Quiedeville Valentin
Launay Mathis

Projet C : Le shing shang

I) Les fonctionnalités implémentées

Lors de ce projet, nous avons été amenés à implémenter différentes fonctionnalités, permettant d'une part la jouabilité du jeu en lui-même et d'autre part des fonctionnalités liées au confort d'utilisation du jeu. Commençons par détailler les fonctionnalités qui permettent de rendre le jeu jouable. Nous avons tout d'abord créer une fonction permettant aux joueurs de pouvoir visualiser le plateau, les pions qu'ils possèdent ainsi que les cases où ils peuvent potentiellement se déplacer. Lors du tour du joueur nous avons établi une fonction qui permet de suivre fidèlement les règles du jeu. En effet, ce n'est pas l'utilisateur qui choisit réellement où il va déplacer son pion, mais c'est le programme qui va analyser « l'environnement » du pion que l'utilisateur a choisi, afin de lui proposer les différentes cases possibles sur lesquelles il pourra déplacer son pion. Détaillons maintenant les fonctionnalités qui permettent un confort d'utilisation pour les joueurs. Nous avons implémentés la couleur au sein du programme qui permet au joueur de mieux différencier ses pions avec ceux du joueur adverse, de mieux visualiser le pion choisi et les cases où il pourra se déplacer, cette fonctionnalité apporte donc une meilleure visibilité au joueur. Nous avons également permis au joueur de pouvoir sauvegarder sa partie afin qu'il puisse la reprendre plus tard. Enfin quelques situations pré définies, peuvent être chargées au moment de la création d'une partie, pour s'assurer que le programme est fonctionnel.

II) Organisation du programme

Dans les fichiers bushi.c et bushi.h, on trouve les fonctions et structures relatives aux bushis. Il y a tout d'abord la structure du bushi. Ce bushi est composé d'un « type » numéroté de -2 à 3 comme ceci :

- (-2) ⇒ portail
- (-1) ⇒ case non jouable
- 0 ⇒ case vide
- 1 ⇒ Singe
- 2 ⇒ Lion
- 3 ⇒ Dragon

Le bushi est également composé des attributs « abs » et « ord » qui sont respectivement l'abscisse et l'ordonnée du bushi sur le plateau. Un bushi possède également un attribut « joueur » qui permet de connaître à quel joueur celui-ci appartient. Enfin il est composé d'un attribut « jouable » qui permet de savoir si le bushi peut être joué. Cet attribut permet également de gérer les déplacements, cf commentaire des fonctions deplacement bushi, déplacement singe.
Bushi.h contient également des macros permettant de créer des cases vides ou bloquées plus rapidement lors de l'initialisation du plateau.

Bushi.c contient quant à lui les fonctions nécessaires au jeu et qui ne font intervenir que les bushis en eux même (pas le plateau).

Ce projet est également constitué d'un fichier couleur.h, qui permet d'inclure de la couleur au jeu. On a définit les différentes couleurs par des valeurs.

Dans les fichiers plateau.c et plateau.h, on trouve tous les éléments qui permettent de mettre en place le plateau. On trouve premièrement la structure plateau qui est constitué un tableau de 2 dimensions de bushi. Cette structure est également composé d'un attribut « quiJoue » qui permet de connaître le joueur dont c'est le tour de jouer. Cet attribut est initialisé par un tirage aléatoire qui est réalisé lors de l'appel à la fonction init_plateaubis.

La fonction init_plateaubis prend en paramètre un pointeur sur le plateau et initialise toutes les cases comme des cases vides ou bloquées.

Ensuite on trouve la fonction bushi_joueur qui prend en paramètre le plateau et le numéro du joueur, cette fonction crée un tableau de bushi qui recense tous les bushis du joueur qui sont jouables durant ce tour. La fonction parcourt donc le plateau à la recherche des pions qui possède un attribut joueur correspondant à celui indiqué en paramètre de la fonction, vérifie que ce pion n'est pas un portail (n'est pas de type -2) et qu'il est jouable, et recense ces pions dans un tableau appelé tab. Si la fonction retourne des pions jouables alors on les liste, tant que le joueur saisie une case inexistante du tableau (<0 ou supérieur à la taille du tableau de pions) on lui redemande quel pion il souhaite jouer. Si aucun de ses pions n'est jouable alors on lui dit.

Les déplacements des bushis sont gérer par un ensemble de fonctions (deplacement_bushi,deplacement_singe, deplacement_lion, deplacement_dragon, effectuer_deplacement).
Déplacement_bushi fait appel à la fonction spécifique de déplacement du bushi que l'on souhaite bouger et renvoie la destination que l'on souhaite atteindre. A noter que dans deplacement_bushi, le joueur peut, s'il le souhaite, passer son tour.
Effectuer_deplacement se charge de modifier le plateau afin de rendre effectif le déplacement.

La fonction tour_joueur permet de mettre en place le tour de jeu du joueur. On initialise une variable passeSonTour qui permet au joueur qui ne souhaite pas jouer ce tour-ci de passer au tour suivant. On affiche donc le plateau en faisant appel à la fonction affiche_plateau. Une variable bushiBouge prend le bushi choisi par le joueur. Si il y a effectivement un bushi on affiche les déplacements possibles en faisant appel à la fonction deplacement_bushi. Si les coordonnées de la destination ne sont pas possibles ou si aucun bushi ne peut être joué on indique que cette destination n'est pas possible en mettant l'attribut « jouable » du bushi destination à 0 et passeSonTour à 1.
La fonction a_perdu prend en paramètre le plateau et un joueur. On parcourt le plateau à la recherche de dragons et de portails. On incrémente nbDragons qui comptabilise les dragons et nbPortails qui comptabilise le nombre de portails à chaque fois que l'on trouve un dragon ou un portail (un bushi ayant donc pour type 3 si c'est un dragon ou -2 si c'est un portail). Si après comptage, il n'y a plus de dragons chez le joueur, il perd et on lui indique pourquoi (il ne possède plus de dragons), si le nombre de portails est inférieur à 2 c'est qu'un des deux ou les deux ont été atteints par les dragons adverses donc il perd à cause de cette ou de ces captures de portails.

III) Organisation et répartition des tâches

Lors de ce projet, nous nous sommes répartis les différentes parties du programme. Valentin a travaillé sur les déplacements, sur les différentes fonctions testant la possibilité ou non des déplacements, sur les fonctions d'initialisation du plateau. J'ai également travaillé sur les déplacements, j'ai aussi développer une fonction d'affichage pour le menu principal qui permet d'accéder au différentes « modes » possibles mais que nous n'avons pas retenu dans le projet final, nous l'avons modifié afin d'arriver au résultat final. J'ai travaillé sur les structures nécessaires au bon fonctionnement du programme.

Valentin	Mathis
Déplacements des bushi	Travail sur la fonction Deplacement_bushi
Affichage Plateau	Travail sur l'affichage du menu (non présent dans la version finale)
Fonction de sauvegarde	Gestion des conditions de victoire
Gestion de la boucle de jeu	Travail pour rendre le jeu plus attrayant

IV) Bilan qualitatif du travail, difficultés rencontrées, points qui nous ont paru intéressants

Nous avons trouvé intéressant le fait de récupérer le signal envoyé par le système lors d'un Ctrl+c. Nous avons donc choisi cette méthode afin de sauvegarder la partie en cours de jeu. Toutefois cette technique impose que le plateau de jeu soit global. Ceci nous a obligé à être très prudent lors de nos interactions avec le plateau de jeu, afin de ne pas le modifier de manière intempestive.

La principale difficulté résidait dans le calcul des déplacements du singe, (sachant que les déplacements des autres pièces en découlent). Nous avons établi une boucle permettant de calculer tous les déplacements possibles d'une pièce donnée. Ensuite nous nous sommes attelés au respect des règles de saut. Le fait de fractionner la tâche en plusieurs fonctions nous a permis de franchir cette étape ardue. Cependant nous ne sommes pas parvenus à mutualiser les fonctions déplacements des différents types de bushi.

Le travail en binôme fut très intéressant, il nous a permis de confronter nos idées et d'échanger sur la manière dont le programme devrait fonctionner. Ceci nous a permis de nous remettre en question plusieurs fois dans la conception du projet. Nous avons réinventer la structure du projet, et son fonctionnement général 2 fois avant d'arriver à quelque chose qui nous conviennent, et qui soit aisé à manipuler. L'utilisation de GitHub fut un vrai plus concernant le confort de travail pour s'échanger les fichiers. De plus, le fait qu'il sauvegarde les différentes versions du projet nous a sauvé plusieurs fois.

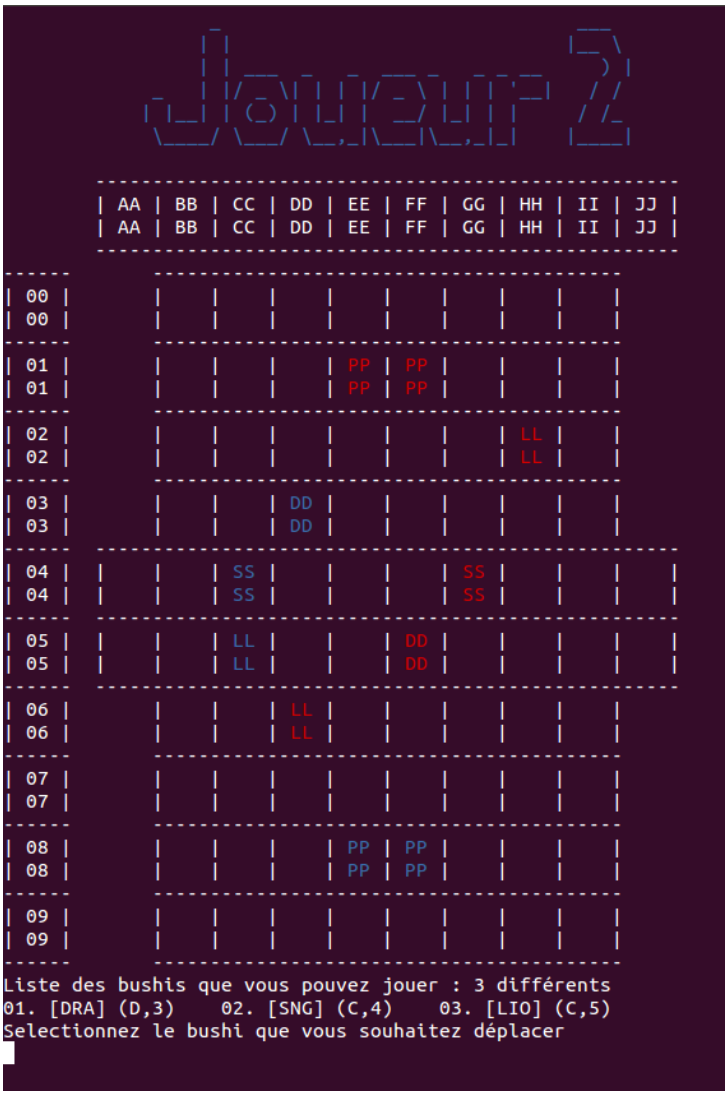
Enfin nous avons découvert le débogueur GDB, et nous nous sommes initiés à son fonctionnement à fin de résoudre quelques de problèmes de "segmentation fault".

V) Mode d'emploi

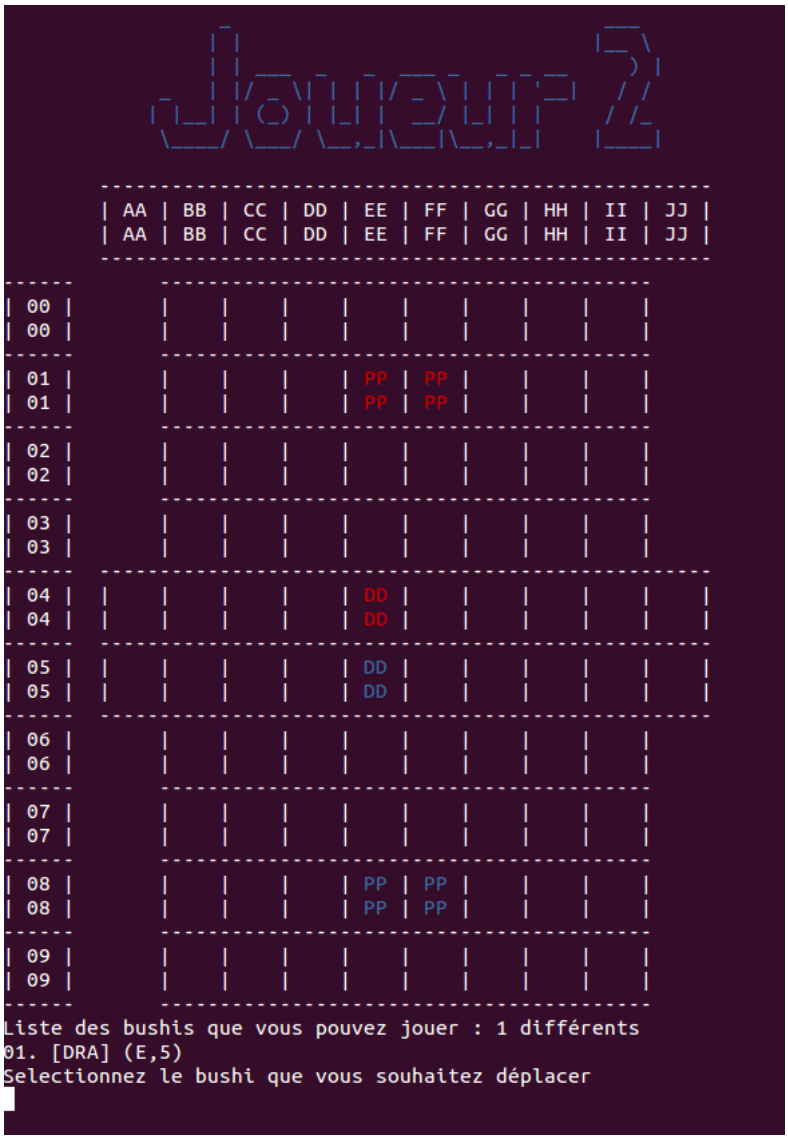
On lance le jeu en tapant dans un terminal « ./ShingShang », si le programme détecte une sauvegarde, il propose au joueur de la charger. L'utilisateur choisit entre charger la partie ou bien en créer une nouvelle. Un tirage est effectué afin de déterminer le joueur qui jouera en premier (ce tirage est effectué seulement lors de la création d'une nouvelle partie et non pas lors du chargement d'une partie existante). Comme une image pertinente vaut mieux que de longs discours voici une image qui illustre le cas en question.

Illustration du menu principal du jeu

Le menu propose également de jouer des situations précises pouvant être rencontrées par le joueur dans le jeu en tapant les identifiant associés à ces situations indiqués dans le menu principal. Certaines de ces situations sont illustrées ci-dessous :



Situation de shing shang pour le dragon bleu



Situation où l'on prend le dernier dragon adverse

Si vous choisissez le lancement d'une partie standard, un plateau sera initialisé, des bushis seront placés soit à leur position initiale en cas de nouvelle partie, soit la position à laquelle il était placés lors de l'arrêt de la partie précédente.

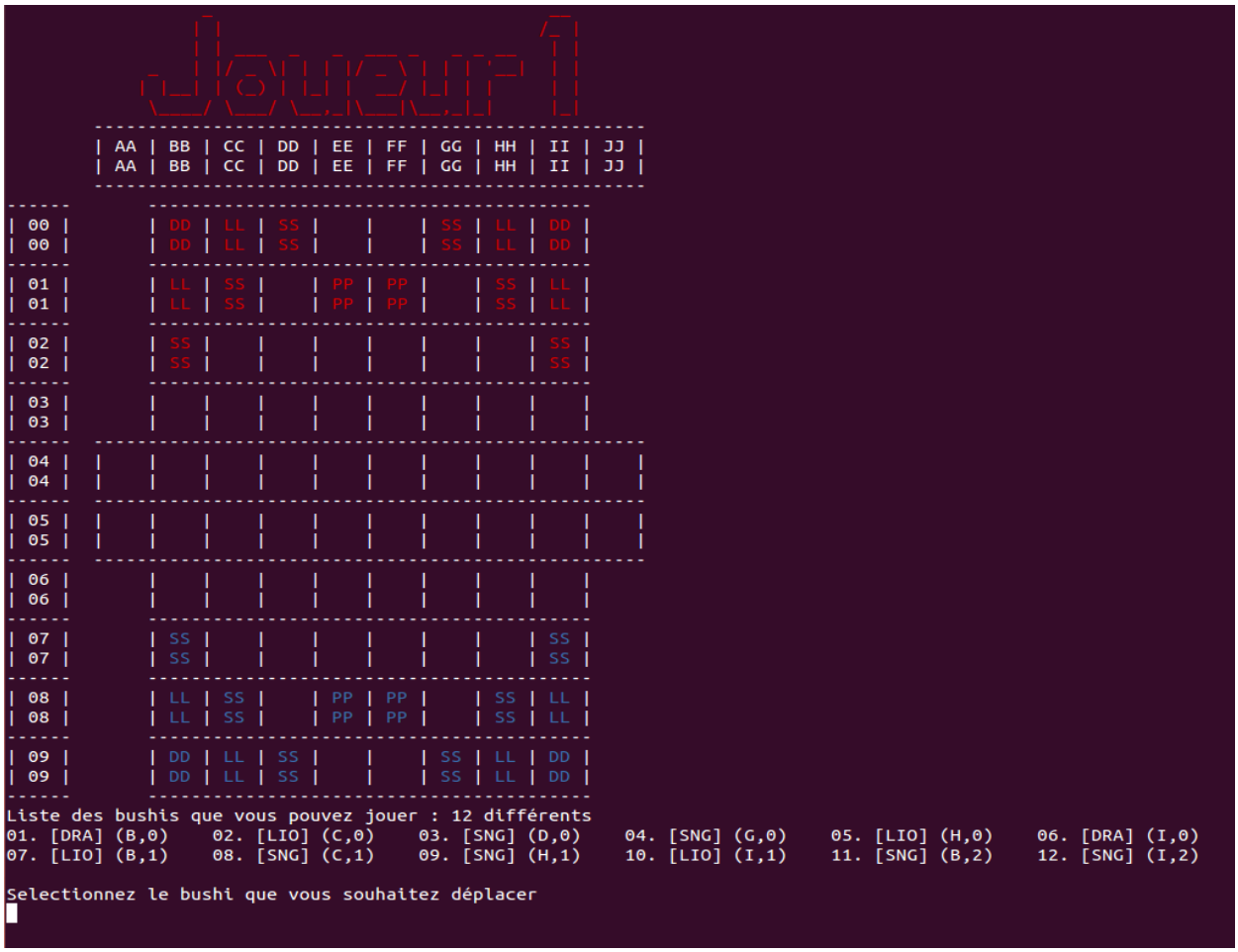


Illustration d'un plateau lors du lancement d'une nouvelle partie

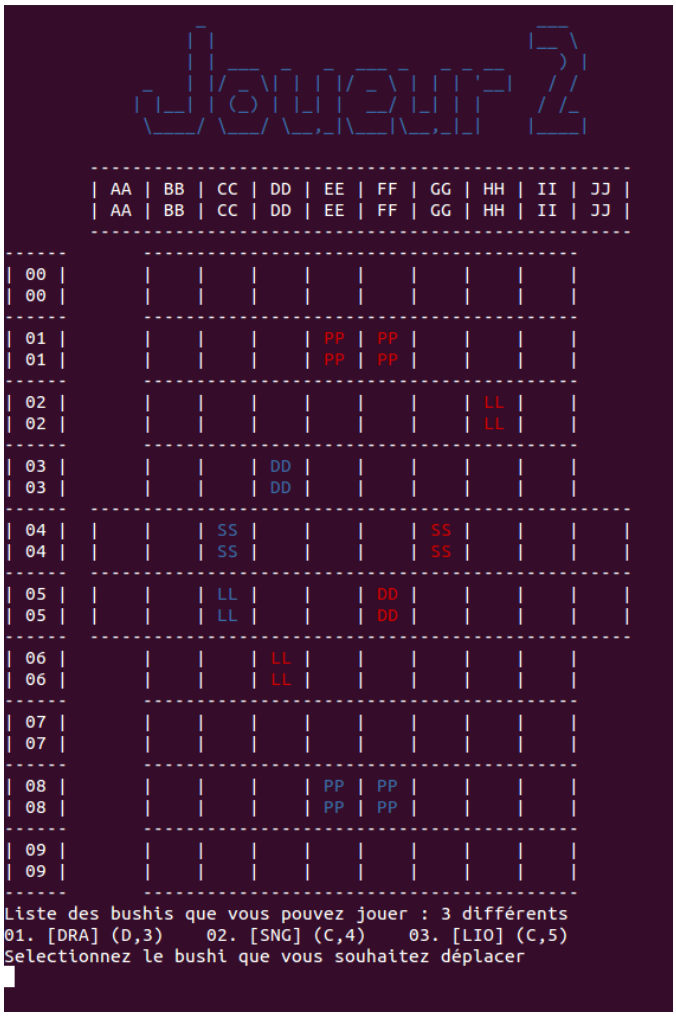
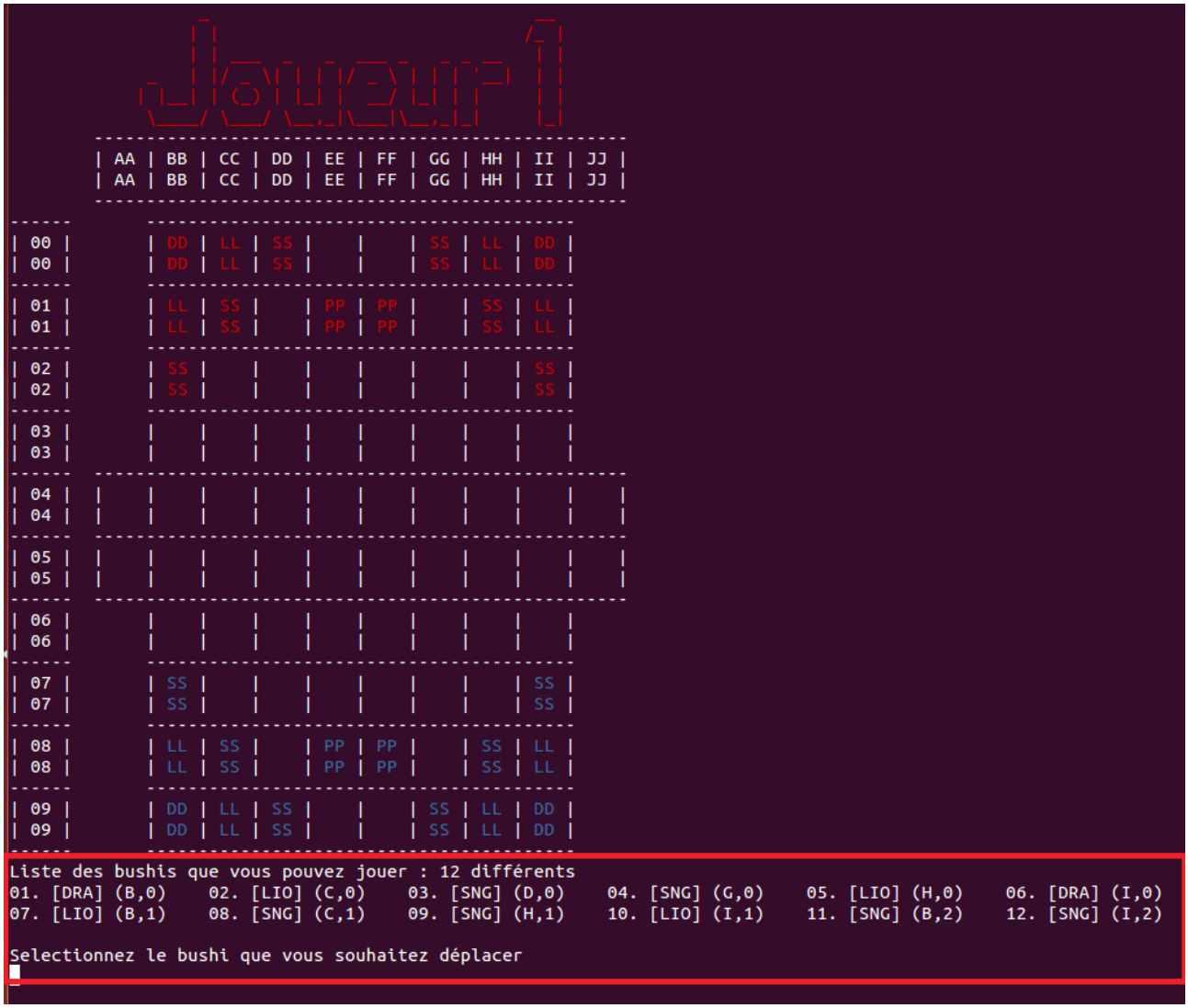
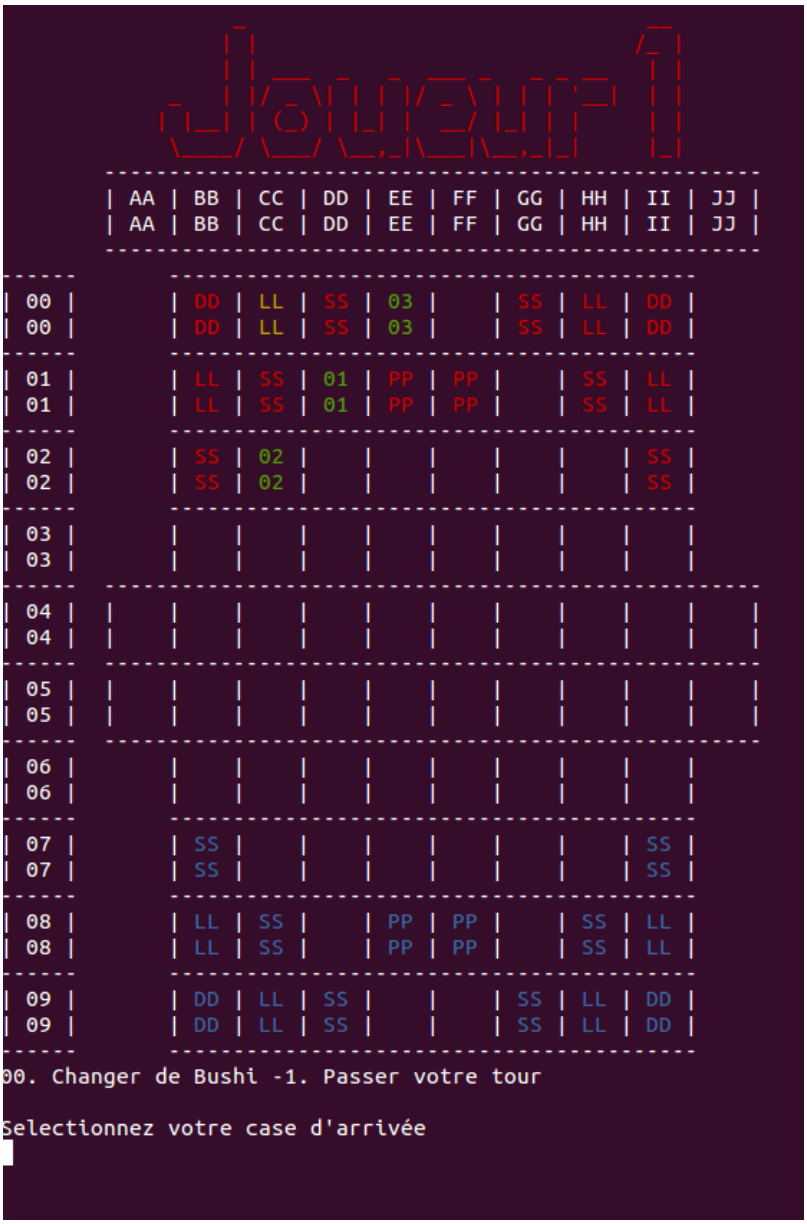


Illustration du lancement d'une partie commencé précédemment.

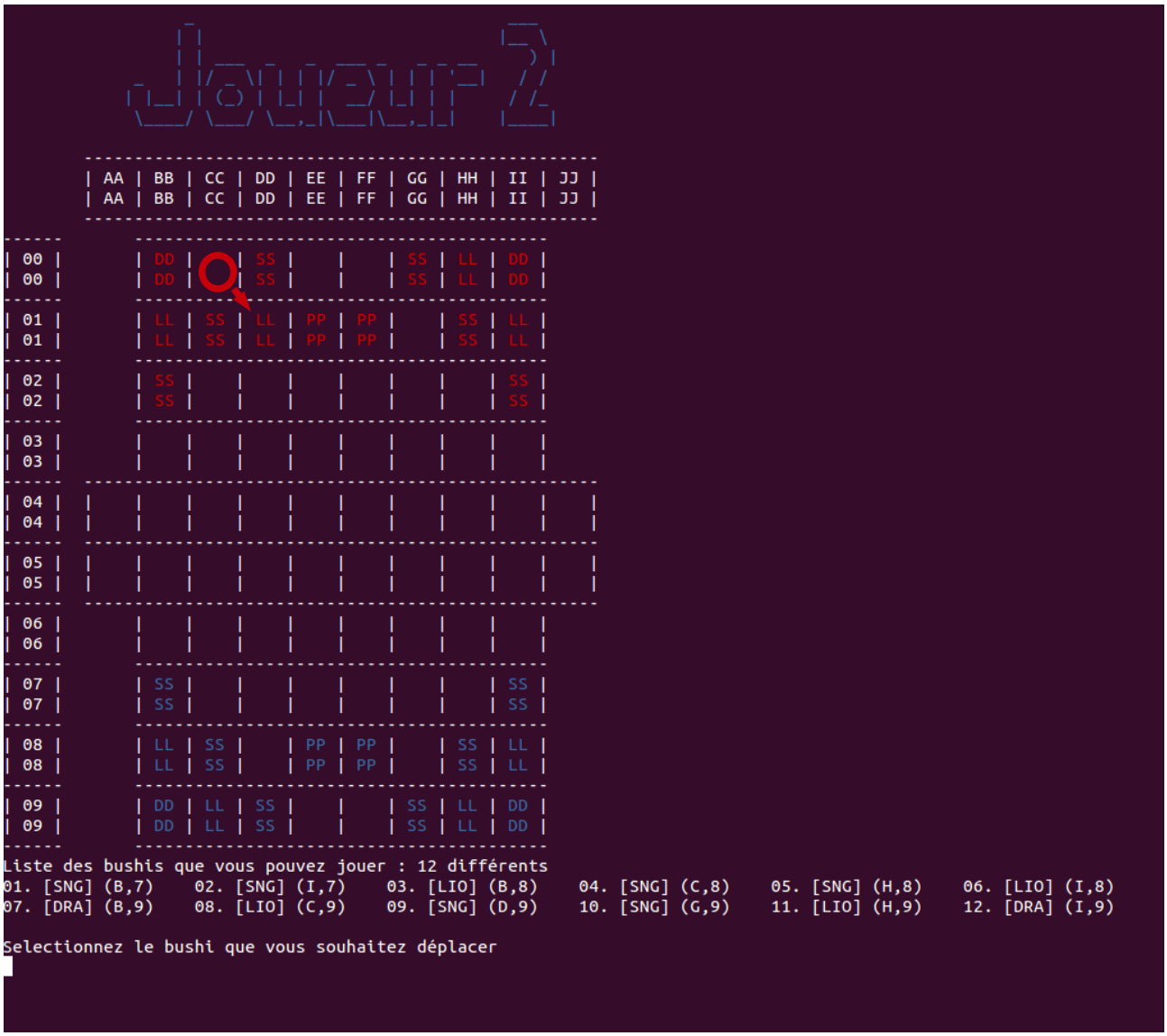
Le programme vous demande quel bushi vous souhaiter déplacer puis vous propose une liste des déplacement possibles.



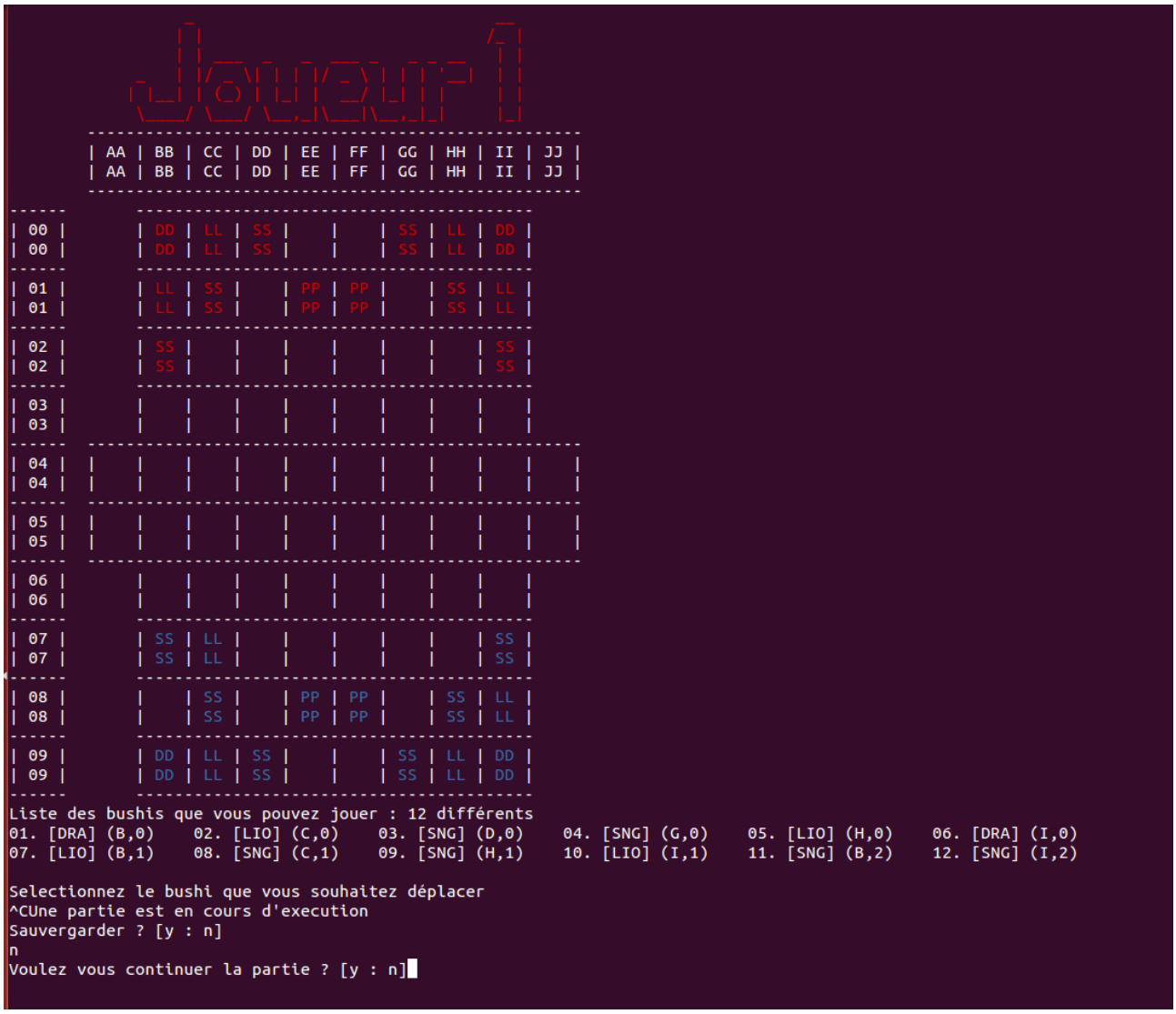
Encadré en rouge, la liste des bushis jouables et la demande du jeu pour savoir quel bushi le joueur souhaite jouer.



Sur l’illustration ci-dessus, le joueur a choisi le lion situé en C0, le jeu lui propose les différentes cases colorées en vert où son bushi lion peut se déplacer. Et une fois la case choisi on obtient ceci:



Lors de la sélection de la destination d'un bushi si vous souhaitez changer de bushi vous pouvez tapez 0, vous pouvez également passez votre tour en tapant -1. Si vous souhaitez sauvegarder votre partie, vous pouvez tapez «ctrl+c» le programme vous propose de sauvegarder la partie ce qui écrasera une partie potentiellement sauvegardé précédemment, puis vous pouvez choisir de continuer la partie ou la quitter. La sauvegarde est illustrée sur l’image ci-dessous:



Une fois tous ces mécanismes pris en main, gérez vos bushis pour vous emparez de la victoire.

Affichage de fin de jeu dans la cas de la victoire du second joueur

V) Conclusion

Ce projet nous a permis de nous améliorer de manière significative en langage C. Nous avons progressé entre autre dans la compréhension et l'utilisation des pointeurs et structures. Nous avons découverts la gestion des entrées sorties ainsi que les pointeurs sur fonctions, et la récupération d'un signal envoyé par le système.

D'un point de vue humain et travail en groupe, la démarche fut une première. En effet ce n'est pas un exposé classique ou chacun s'occupe d'une partie presque indépendamment des autres. Il faut constamment se parler, échanger sur nos difficultés et sur les modifications que l'on a dû faire par rapport à l'idée que l'on s'était faite de la fonction que nous sommes en train de coder, c'est selon nous, ce qui fait la richesse, du point de vue humain, du projet d'algorithmique en langage C.