
Rapport du projet informatique

Langage C - jeu Picross

Projet réalisé par

Thierry KHAMPHOUSONE
Nimrod Ndoudi

Projet encadré par

Mme RAQUEL MARTINS
M GONZALEZ

Table des matières

1. Présentation du jeu	5
1.1. Le jeu PICROSS	5
1.2. Les règles du jeu	5
1.3. Le déroulement du jeu	7
1.4. Le menu du jeu	9
2. Implémentation du jeu	11
2.1. La bibliothèque NCURSES	11
2.2. La bibliothèque SOX	13
2.3. Le contexte logique appliqué	14
2.4. Les fonctions principales	20
2.5. Le design	30
2.6. Les fonctionnalités cachées du jeu	35
3. Crédits du jeu	40
4. Conclusion	41

1. Présentation du jeu

1.1. Le jeu PICROSS

Le jeu PICROSS est un jeu classé dans le genre "Casse-tête". Son but est de faire apparaître une image cachée en coloriant les cases correspondantes aux nombres situés à gauche et au-dessus de la grille de jeu.

Nous avons inventé de nombreuses images que vous pourrez découvrir durant votre expérience au sein du jeu.

Mettez-vous également au défi avec le mode aléatoire à dimension sur demande !



Une image secrète est dissimulée quelque part dans le jeu...

... Saurez-vous la retrouver ?



1.2. Les règles du jeu

Pour pouvoir compléter nos grilles PICROSS, mieux vaut prendre connaissance des règles du jeu.

Une fois ces règles assimilées, plus aucune grille PICROSS ne vous résistera !

✱ Les nombres situés sur le **côté gauche** de la grille PICROSS permettent d'identifier le nombre **total de cases à colorier à la suite sur la ligne** correspondante.

✱ Les nombres situés sur le **côté supérieur** de la grille PICROSS permettent d'identifier le nombre **total de cases à colorier à la suite sur la colonne** correspondante.

✱ Lorsqu'il y a plus d'un nombre dans une ligne ou colonne, cela signifie qu'il existe au moins 1 espace vide non colorié entre chaque suite de cases coloriées à la suite sur la ligne ou colonne.

✱ Les cases considérées comme "non remplies" devront être complétées par des croix afin d'être considérées comme vides par le joueur.

Voici quelques exemples...

Nous avons ici 9 puis 4, cela signifie que nous devons d'abord colorier 9 cases à la suite, cependant il est probable que des cases vides existent avant ces 9 cases à colorier.

Ensuite, nous devons laisser au moins 1 espace et enfin colorier 4 cases à la suite.



Il existe donc plusieurs méthodes de remplissage...



Ou bien...



Ou encore...



Il existe cependant une seule bonne combinaison possible, et pour la découvrir, il faut se référer aux nombres situés au-dessus de la grille PICROSS permettant de remplir les colonnes.

Note : Lorsque les lignes et colonnes ont été correctement remplies, un affichage apparaît afin de valider la ligne ou la colonne correspondante.

1.3. Le déroulement du jeu

Le joueur débute facilement une grille PICROSS du jeu en coloriant les cases les plus faciles de la grille.

Par exemple, notre grille PICROSS est de taille 3x3. Sur la troisième ligne et troisième colonne est demandé de colorier 3 cases à la suite.

RETURN MENU		2	1	3
1				
1 1				
3				

Il est donc plus facile de résoudre la grille PICROSS en commençant par colorier à la suite les cases "remarquables".

Nous remarquons par la suite que sur la première ligne et la deuxième colonne est demandé de colorier qu'une seule case à la suite.

RETURN MENU		2	1	3
1		X	X	
1	1		X	
3				

Cependant nous avons déjà une case coloriée. Nous pouvons donc en déduire que les cases restantes sont des cases "vides" et y ajouter des croix.

Enfin, sur la deuxième ligne est demandé [1 1], soit 1 case coloriée à la suite, puis au moins 1 espace, puis 1 case coloriée à la suite.

Dans notre cas, il est très facile d'en déduire la dernière case à colorier pour clôturer l'affichage de notre grille PICROSS 3x3.

Afin de remplir toute la grille PICROSS et de la valider, toutes les cases doivent être remplies soit par le coloriage des cases en y ajoutant un carré, soit par l'élimination des cases en y ajoutant une croix.

1.4. Le menu du jeu

Accompagné d'un fond d'écran animé recouverts de neige, laissez-vous emporter par la musique douce du menu principal...



Nous avons ici un large choix de mode de jeu...

RANDOM MODE : Mode de jeu permettant de jouer sur une grille PICROSS à cases aléatoires d'une taille souhaitée comprise entre 1x1 et infinie x infinie. Il est tout de même recommandé d'opter pour une grille de taille minimale 4x4 et de taille maximale 17x25 pour un écran de taille 15 pouces par exemple.

Cependant, rien ne vous empêche de réduire la taille d'affichage du terminal, d'opter pour une grille PICROSS de taille 75x150 et donc de remplir les 11 250 cases...

ΣΧ3

Avez-vous déjà essayé de résoudre l'énigme proposé dans les CREDITS ... ?

ΣΧ3

PICTURES MODE : Mode de jeu permettant de jouer sur une grille PICROSS de taille prédéfinie. Une image cachée est à découvrir pour chacun des niveaux proposés !

LETTERS MODE : Mode de jeu permettant de jouer sur une grille PICROSS de taille prédéfinie. Un mot caché est à découvrir pour chacun des niveaux proposés !

CREDITS : Profitez d'une musique rétro tout en lisant les crédits.

LEAVE THE GAME : Malgré sa couleur sombre, cette touche s'avère plutôt utile, elle permet au joueur de quitter le jeu. Par son addiction, le joueur va devoir relancer le jeu de nouveau.

2. Implémentation du jeu

2.1. La bibliothèque NCURSES

Nous avons décidé d'utiliser la bibliothèque Ncurses pour la création de notre jeu sur terminal. Ce dernier nous permet d'ouvrir une terminal mode NCURSES dans notre terminal.

La bibliothèque Ncurses nous a été très utile pour son aspect graphique. En effet, il nous a permis d'ajouter des caractères aux différents emplacements souhaités sur le terminal NCURSES et donc de nous permettre de dessiner notre grille PICROSS.

L'utilisation de la NCURSES.H se rapproche énormément du code en langage C des bibliothèque STDLIB.H et STDIO.H.

Voici quelques exemples ...

STDLIB.H/STDIO.H	NCURSES.H
printf	printw
scanf	scanw

Elle possède également des commandes propres à son utilisation.

NCURSES	UTILISATION
<code>move(y,x)</code>	Déplace le curseur à la position (y,x)
<code>addch(CHAR_TYPE)</code>	Ajoute un caractère
<code>initscr()</code>	Démarre le mode NCURSES
<code>endwin()</code>	Arrête le mode NCURSES
<code>getch()</code>	Met en pause le jeu et attend
<code>refresh()</code>	Rafraîchis l'affichage du terminal
<code>attron(ATTRIBUTE_TYPE)</code>	Active l'attribut
<code>attroff(ATTRIBUTE_TYPE)</code>	Désactive l'attribut
<code>start_color()</code>	Démarre le mode couleur
<code>init_color(COLOR_TYPE, x, y, z)</code>	Initialise une couleur à une intensité x, y, z
<code>assume_default_color(COLOR_TYPE, COLOR_TYPE)</code>	Défini une palette de couleur par défaut au terminal
<code>init_pair(x, COLOR_TYPE, COLOR_TYPE)</code>	Défini une palette de couleur x
<code>curs_set(TRUE)</code>	Active l'affichage du curseur
<code>curs_set(FALSE)</code>	Désactive l'affichage du curseur
<code>echo()</code>	Active l'affichage des caractères saisis
<code>noecho()</code>	Désactive l'affichage des caractères saisis
<code>keypad(stdscr, TRUE)</code>	Active les touches spécifiques
<code>keypad(stdscr, FALSE)</code>	Désactive les touches spécifiques
<code>cbreak()</code>	Désactive la mise en buffer

2.2. La bibliothèque SOX

Nous avons utilisé la bibliothèque SOX pour y ajouter des musiques, des sons ainsi que des effets sonores.

Voici quelques exemples ...

SOX & KILL	UTILISATION
play FILE.mp3	Joue FICHER.mp3 une seule fois
play FILE.mp3 repeat 99	Joue FICHER.mp3 99 fois
play -q FILE.mp3	Joue FICHER.mp3 sans affichage (quiet)
play FILE.mp3 vol-6db	Joue FICHER.mp3 et diminue de 6db
pkill -n	Permet d'arrêter le dernier processus lancé
pkill play	Permet d'arrêter les processus du nom de "play"

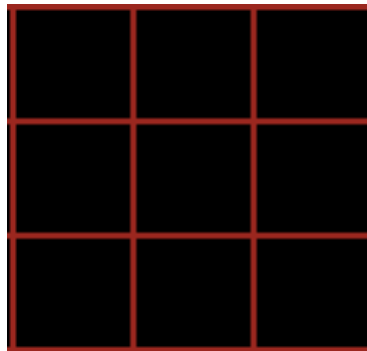
2.3. Le contexte logique appliqué

Au commencement de la conception du jeu, nous avons tout d'abord dessiné la plateforme de jeu. En effet, débiter par le plateau de jeu nous a permis de mieux visualiser nos premiers objectifs et de pouvoir répondre aux problèmes.

"Une grille capable de contenir des éléments"

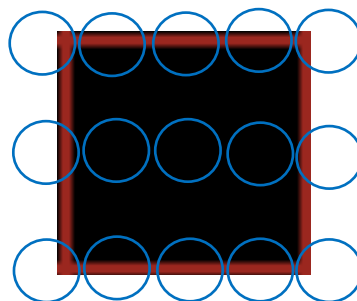
Notre premier problème était de dessiner des cases tout considérant un caractère vide au centre.

Voici une grille PICROSS 3x3



Solution : Pour chacune des cases que nous avons dessinées, nous avons utilisé des `[addch(ACS_TYPE)]` pour pouvoir tracer les lignes et les angles.

Voici une case de la grille PICROSS 3x3



Notre case possède 15 coordonnées différentes dont celles situées au centre qui nous permettent d'afficher des caractères.

Note : les cercles bleus permettent de distinguer les différentes positions ainsi que les `addch(ACS_TYPE)` utilisés pour les lignes rouges.

"Une grille capable d'afficher un élément"

Notre second problème était d'afficher les éléments uniquement dans la grille de jeu et entre les lignes de la grille.

En effet, lorsque nous cliquons, nous faisons afficher un caractère à la position du click. Cependant, nous n'avions pas délimité le tableau et avons affiché des caractères en dehors ainsi que sur les lignes du tableau.

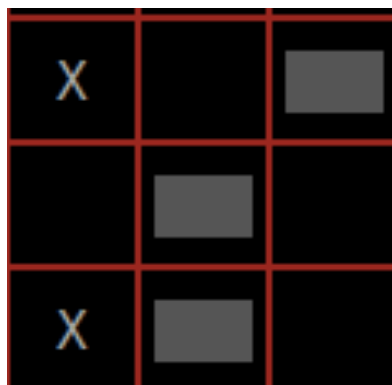
Solution : Pour répondre à ce problème, il nous suffisait d'autoriser l'affichage d'un caractère uniquement si le click appartenait au tableau et si seulement le caractère appartenait à une case du tableau et non une ligne. Pour cela, nous nous sommes référés aux coordonnées des lignes du tableau.

"Une grille capable d'afficher différents éléments entre chaque click"

Notre troisième problème était de pouvoir afficher 3 caractères différents entre chaque click. Nous avons besoin d'une croix, un rectangle (permettant de colorier la case) ainsi qu'un caractère vide.

Solution : Pour cela, nous avons utilisé une incrémentation d'une variable initialisée à 0 à chaque click et nous avons ajouté un modulo 3 à cette incrémentation afin d'obtenir 3 cas différents.

C'est-à-dire 0 pour le vide, 1 pour le rectangle et 2 pour la croix.



"Une grille capable d'afficher le nombre de cases à colorier"

Notre quatrième problème à résoudre était de pouvoir dans un premier temps de compter les nombres de cases à colorier, puis dans un second temps de les afficher dans les lignes et colonnes à côté de la grille PICROSS.

Solution : Pour résoudre le problème, nous avons initialisé un compteur et utilisé deux boucles itératives l'une dans l'autre. Ce compteur incrémentait à chaque fois qu'il rencontrait le chiffre 1. Si par la suite il rencontrait le chiffre 0, alors il affichait le compteur dans la ligne/colonne correspondante.

Il existe différent cas, par exemple s'il ne rencontre que des 0, et que le compteur est égal au nombre de colonnes/lignes, alors il affiche 0 dans la ligne/colonne correspondante.

Ou encore, s'il n'y a que des 1 sur toute une ligne/colonne ou sur une grande longueur et ne rencontre pas de 0 à la fin, alors de même, il affiche le compteur dans la ligne/colonne correspondante.

2
3
2

2	3	2
---	---	---

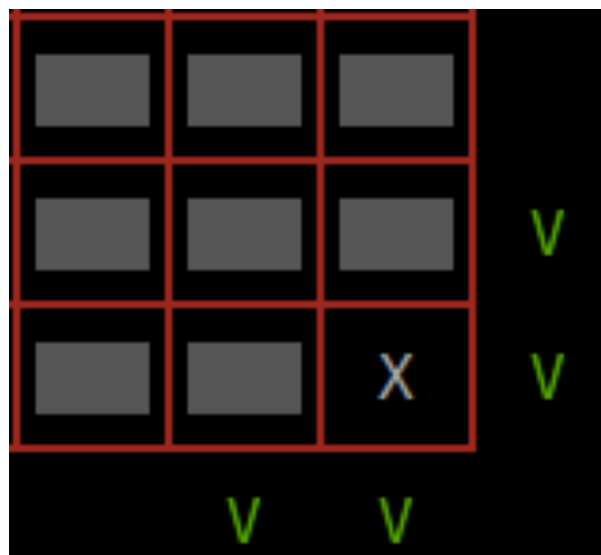
"Une grille capable de signaler au joueur qu'une ligne/colonne a été validée"

Notre cinquième problème était de pouvoir vérifier si le joueur a bien complété une ligne/colonne.

Solution : Pour résoudre ce problème, nous avons utilisé deux boucles itératives qui compare le tableau du résultat (TAB[i][j]) avec le tableau du joueur (FLAGTAB[i][j]). Si les deux tableaux aux cases [i][j] sont identiques, alors un compteur initialisé à 0 incrémente. Si le compteur incrémenté est égal au nombre total de colonne, cela veut dire que le joueur à bien complété la ligne/colonne.

Pour comparer en ligne, nous faisons incrémenter la variable j de 0 à la taille maximale de la colonne-1 inclus le tout dans la boucle itérative i allant de 0 à la taille maximale de la ligne-1 inclus.

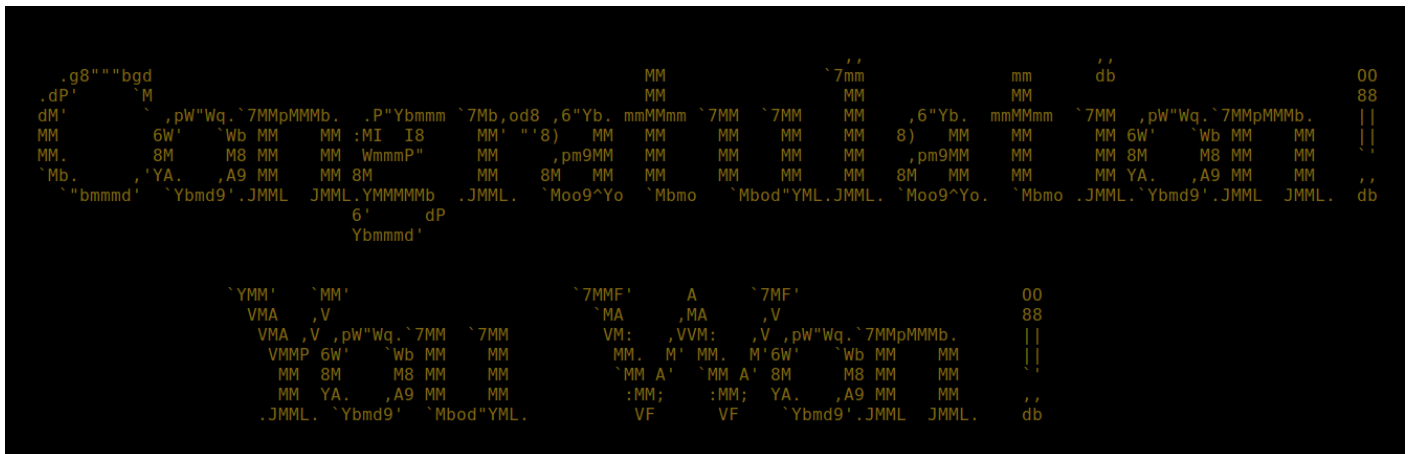
Pour comparer en colonne, nous faisons incrémenter la variable i de 0 à la taille maximale de la ligne-1 inclus, le tout dans la boucle itérative j allant de 0 à la taille maximale de la colonne-1 inclus.



"Une grille capable d'annoncer la victoire !"

Notre sixième problème était de pouvoir déterminer le moment où le joueur a totalement complété la grille PICROSS avec succès et d'afficher un message "Congratulation ! You Won !".

Solution : Pour résoudre ce problème, il nous suffisait de comparer le tableau du résultat (TAB[i][j]) avec le tableau du joueur (FLAGTAB[i][j]) grâce à deux boucles itératives. Si les deux tableaux sont totalement identiques, alors on affiche "Congratulation ! You Won !".



"Une grille capable d'afficher la réponse durant le mode image"

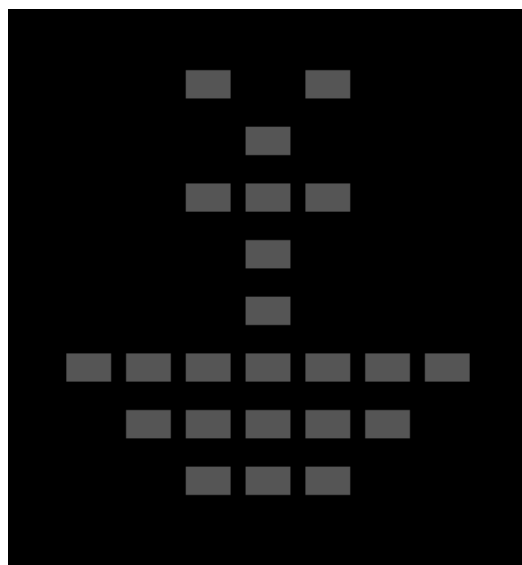
Notre septième problème consistait à afficher l'image modèle pendant que le joueur joue. Cependant, lorsque le joueur joue et qu'il a déjà commencé à remplir des cases, lorsque nous affichions le modèle, la grille de jeu avait totalement disparu, les cases déjà coloriées également ainsi que les validations des lignes/colonnes.

Solution : Pour résoudre ce problème, nous avons créé un bouton, qui lorsque nous appuyons dessus, cela nous demande si on souhaite vraiment afficher le modèle. Pour pouvoir réafficher les éléments disparus, nous avons réaffiché le tableau, créé une fonction qui permet de tester les cases déjà coloriées par le joueur et de les réafficher, et réafficher les encoches vertes pour signaler une ligne/colonne déjà validée.

RETURN MENU		1	2	1 1 3	7	1 1 3	2	1
PATTERN								
1 1		X			X			
1								
3								
1								
1			X		X			
7								
5		X						X
3		X	X				X	X
V								

Do you want to see the pattern ?

YES NO



Le modèle a été affiché, rien n'a été effacé lorsque nous retournons sur le jeu.

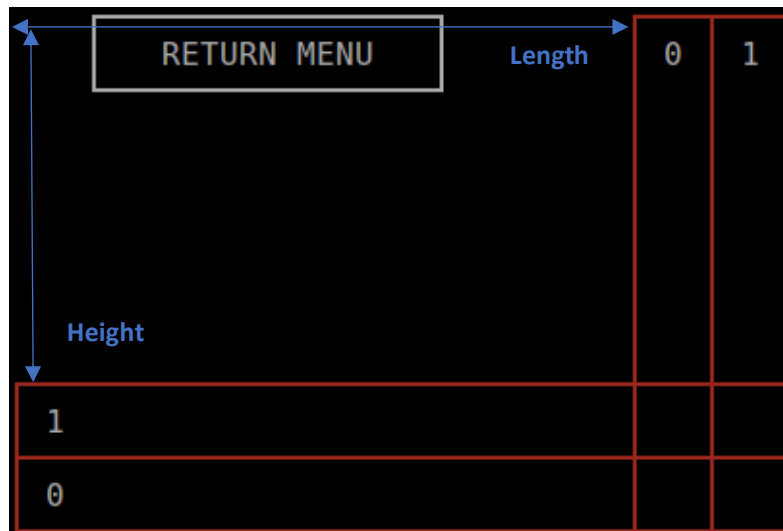
2.4. Les fonctions principales

Le jeu PICROSS possède des fonctions principales qui sont les plus importantes pour le fonctionnement du jeu.

Commençons par la fonction DRAW_TAB

```
771 // Affichage graphique du tableau de jeu PICROSS
772 void DRAW_TAB(int lin, int col, int TAB[lin][col])
773 {
774     int i,j;
775
776     for(i=0; i< (Height + NBboxlin)+1; i++)
777     {
778
779         // Initialisation de la première ligne du tableau graphique "le couvercle" de la box a nombre n°1
780         if(i == 0)
781         {
782             for(j=0; j<NBboxcol; j++)
783             {
784                 if(j == 0)
785                 {
786                     move(i,Length);
787                     addch(ACS_ULCORNER);
788                 }
789
790                 if(j%2 == 1 && j>0 && j<(NBboxcol*3))
791                 {
792                     addch(ACS_HLINE);
793                     addch(ACS_HLINE);
794                     addch(ACS_HLINE);
795                 }
796
797                 if(j%2 == 0 && j>0 && j<NBboxcol-2)
798                 {
799                     addch(ACS_TTEE);
800                 }
801
802                 if(j == NBboxcol-1)
803                 {
804                     addch(ACS_URCORNER);
805                 }
806             }
807             printw("\n");
808         }
809
810         // Initialisation des bordures et inter lignes entre le couvercle et le tableau de jeu
811         if(i != 0 && i<Height)
812         {
813             for(j=0; j<NBboxcol; j++)
814             {
815                 if(j == 0)
816                 {
817                     move(i,Length);
818                     addch(ACS_VLINE);
819                 }
```

Située entre les lignes 772 et 1005, cette fonction permet de dessiner l'affichage de la grille PICROSS en fonction de la taille donnée par lin et col.



Le principe est le suivant, nous utilisons deux boucles itératives pour j allant de 0 à col-1 inclus dans la boucle pour i allant de 0 à lin-1 inclus. **Nous débutons donc par la ligne 0** et nous parcourons de gauche à droite les colonnes sur cette même ligne.

Voici quelques exemples du principe

Ligne 0 :

- si j est égal à Length, nous ajoutons le caractère **ULCORNER**.
- si j est divisible par deux, nous ajoutons le caractère **BTEE**, sinon nous ajoutons les caractères **HLINE**.
- si j est égal à Length + taille maximale colonne, alors nous ajoutons le caractère **URCORNER**.

Ligne 1 à Height-1 inclus:

- si j est égal à Length, nous ajoutons le caractère **VLINE**.
- si j est divisible par deux, nous ajoutons le caractère **VLINE**, sinon nous ajoutons les **caractères vides**.
- si j est égal à Length + taille maximale colonne, alors nous ajoutons le caractère **VLINE**.

Même principe jusqu'à la fin de notre grille PICROSS.

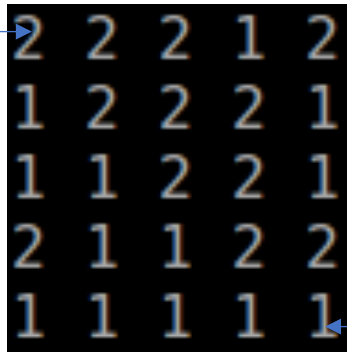
Continuons avec la fonction `FILL_TAB`

```
1066 void FILL_TAB(int lin, int col, int TAB[lin][col])
1067 {
1068     int i,j;
1069     int compteur1 = 0;    //pour les carrés
1070     int compteur2 = 0;    //pour les croix
1071     int compteur0 = 0;    //pour le cas où il n'y a rien sur lignes/colonnes
1072     int compteur0bis = 0;
1073     int LLEFT = 11;       //position de base du curseur pour écriture gauche
1074     int CLEFT = 2;
1075     int LUP = 1;          //position de base du curseur pour écriture haut
1076     int CUP = 34;
1077
1078     for(i=0; i<lin; i++)
1079     {
1080         for(j=0; j<col; j++)
1081         {
1082             if(TAB[i][j]==2)                //cas où = 2
1083             {
1084                 compteur0++;
1085
1086                 if(compteur0 == col-1)
1087                 {
1088                     compteur0 = 0;
1089                     SHOW_LEFT(LLEFT, CLEFT, compteur0);
1090                 }
1091
1092                 if (compteur1 == 0 && j == col-1) //cas où fin = 0 et compteur NON
1093                 {
1094                     //équivalent à un printf("\n");
1095                     LLEFT+=2;
1096                     CLEFT = 2;
1097                     JUMP_LEFT(LLEFT, CLEFT);
1098                     compteur1 = 0;
1099                     compteur0 = 0;
1100                 }
1101
1102                 if (compteur1 >= 1 && j == col-1) //cas où fin = 0 et compteur OUI
1103                 {
1104                     //équivalent à un printf("%d ",compteur1);
1105                     SHOW_LEFT(LLEFT, CLEFT, compteur1);
1106                     compteur1 = 0;
1107                     compteur0 = 0;
1108
1109                     //équivalent à un printf("\n");
1110                     LLEFT+=2;
1111                     CLEFT = 2;
1112                     JUMP_LEFT(LLEFT, CLEFT);
1113                 }
1114             }
1115         }
1116     }
1117 }
```

La fonction `FILL_TAB` est située entre les lignes 1066 et 1214. Cette fonction permet de compter les cases à colorier et remplir les lignes/colonnes par des nombres correspondants au nombre de cases à colorier à la suite dans la ligne/colonne correspondante situées en bordure de la grille `PICROSS`.

Voici le tableau TAB[lin][col] (qui est la solution d'une grille aléatoire)

[2] : Case à
supprimer (X)



2	2	2	1	2
1	2	2	2	1
1	1	2	2	1
2	1	1	2	2
1	1	1	1	1

[1] : Case à colorier

Le principe du comptage consiste à parcourir tout le tableau deux fois. Une fois en ligne grâce à deux boucles itératives en parcourant les colonnes puis une seconde fois grâce à deux boucles itératives en parcourant les lignes. Selon les valeurs rencontrées, il va afficher ou non le compteur.

- S'il rencontre un [1], **le compteur incrémente**, puis si le compteur est non nul et qu'il rencontre un [2], alors il affiche le compteur avant de le réinitialiser à 0 pour préparer son passage à la ligne/colonne suivante.

- S'il rencontre un [2], **un second compteur incrémente**.

- S'il rencontre un **[1] à la dernière case du tableau** et que le compteur est non nul, alors il affiche le compteur avant de le réinitialiser.

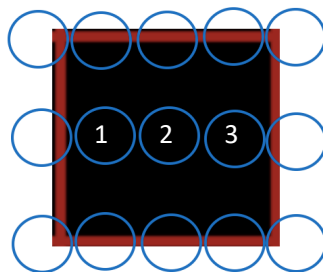
- S'il rencontre **uniquement des [2] sur la ligne/colonne**, alors cela voudrait dire que le second compteur est égal à la taille de la ligne/colonne. Si c'est le cas, il affiche 0 sur la grille.

- S'il rencontre **uniquement des [1] sur la ligne/colonne**, alors il affiche le compteur sur la grille avant de la réinitialiser à 0 pour préparer son passage à la ligne/colonne suivante.

Continuons avec la fonction `MID_CLICK`

```
1220 int MID_CLICK(int L, int C)
1221 {
1222     int Ctemp = C;
1223
1224     if(L%2==1 && C%2==1)
1225     {
1226         if(C<C+1 && (C+1)%4==2)
1227         {
1228             return Ctemp++;
1229         }
1230
1231         if(C>C-1 && (C-1)%4==2)
1232         {
1233             return Ctemp--;
1234         }
1235     }
1236
1237     else if(L%2==1 && C%2==0)
1238     {
1239         return Ctemp;
1240     }
1241     return Ctemp;
1242 }
```

Située entre les lignes 1220 et 1242, la fonction `MID_CLICK` permet de recentrer les click de la souris. En effet, nous avons vu dans les parties précédentes que notre case de jeu possédait 3 positions de click possibles.



Principe de la fonction :

Lorsque nous cliquons sur (1) à la position (L, C), C prend la valeur C+1 pour retomber sur (2) et renvoie la case du milieu **Ctemp**.

Lorsque nous cliquons sur (3) à la position (L, C), C prend la valeur C-1 pour retomber sur (2) et renvoie la case du milieu **Ctemp**.

Poursuivons avec la fonction `FORM_CONDITION`

```
1249 int FORM_CONDITION(int L, int Ctemp, int lin, int col, int FLAGTAB[lin][col])
1250 {
1251     if((L%2 == 1 && C%2 == 0 && C%4 != 0) || (L%2 == 1 && C%2 == 1)) //Click sur [ ][x][ ] ou Click sur [X][ ][X]
1252     {
1253         if((FLAGTAB[((L-Height)/2)][(Ctemp-Length)/4])%3 == 0)
1254         {
1255             FLAGTAB[((L-Height)/2)][(Ctemp-Length)/4] = 1;
1256             return 1;
1257         }
1258         else if((FLAGTAB[((L-Height)/2)][(Ctemp-Length)/4])%3 == 1)
1259         {
1260             FLAGTAB[((L-Height)/2)][(Ctemp-Length)/4] = 2;
1261             return 2;
1262         }
1263         else if((FLAGTAB[((L-Height)/2)][(Ctemp-Length)/4])%3 == 2)
1264         {
1265             FLAGTAB[((L-Height)/2)][(Ctemp-Length)/4] = 0;
1266             return 0;
1267         }
1268     }
1269     return 1;
1270 }
1271 }
1272 }
```

Cette fonction est située entre les lignes 1249 et 1272. Elle permet de définir un caractère à chaque click.

Principe de la fonction :

0	Caractère vide
1	ACS_CKBOARD (pour colorier)
2	X

FLAGTAB correspond au tableau du joueur qui est initialisé à 0 dans toutes les cases au début. Il se modifie au fur et à mesure des clicks.

Ctemp provient de la fonction **MID_CLICK** que nous avons vu précédemment qui permet d'obtenir la position C centrée sur une case.

Dans le tableau FLAGTAB, Si la case cliquée possède un 0, alors cette case prend la valeur 1. Si la case cliquée possède un 1, alors cette case prend la valeur 2. Si la case cliquée possède un 2, alors cette case prend la valeur 0.

Note : ici le modulo 3 (%3) est inutile car au lieu d'incrémenter une variable et de prendre son modulo 3 (ce que nous faisons auparavant), nous affectons directement les valeurs 0, 1 et 2 aux cases du tableau.

Continuons avec la fonction `DRAW_CLICK`

```
1278 void DRAW_CLICK(int L, int C, int temp)
1279 {
1280     if(L%2 == 1 && C%2 == 0 && C%4 != 0)
1281     {
1282         if(temp == 0) //Click sur [ ][x][ ]
1283         {
1284             mvaddch(L, C-1, ' ');
1285             mvaddch(L, C, ' ');
1286             mvaddch(L, C+1, ' ');
1287             refresh();
1288         }
1289
1290         else if(temp == 1)
1291         {
1292             system("play -q ./Music/V_click.mp3 vol -12db&");
1293             mvaddch(L, C-1, ACS_CKBOARD); //
1294             mvaddch(L, C+1, ACS_CKBOARD); //
1295             mvaddch(L, C, ACS_CKBOARD);
1296             refresh();
1297         }
1298
1299         else if(temp == 2)
1300         {
1301             system("play -q ./Music/X_click.mp3 vol -12db&");
1302             mvaddch(L, C-1, ' ');
1303             mvaddch(L, C, 'X');
1304             mvaddch(L, C+1, ' ');
1305             refresh();
1306         }
1307     }
1308
1309     if(L%2 == 1 && C%2 == 1) //Click sur [X][ ][X]
1310     {
1311
1312         if(C < C+1 && (C+1)%4 == 2) //Click sur [X][ ][ ]
1313         {
1314             if(temp == 0)
1315             {
1316                 mvaddch(L, C+1, ' ');
1317                 mvaddch(L, C, ' ');
1318                 mvaddch(L, C+2, ' ');
1319                 refresh();
1320             }
1321             else if(temp == 1)
1322             {
1323                 system("play -q ./Music/V_click.mp3 vol -12db&");
1324                 mvaddch(L, C+1, ACS_CKBOARD);
1325                 mvaddch(L, C, ACS_CKBOARD); //
```

Cette fonction est située entre les lignes 1278 et 1366. Elle permet d'afficher des caractères selon la valeur de **temp** qui est la valeur retournée par la fonction **FORM_CONDITION** que nous avons vue précédemment.

Principe de la fonction : Selon la valeur de **temp**, la fonction va afficher 3 caractères à la fois afin de compléter une case entière dans la grille de jeu.

Poursuivons avec la fonction `LINE_TEST` et `COL_TEST`

```
1438 void LINE_TEST(int lin, int col, int TAB[lin][col], int FLAGTAB[lin][col])
1439 {
1440     int i,j;
1441     int temp=0;
1442
1443     for(i=0; i<lin ; i++)
1444     {
1445         for(j=0; j<col ; j++)
1446         {
1447             if(TAB[i][j]==FLAGTAB[i][j])
1448             {
1449                 temp++;
1450             }
1451
1452             if(temp == col)
1453             {
1454                 if(Height+1+i*2 < Height + lin*2)
1455                 {
1456                     move(Height+1+i*2, Length+col*4+2);
1457                     attron(COLOR_PAIR(3));
1458                     addch('V');
1459                     attroff(COLOR_PAIR(3));
1460                 }
1461             }
1462
1463             else
1464             {
1465                 if(Height+1+i*2 < Height + lin*2)
1466                 {
1467                     move(Height+1+i*2, Length+col*4+2);
1468                     addch(' ');
1469                 }
1470             }
1471         }
1472     }
1473     temp=0;
1474 }
1475 temp = 0;
1476 }
```

La fonction est située entre les lignes 1438 et 1476. Elle permet de vérifier si une ligne remplie par le joueur dans la grille de jeu a bien été complétée.

Principe de la fonction : Nous parcourons tous les deux tableaux entièrement ligne par ligne grâce à deux boucles itératives l'une dans l'autre pour comparer la grille de jeu **TAB[i][j]** et la grille du joueur **FLAGTAB[i][j]**.

Lorsque les deux tableaux possèdent un nombre qui est identique dans la case correspondante, un compteur initialisé à 0 est incrémenté.

Si le compteur est égal à la taille maximale de la colonne-1, alors on affiche le "V".

```

1482 void COL_TEST(int lin, int col, int TAB[lin][col], int FLAGTAB[lin][col])
1483 {
1484     int i,j;
1485     int temp=0;
1486
1487     for(j=0; j<col ; j++)
1488     {
1489         for(i=0; i<lin ; i++)
1490         {
1491             if(TAB[i][j]==FLAGTAB[i][j])
1492             {
1493                 temp++;
1494             }
1495
1496             if(temp == lin)
1497             {
1498                 if(Length+2+j*4 < Length + col*4)
1499                 {
1500                     move(Height+lin*2+1 , Length+2+j*4);
1501                     attron(COLOR_PAIR(3));
1502                     addch('V');
1503                     attroff(COLOR_PAIR(3));
1504                 }
1505             }
1506
1507             else
1508             {
1509                 if(Length+2+j*4 < Length + col*4)
1510                 {
1511                     move(Height+lin*2+1 , Length+2+j*4);
1512                     addch(' ');
1513                 }
1514             }
1515         }
1516         temp = 0;
1517     }
1518     temp = 0;
1519 }
1520 }

```

Située entre les lignes 1482 et 1520, COL_TEST fonctionne de la même manière que **LINE_TEST** excepté le fait que nous parcourons les deux tableaux entièrement colonne par colonne.

De même, lorsque le compteur est égal à la taille maximale de ligne-1, alors nous affichons "V".

Poursuivons avec la fonction `END_GAME`

```
1526 int END_GAME(int lin, int col, int TAB[lin][col], int FLAGTAB[lin][col])
1527 {
1528     int i,j;
1529     int win = 1;
1530
1531     for(i=0; i<lin ; i++)
1532     {
1533         for(j=0; j<col ; j++)
1534         {
1535             if(FLAGTAB[i][j] != TAB[i][j])
1536             {
1537                 win = 0;
1538             }
1539         }
1540     }
1541     return win;
1542 }
```

Située entre les lignes 1526 et 1542, `END_GAME` permet de déterminer le moment où le joueur a gagné.

Principe de la fonction : Nous initialisons une variable **win** à 1. Nous parcourons la grille de jeu **TAB[i][j]** ainsi que la grille du joueur **FLAGTAB[i][j]** entièrement. Tant que les deux tableaux sont différents, **win** prend la valeur de 0. Si les deux tableaux sont identiques, **win** initialisé à la valeur de 1 est renvoyé à une fonction qui gère un mode de jeu.

Par exemple dans le mode aléatoire :

```
1837     if(END_GAME(lin, col, TAB, FLAGTAB) == 1)
1838     {
1839         win = 1;
1840         AFFICHAGE_CONGRATULATION(Congratulation, MaxY, MaxX);
1841         getch();
1842         EXIT_MENU(lin, col, TAB, FLAGTAB, MaxY, MaxX, 1, dev); //pour le cas "Replay"
1843     }
```

Lorsque la valeur retournée est égale à 1, nous affichons "CONGRATULATION" et attendons un click de la part du joueur avant d'afficher un menu lui demandant s'il souhaite rejouer.

2.5. Le design

Afin d'obtenir un meilleur rendu du jeu, nous avons décidé de retoucher le jeu PICROSS.

[Welcome] et [Congratulation ! You Won !]



L'affichage de **"WELCOME IN THE PICROSS GAME"** et **"CONGRATULATION ! YOU WON !"** ont été conçu en utilisant des **fichiers.txt**.

```

4832 void REPLISSAGE_CONGRATULATION()
4833 {
4834     int i,j;
4835     FILE * congratulation = NULL;
4836     congratulation = fopen("./Picture/Congratulation.txt", "r");
4837
4838     for(i=0; i<22; i++)
4839     {
4840         for(j=0; j<135; j++)
4841         {
4842             Congratulation[i][j]= fgetc(congratulation);
4843         }
4844         fgetc(congratulation);
4845     }
4846
4847     fclose(congratulation);
4848
4849     if(congratulation == NULL)
4850     {
4851         fprintf(stderr, "Erreur lors de l'ouverture du fichier : %s ", "./Picture/Congratulation.txt");
4852         endwin();
4853         exit(EXIT_FAILURE);
4854     }
4855     rewind(congratulation);
4856 }
4857
4858 /*
4859
4860 void AFFICHAGE_CONGRATULATION(char TAB[22][135], int MaxY, int MaxX)
4861 {
4862     system("play -q ./Music/Win.mp3 vol -6db&");
4863     system("pkill -n");
4864
4865     attron(COLOR_PAIR(5));
4866     int i,j;
4867     system("clear");
4868     move(4,0);
4869
4870     for(i=0; i<22; i++)
4871     {
4872         for(j=0; j<135; j++)
4873         {
4874             printw("%c", TAB[i][j]);
4875         }
4876     }
4877     attroff(COLOR_PAIR(5));
4878
4879     mvprintw(MaxY-2,MaxX/2+10, "Press a Key to continue ...\n");
4880 }
4881

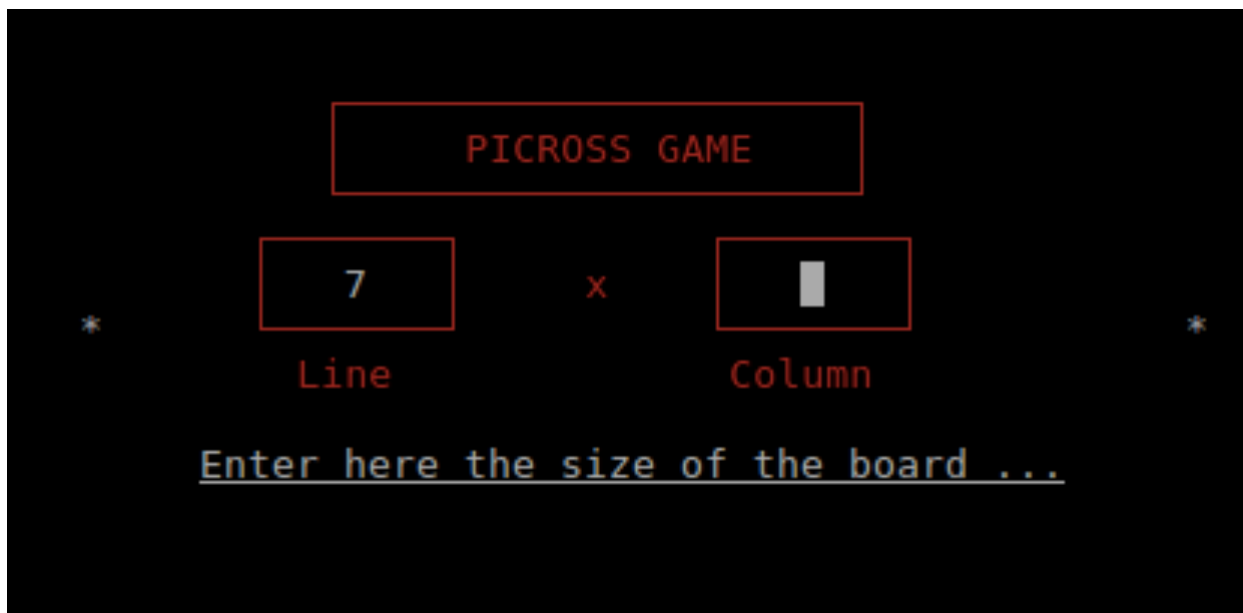
```

Les quatre procédures permettant nos deux affichages sont situées entre les lignes 4786 et 4881.

REPLISSAGE_CONGRATULATION est une procédure permettant de lire, de récupérer les caractères du fichier le fichier Congratulation.txt situé dans la dossier Picture et de les stocker dans un tableau de char.

AFFICHAGE_CONGRATULATION est une procédure permettant d'afficher le tableau de char.

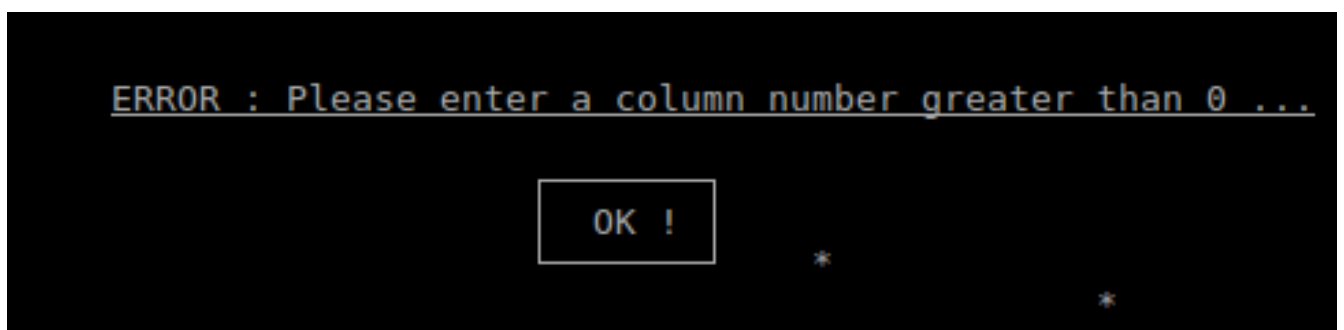
L'affichage d'initialisation des valeurs demandées lors du mode aléatoire



Nous avons créé un affichage permettant à l'utilisateur d'entrer les valeurs pour la ligne et la colonne lors de l'initialisation de la grille PICROSS.

Nous n'avons pas délimité la taille maximale car il se pourrait qu'un joueur possède un très grand écran pour pouvoir jouer sur une très grande grille de jeu PICROSS.

Nous avons eu de nombreuses difficultés lors de la création. Après avoir entré la valeur de la ligne, nous devons entrer la valeur de la colonne. Cependant, si nous entrons une valeur inférieure à 0 ou encore un caractère, la gestion d'erreur se fait.



Cependant lorsque nous affichons le message d'erreur, l'affichage précédent est supprimé. Nous nous sommes donc retrouvés avec un affichage vide alors que nous avions entré le nombre 7 dans la ligne.

De plus nous devons entrer une nouvelle fois le nombre de la ligne, ce qui nous donnait une erreur de segmentation.

Solution au problème :

```
2195 void RANDOM_SIZE_CALL(int *nl, int *nc, int MaxY, int MaxX)
2196 {
2197     RANDOM_SIZE_AFFICHAGE(MaxY, MaxX);
2198     RANDOM_SIZE_LINE(nl, MaxY, MaxX);
2199     RANDOM_SIZE_COL(nl, nc, MaxY, MaxX);
2200     noecho();
2201     curs_set(FALSE);
2202     refresh();
2203 }
```

Nous réaffichons la ligne (nl) qui a été inscrite dans la case ligne lors de la demande **RANDOM_SIZE_COL**.

```
2145 void RANDOM_SIZE_LINE(int *nl, int MaxY, int MaxX)
2146 {
2147     RANDOM_SIZE_AFFICHAGE(MaxY, MaxX);
2148     echo();
2149     curs_set(TRUE);
2150
2151     while(mvscanw(MaxY/8+4, MaxX/2-10, "%d", nl)!=1 || *nl <= 0)
2152     {
2153         clear();
2154         noecho();
2155         curs_set(FALSE);
2156         attron(A_UNDERLINE);
2157         mvprintw(MaxY/8+7,MaxX/2-23,"ERROR : Please enter a line number greater than 0 ..." );
2158         attroff(A_UNDERLINE);
2159         RANDOM_SIZE_OK(MaxY, MaxX);
2160
2161         RANDOM_SIZE_AFFICHAGE(MaxY, MaxX);
2162         echo();
2163         curs_set(TRUE);
2164     }
2165 }
2166
2167
2168 /*
2169
2170 void RANDOM_SIZE_COL(int *nl, int *nc, int MaxY, int MaxX)
2171 {
2172     echo();
2173     curs_set(TRUE);
2174
2175     while(mvscanw(MaxY/8+4, MaxX/2+9, "%d", nc)!=1 || *nc <= 0)
2176     {
2177         clear();
2178         noecho();
2179         curs_set(FALSE);
2180         attron(A_UNDERLINE);
2181         mvprintw(MaxY/8+7,MaxX/2-23,"ERROR : Please enter a column number greater than 0 ..." );
2182         attroff(A_UNDERLINE);
2183         RANDOM_SIZE_OK(MaxY, MaxX);
2184
2185         RANDOM_SIZE_AFFICHAGE(MaxY, MaxX);
2186         echo();
2187         mvprintw(MaxY/8+4, MaxX/2-10, "%d", *nl);
2188         curs_set(TRUE);
2189     }
2190 }
```

RANDOM_SIZE est situé entre les lignes 1873 et 2203.

La neige et les étoiles



Afin d'obtenir quelques animations sur notre jeu PICROSS, nous avons créé une neige qui sont des caractères d'étoiles placés au hasard sur tout le menu.

La neige est située dans l'affichage de fin de jeu lorsque l'on quitte le jeu. Cette animation et chutent vers le bas.

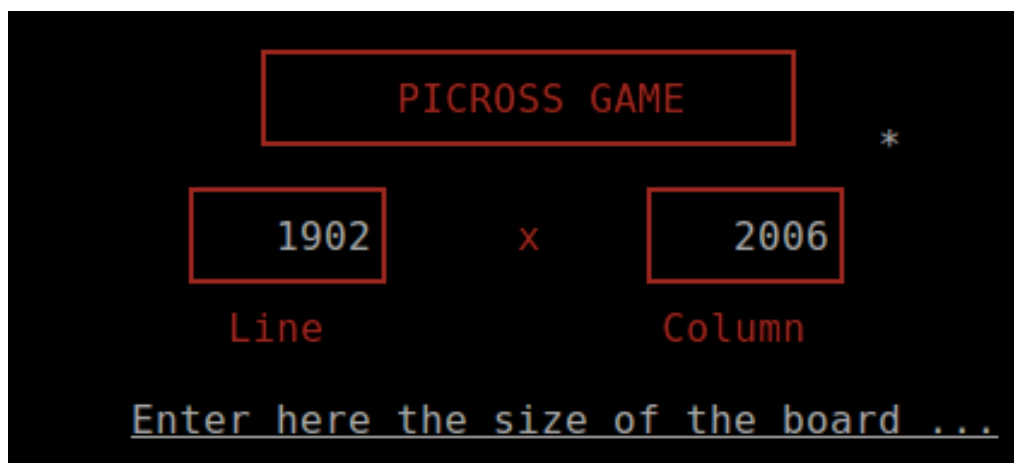
2.6. Les fonctionnalités cachées du jeu

Notre jeu Picross est désormais opérationnel !

Afin d'optimiser l'ambiance secrète du jeu, nous avons créé un mode de jeu caché appelé "**SECRET MODE**".

Il s'agit enfaite du mode de jeu image, mais il n'est accessible qu'à partir du **RANDOM_MENU**.

Pour pouvoir y jouer : Rien de plus simple, il suffit de consulter les Crédits et de résoudre la petite énigme. Enfin, il faut entrer le code secret dans le RANDOM_MODE pour y accéder.

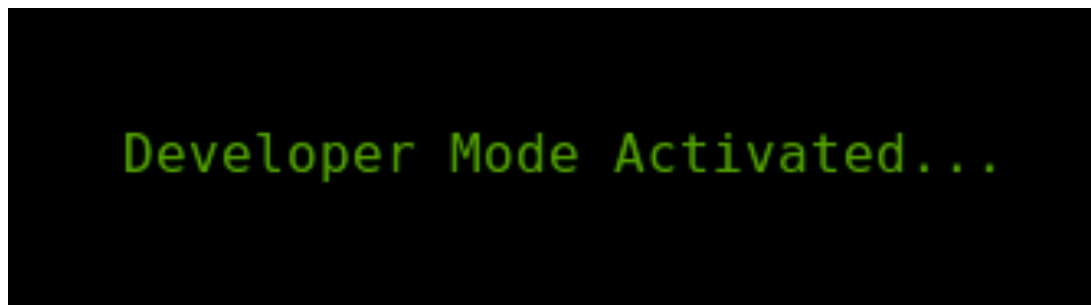
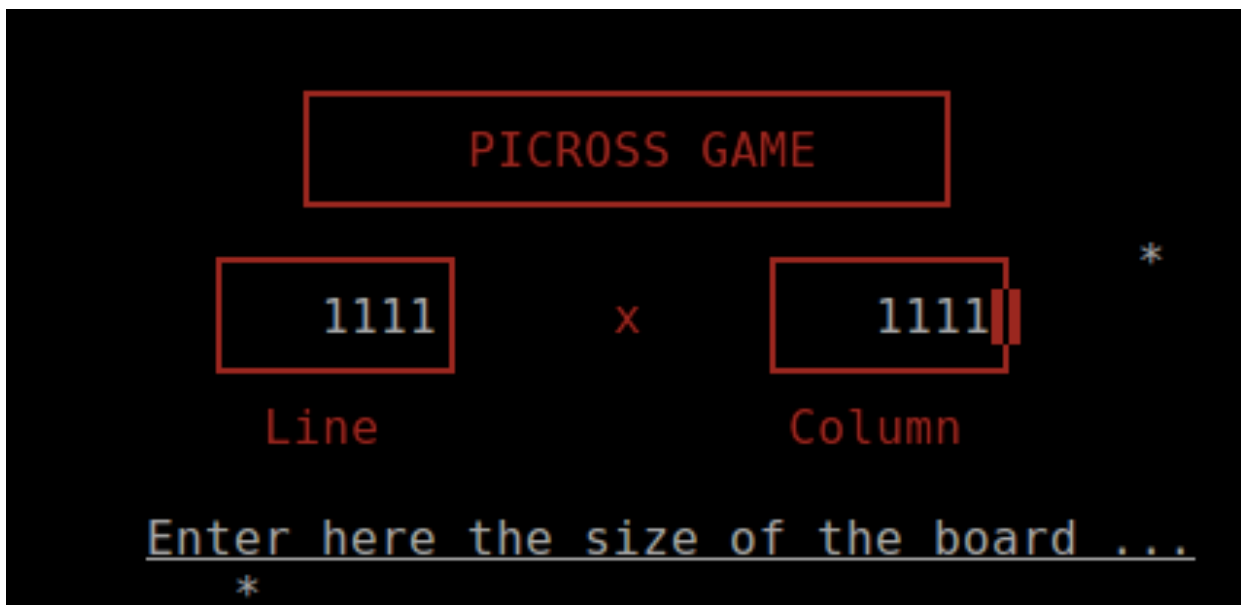


Nous entrons par la suite, dans le SECRET_MODE ...

Le mode Développeur

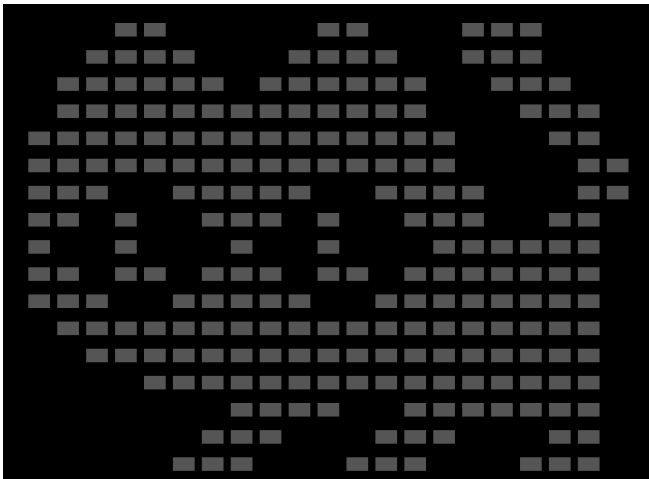
Afin de faciliter notre présentation le jour de la soutenance, nous avons créé le mode Développeur qui nous permet d'obtenir l'affichage de toutes les images créées dans leurs modes respectifs.

Afin d'activer le mode Développeur : Il suffit d'entrer le code [1111 - 1111]



Maintenant que le mode développeur est activé... allons voir les changements en jeu !

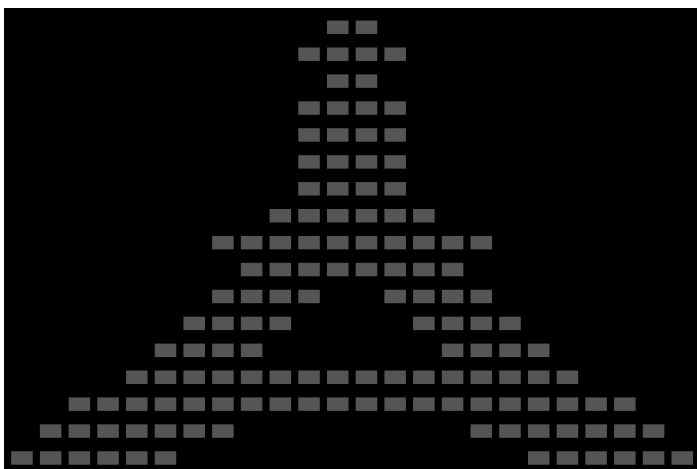
Dans le PICTURES_MODE ...



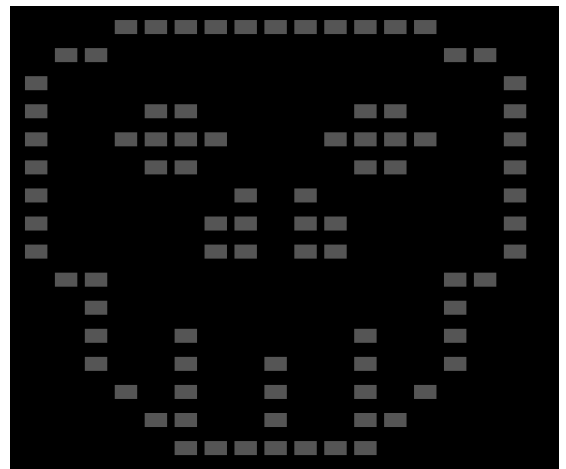
Le Chat



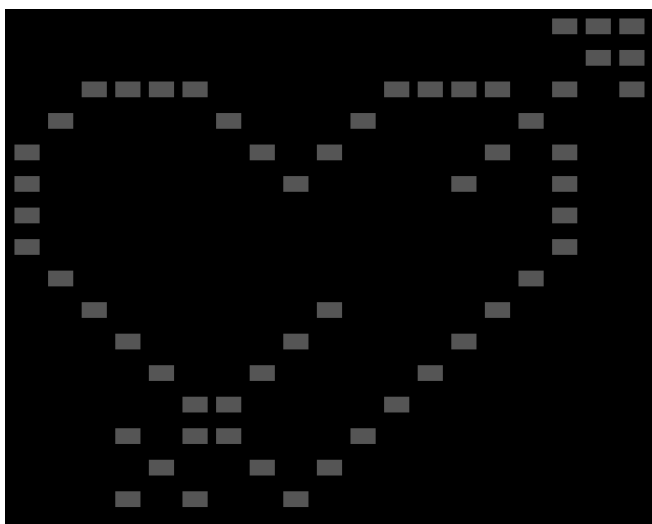
La baleine



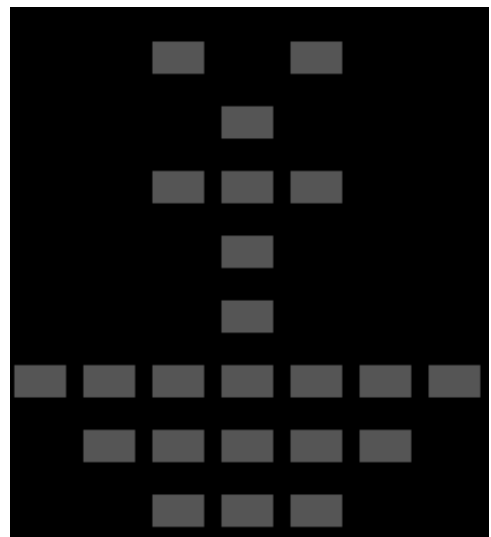
La Tour Eiffel



Le Crâne

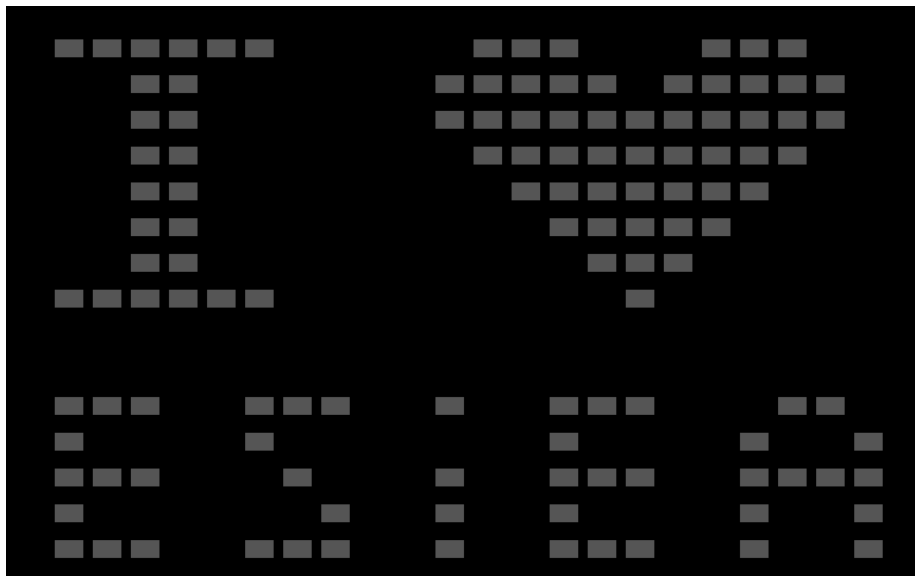
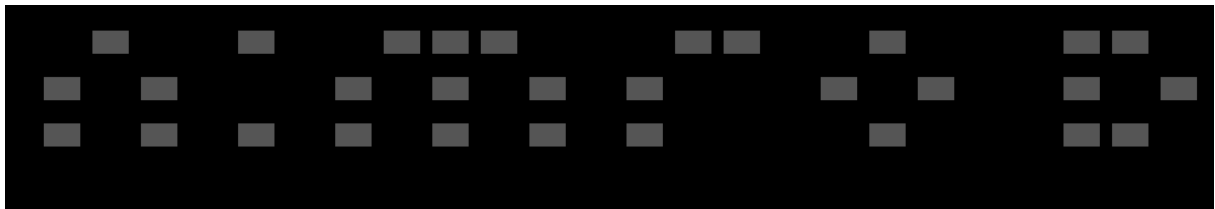
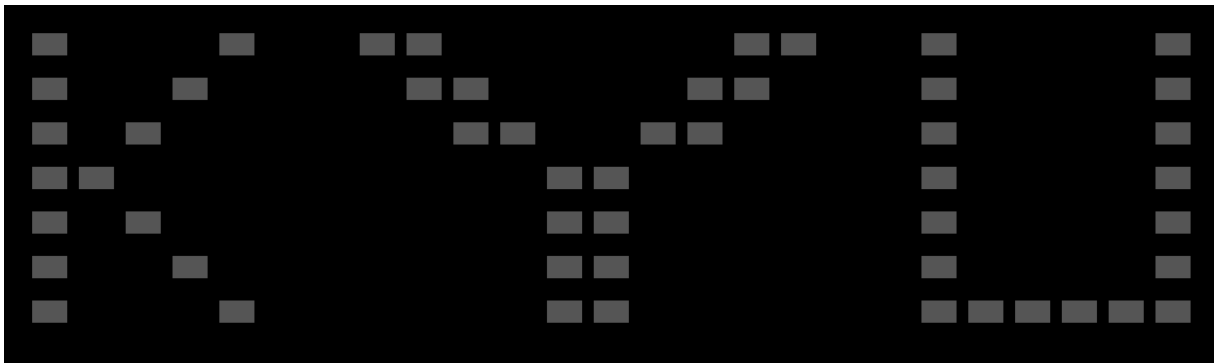


Le Cœur

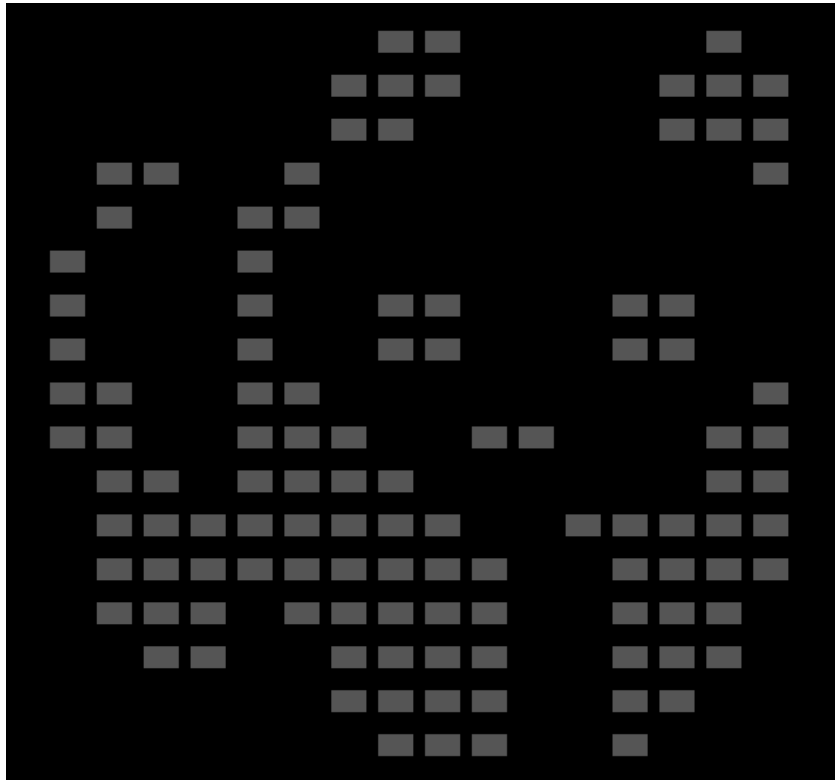


La Plante

Dans LETTERS_MODE ...

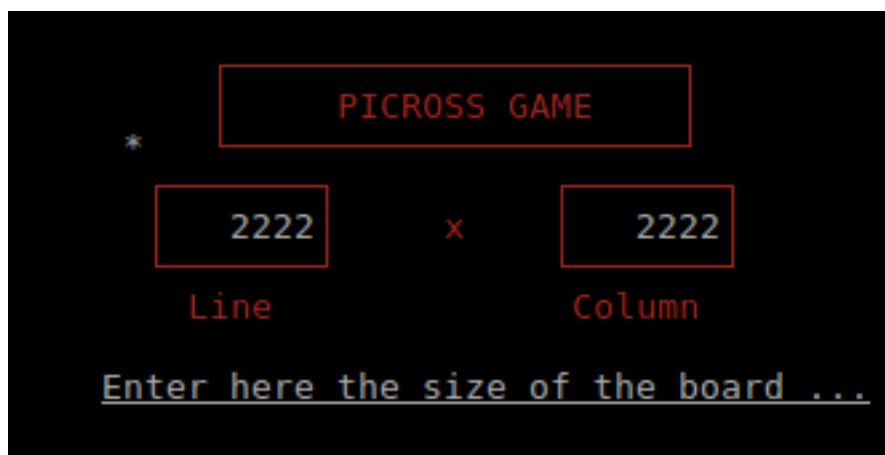
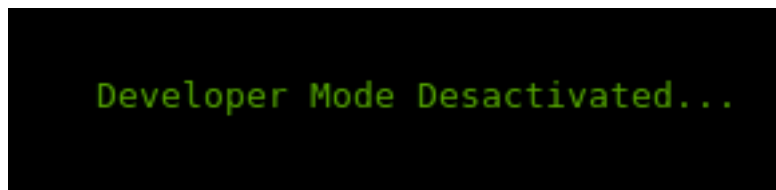


Dans SECRET_MODE ...



Et oui ! c'est un Panda ...

Afin de désactiver le mode Développeur : Il suffit d'entrer le code [2222 - 2222]



3. Crédits du jeu

Nous pouvons accéder aux crédits du jeu à partir du menu dans le jeu PICROSS.

Les crédits contiennent l'énigme à résoudre pour le mode secret.

Projet 1A [ESIEA 2017-2018]

Créé par :

Thierry KHAMPHOUSONE & Nimrod NDOUDI.

Suivi par :

Mme RAQUEL MARTINS et M. GONZALEZ.

Sons et musiques :

Yiruma - River Flow in you (Menu theme) by Thierry KHAMPHOUSONE

Tokyo Ghoul - White Silence (Opening/Ending theme) by Thierry
KHAMPHOUSONE

Hannes Hofkind - Game Over (Credits theme)

Music For Your Media - A Hard Decision

Funguypiano - Stay With Me

Funguypiano - I believe

Funguypiano - Beautiful

Soundclick effect (Menu/Fill/Cross) by Thierry KHAMPHOUSONE

Technologies utilisées :

C language

Ncurses library

P-255 Piano

3. Conclusion

La création du jeu PICROSS nous a permis de mieux consolider nos connaissances et bases dans le langage de programmation C. Nous avons su prendre connaissance et utilisé la bibliothèque NCURSES tout au long de notre projet C.

Nous avons amélioré notre capacité à travailler en équipe en nous répartissant les tâches et en s'entraîdant.

Nous avons rencontré de nombreuses difficultés. (Expliqué dans les précédents paragraphes)

Notre jeu fonctionne totalement, et possède aucun problème.

Une répartition des tâches au sein du binôme a été de fait.

Nimrod : Je me suis occupé de la création du menu du jeu ainsi que la création des images pour le mode image.

Thierry : Je me suis occupé de la création du mode de jeu aléatoire, du mode de jeu secret. J'ai également ajouté de nombreux designs au jeu, y compris pour certaines musiques jouées au piano P-255 YAMAHA.