Rapport de Projet sur le jeu Yahtzee

Participants : Benoit COMBASTEIX, Simon FERNANDES
et Nathan OUALET

Dépot Github sur le Projet :
https://github.com/FernandesSimon/Projet_Etudiant_C_2016

Encadrants : Dominique Py, Claudine Piau-Toffolon
et Thierry Lemeunier.

Table des matières :

- **Introduction**

1. Présentation du jeu
2. Déroulement d'une partie

- **Organisation du projet**

1. Détails des actions faites par chacun.
2. Analyse des différentes des fonctions.

- **Conception**

1. Explication détaillée des règles
2. Environnement de travail.

- **Développement**
- **Résultats**
- **Conclusion**
- **Annexes**

• Introduction :

1. Présentation du jeu : le Yahtzee.

Le Yahtzee (aussi nommé le Yam's) est un jeu de dés inventé en 1954 au Canada. Il fut inventé par un couple de riches canadiens qui s'amusait à faire jouer leur amis sur leur Yacht, d'où le « Yacht Game ». Le jeu plut tellement aux amis du couple, qu'il décida de contacter le fabriquant de jeux Edwin S. Lowe afin qu'il crée des versions du jeu. Lowe eut un coup de cœur et commercialisa le jeu de dés.

En 1973, Milton Bradley – firme de Hasbro – achète la E. S. Lowe Company et obtient les droits du jeu.

2. Déroulement d'une partie :

Le Yam's se joue à un ou plusieurs joueurs. Jouer en solitaire signifie que le seul but est d'améliorer son score au fil des différentes parties.

Le jeu se joue avec 5 dés et un tableau de score.
Au début de la partie, chaque joueur lance une fois les dés. Celui qui possède plus grand score commence la partie.
A chaque tour, les joueurs lancent les dés au maximum 3 fois. La première fois, tous les dés doivent obligatoirement être lancés.

Si le joueur décide de lancer les dés une seconde (et troisième fois), il peut relancer les dés de son choix, en sélectionnant les dés qu'il souhaite garder.
Après le troisième lancer, les dés ne peuvent plus être lancés et les points obtenus sont validés. C'est à ce moment que le score doit être inscrit dans l'une des 13 cases, selon les combinaisons obtenues dans le dernier lancer de dés effectué. Selon le choix de catégorie le score sera différent mais attention vous pouvez choisir une catégorie sans avoir la combinaison mais vous obtenez un score de 0 il faut donc être stratégique dans vos choix.
Une case doit obligatoirement être remplie à la fin du tour.

Pour gagner une partie a plusieurs joueurs il faut obtenir le meilleur score parmi tout les joueurs. Jouer en solitaire permet de perfectionner ses techniques et de chercher a avoir un meilleur score (il sert donc plus d'entraînement que de réel partie).

• Organisation du travail

1. Détails des actions faites par chacun :

Benoit	Affichage des dés, création de fonction de calcul et de fonction booléenne, tests des fonction et assemblage.
Nathan	Affichage des dés, création de fonction de calcul et de fonction booléenne, commentaire Doxygen.
Simon	Affichage des dés, création de fonction de calcul et de fonction booléenne, interface graphique Ncurses.

Le partage des fichiers du projet a été fais sur un répertoire Github, la communication entre chaque membres du projet s'est faite par message, et logiciel de communication comme curse et skype.

2. Fonctions principales :

Fonction de calcul:

```
void CalculScore(int Joueur, int Categorie)

/* Calcule le Score de la catégorie choisie et vérifie si la combinaison est présente dans les dés obtenue*/

void CalculOccurrences()

/*parcourt les 5 dés lancés, on incremente le nombre de valeurs trouvés dans le tableau d'occurrences correspondant à la valeur du dé */
```

Fonction booléenne :

```
int isFull()

/* Vérifie si il y a un full dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */

int isCarre()

/* Vérifie si il y a un carré dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */
```

```
int isYahtzee()

/* Vérifie si il y a un yahtzee dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */

 int isBrelan()

/* Vérifie si il y a un brelan dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */

int isPtSuite()

/* Vérifie si il y a une petite suite dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */
int isGdSuite()

/* Vérifie si il y a une grande suite dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */
```

Fonction d'affichage :

```
void EcranTitre()

/* Affiche le titre du jeu */

void FicheDeScore(WINDOW *Fenetre, int y, int x)

/* Affiche le squelette de la fiche de score */

void de_un, de_deux…, de_six, de_vide()

/* Dessine un dés de la valeur obtenue duant le lancer */

void AffichageDe(int De[5], WINDOW *Fenetre)

/* Permet d'afficher les dés obtenue */

void MiseEnPlace()

*/ Créer les differentes zones nécessaires à la partie */

int Resultat(int * isRelancer)

/* Detruis les précedentes zones pour n'afficher que ZoneResultat */
/* Ajoute les différentes primes aux scores des joueurs */
/* Compare les différents scores pour chercher le score le plus élevé */
/* Affiche les scores des joueurs, leuint isFull()
```

/* Vérifie si il y a un full dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */

int isCarre()

/* Vérifie si il y a un carré dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */

int isYahtzee()
/* Vérifie si il y a un yahtzee dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */

int isBrelan()

/* Vérifie si il y a un brelan dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */

int isPtSuite()

/* Vérifie si il y a une petite suite dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */
int isGdSuite()

/* Vérifie si il y a une grande suite dans les dés obtenue */
/* Renvoie vrai si oui faux sinon (booléen) */
void FicheDeScore(WINDOW *Fenetre, int y, int x)

/* Affiche le squelette de la fiche de score */

de_un, de_deux…, de_six, de_vide :

/* Dessine un dés de la valeur obtenue duant le lancer */

void AffichageDe(int De[5], WINDOW *Fenetre)

/* Permet d'afficher les dés obtenue */

void MiseEnPlace()

*/ Créer les differentes zones nécessaires à la partie */

int Resultat(int * isRelancer)

/* Detruis les précedentes zones pour n'afficher que ZoneResultat */
/* Ajoute les différentes primes aux scores des joueurs */
/* Compare les différents sc
int Partie()

/* Appelle les fonctions MiseEnPlace , Aide, Lancer, AffichageDe, Garder, EcrireScore et ChangerJoueur pour effectuer les actions nécessaire au déroulement du jeu de yahtzee */

void Page_Regles(WINDOW *Fenetre, int y, int x)

/* Affiche les règles dans la zone indiquée au coordonnées indiquée */

int Regles()

/* Créer deux nouvelles Zones ZoneRegles et ZoneMessage */
/* Affiche les règles dans la ZoneRegles que l'utilisateur fais defiler grâce aux touches directionnne */

void EcranTitre()

/* Affiche l'ecran titre jusqu'à se qu'une touche soit rentrée */

void Menu()

/* Affiche le menu dans la ZoneMenu et demande a l'utilisateur de faire un choix grâce aux touches directionnelles*/
/* Met à jour l'affichage pour ne laisser que la zones utile celon le choix de l'utilisateur (supprime donc ZoneMenu) */

void Nettoyer(WINDOW * Fenetre, int DebutY, int DebutX, int FinY, int FinX)

/* Vide l'interieur d'une fenêtre à partir et jusqu'aux coordonées indiquée */

void Garder(WINDOW *Fenetre, int Garde[5])

/* Demande à l'utilisateur quel dés il ne veut pas relancer */

void Lancer()

/* Effectue une donne aléatoire de 5 dés de six */
/* Permet de sauvegarder et quitter le jeu */

void DetruireFenetre(WINDOW * Fenetre)

/* Supprime la fenêtre indiquée */

WINDOW *CreerFenetre(int height, int width, int starty, int startx)

/* Créer la fênetre indiquée dans les dimensions voulue à l'endroie voulue */

```
int Initialisation()

/* Initialise les tableaux de valeurs et les variables Joueur NbTours à 0  et NbLancer à 1 */


void FinDePartie()

/* Affiche un message de remerciement et quitte le programme */

void ChargementAffichage()


/* Met à jour l'affichage du tableau des scores */
/* Ne fait rien si la partie n'a pas été chargée à partir d'un fichier */
```

Fonction de gestion:

```
int Charger()

/* Affiche les sauvegarde disponible */
/* Charge la partie séléctionnée */

void Sauvegarder()

/* Sauvegarde la partie en écrivant les differents scores dans un fichier sav_ */

void ChoixCategorie (int *Categorie, WINDOW *Fenetre, int Joueur)

/* Demande à l'utilisateur de choisir une catégorie qui n'a pas encore été sélectionnée */

void EntrerNbJoueur()

/* Demande a l'utilisateur de choisir le nombre de joueur qu'il valide avec la touche entrée */

void EntrerPseudo()

/* Demande un pseudo qui respecte les conditions à l'utilisateur */

void ChainePseudo (char Pseudos[40])

/* Concatene les caractères les caractères avec des espaces pour les afficher */

void EcrireScore(int Joueur)

/* Rentre le score obtenue dans la catégorie */
```

```
int ChangerJoueur(int Joueur, int NbJoueurs)

/* Renvoie plus un à la variables Joueur */
/* Retourne la variable global Joueur à zero si Joueur est égal aux nombre de joueurs */

void Aide(int isAide[4], int Joueur)

/* Affiche l'aide si l'aide à été acceptée */
/* Met à jour l'affichage de l'aide en indiquant les combinaison disponibles*/

void DemandeAide()

/* Demande à l'utilisateur s'il veut afficher l'aide */

void ActiverAide(int i)

/* Permet de désactiver ou d'activer l'aide pendant la partie */

void Tri(int iTab[5], char cTab[5][10])

/* Trie un tableau d'entier dans l'ordre décroissant et effectue les même permutation sur le tableau
de caratères correspondant */

int LectureHS(FILE * HighScore, char PseudoHS[5][10], int HS[5])

/* Ouvre le fichier des Highscores et récupère leur valeur */

int VerifHS(int Total[4])

/* Vérifie si les résultats de fin de partie peuvent être entrés comme highscores */

int AffichageHS()

/* Créer une nouvel fenêtre et affiche les highscore à l'interieur */

void EcritureHS(char PseudoJ[4][10], int Total[4])

/* Modifie le fichier avec les nouveaux highscores */
```

• Conception

1. Explication détaillée des règles :

La feuille de Yahtzee est divisée en deux parties :

- La partie supérieure :

de l'As au Six :

Vous multipliez le nombre d'occurrences d'une valeur par cette valeur et l'écrivez dans la case correspondante. C'est à dire que si vous obtenez 5 Deux, vous inscrivez 10 dans la case.

- La partie inférieure :

Brelan : Vous obtenez trois fois le même chiffre et deux autres chiffres. Vous additionnez le total des dés et l'écrivez dans la case « Brelan »

Carré : Vous obtenez quatre fois le même chiffre et un autre. Vous additionnez le total des dés et l'écrivez dans la case « Carré ».

Full : Vous obtenez trois dés identiques et 2 dés identiques entre eux. Vous additionnez le total des dés et l'écrivez dans la case « Full », vous gagnez 25 points bonus.

Petite suite : Vous obtenez une suite de quatre chiffre. Vous gagnez 30 points.

Grande suite : Vous obtenez une suite de cinq chiffres. Vous gagnez 40 points.

Yahtzee : Vous obtenez 5 dés identiques. Vous marquez 50 points.

Chance : Vous additionnez la valeur de chaque dé et l'écrivez dans la case.

Attention ! Plusieurs résultats peuvent correspondre à plusieurs catégories mais vous ne pouvez remplir une case qu'une seule fois, soyez stratégiques !

Prime des 35 points: Lorsque vous totalisez un score de 63 points juste avec la partie supérieure, vous obtenez une prime de 35 points bonus.

Prime des Yahtzee : Lorsque vous avez obtenu un yahtzee dans votre partie et qu'il a été inscrit dans la case yahtzee vous obtenez un bonus de 100 points pour chaque yahtzee suivant.

2. Environnement de travail :

Les parties du jeu se feront sur une interface graphique en console.

Le déroulement se produit sur plusieurs pages de console :

- 1ere page : le titre du jeu ;

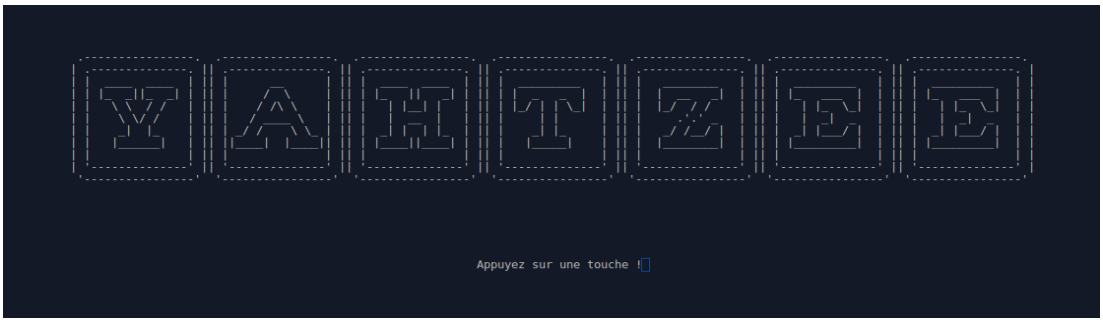


Illustration 1: Titre du jeu.

- 2 eme page : Menu principal du jeu : Nouvelle partie, Regles, Meilleur Score et Quitter le Jeu.

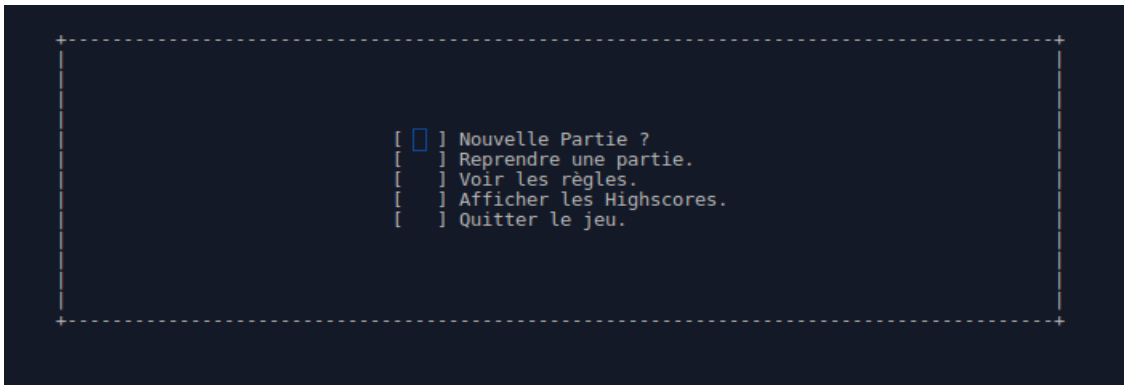


Illustration 2: Menu principal du jeu

- 2eme page bis : Affichage déroulant des règles.

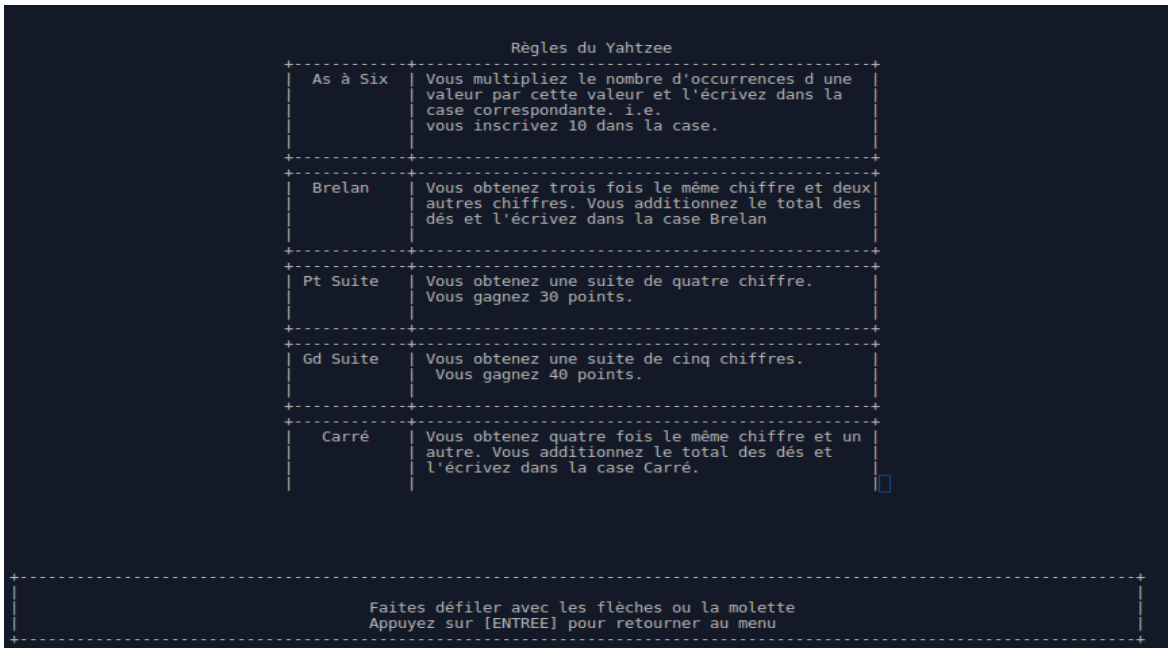
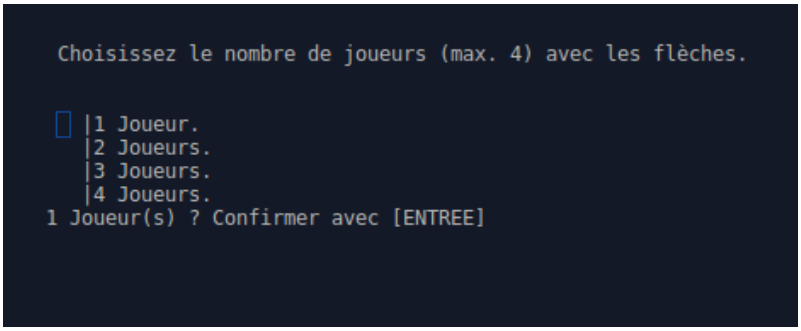
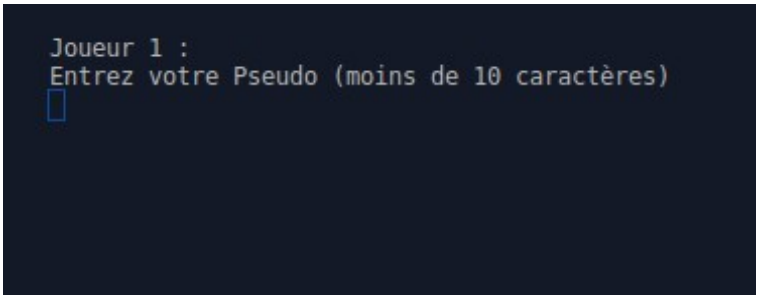


Illustration 3: Règles du jeu

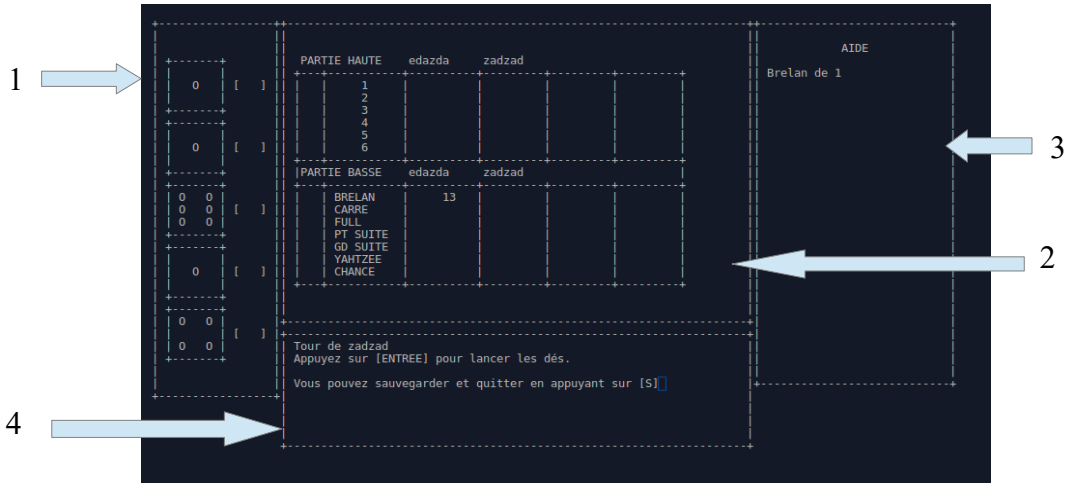
- 2eme page trio : Affichage du meilleur score réalisé sur le jeu.
- 3eme page : Nouvelle partie : -Combien de joueurs ? Les pseudos des joueurs ?





-

- 4eme page : Après sélection des pseudos, la partie commence… :



- 1) Zone d'affichage des 5 dés, où se fait la sélection des dés à garder .
- 2) Zone de Score : avec l'affichage du score selon les catégories et les différents joueurs.
- 3) Zone Aide : Affichage des différentes combinaisons des lancers si le joueur a demandé l'aide en début de partie.
- 4) Zone Message : qui permet d'afficher les instructions pour le bon déroulement de la partie.

- 5eme page : Après qu'un joueur ait rempli sa partie de tableau, la page de score s'affiche automatiquement, demandant si on veut recommencer une partie ou retourner au menu.

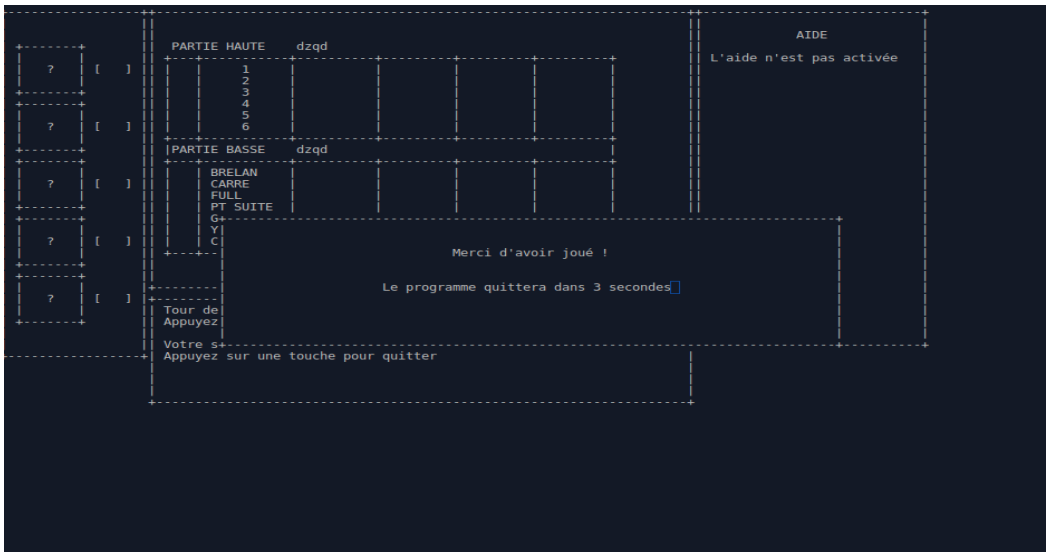


Illustration 4: Message de fin de jeu

- 6eme page : A partir du menu l'utilisateur peut afficher la page des highscore affichant les meilleurs score effectuer par les joueurs.

Illustration 5: affichage des highscores

- 7eme page : On peut accéder à la page des sauvegarde grâce a la page du menu, cette page affiche les différentes sauvegardes disponibles afin de reprendre les parties laissés en cours.

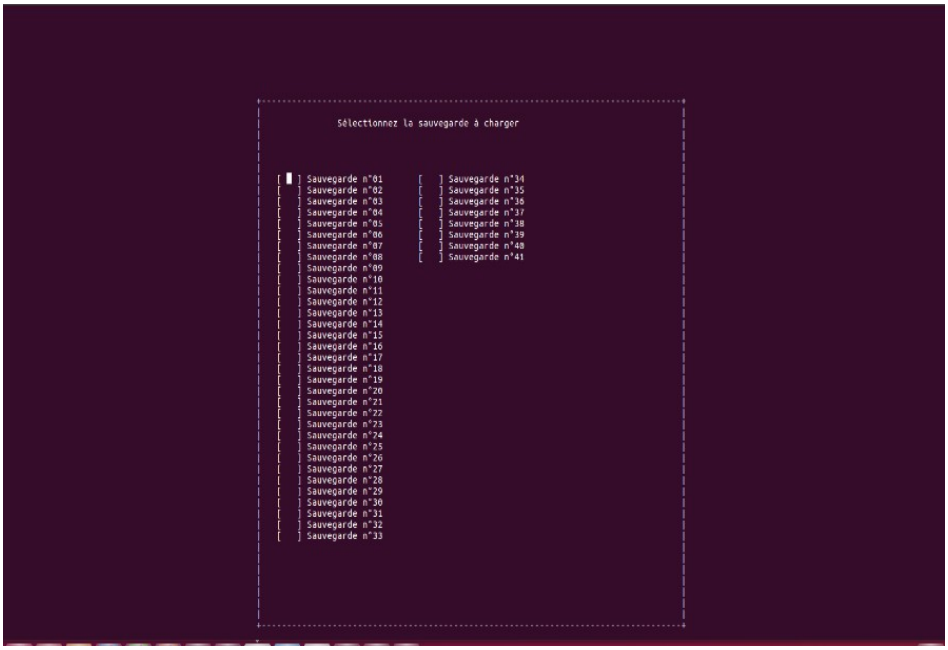


Illustration 6: Affichage de la page de sauvegarde

Les commandes demandées a chaque instruction s'exécute grâce à la bibliothèque Ncurses ou a une demande d'entrée d'un caractère spécial caractérisant la plupart du temps la réponse « oui » ou « non » ou alors de monter ou de descendre dans l'affichage en effectuant la commande « key_down »(flèche du bas) ou « key_up »(flèche du haut).

Ncurses :

La bibliothèque Ncurses permet de se déplacer dans l'interface graphique de la console, par exemple si on affiche un tableau dans la console grâce à diverses commandes on pourra se déplacer dans ce tableau et écrire à l'intérieur.
Cette bibliothèque permet aussi d'afficher des objets à l'endroit ou on veut c'est à dire qu'on choisis la colonne et la ligne ou l'objet apparaîtra.

L'intelligence artificielle:

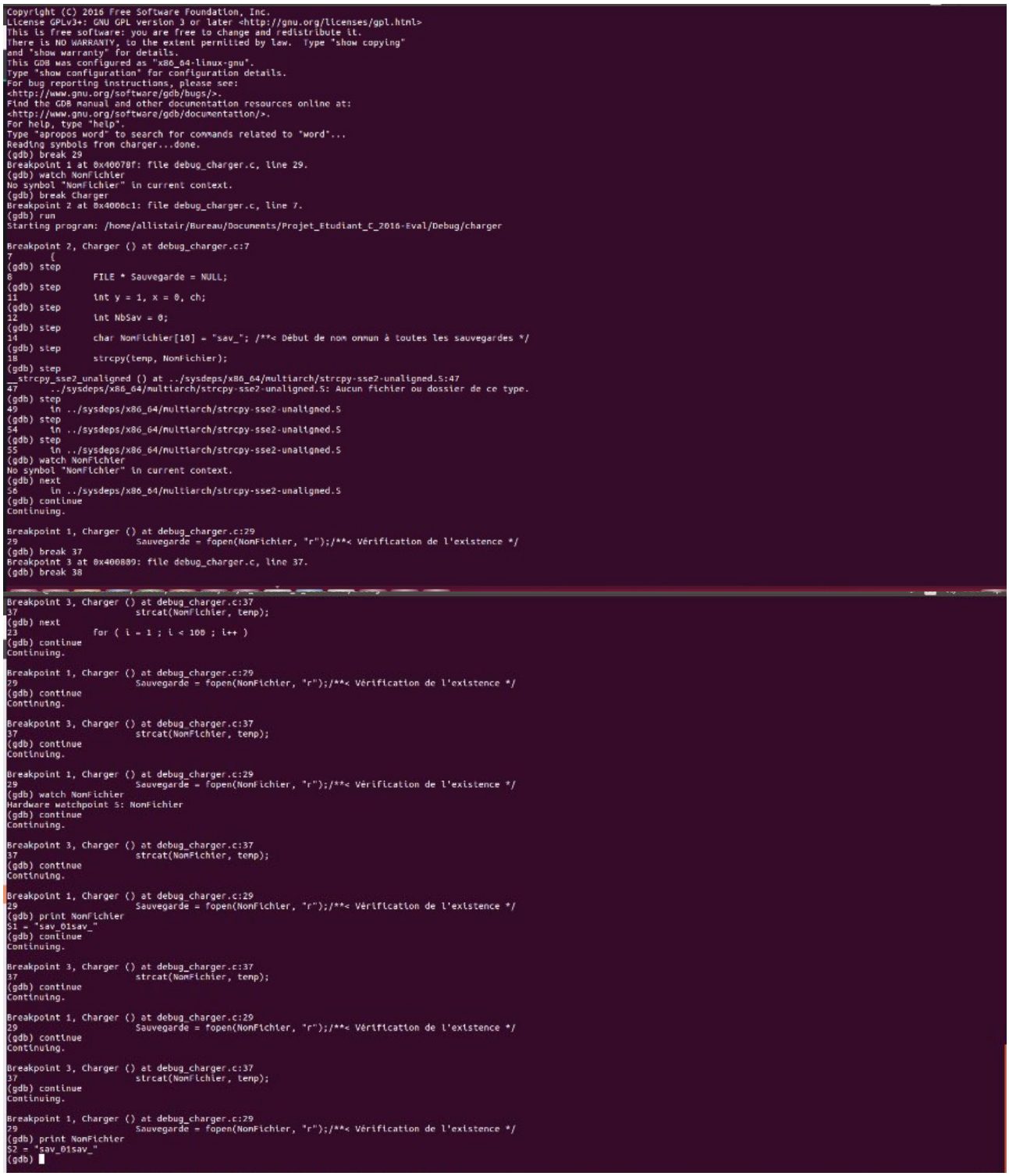
L'IA sera utilisée pour jouer comme un joueur. Pour lancer ses 5 dés de façon optimale, nous calculerons le score qu'il a obtenu dans les précédents lancers. L'utilisation des probabilités sera essentielle pour connaître la meilleure combinaison de dés parmi celle possible selon les dés obtenus dans le lancer. Le programme choisira la meilleure catégorie qui n'a pas été encore rempli et a une des meilleurs probabilités selon les catégories restantes.

- # Tests :

Nous avons effectuer plusieurs phase de debogage dont l'une d'elle sur la fonction CalculScore() pour vérifier si la fonction donnait le résultat attendue vous pouvez voir ces tests sur le lien ci-dessous :

https://raw.githubusercontent.com/FernandesSimon/Projet_Etudiant_C_2016/Eval/tests/Occurrences/test_occurrences.c

Il y a ensuite eu un problemes de coredump dans la recherches des noms de sauvegarde dans la fonction Charger() il se trouve que le problème était dans un strcat qui devait à la base être un strcopy les images de ce test sont ci-dessous :


Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from charger...done.
(gdb) break 29
Breakpoint 1 at 0x40078f: file debug_charger.c, line 29.
(gdb) watch NonFichier
No symbol "NonFichier" in current context.
(gdb) break Charger
Breakpoint 2 at 0x4006c1: file debug_charger.c, line 7.
(gdb) run
Starting program: /home/allistair/Bureau/Documents/Projet_Etudiant_C_2016-Eval/Debug/charger
Breakpoint 2, Charger () at debug_charger.c:7
7 {
(gdb) step
8 FILE * Sauvegarde = NULL;
(gdb) step
11 int y = 1, x = 0, ch;
(gdb) step
12 int NbSav = 0;
(gdb) step
14 char NomFichier[10] = "sav_"; /**< Début de nom commun à toutes les sauvegardes */
(gdb) step
18 strcpy(temp, NomFichier);
(gdb) step
__strcpy_sse2_unaligned () at ../sysdeps/x86_64/multiarch/strcpy-sse2-unaligned.S:47
47 ../sysdeps/x86_64/multiarch/strcpy-sse2-unaligned.S: Aucun fichier ou dossier de ce type.
(gdb) step
49 in ../sysdeps/x86_64/multiarch/strcpy-sse2-unaligned.S
(gdb) step
54 in ../sysdeps/x86_64/multiarch/strcpy-sse2-unaligned.S
(gdb) step
55 in ../sysdeps/x86_64/multiarch/strcpy-sse2-unaligned.S
(gdb) watch NonFichier
No symbol "NonFichier" in current context.
(gdb) next
56 in ../sysdeps/x86_64/multiarch/strcpy-sse2-unaligned.S
(gdb) continue
Continuing.
Breakpoint 1, Charger () at debug_charger.c:29
29 Sauvegarde = fopen(NomFichier, "r");/**< Vérification de l'existence */
(gdb) break 37
Breakpoint 3 at 0x400889: file debug_charger.c, line 37.
(gdb) break 38
Breakpoint 3, Charger () at debug_charger.c:37
37 strcat(NomFichier, temp);
(gdb) next
23 for (i = 1 ; i < 100 ; i++)
(gdb) continue
Continuing.
Breakpoint 1, Charger () at debug_charger.c:29
29 Sauvegarde = fopen(NomFichier, "r");/**< Vérification de l'existence */
(gdb) continue
Continuing.
Breakpoint 3, Charger () at debug_charger.c:37
37 strcat(NomFichier, temp);
(gdb) continue
Continuing.
Breakpoint 1, Charger () at debug_charger.c:29
29 Sauvegarde = fopen(NomFichier, "r");/**< Vérification de l'existence */
(gdb) watch NonFichier
Hardware watchpoint 5: NonFichier
(gdb) continue
Continuing.
Breakpoint 3, Charger () at debug_charger.c:37
37 strcat(NomFichier, temp);
(gdb) continue
Continuing.
Breakpoint 1, Charger () at debug_charger.c:29
29 Sauvegarde = fopen(NomFichier, "r");/**< Vérification de l'existence */
(gdb) print NomFichier
$1 = "sav_01sav_"
(gdb) continue
Continuing.
Breakpoint 3, Charger () at debug_charger.c:37
37 strcat(NomFichier, temp);
(gdb) continue
Continuing.
Breakpoint 1, Charger () at debug_charger.c:29
29 Sauvegarde = fopen(NomFichier, "r");/**< Vérification de l'existence */
(gdb) continue
Continuing.
Breakpoint 3, Charger () at debug_charger.c:37
37 strcat(NomFichier, temp);
(gdb) continue
Continuing.
Breakpoint 1, Charger () at debug_charger.c:29
29 Sauvegarde = fopen(NomFichier, "r");/**< Vérification de l'existence */
(gdb) print NomFichier
$2 = "sav_01sav_"
(gdb)

• Résultats

A la fin du projet, nous avons fait tout ce qu'on a désiré au début du projet. Le fait de travailler ensemble point par point, nous a permis de trouver de grandes erreurs à l'assemblage de toutes les fonctions faites précédemment. Nous avons bien respecté les règles du jeu de base. Au cours du projet, nous avons changé d'avis sur la façon de faire. Mais, globalement, nous sommes restés dans le droit chemin.

Nous voulions intégrer une intelligence artificielle dans le programme, qui aura un rôle de joueurs se battant contre un joueur humain ou une autre IA (qui serait sa jumelle). Une interface graphique SDL est aussi possible. Mais par manque de temps, nous n'avons pas pu le faire avant les vacances.

• Conclusion

Durant ce projet nous avons rencontrer divers problèmes surtout lier à Ncurses puisqu'en effet la bibliothèque Ncurses ne fonctionnnaient pas sur window mais seulement sur linux et mac se qui à été source de recherche sur une autre solution afin que cette solution puisse fonctionner sous tout les appareil et surtout pour que l'on puisse travailler sur Ncurses chez nous avec nos ordinateur fonctionnant sur window.
Malgré ces problèmes rencontrer et la difficultés de Ncurses passés nous avons tout de même réussi a créer un programme qui permet de jouer au jeu du yahtzee, pouvant effectuer des parties avec jusqu'à 4 joueurs, possédant un système de highscore, de sauvegarde, différent menu et explication qui améliore le bon déroulement du jeu.
Par rapport aux objectif premiers qui était de créer un jeu sous SDL nous avons préférer améliorer l'experience de jeu plutôt que le point de vue graphique grâce à Ncurses, nous avions tout d'abord penser mettre les deux mais nous avons remarquer que nous étions dans l'imossibilités d'allier Ncurses et SDL.
Concernant les limites de notre jeu nous pourions dire qu'il est bon pour un projet de 2ème année mais malgré tout il n'est pas suffisant pour le mettre en distribution.
Ce projet nous à apporter une experience de travail de groupe pour un projet de grande envergure (environ 2 mois) il nous à montrer que nous avions une certaine liberté dans le choix de nos fonction qu'il y avait beaucoup de façon d'accomplir un projet tel que celui là mais que la date limite nous empêchais de réaliser toutes nos idées (il fallait apprendre à rester réaliste), il nous à appris à trouver des solutions en groupes à nos problèmes, permis de trouver nos préferences dans les différentes partie à faire du projet.