la plus grande sélection. Puis elle retourne un tableau de booleen de la meilleur sélection grâce aux coordonnées.

-function Aide_2(grille:T_grille;x,y:integer) return T_grille_bool; -- fait par Richard Guerci

C'est une fonction qui a pour but renvoyer la meilleur sélection en fonction des coups à venir.
Ici on initialise res à faux et on récupère l'horloge (expliqué dans la procédure simulation).
Ensuite on appel Simulation qui fait le travaille et on retourne res qui contient le meilleur coup pour obtenir le meilleur score

-procedure Simulation(TempsDepart:Time;grille:T_grille;x,y,profondeur:integer;
points: in out integer;grille_bool: in out T_grille_bool;fond: out boolean);

C'est pour moi la procédure la plus intéressante du projet.
C'est une procédure récursive qui renvoie la sélection de profondeur 0 si le score associé a cette sélection a été le plus grand score trouvé trouvé parmi tous les possibilité de profondeur n.
Il y a plus plusieurs conditions d'arret:

- la partie est finie
- la profondeur maximal (en fonction du nombre de bille restante) a été atteinte
- ou le délais d'exécution a été dépassé

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

b. Fonctions et procédures réalisées Yann LEGUILLE

-procedure Tri_colonne(C : in out T_colonne;Y : integer);

Cette procédure permet de "trier" les colonnes, de placer les cases vides, lorsque les billes sont supprimées, en haut de la grille et ainsi faire redescendre les billes de couleur. On a 'y' qui correspond à la longueur de la colonne et 'C' qui représente la colonne contenant les billes à trier.

-procedure Tri_grille(G : in out T_grille;W : integer);

Cette procédure permet de décaler les colonnes vers la gauche lorsque l'une d'elle est vide, soit remplie que de cases vides. 'G' représente la grille à trier et 'w' représente le nombre de colonnes contenues dans la grille.

-procedure Traitement_supprime(x,y : in integer; Tab_bool : in T_grille_bool; Grille : in out T_grille);

Cette procédure permet de supprimer les billes sélectionnées, que l'on retrouve dans le tableau Tab_bool, on les transforme en case grise. 'x' et 'y' représentent les coordonnées de la bille à supprimer, 'tab_bool' contient les coordonnées des billes sélectionnées et 'grille' représente la grille de jeu.

-function Nb_selection(T :T_grille_bool; x,y : integer) return integer;

Cette fonction retourne le nombre de billes sélectionnées. 'T' contient les coordonnées des billes sélectionnées, 'x' et 'y' représentent le nombre de lignes et de colonnes.

-procedure Gravite(Grille : in out T_grille; x,y : integer);

Cette procédure permet de trier toute la grille après la suppression, elle fait appel aux procedures tri_colonne et tri_grille ce qui permet de mettre les billes au bon endroit après avoir joué un coup. 'Grille' correspond à la grille de jeu et 'x' et 'y' correspondent au nombre de lignes et de colonnes.

-function Fin_partie(Grille : T_grille; x,y : integer) return boolean;

Cette fonction teste si la partie est finie en fonction des billes qu'il reste. 'Grille' est la grille de jeu et 'x' et 'y' correspondent au nombre de lignes et de colonnes.

c. Fonctions et procédures réalisées Matthieu LAUER

-function Points(pts, selection:integer) return integer ;

Cette fonction permet de calculer le score du joueur en fonction du nombre de bille détruite.

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

-function Meilleur_score(score : Tab_Score; nb,pts : Integer) return boolean ;

Cette fonction permet de savoir si le score fait appartient aux 10 meilleurs scores.
La table 'Tab_score', contient les scores déjà enregistré, ainsi que le classement des scores.
La variable 'pts' correspond aux points effectué par le joueur actuel, et la variable 'nb' détermine la position.

-procedure Ajout_score(score : in out Tab_Score; nb: in out Integer;nom:string;pts:integer ;

Cette procédure va permettre d'ajouter un score, au tableau des scores. Le score qui va être ajouté, va être comparé aux autres scores existants afin de l'insérer à la bonne place. Ainsi, dans le tableau figurera :
-la position du score (la case du tableau Tab_score).
-le score du joueur.
-le nom du joueur.

-function Position_score(score:Tab_Score; nb,pts:integer) return integer ;

Cette fonction retourne la position du score effectué. Cela permet ainsi de connaître si on est dans les 10 meilleurs joueurs, ou non.

3) P_fichier (Nicolas FERRARI)

J'ai créé le paquetage P_FICHIER, qui gère les fichiers textes utilisés pour sauvegarder et charger une partie, ainsi que le fichier contentant les 10 meilleurs scores réalisés.

- function Existe(nom : string) return boolean;

Cette fonction teste si un fichier portant le nom entré en paramètre existe ou non.

- function Conforme(nom : string) return boolean;

Cette fonction teste si le fichier portant le nom "nom" est conforme aux fichiers texts contenant les informations d'une partie : nombre de lignes, de colonnes, points, grille textuelle.

- procedure Chargement(nom : in string; z,y,points : out integer; Grille : out T_Grille);

Cette procédure charge un fichier texte contenant les informations d'une partie. Il charge la grille textuelle pour en faire une grille graphique.

- procedure Sauvegarde(nom : in string; x,y,points : in integer; Grille : in T_Grille);

Cette procédure sauvegarde une partie dans un fichier texte.

- procedure Sauve_Score(score : Tab_Score; nb : Integer);

Cette procédure sauvegarde le score dans un fichier texte nommé “highscore.txt”.

- procedure Charge_Score (score : out Tab_Score; nb: out Integer);

Cette procédure charge le tableau des scores obtenu après chargement du contenu du fichier texte “highscore.txt”.

Dans chacune de ces procédures et fonctions, je créé une variable F de type FILE_TYPE (fichier) permettant la manipulation des fichiers et le bon fonctionnement de ces procédures et fonctions.

III. Conclusion

Le projet Algo - ADA nous a permis de mieux nous familiariser avec le langage ADA mais aussi avec les logiques de programmation étudiées tout au long de l’année.

Ce projet nous a permis d’entreprendre les démarches d’un informaticien, de la conception à la rédaction d’un mode d’emploi en passant par l’analyse de l’application et la programmation.

Bien que le travaille fourni par chacun n’ai pas été équivalent, le programme obtenu est fonctionnel, sans bug connu et amusant.

Pour conclure, le projet Algo - ADA a été une très bonne expérience.

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

MODE D'EMPLOI

I. Présentation du jeu

Hécatombe est un jeu destiné à tout le monde, sans limite d'âge.
C'est un jeu de stratégie où il faut éliminer des groupes de billes de couleurs (au minimum 2) jusqu'à ce qu'il n'en reste plus du tout.
La partie s'arrête donc quand il n'y a plus de billes ou qu'il n'y plus de coup possible.

II. Le score

Le but du jeu est de faire le meilleur score possible.
Pour cela, supprimer les plus grands groupes de billes. Les groupes de deux billes ne rapportent pas de points.
Si la grille est totalement vide a la fin du jeu il vous est attribué un bonus de 1000 points

III. Comment jouer

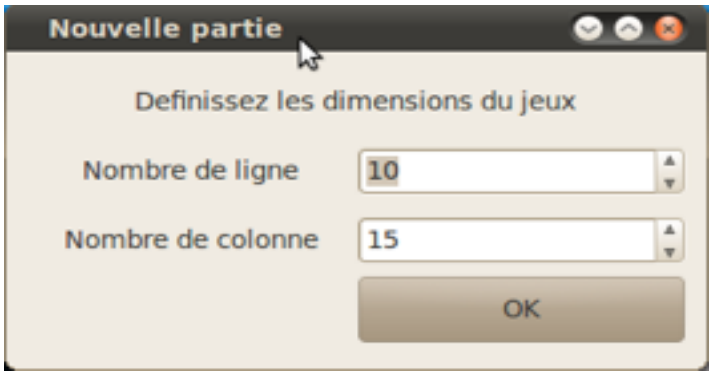
A. Lancement du jeu

Pour commencer à jouer à hécatombe double-cliquez sur le jeu (hecatombe.exe).
Après cela une fenêtre va s'ouvrir vous demandant si vous voulez commencer une nouvelle partie ou bien reprendre une partie enregistrée.

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

B. Commencer une nouvelle partie

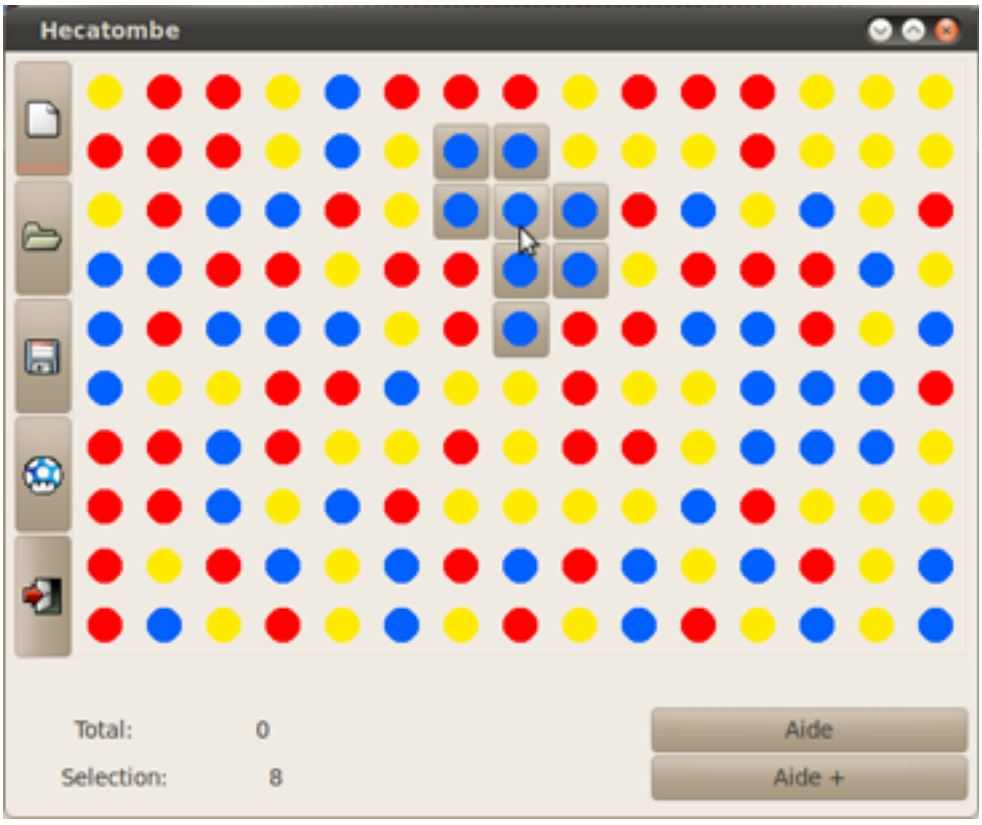
Pour débuter une nouvelle partie cliquez donc sur le bouton « Nouvelle partie », à ce moment une nouvelle fenêtre va s'ouvrir pour vous demander de choisir les dimensions de la grille (entre 5 et 20)



C. Sélection des billes

Une fois que le choix des dimensions a été effectué le fenêtre de jeu s'ouvre et vous pouvez commencer à jouer.

Pour jouer il suffit de passer le curseur de la souris sur une bille et un groupe de billes de même couleur va être sélectionné. Le nombre de billes sélectionnées sera affiché en bas à gauche.

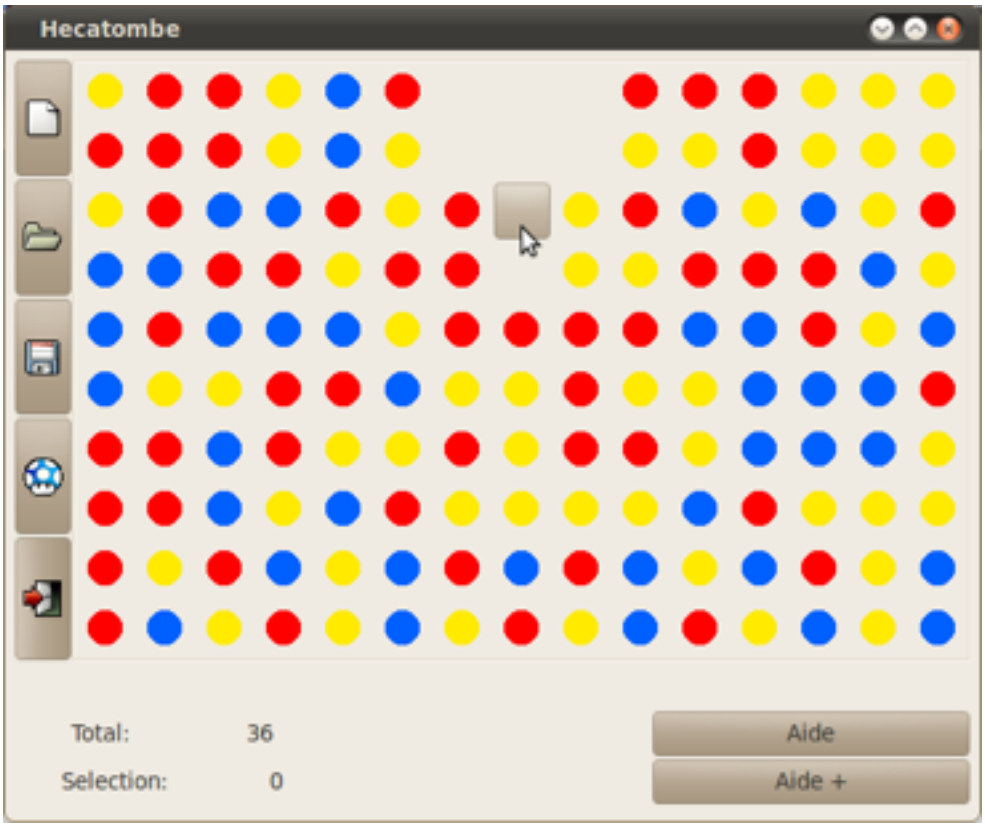


GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

La sélection comprend 8 billes

D. Premier coup

Une fois que vous avez choisi quel groupe de billes vous voulez éliminer cliquez dessus.
La sélection va disparaitre et votre score augmentera.
Celui-ci est affiché en bas à gauche en face de « Total: ».



Répétez ces étapes jusqu'à ce qu'il n'y ait plus de billes.

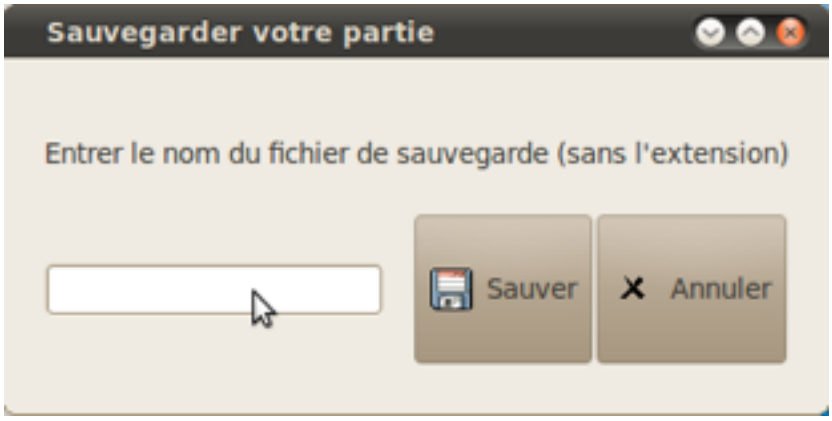
GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

E. Sauvegarde de la partie

Si vous voulez arrêter de jouer et reprendre votre partie plus tard il vous faudra cliquer sur la petite disquette située sur la gauche de la fenêtre.

Lorsque vous cliquez dessus une fenêtre apparaît et vous demande d'entrer un nom de sauvegarde. Une fois le nom entré cliquez sur « Sauver », votre partie sera enregistrée et vous pourrez la continuer plus tard ou bien cliquez sur annuler pour annuler l'opération.

Remarque: il est impossible de sauvegarder une partie finie.

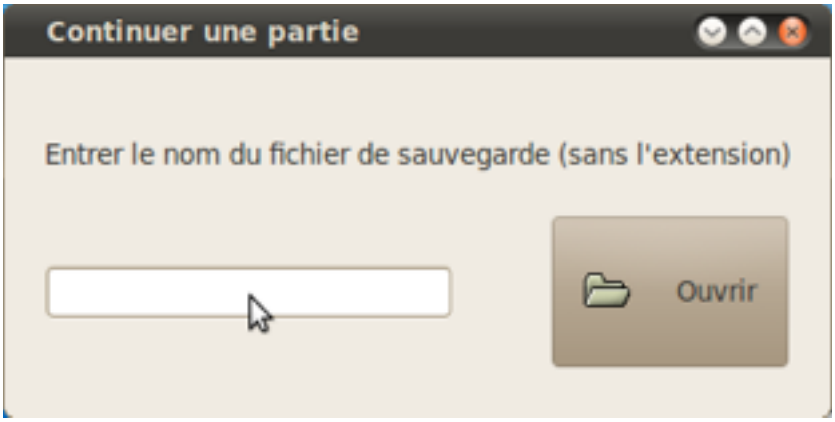


GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

F. Chargement d'une partie

Si vous désirez reprendre une partie enregistrée il vous suffit de cliquer sur le bouton ci-dessous.

Une fenêtre va s'ouvrir vous demandant d'entrer le nom du fichier de sauvegarde (sans extension).



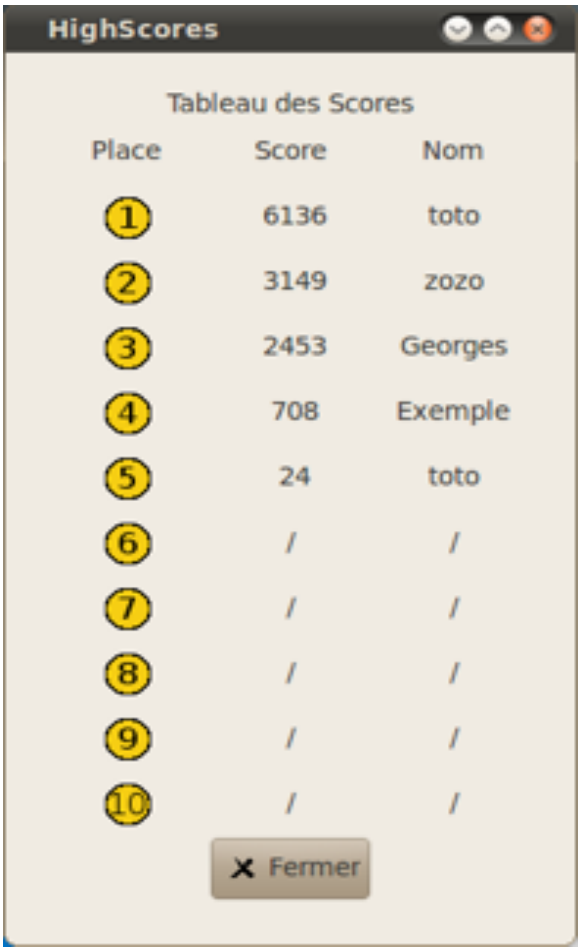
Une fois le nom entré, cliquez sur « Ouvrir » et vous pourrez reprendre votre partie.

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

G. Affichage des meilleurs scores

Vous pourrez consulter le classement des 10 meilleurs scores à n'importe quel moment du jeu en cliquant sur le petit champignon.

Une fenêtre apparaît et vous dévoile le Top 10 des meilleurs joueurs

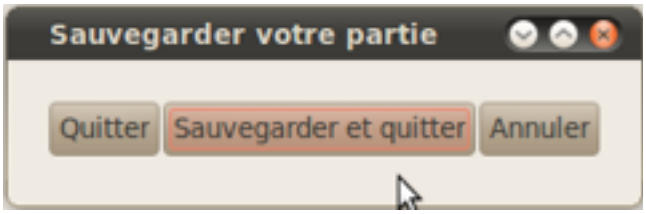


GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

H. Quitter le jeu

Si vous souhaitez arrêter le jeu il vous suffit de cliquer sur la petite porte.

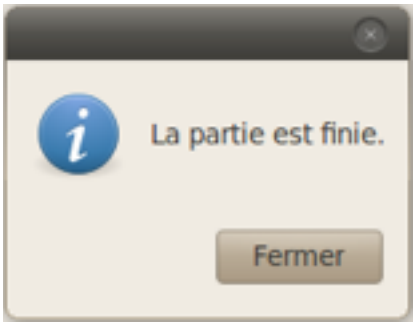
Une fenêtre s'ouvre pour vous demander si vous voulez quitter le jeu sans sauvegarder, ou bien en sauvegardant la partie, ou annuler.

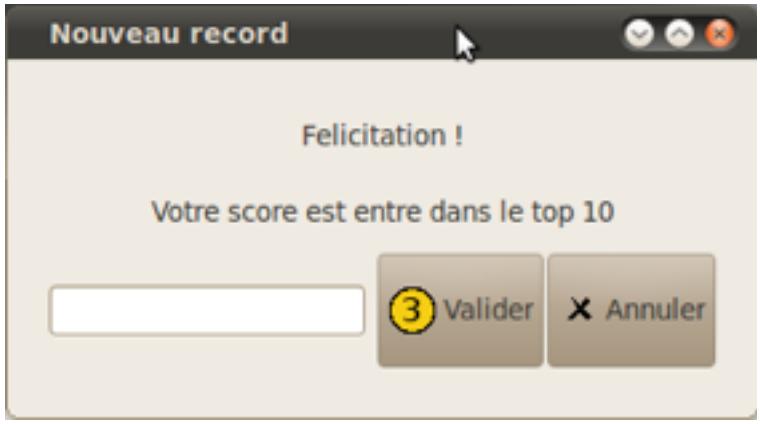


Si vous cliquez sur « Quitter » la fenêtre se ferme et votre partie courante sera perdue.
En cliquant sur « Sauvegarder et quitter » la fenêtre de sauvegarde apparaît (voir E. Sauvegarde de la partie). Après l'enregistrement la fenêtre se ferme.

I. Fin de partie

Lorsque la partie est finie, c'est-à-dire qu'il n'y a plus de billes ou plus de coup possible, une fenêtre vous avertit que la partie est finie, cliquez sur « Fermer » et entrez votre nom dans la fenêtre « Nouveau record » si votre score vous le permet. Valider votre nom ou annuler si vous ne voulez pas enregistrer votre score.





GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

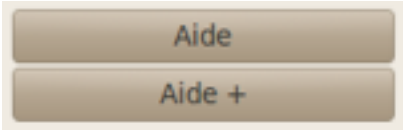
J. Nouvelle partie

Maintenant vous pouvez quitter le jeu ou bien recommencer une nouvelle partie en cliquant sur le bouton.

Une fenêtre apparaîtra vous demandant les dimensions de la nouvelle partie (voir B. Commencer une nouvelle partie).

K. Triche

Pour ceux qui auraient un peu de mal à faire en sorte qu'il n'y ait plus de billes, il y a une petite « Aide ». En bas à droite il y a deux boutons



Ceux-ci permettent de vous donner le meilleur coup à jouer. Il met en surbrillance le groupe de bille à éliminer.
« Aide » vous donnera le groupe de billes qui vous rapportera le plus de points.
« Aide + » vous donne le groupe de billes qui vous rapportera le plus de points en fonction des coups à venir (il donne le coup permettant d'obtenir le meilleur score possible de la grille)

GUERCI Richard *Groupe 1*
FERRARI Nicolas
LEGUILLE Yann
LAUER Mathieu

CODE SOURCE